SE 463
Final System Requirements Specification Deliverable
# Table of Contents

1 Introduction ................................................................................................................................. 5  
  1.1 Purpose .................................................................................................................................. 5  
  1.2 Scope .................................................................................................................................... 5  
  1.3 Definitions, acronyms, abbreviations .................................................................................. 6  
  1.4 References ............................................................................................................................ 6  
  1.5 Overview ............................................................................................................................... 6  
2 Overall description ........................................................................................................................................ 7  
  2.1 Product perspective ............................................................................................................. 7  
  2.2 Product features .................................................................................................................... 8  
  2.3 User characteristics ............................................................................................................... 9  
  2.4 Constraints ............................................................................................................................ 9  
  2.5 Assumptions and dependencies .......................................................................................... 10  
3 Specific Requirements .................................................................................................................................. 11  
  3.1 External Interfaces .............................................................................................................. 11  
    3.1.1 User Interfaces ............................................................................................................. 11  
    3.1.2 Hardware Interface Event Mapping ............................................................................. 27  
      3.1.2.1 Hardware Input Signals ......................................................................................... 27  
      3.1.2.2 Hardware Output Signals ....................................................................................... 27  
      3.1.2.3 Hardware Output Activities ................................................................................... 28  
  3.2 Functional Requirements ..................................................................................................... 29  
    3.2.1 Use Cases ..................................................................................................................... 29  
      Making a Call ..................................................................................................................... 29  
      Charging a Monthly Service Fee ....................................................................................... 31  
      Recording a Bill Payment ................................................................................................. 32  
      Editing a Phone Account ................................................................................................. 33  
      Deleting a Phone Account ............................................................................................... 34  
      Adding a User Account ................................................................................................. 35  
      Deleting a User Account ............................................................................................... 36  
      Editing a Billing Plan ................................................................................................. 37  
      Adding an Administrator Account ................................................................................ 38  
      Deleting an Administrator Account ............................................................................. 38  
      Adding Filter Expressions .............................................................................................. 39  
      Adding a Blocked Extension ............................................................................................ 40  
      Adding an IP Address ....................................................................................................... 41  
      Running a Hardware Test ............................................................................................... 41  
      Starting the System .......................................................................................................... 42  
    3.2.2 Domain Model ............................................................................................................. 43  
    3.2.3 Functional Specifications ............................................................................................ 44  
      Making a Call .................................................................................................................. 44  
      Ending a Call .................................................................................................................... 45  
      Charging For Part of a Call .............................................................................................. 46  
      Charging Monthly Service Fee ....................................................................................... 48  
      Recording Bill Payment ................................................................................................. 49  
      Prorating a Service Fee ................................................................................................. 50  
      Editing a Phone Account ............................................................................................... 51  
      Deleting a Phone Account ............................................................................................ 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding a User Account</td>
<td>55</td>
</tr>
<tr>
<td>Deleting a User Account</td>
<td>56</td>
</tr>
<tr>
<td>Editing a Billing Plan</td>
<td>58</td>
</tr>
<tr>
<td>Adding an Administrator Account</td>
<td>59</td>
</tr>
<tr>
<td>Deleting an Administrator Account</td>
<td>60</td>
</tr>
<tr>
<td>Adding an Incoming Filter Expression</td>
<td>61</td>
</tr>
<tr>
<td>Adding a Blocked Extension</td>
<td>62</td>
</tr>
<tr>
<td>Adding an IP Address</td>
<td>63</td>
</tr>
<tr>
<td>Running a Hardware Test</td>
<td>63</td>
</tr>
<tr>
<td>3.2.4 State Machine Models</td>
<td>65</td>
</tr>
<tr>
<td>Phone System</td>
<td>65</td>
</tr>
<tr>
<td>Administration</td>
<td>66</td>
</tr>
<tr>
<td>Administrator’s Account</td>
<td>67</td>
</tr>
<tr>
<td>Editing an Administrator Account</td>
<td>68</td>
</tr>
<tr>
<td>Remove Administrator Account</td>
<td>69</td>
</tr>
<tr>
<td>Add New Administrator</td>
<td>70</td>
</tr>
<tr>
<td>Configure Billing Plans</td>
<td>71</td>
</tr>
<tr>
<td>Enter Billing Plan Details</td>
<td>72</td>
</tr>
<tr>
<td>Configuring System</td>
<td>73</td>
</tr>
<tr>
<td>Request System Test</td>
<td>74</td>
</tr>
<tr>
<td>Calls in System</td>
<td>75</td>
</tr>
<tr>
<td>Configuring Maximum Number of Calls</td>
<td>76</td>
</tr>
<tr>
<td>IPs</td>
<td>77</td>
</tr>
<tr>
<td>Configure Emergency Extension</td>
<td>78</td>
</tr>
<tr>
<td>Modify User Account</td>
<td>79</td>
</tr>
<tr>
<td>Add User Account</td>
<td>80</td>
</tr>
<tr>
<td>Modify Phone Account</td>
<td>81</td>
</tr>
<tr>
<td>Add Phone Account</td>
<td>82</td>
</tr>
<tr>
<td>Configure Filtered Extensions</td>
<td>83</td>
</tr>
<tr>
<td>Phones</td>
<td>84</td>
</tr>
<tr>
<td>Phones In Service</td>
<td>85</td>
</tr>
<tr>
<td>Call Blocking</td>
<td>86</td>
</tr>
<tr>
<td>Testing</td>
<td>87</td>
</tr>
<tr>
<td>3.3 Performance</td>
<td>88</td>
</tr>
<tr>
<td>3.4 Design Constraints</td>
<td>90</td>
</tr>
<tr>
<td>3.5 Quality Attributes</td>
<td>93</td>
</tr>
<tr>
<td>Appendix A - Glossary</td>
<td>97</td>
</tr>
<tr>
<td>Basic Terms</td>
<td>97</td>
</tr>
<tr>
<td>Domain Model</td>
<td>99</td>
</tr>
<tr>
<td>Condition Functions</td>
<td>102</td>
</tr>
<tr>
<td>Action</td>
<td>104</td>
</tr>
<tr>
<td>Events</td>
<td>109</td>
</tr>
<tr>
<td>GUI Screenshots</td>
<td>111</td>
</tr>
<tr>
<td>Functions</td>
<td>114</td>
</tr>
<tr>
<td>States</td>
<td>116</td>
</tr>
<tr>
<td>Variables</td>
<td>125</td>
</tr>
<tr>
<td>Appendix B – Meeting Minutes</td>
<td>127</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1 – Navigability diagram................................................................................................... 11
Figure 2 – Use Cases Diagram................................................................................................... 29
Figure 3 - Domain class diagram.............................................................................................. 43
Figure 4 - Phone System SSD.................................................................................................. 65
Figure 5 - Administration SSD............................................................................................... 66
Figure 6 - Administrators Account SSD.................................................................................. 67
Figure 7 - Editing an Admin Account SSD............................................................................. 68
Figure 8 - Remove Admin Account SSD................................................................................ 69
Figure 9 - Add New Administrator SSD.................................................................................. 70
Figure 10 - Configure Billings Plans SSD................................................................................ 71
Figure 11 - Enter Billing Plan Details SSD............................................................................. 72
Figure 12 - Configuring System SSD...................................................................................... 73
Figure 13 - Request System Test SSD.................................................................................... 74
Figure 14 - Calls In System SSD............................................................................................ 75
Figure 15 - Configuring Maximum Number of Calls SSD...................................................... 76
Figure 16 - IPs SSD.................................................................................................................... 77
Figure 17 - Configure Emergency Extension SSD................................................................. 78
Figure 18 - Modify User Account SSD................................................................................... 79
Figure 19 - Add User Account.................................................................................................. 80
Figure 20 - Modify Phone Account SSD.................................................................................. 81
Figure 21 - Add Phone Account SSD...................................................................................... 82
Figure 22 - Configure Filtered Extensions SSD...................................................................... 83
Figure 23 - Phones SSD............................................................................................................ 84
Figure 24 - Phone_inService SSD.......................................................................................... 85
Figure 25 - Call Blocking SSD............................................................................................... 86
Figure 26 - Testing SSD............................................................................................................ 87
1 Introduction

1.1 Purpose
This document describes the software requirements and specification for a small telephone exchange and its associated information system.

The document is intended to be viewed by the customer and the developers of the system (designers, testers, maintainers). Readers are assumed to have a sound understanding of the workings of the underlying phone system. Readers also require an understanding of UML 2.0 diagrams.

Once approved by all primary stakeholders, this document serves as a contract specifying the functional and non-functional requirements of the telephone exchange information system.

1.2 Scope
The described software system, entitled “SkyNet”, will ensure that the underlying telephone system operates in such a way that all customer-provided requirements are satisfied. The system facilitates basic call processing and ensures that the appropriate charges are applied to customers. Customers have individual accounts in the system and will be able to have multiple phone accounts associated to them. Customers will be able to use blocked lists and incoming/outgoing filters in order to customize their phone usage experience.

The system will uniquely identify phones by their assigned extension and IP address. The system can handle emergency extensions and assign them to available phones.

The system allows administrators to manage all customer accounts in the system and their associated phones. Administrators can issue bills and record customer payments, and also manage the billing plans in the system. The system must handle concurrent access to customer and phone accounts appropriately.

The system is not concerned with connecting to any other systems. The system does not implement any form of replication or redundancy. The security features of the administration interface are limited to an administrator username and password.

This document does not describe the physical interface of the phones in the telephony exchange. Payment processing and the delivery mechanism used for bills are also out of the scope of this system.
1.3 Definitions, acronyms, abbreviations

**Dedicated server:** A computer used exclusively as a network server.

**GUI:** Graphical User Interface - An UI based on visual shapes and icons rather than only text.

**Payment:** An amount of money that is transferred from a system customer to the system operator.

**SRS:** Software Requirements Specification - a document that specifies the functional and non-functional requirements of a system, often serving as a contract between customer and software provider.

**SSD:** System State Diagram - models the transitions and states within the system

**UC:** Use Case - a typical scenario of how an end user may use the system

**UI:** User Interface - provides affordances for the user to interact with the system.

**VoIP:** Voice over Internet Protocol - a process for transmitting telephone calls and voice data over an Internet Protocol based network.

1.4 References

**Customer Meetings**
The set of minutes taken during private group meetings with the customer/TA. Please see Appendix B.

**Customer Press-conference Minutes**
The set of minutes taken during the press-conference style meetings with the customer. Please see Appendix B.

**Software Engineering Sequence SE1 SE2 SE3: Overview of the Course Project**
Project overview document made available on the course website. Please see: http://www.student.cs.uwaterloo.ca/~cs445/Fall2006/Project/se1-2-3-project-overview.pdf

1.5 Overview

Going forward, this document is structured as follows:
Section 2 contains an overall description of the aforementioned software system, including a list of features. This section also contains a list of the various factors affecting the requirements placed on the system.

Section 3 contains a set of functional and non-functional system specifications. These include use cases, user interface diagrams, function tables and a collection of non-functional requirements.

Appendix A contains the document glossary, which lists and defines various domain-specific terms, states, classes, attributes, operations, events, activities, and variables used throughout this specification.

Appendix B contains a complete set of minutes taken during press conferences and customer/TA meetings.

The document appendices are used as supporting material throughout the document and do not constitute extra requirements of the software.

2. Overall description
This section of the document describes the context within which the system will operate, as well as the various factors that affect the requirements of the system.

2.1 Product perspective
The software system works together with the customer-selected telephony VOIP hardware in order to perform its functions. The software will consist of two parts – one which resides on a central server and a second part which resides on each phone subsystem.

The software interacts with the following hardware components:

- Phones – Software is embedded within each phone subsystem to facilitate call processing. The phone subsystem software transmits voice data via the central server.
- Central Server – This server contains the majority of the software solution. It communicates with each phone subsystem to connect calls and to ensure that calls are properly recorded in the system for billing purposes. This part of the system also keeps track of all user accounts, phone accounts and billing information.
- Terminals – The terminals communicate with the central server in order to display the system console to the administrators.

The only user interface component provided by the software will be the system console. The system console is displayed on terminals connected to the central server.

The context in which SkyNet operates is shown below in Figure 1.
2.2 Product features

The primary functions of the system include call processing, billing, and administrator access to the system. In more detail, the system must support the following:

- Basic call processing – The software facilitates the establishing of calls between phones in the telephone system. This involves the establishment and termination of audio connections between phones as well as dealing with any problems which might arise.

- Phone mapping – Phone IP addresses are mapped to four digit extensions for use by customers. The software performs the necessary conversions between IPs and extensions in order to successfully connect customer calls. Administrators can add and remove specific IP addresses from the system.

- Emergency extensions - Some extensions are used as special numbers and can not be associated with customers.

- Administrator console – The software provides a graphical user interface to system administrators which will allow them to manage the telephone system. Administrators must log into the console in order to perform any functionality.

- User and phone accounts – The system represents all customers by associating user accounts to them (in such a way that one customer can have only one user account). Each user account can have multiple phone accounts and each phone account can be associated with at most one phone.

- Adding users to the system – Administrators are able to add new users to the system by creating user accounts for them

- Deleting users from the system – Administrators are able to remove user accounts from the system

- Editing phone accounts – Administrators are allowed to edit any phone accounts belonging to a user.

- Deleting phone accounts – Administrators may remove a phone account that is associated with a particular user

- Performing system tests – The system is able to run hardware tests (both automatically and on administrator request) in order to determine if there are phones in the system with malfunctioning hardware. If such phones are found, the system alerts the administrators as soon as possible.
• View system error logs – Administrators can view the complete set of errors that have occurred in the system
• System maintenance – Administrators can change the state of phones within the system by enabling or disabling them.
• System state management – Administrators can start and stop the telephone system at will.
• Ongoing call termination – Administrators can terminate any currently ongoing calls.
• Load balancing – The system allows administrators to set a maximum allowable concurrent calls boundary. If this maximum is reached, the system disallows the processing of any new calls.
• Manage billing plans – Administrators can add, edit and remove billing plans. A billing plan is assigned to each phone account, giving the base rate to be used for charging calls and the service charge to be levied against the phone account. A billing plan may also have one or more discount periods within which the base call rate is discounted by a given percent.
• Charging for calls – user’s phone accounts are charged for any calls
• Viewing bills - An administrator may view the bill for any billing period and any phone account belonging to a user.
• Record bill payments – The system allows an administrator to record bill payments against a user’s phone account.
• Apply service fees – The system applies service charges to users at the end of each billing period. Phone accounts with outstanding balances for longer than 90 days are automatically suspended.
• Manage users’ filter lists – Administrators are able to add and remove phone extensions from the incoming and outgoing filter lists of specific phone accounts.
• Manage administrators – Administrators are able to add, edit and delete other administrator accounts within the system.
• Blocked number management – Users are able to specify a set of extensions which are blocked from calling the user. Users are able to manually add and remove extensions from their blocked list.

2.3 User characteristics

The following two kinds of users are envisioned to be using the system:
• Customers – these are users with sufficient knowledge of proper phone usage. Besides the knowledge of how to perform basic phone functionality (such as initiating and receiving a call), these users may have some training if they have call blocking as an option for at least one phone accounts.
• Administrators – these are individuals who are entrusted by the customer to manage the telephone system. They are assumed to have a relatively advanced knowledge of telephony, computer systems and the chosen hardware platform in particular. These individuals have been trained in how to use the software (particularly the administrator user interface) and are familiar with all its aspects.

2.4 Constraints

The performance of the system is restricted by the capabilities of the underlying server hardware.
Multiple administrators must be able to use the system concurrently. The system ensures that concurrent access does not corrupt the integrity of the system data.

Administrators must authenticate with the system by using a username and password.

Administrators can not be logged in at two terminals at the same time.

Administrators must be automatically logged out after 10 minutes of inactivity.

The source code of the system must always be available to the customer.

2.5 Assumptions and dependencies
The software system specifications presented in this document are based on the following assumptions:

- The computer systems running the server and the system administration console never fail
- The customer is responsible for the deployment of the software and all related hardware components
- All the necessary external precautions against physical damage have been taken
- The telephone system can never experience a sudden loss of power
- Daylight savings time does not affect the billing of the system
- No person will maliciously try to gain access to the administrator console
3 Specific Requirements
This section describes in detail the requirements and specifications for the telephone system software.

3.1 External Interfaces

3.1.1 User Interfaces

1) **Input Button (Turn on)** - Button which initiates the startup of the phone system.
   
   *Events Triggered: turnSystemOn (sent to all administrators)*

2) **Input Button (Log Out)** - Button which logs out the current administrator
   
   *Events Triggered: clickLogout*
2: Login administrator

![Login Screen](image)

1) **Text box (Username)** - Text box for the administrator to type in their user name.
2) **Text box (Password)** - Holds the password entered by the administrator. Characters are denoted by *'s to hide the exact entry.
3) **Input Button (Login)** - Clicked by an administrator to initiate login to the system.
   
   *Events Triggered: clickLogin(loginInfo*)

   *loginInfo refers to the username/password pair

3: Generic Error/Warning Message - Omitted for space reasons.

4: Main screen – Omitted for space reasons.

5: Administrator Management Screen – Omitted for space reasons.

6: Edit administrator list

![Edit Administrator Screen](image)

1) **Combo Box (UserName)** - Dropdown is populated with the username of all administrator accounts currently in the system.
2) **Input Button (Edit Administrator)** - Initiates the modification of an administrator account.

   *Events Triggered: clickChange(adminName*)

3) **Input Button (Cancel)** - Cancels the edit an administrator accounts.

   *Events Triggered: clickCancel

   * Parameter adminName is provided by the selection in combo box 1.

7: Edit administrator details

![Edit Administrator Details Screen](image)

1) **Text Box (Full Name)** - Contains the full name of the administrator allowing for it to be changed

   *
2) **Text Box (Username)** - Contains the username of the administrator allowing for it to be changed
3) **Text Box (Password)** - The new password for the administrator.
4) **Input Button (Save)** - Initiates the commitment of the change to the administrator account information.
   
   *Events Triggered: clickSave(adminDetails*)
5) **Input Button (Cancel)** - Initiates a cancellation of the modification of an administrator account.
   
   *Events Triggered: clickCancel

*adminDetails is the (possibly) new full name (1), username (2) and password (3) of the admin.

8: **Remove administrator list**

![Remove Administrator List Screen]

1) **List Box (List of Usernames)** - Displays a list of all administrators excluding the administrator who invoked the remove administrator functionality.
2) **Input Button (Remove Selected)** - Initiates process to remove selected administrators
   
   *Events Triggered: clickDeleteAdmin(adminRemoveArray[]*)
3) **Input Button (Cancel)** - Cancel initiates canceling the remove administrator operations.
   
   *Events Triggered: clickCancel

* Parameter adminRemoveArray is defined by the selections in element 1

9: **Remove Administrator Confirmation** – *Omitted for space reasons.*

10: **Add new administrator**

![Add New Administrator Screen]

1) **Text Box (Username)** - Contains the entry of the administrator's username.
2) **Text Box (Password)** - Defines the password for the administrator.
3) **Text Box (Full Name)** - Contains the full name of the administrator.
4) **Input Button (Add)** - Initiates the addition of the defined administrator account.
Events Triggered: clickAdd(adminData*)

5) **Input Button (Cancel)** - Initiates a cancellation of the addition of the administrator account.
   *Events Triggered: clickCancel*
   *adminData includes the username (1), password(2) and full name(3)*

11: **Add New Administrator Confirmation** – Omitted for space reasons.

12: **Configure Billing Plans**

1) **Combo Box (Billing Plans)** - A list containing the names of all the available billing plans in the system.

2) **Input Button (Edit)** - Initiates the modification of a billing plan.
   *Events Triggered: clickEdit(billingPlan*)*

3) **Input Button (Delete)** - Initiates the deletion of a billing plan.
   *Events Triggered: clickDeleteBillingPlan(billingPlan*)*

4) **Input Button (Add Billing Plan)** - Initiates the addition of a billing plan.
   *Events Triggered: clickAddNewBillingPlan*

5) **Input Button (Cancel)** - Cancels the selection of billing plan options.
   *Events Triggered: clickBack*

*billingPlan is the billing plan selected in combo box 1.*
13: Edit Billing Plan

1) **Text Box (Charge Rate)** - Defines the charge per minute for the billing plan.
2) **Text Box (Service Charge)** - Defines the service charge for the billing plan.
3) **Check Box (Call Blocking)** - Specifies whether or not the billing plan has the call blocking feature.
4) **Period Definition Widget (Discount Periods)** - Defines a day of the week and a range of hours on that day (the time period for the discount plan).
5) **Input Button (Add discount period)** - Initiates the addition of the discount period defined in 4 & 6 to the billing plan.
   *Events Triggered:* `clickAddNewDiscountPeriod(period*)`
6) **Text Box (Discount Rate)** - Defines the percentage discount rate for the discount period.
7) **List Box (Discount Periods)** - A list containing all of the discount periods associated to the billing plan.
8) **Input Button (Remove Discount Period)** - Initiates the removal of the discount period defined in 7 from the billing plan.
   *Events Triggered:* `clickDeleteDiscountPeriod(period*)`
9) **Input Button (Cancel)** - Initiates the cancellation of billing plan modifications.
   *Events Triggered:* `clickCancel`
10) **Input Button (Save)** - Initiates a save of the modifications made to the billing plan.
    *Events Triggered:* `clickSaveBillingPlan(billingPlan**)`
11) **Text box (Billing Plan Name)** - Defines the name of the billing plan.

* period includes a time from, a time to, a day of the week (4) and a discount rate (6).
** billingPlan includes the set of discount periods(7), name (11), regular charge rate(1), monthly service charge(2) and call blocking available(3)
14: System Management

![Screen 14 - System Management](image)

1) **Input Button (Request Hardware Test)** - Moves the user to the request hardware test UI screen.
   
   *Events Triggered:* clickRequestSystemTest

2) **Input Button (Ongoing Phone Calls)** - Moves the user to the ongoing phone calls UI screen.
   
   *Events Triggered:* clickCallsInSystem

3) **Input Button (Change Maximum Number of Calls)** - Moves the user to the change maximum number of calls in system UI screen.
   
   *Events Triggered:* clickMaxCallsInSystem

4) **Input Button (View Error Logs)** - Moves the user to the view error logs UI screen.
   
   *Events Triggered:* clickViewErrorLogs

5) **Input Button (Configure IPs)** - Moves the user to the configure IPs UI screen.
   
   *Events Triggered:* clickIPs

6) **Input Button (Configure Emergency Extensions)** - Moves the user to the configure emergency extensions UI screen.
   
   *Events Triggered:* clickConfigureEmergencyExtensions

7) **Input Button (Back to main)** - Moves the user back to the main screen
   
   *Events Triggered:* clickBack

15: View System Error Logs – *Omitted for space reasons.*

16: Select Telephone for Hardware Testing

![Screen 16 - Selects Telephone Accounts for Hardware Testing](image)

1) **Checkbox list (Phone Accounts)** - A list of all extensions of all phone accounts in the system.
2) **Input Button (RunTests)** - Initiates the start of hardware tests on phones defined in element 1.
   *Events Triggered: clickRunTests(adminID*, IPs**)  
3) **Input Button (Cancel)** - Cancels the selection of tests to run.
   *Events Triggered: clickCancel  
4) **Input Button (Select All)** - Initiates the selection of all checkboxes in element 1.  
5) **Input Button (Unselect All)** - Initiates the deselection of all checkboxes in element 1.  
   *adminID is the username of the currently logged in admin  
   **IPs holds all phones defined in element 1

17: **View Max Calls in System**

![Screen 17 - View Number of Calls in System](image)  

1) **Text field (Maximum number of calls)** - This box displays the current maximum number of allowed calls in the system.  
2) **Input Button (Edit)** - Takes the user to the edit maximum number of calls page.  
   *Events Triggered: clickEdit  
3) **Input button (Go back)** - Go to the previous screen.  
   *Events Triggered: clickBack

18: **Set max calls in system**

![Screen 18 - Setting Maximum Number of Calls in System](image)  

1) **Text box (Maximum number of calls)** - The user enters the new maximum number of calls in this text box.  
2) **Input button (Ok)** - Attempts to set a new maximum number of allowed calls in the system.  
   *Events Triggered: clickOK(num*)  
3) **Input button(Cancel)** - Cancels modification and returns to the page that displays the maximum number of calls.  
   *Events Triggered: clickCancel  
   * Parameter num is the number value entered in Text box 1.

19: **Terminating an Ongoing Call**
1) **Selection List (List of active calls)** - This box contains a list of all active calls. Exactly one active call can be selected at a time.

2) **Input Button (Terminate)** - Initiates termination of the selected call.
   
   *Events Triggered:* clickTerminate(call*)

3) **Input Button (Done)** - Returns to the previous screen.
   
   *Events Triggered:* clickDone

* Parameter call consists of the two extensions that are participating in call selected in Selection list 1

### 20: Configure Emergency Extensions

1) **Selection List (Emergency Extensions)** - List of possible emergency extensions that can be selected (there will only be exactly 2)

2) **Input Button (Select)** - Initiates the editing of an emergency extension.
   
   *Events Triggered:* clickEdit(index*)

3) **Input Button (Cancel)** - Moves the user back to the system management page.
   
   *Events Triggered:* clickBack

* index is the extension to be modified (emergency extension 1 or emergency extension 2)
21: Modify Emergency Extension

![Screen 21 - Modify Emergency Extension](image)

1) **Text field (Extension)** - Current extension assigned to emergency extension.
2) **Dropdown Box (IP list)** - List of all unassigned IPs in the system. Widget is used to select a new IP for the emergency extension. The dropdown box defaults to the IP currently assigned to the emergency extension.
3) **Input Button (Save)** - Initiates a save of the emergency extension.
   
   *Events Triggered:* clickSave(extension*, IP**)
4) **Input Button (Cancel)** - Initiates a cancel of modification to the emergency extension.
   
   *Events Triggered:* clickCancel
5) **Extension Selector (Extension)** - UI element enabling the selection of an extension. Has 3 options, select randomly, enter manually, or pick from dropdown box. Used to assign a new extension to the emergency extension. The default selection is the current extension.
   
   *Events Triggered:* clickPickRandomExt

* extension is the extension entered by element 5
** IP is the IP selected by element 2

22: Change Emergency Extension Confirmation Screen – Omitted for space reasons.

23: List User Accounts

![Screen 23 - List of User Accounts](image)

1) **TextBox (User Account #)** - Text box used to search user accounts.
2) **Dropdown Box (User Accounts)** - Dropdown box containing all user accounts narrowed by the search criteria in element 1.
3) **Input Button (Edit)** - Initiates and edit of the user account selected in element 2.
   
   *Events Triggered:* clickEdit(userAccount*)
4) **Input Button (Done)** - Returns to main menu.
   
   *Events Triggered: clickDone*

5) **Input Button (Delete)** - Initiates a deletion of the user account selected in element 2.
   
   *Events Triggered: clickDelete(userAccount*)

6) **Input Button (Add new user)** - Initiates the addition of a new user account to the system.
   
   *Events Triggered: clickAddNewUser*

   *userAccount is the user ID as selected by element 2*

---

24: **Remove User Account Prompt** – *Omitted for space reasons.*

25: **User Deletion Success** – *Omitted for space reasons.*

26: **Edit User Account**

![Screen 26 - Editing User Account](image)

1) **Radio buttons (Suspend)** - Choice of whether or not the user's phone accounts should be/are suspended.

2) **Text box (User Name)** - Text box with alphanumeric characters for user name.

3) **Text box (Address)** - Text box which holds the user's street address.

4) **Text box (Alternate Phone)** - Text box which holds the alternate phone number of the user.

5) **Text box (birth date)** - Widget which holds the date of birth of the user.

6) **Input Button (Edit phone account)** - Moves administrator to the page where they can edit phone accounts.

   *Events Triggered: clickEditPhoneAccounts(userAccount*)

7) **Input button (Cancel)** - Initiates a cancellation of the modification of a particular user.

   *Events Triggered: clickCancel*

8) **Input button (Save)** - Initiates a save of the entered user information.

   *Events Triggered: clickSave(userAccount*)

9) **Text field (Account#)** - Non-editable text field of the user's account number

   *Parameter userAccount consists of the values in the text field 1 and text boxes 26*

---

27: **Editing User Account Success** – *Omitted for space reasons.*
28: User Account's Phone Account List

![User Account's Phone Account List Screen](image)

1) **Text Box (Search)** - Text box containing the search criteria for phone accounts.
2) **Input Button (Edit)** - Initiates the modification of the phone account selected in 5.
   
   *Events Triggered:* `clickEditPhoneAccount(phoneAccount*)`
3) **Input Button (Delete)** - Initiates the deletion of the phone account selected in 5.
   
   *Events Triggered:* `clickDeletePhoneAccount(phoneAccount*)`
4) **Input Button (Edit Filtered Extensions)** - Moves the administrator to the page where they can edit filtered extensions for the phone account selected in 5.
   
   *Events Triggered:* `clickFilteredExt(phoneAccount*)`
5) **List Box (Phone Account)** - List box containing entries for all phone accounts associated to the current user that meet the search criteria in 1.
6) **Text Box (Bill Payment)** - The bill payment amount which can be applied to the chosen phone account.
7) **Input Button (Apply Bill Payment)** - Initiates the application of a bill payment in 6 to the selected phone account in 5.
   
   *Events Triggered:* `clickRecordBillPayment(amount**, phoneAccount*)`
8) **Input Button (Add New Phone Account)** - Initiates the addition of a new phone account to the user.
   
   *Events Triggered:* `clickAddPhoneAccount`
9) **Input Button (Done)** - Goes back to the user account page.
   
   *Events Triggered:* `clickDone`

* phoneAccount is the phone account selected in list box 5.
** amount is the numeric amount entered in text box 6.

29: Delete Phone Account Prompt – Omitted for space reasons.

30: Delete Phone Account Success – Omitted for space reasons.
31: Edit Phone Account Details

1) **Text field (Extension)** - Text indicating the extension of the phone account being edited.
2) **Radio Button (Suspended)** - Radio buttons denoting the state of the phone account (Active or Suspended).
3) **Text field (Billing Period)** - Text indicating the billing period which is being displayed in 7 and 10.
4) **Dropdown Box (Billing Periods)** - Box populated with all billing periods where the phone account has a bill.
5) **Input Button (View)** - Initiates the viewing of the bill from the billing period selected in 4.
   
   *Events Triggered:* `clickViewBillingPeriod(period*)`
6) **Dropdown Box (Billing Plans)** - List populated with all active billing plans in the system. The selection in this dropdown denotes the billing plan of the phone account.
7) **Selection list (Phone IP)** - Selection list populated with all available IPs. Used to select IP for a phone account.
8) **Send/Receive Checkbox Widgets (Send/Receive)** - Checkboxes denoting the phone account's ability to send and receive calls.
9) **Text Fields (Bill Information)** - Field denoting the total cost of service charges, calls, total charges, and outstanding charges for a billing period specified in 4.
10) **Text Fields (Call Information)** - Text of the bill denoting call start and end times. It also includes changes to the phone account's billing plan.
11) **Extension Selector (Extension)** - UI element enabling the selection of an extension. Has 3 options, select randomly, enter manually, or pick from dropdown box. Used to assign a new extension to the emergency extension. The default selection is the current extension.
   
   *Events Triggered:* `clickPickRandomExt`
12) **Input Button (Save)** - Initiates the modification of the phone account.
   
   *Events Triggered:* `clickSave(phoneAccount*)`
13) **Input Button (Cancel)** - Initiates a cancel of modification to the phone account.
   
   *Events Triggered:* `clickCancel`

*period is the billing period selected in combo box 4.
**phoneAccount includes fields 6-8, 11
32: Phone Account Save Success – Omitted for space reasons.

33: View Filtered Extensions

![Screen 33 - View Filtered Extensions](image)

1) **Selection list (List of incoming filtered extensions)** - Displays a list of all extensions that cannot call the current phone account.
2) **Input button (Delete incoming filtered extension)** - Deletes the selected incoming filtered extension.
   
   *Events Triggered:* `clickDeleteIncoming(incExt*)`
3) **Input button (Add incoming filtered extension)** - Goes to a screen where a new incoming filtered extension can be added.
   
   *Events Triggered:* `clickAddIncoming`
4) **Selection list (List of outgoing filtered extensions)** - Displays a list of all extensions the current phone account cannot call.
5) **Input button (Delete outgoing filtered extension)** - Deletes the selected outgoing filtered extension.
   
   *Events Triggered:* `clickDeleteOutgoing(outExt**)`
6) **Input button (Add outgoing filtered extension)** - Goes to a screen where a new outgoing filtered extension can be added.
   
   *Events Triggered:* `clickAddOutgoing`
7) **Input button (Done)** - Returns to the edit phone account page.
   
   *Events Triggered:* `clickDone`

* Parameter `incExt` consists of the incoming filtered extension (which may include wildcards) that is selected in Selection list 1.
* Parameter `outExt` consists of the outgoing filtered extension (which may include wildcards) that is selected in Selection list 4.

34: Add incoming filtered extensions

![Screen 34 - Add New Incoming Filtered Extension](image)

1) **Text box (New incoming filter)** - The user enters a new incoming filtered extension here.
2) **Input button (Add)** - Adds the entered extensions to the list of the incoming filtered extensions.
**Events Triggered:** clickIncomingOK(ext*)

3) **Input button (Cancel)** - Cancels addition and returns to the page that displays the incoming and outgoing filtered extensions.

   * The parameter ext is the string entered into Text box 1

**35: Add outgoing filtered extensions** – *Omitted for space reasons. Identical to 34.*

**36: Create Phone Account Success** – *Omitted for space reasons.*

**37: New User Account**

![Screen 37 - New User Account](image)

1) **Input button (Cancel)** - Cancels addition and returns to the edit user page.

   * The parameter userAccount is the data in fields 2-5, 7

2) **Text box (User Name)** - Text box with alphanumeric characters for user name.

3) **Text box (Address)** - Text box which holds the user's street address.

4) **Text box (Alternate Phone)** - Text box which holds the alternate phone number of the user.

5) **Text box (birth date)** - Widget which holds the date of birth of the user.

6) **Input Button (Next)** - Initiates the addition of a new user account.

   * The parameter userAccount is the data in fields 2-5, 7

7) **Text Field (Account #)** - The automatically assigned account # of the user account to be added.

**38: New User Account Creation Successful** – *Omitted for space reasons. Identical to 34.*
39: Display IP List

1) **Check box (Enabled filter)** - If checked, enabled IPs are eligible to be displayed in the list of IPs.
2) **Check box (Disabled filter)** - If checked, disabled IPs are eligible to be displayed in the list of IPs.
3) **Check box (Extension mapped filter)** - If checked, IPs with extensions mapped are eligible to be displayed in the list of IPs.
4) **Check box (Extension unmapped filter)** - If checked, IPs without extensions mapped are eligible to be displayed in the list of IPs.
5) **Text box (IP filter)** - When an IP or partial IP is entered into this box, the entries in combo box 9 are filtered.
6) **Input button (Delete)** - Initiates the deletion of the IP selected in combo box 9.
   *Events Triggered: clickDeleteIP(IP*)
7) **Input button (Change state)** - Allows the user to change the state of the IP selected in combo box 9.
   *Events Triggered: clickChangeState(IP*)
8) **Input button (Run hardware test)** - Initiates the hardware testing of the phone mapped to the IP selected in combo box 9.
   *Events Triggered: clickRequestSystemTest(IP*)
9) **Combo box (IP list)** - Lists all of the IPs that exist in the system, as well as their state and extension (if mapped)
10) **Input button (Add new IP)** - Initiates the addition of a new IP to the system
    *Events Triggered: clickAddIP
11) **Input button (Done)** - Returns to the system management screen.
    *Events Triggered: clickDone

* IP is the IP selected in combo box 9.
**40: Change IP State**

1) **Text Field** - Text displaying the IP of which the state is being changed.
2) **Radio buttons** - Allows the user to choose the state of the IP address.
3) **Input button (Done)** - Changes the state of the IP to the value selected using the radio buttons.

   *Events Triggered:* clickDone(state*)
   *the state selected in radio buttons 2*

**41: Add New IP Address**

1) **Text box (New IP address)** - The user enters the new IP address in this text box.
2) **Input button (Ok)** - Attempts to add a new IP to the system.

   *Events Triggered:* clickOK(IP*)

3) **Input button (Cancel)** - Cancels modification and returns to the page that lists all of the IPs in the system.

   *Events Triggered:* clickCancel

   *Parameter IP is the IP value entered in Text box 1.*
3.1.2 Hardware Interface Event Mapping
Every hardware interface event is implicitly associated to a particular phone's state machine. When the hardware generates a signal, it is sent to the associated phone's state machine. When a phone's state machine generates a hardware output signal or activity, the event is sent to the physical phone associated with the state machine.

3.1.2.1 Hardware Input Signals

keyPressed(key)
Sent to a phone's state machine whenever a key is pressed on the phone's keypad. The character printed on the physical key is passed as a parameter. This event corresponds to the hardware interface KeyPressed message. It is assumed that pressing a key will automatically play the tone associated with the key.

keyReleased(key)
Sent to a phone's state machine whenever a key is released on the phone's keypad. The character printed on the physical key is passed as a parameter. This event corresponds to the hardware interface KeyReleased message. It is assumed that releasing a key will automatically stop playing the tone associated with the key.

pickUp
Sent to a phone's state machine whenever the receiver is taken off the cradle. This event corresponds to the hardware interface OffHook message.

hangUp
Sent to a phone's state machine whenever the receiver is placed on the cradle. This event corresponds to the hardware interface OnHook message.

3.1.2.2 Hardware Output Signals

startRinging
Sent by a phone's state machine to make the physical phone ring. This maps to the StartRinging message on the hardware interface.

stopRinging
Sent by a phone's state machine to make the physical phone stop ringing. This maps to the StopRinging message on the hardware interface.

connect(myPhone, otherPhone)
Sent by a phone's state machine to cause the phone to start sending and receiving data to/from the given phone IP. This corresponds to sending the two hardware interface messages StartAudioSend and StartAudioReceive.
disconnect(myPhone, otherPhone)
Sent by a phone's state machine to cause the phone to stop sending and receiving data to/from the
given phone IP. This corresponds to the sending of two hardware interface messages
StopAudioSend and StopAudioReceive.

handsetOn
Sent by a phone's state machine to turn the handset of the phone on. This corresponds to the
HandsetOn message in the hardware interface.

handsetOff
Sent by a phone's state machine to turn the handset of the phone off. This
corresponds to the HandsetOff message in the hardware interface.

3.1.2.3 Hardware Output Activities

playDialTone
Causes a phone to play its dial tone. When this activity is interrupted, the phone returns to being silent. This maps to two events on the hardware interface: PlayTone(DialTone, Constant) to start playing and PlayTone(ToneOff, Constant) to stop.

playBusyTone
Causes a phone to play a busy tone. When this activity is interrupted, the phone returns to being silent. This maps to two events on the hardware interface: PlayTone(Busy, Medium) to start playing and PlayTone(ToneOff, Constant) to stop.

playFastBusyTone
Causes a phone to play a fast busy tone. When this activity is interrupted, the phone returns to being silent. This maps to two events on the hardware interface: PlayTone(Busy, Fast) to start playing and PlayTone(ToneOff, Constant) to stop.

playErrorTone
Causes a phone to play an error tone. When this activity is interrupted, the phone returns to being silent. This maps to two events on the hardware interface: PlayTone(Busy, Constant) to start playing and PlayTone(ToneOff, Constant) to stop.

playRingbackTone
Causes a phone to play a ringback tone. When this activity is interrupted, the phone returns to being silent. This maps to two events on the hardware interface: PlayTone(Ringing, Ringing) to start playing and PlayTone(ToneOff, Constant) to stop.
3.2 Functional Requirements

3.2.1 Use Cases

Name: Making a Call
Number: UC01
Authors: SE1_06
System: A VoIP telephone system
Actors: Caller (Initiator)
  • Callee
Event/Precondition: The caller has a phone on the hook and picks up the receiver.
Overview/Postcondition: Both parties hang up and the caller is charged for the call.
References:
  Course Project Overview: 7, 8
  Minutes #1: 1.1, 1.2, 1.7, 1.9, 1.11
  Minutes #2: 1.2, 1.6, 1.8, 1.14
  Minutes #3: 1.7, 1.13
**Related Use Cases:**

UC03

**Typical process description:**

<table>
<thead>
<tr>
<th>Caller Actions</th>
<th>System Response</th>
<th>Callee Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Caller picks up the receiver</td>
<td>2. System signals caller’s phone to play a dial tone</td>
<td></td>
</tr>
</tbody>
</table>
| 3. While caller has pressed fewer than 4 digits  
  Caller enters a digit  
  Caller waits less than ten seconds before entering the next digit | 4. System checks that the maximum number of simultaneous calls has not been reached  
  5. System checks that the number dialed corresponds to a phone that is in the “in service” state and a phone account that is not suspended  
  6. System verifies that the caller can call the callee  
  7. System signals callee’s phone to start ringing, signals the caller’s phone to play a ringback tone, and updates the number of calls currently in progress | |
| 8. Callee picks up the receiver | | |
| 9. System signals callee’s phone to stop ringing and signals caller’s phone to stop playing ringback tone | | |
| 10. System establishes audio connection between caller and callee, and begins timing the call | | |
| 11. System adds the call between the caller and the callee to the list of ongoing calls. The information stored includes the two extensions and the duration of the call. | | |
| 12. While neither party has hung up  
  If discount period ends or billing period ends  
  Split the call and charge the caller for the portion that has elapsed | | |
| 13. Caller hangs up phone | 14. System breaks audio connection between caller and callee, updates the number of calls in system, and removes the call from the list of current calls | |
| | 15. The system stops timing the call and charges the caller | |
### Notes:

1. Phone account A can call phone account B when B’s extension does not match an entry in A’s outgoing filter list, A’s extension does not match an entry in B’s incoming filter list, B does not have A in its block list, A is allowed to make outgoing calls, and B is allowed to receive incoming calls.

2. A ringback tone is the tone played on a caller's phone while the callee's phone is ringing.

3. An extension matches an entry in the list if all digits, other than the wild character (*), are the same.

4. If the call has been split, then the system will charge for the final portion of the split call during this step. The charge for the previous portions would have been applied when the call was split.

### Name:

**Charging a Monthly Service Fee**

**Number:** UC02  
**Authors:** SE1_06  
**System:** A VoIP telephone system  
**Actors:** Time (Initiator)

**Event/Precondition:** An unsuspended phone account exists in the system, and its current billing period has ended.

**Overview/Postcondition:** The service fee has been charged to the phone account. If the account has an outstanding charge on a bill over 90 days old, it is automatically suspended.

**References:**  
Minutes #1: 7.13, 7.15

**Related Use Cases:**  
UC03

<table>
<thead>
<tr>
<th><strong>Time Actions</strong></th>
<th><strong>System Response</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A billing period for the unsuspended phone account ends.</td>
<td>2. System adds a service fee for that phone account.</td>
</tr>
</tbody>
</table>
| | 3. If the phone account has an outstanding charges on a bill over 90 days old  
System suspends the phone account |

**Notes:**

1. A billing period is 30 days long. This billing period ends 30 days after the previous billing period has ended.

2. If the phone account’s billing plan was changed over the last 30 days, then the service fee is prorated for the new billing plan. If the phone system was down for some time over the past 30 days, then the service fee is prorated for the amount of time that the system was up.
Recording a Bill Payment

Number: UC03
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)
Event/Precondition: The administrator is logged into the system. The user has submitted a bill payment for a particular phone account.
Overview/Postcondition: The amount paid has been credited to the phone account.
References: Course Project Overview: 51
Related Use Cases: UC01, UC02

<table>
<thead>
<tr>
<th>Typical process description:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Administrator Actions</strong></td>
</tr>
<tr>
<td>1. Administrator submits the existing user whose bill payment they want to record</td>
</tr>
<tr>
<td>3. System displays user’s list of phone accounts</td>
</tr>
<tr>
<td>4. Administrator selects the phone account for which to record a payment</td>
</tr>
<tr>
<td>6. System displays phone account’s current outstanding balance and prompts for payment amount</td>
</tr>
<tr>
<td>7. Administrator submits payment amount</td>
</tr>
<tr>
<td>9. System records bill payment applies the credits to the phone account’s bills²</td>
</tr>
</tbody>
</table>

Notes:
¹ A valid payment amount is a positive real number, and can have up to two decimal places.
² The heuristic is to apply credits to the phone account’s oldest bills first. If the credits exceed the total amount owed, a credit is added to the user account.
Name: Editing a Phone Account
Number: UC04
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)

Event/Precondition: The administrator is logged into the system. The administrator chooses to edit a phone account. The input is the phone account that the administrator wants to modify.

Overview/Postcondition: The changes to the phone account have been saved. If a billing plan change occurred or the account was suspended, its service fee has been prorated.

References:
   Course Project Overview: 24
   Minutes #1: 4.2, 4.3, 4.13, 6.1
   Minutes #3: 5.3, 5.6

Related Use Cases: UC05, UC06, UC07

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator submits choice to view a specific user account</td>
<td>2. System verifies that no administrator is viewing/editing the user account</td>
</tr>
<tr>
<td></td>
<td>3. System displays the list of phone accounts associated with the user</td>
</tr>
<tr>
<td>4. Administrator submits choice to modify a specific phone account of that user</td>
<td>5. System verifies that no administrator is viewing/editing the phone account</td>
</tr>
<tr>
<td></td>
<td>6. System displays the phone account’s details to the administrator and prompts the administrator to enter new information¹ for the phone account</td>
</tr>
<tr>
<td>7. Administrator enters new information for the phone account²</td>
<td>8. System validates³ phone account information</td>
</tr>
<tr>
<td></td>
<td>9. System updates the phone account with the new information</td>
</tr>
<tr>
<td></td>
<td>10. If the administrator suspended the phone account or changed its billing plan System prorates⁴ the service fee for the phone account</td>
</tr>
</tbody>
</table>
Notes:
1 The information for a phone account is the associated telephone IP, the billing plan, the extension, suspended/unsuspended and a combination of send and/or receive privileges
2 If the administrator is changing the extension, he has the option of picking extensions from a list of available extensions, specifying an extension, or using a randomly generated extension
3 The new phone account information is valid if
   - the extension did not change, or the new extension is not already mapped, and
   - the IP did not change, or the new IP exists and is not already mapped, and
   - the billing plan exists in the system
4 The prorated service fee is the service fee for a billing plan for the entire 30 days scaled by the proportion of time that the phone account was actually using that billing plan

Name: Deleting a Phone Account
Number: UC05
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)

Event/Precondition: The administrator is logged into the system. The administrator chooses to delete a phone account. The input is the phone account that the administrator wants to delete.

Overview/Postcondition: The phone account has been marked as deleted, its extension has been freed, and a final service fee has been applied.

References:
   Course Project Overview: 24
   Minutes #1: 4.2, 4.3, 4.13, 6.1
   Minutes #3: 5.3, 5.6

Related Use Cases:
   UC04, UC06, UC07

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator submits choice to view a specific user account.</td>
<td></td>
</tr>
<tr>
<td>2. System verifies that no administrator is viewing/editing the user account.</td>
<td></td>
</tr>
<tr>
<td>3. System displays the list of phone accounts associated with the user.</td>
<td></td>
</tr>
<tr>
<td>4. Administrator submits choice to delete a specific phone account.</td>
<td></td>
</tr>
<tr>
<td>5. System verifies that no administrator is viewing/editing the phone account and that the account does not have an outstanding balance.</td>
<td></td>
</tr>
<tr>
<td>6. System marks the selected phone account as deleted, frees up its extensions, and charges the final service fee.</td>
<td></td>
</tr>
</tbody>
</table>

Name:
## Adding a User Account

**Number:** UC06  
**Authors:** SE1_06  
**System:** A VoIP telephone system  
**Actors:** Administrator (Initiator)

### Event/Precondition:
The administrator is logged into the system. The administrator chooses to set up a new user account and an associated phone account.

### Overview/Postcondition:
The new user account and its associated phone account has been saved in the system.

### References:
- Course Project Overview: 6, 15, 21, 22, 24
- Minutes #1: 4.2, 4.3, 4.4, 4.8, 5.1, 7.6
- Minutes #2: 2.4, 2.8, 6.3
- Minutes #3: 4.2, 4.3

### Related Use Cases:
- UC04, UC05, UC07

### Typical process description:

<table>
<thead>
<tr>
<th><strong>Administrator Actions</strong></th>
<th><strong>System Response</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator submits selection choice to set up a new user account</td>
<td>2. System prompts administrator for user information</td>
</tr>
<tr>
<td>3. Administrator enters user information</td>
<td>4. System prompts administrator to choose an IP from a list of all unassigned telephone IPs</td>
</tr>
<tr>
<td>5. Administrator selects an unassigned telephone IP</td>
<td>6. System prompts administrator to choose how he/she would like to enter the telephone extension</td>
</tr>
<tr>
<td>7. Administrator selects to see a list of all available telephone extensions</td>
<td>8. System displays a list of all available telephone extensions</td>
</tr>
<tr>
<td>9. Administrator selects an extension from the list</td>
<td>10. System prompts the administrator to select a billing plan</td>
</tr>
<tr>
<td>11. Administrator selects an available billing plan</td>
<td>12. System prompts administrator to enable send/receive on the new phone account</td>
</tr>
<tr>
<td>13. Administrator enters a combination of send/receive for the new phone account</td>
<td>14. System creates a new user account with the entered user information</td>
</tr>
<tr>
<td>15. System creates a new phone account with the</td>
<td></td>
</tr>
</tbody>
</table>
selected telephone IP, telephone extension, billing plan and send/receive options and associates the phone account with the new user account

<table>
<thead>
<tr>
<th>Alternative 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Administrator selects to manually enter an extension</td>
<td>8. System prompts for a telephone extension</td>
</tr>
<tr>
<td>9. Administrator enters telephone extension</td>
<td>10. System checks that the entered extension is valid(^2) and unassigned</td>
</tr>
<tr>
<td></td>
<td>11. Return to main flow step 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Alternative 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Administrator selects to assign a randomly generated available extension to the telephone</td>
<td>8. System assigns a randomly generated available extension to the telephone</td>
</tr>
<tr>
<td></td>
<td>9. Return to main flow step 10</td>
</tr>
</tbody>
</table>

Notes:
1 User information contains name, address, alternate phone number, account number and date of birth.
2 Valid extensions are exactly 4 digits.

Name: Deleting a User Account

Number: UC07
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)

Event/Precondition: The administrator is logged into the system. The administrator chooses to delete a user account. The input is the user account that the administrator wants to delete.

Overview/Postcondition: The user account has been marked as deleted, and its phone accounts have been marked as deleted.

References:
- Course Project Overview: 24
- Minutes #1: 4.2, 4.3, 4.13, 6.1
- Minutes #3: 5.3, 5.6

Related Use Cases:
- UC04, UC05, UC06

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator submits choice to view a specific user account</td>
<td>2. System verifies that no administrator is viewing/editing the user account</td>
</tr>
<tr>
<td></td>
<td>3. System shows the list of phone accounts associated with</td>
</tr>
</tbody>
</table>
4. Administrator submits choice to delete the user account

5. System verifies that no administrator is viewing/editing the phone accounts of that user and that all phone accounts of that user have no outstanding balance

6. System charges the final service fee to all phone accounts associated with that user, and deletes all phone accounts associated to the selected user account

7. System marks the selected user account as deleted

Name:

**Editing a Billing Plan**

Number: UC08

Authors: SE1_06

System: A VoIP telephone system

Actors: Administrator (Initiator)

Event/Precondition: The administrator is logged into the system. The administrator knows which billing plan he/she would like to change.

Overview/Postcondition: The requested modifications to the billing plan have been committed.

References:

- Course Project Overview: 54, 56, 57, 58
- Minutes #1: 7.6
- Minutes #2: 6.2
- Minutes #3: 1.2, 6.2, 6.3, 6.4

Related Use Cases:

None

Typical process description:

<table>
<thead>
<tr>
<th><strong>Administrator Actions</strong></th>
<th><strong>System Response</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator requests a list of billing plans</td>
<td>2. System displays the list of billing plans in the system</td>
</tr>
<tr>
<td>3. Administrator selects a billing plan to edit</td>
<td>4. System verifies that no administrator is viewing/editing that billing plan</td>
</tr>
<tr>
<td>6. Administrator defines and submits new billing plan information</td>
<td>5. System displays that billing plan, and prompts the administrator to enter information(^1) related to the billing plan</td>
</tr>
<tr>
<td></td>
<td>7. System verifies(^2) entered billing plan information</td>
</tr>
<tr>
<td></td>
<td>8. System commits modifications to billing plan</td>
</tr>
</tbody>
</table>

Notes:

\(^1\) Each billing plan specifies:
(a) the regular charge rate for calls
(b) one or more time periods, such as days of the week and times of the day
(c) the discount rate for each of these periods (percentage of regular rate)  
(d) monthly flat fee; and  
(e) call blocking
2 Validation checks that:  
(a) Regular charge rate is a valid positive dollar amount  
(b) Discount days are days of the week, and discount times are times of the day  
(c) Discount rates must be valid positive percentages  
(d) No two discount periods overlap  
(e) Monthly flat fee must be a valid non-negative dollar amount.

Name:  
Adding an Administrator Account
Number: UC09
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)
Event/Precondition: An administrator chooses to add a new administrator account to the system.
Overview/Postcondition: The new administrator account has been added to the system.
References:
  Course Project Overview: 20
  Minutes #1: 3.2
  Minutes #2: 5.5
Related Use Cases:  
UC10

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator selects to add a new administrator account</td>
<td>2. System prompts the administrator to enter the username and password of the new administrator</td>
</tr>
<tr>
<td>3. Administrator enters the username and password of the new administrator</td>
<td>4. System checks that the username is unique and that the username and password are valid(^1). The system creates the new administrator account</td>
</tr>
</tbody>
</table>

Notes:
\(^1\) A valid password must consist of alphanumeric characters, or characters in the set \{!@#$%^&*()\}. Passwords must be at least 8 characters long and at most 256 characters long. A valid username must be at least 8 character long and at most 256 characters long, consisting only of alphanumeric characters.

Name:  
Deleting an Administrator Account
Number: UC10
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)
Event/Precondition: Administrator chooses to delete an administrator account from the system.
Overview/Postcondition: The administrator account has been removed from the system, and the deleted administrator has been logged out.
References:
Minutes #1: 3.2, 4.15
Minutes #3: 5.4
Related Use Cases:
UC09

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator chooses to delete an administrator account</td>
<td>2. System displays a list of existing administrator accounts, excluding the account of the logged in administrator</td>
</tr>
<tr>
<td>3. Administrator chooses which administrator account to remove</td>
<td>4. System verifies that no administrator is viewing/editing that administrator account</td>
</tr>
<tr>
<td>5. System removes the selected administrator account</td>
<td>6. If the deleted administrator is logged in System logs out the deleted administrator</td>
</tr>
</tbody>
</table>

Name:
Adding Filter Expressions
Number: UC11
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)
Event/Precondition: The administrator is logged into the system. The administrator chooses to add filter expressions to a phone account. The input will be the phone account to which the administrator wants to add filtered expressions.
Overview/Postcondition: The new filter expressions have been added to the phone account.
References:
Minutes #1: 4.1, 4.10, 4.11, 4.12
Minutes #3: 1.3, 1.12, 4.1
Related Use Cases:
None

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator submits the phone account to edit</td>
<td>2. System verifies that no administrator is viewing/editing that phone account</td>
</tr>
<tr>
<td>3. System displays the lists of currently filtered incoming/outgoing expressions</td>
<td>4. Administrator chooses to add a new</td>
</tr>
</tbody>
</table>
expression to either of the filter lists 5. System prompts for the expressions to be added
6. Administrator enters list of expressions to be added 7. System checks that the entered expressions are valid
7. System adds all valid expressions to the filtered list and displays errors about invalid input
8. System adds all valid expressions to the filtered list and displays errors about invalid input

Notes:
1 Valid expressions are exactly 4 digits in length. A wildcard ‘*’ character can be used in place of any digit to represent that all digits will match that spot. For example, 0000, 1234, ****, and 1*2* are valid entries, but 123, 12345, *1234, and * are not.

Name: Adding a Blocked Extension
Number: UC12
Authors: SE1_06
System: A VoIP telephone system
Actors: User (Initiator)
Event/Precondition: The user is at their telephone and has picked up the receiver.
Overview/Postcondition: The extensions that the user wish to add/remove have been added/removed from his blocked extension list.
References:
  Minutes #1: 8.1, 8.4
  Minutes #2: 7.1, 7.3, 7.4, 7.5, 7.6
  Minutes #3: 2.1, 2.2
Related Use Cases: None

Typical process description:

<table>
<thead>
<tr>
<th>User Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. User dials #70 to edit their call blocking list</td>
<td>2. System prompts the user to press 1 for adding extensions to their blocked extensions list, to press 2 to remove extensions from their blocked extensions list, or to press 3 to clear all blocked extensions</td>
</tr>
<tr>
<td>3. User presses 1 to add blocked extensions from their list</td>
<td>5. System validates 1 the extension that the user entered</td>
</tr>
<tr>
<td>4. User enters an extension, followed by the delimiter key (*)</td>
<td>6. System adds the extension to the user’s blocked extension list</td>
</tr>
<tr>
<td>7. If user does not press the terminate key (#)</td>
<td>Go to main flow step 4</td>
</tr>
</tbody>
</table>

Notes:
A valid sequence is four digits followed by a #. A valid sequence can be terminated by pressing the * after pressing the #. All other sequences are invalid.

Name: Adding an IP Address
Number: UC13
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)
Event/Precondition: The administrator is logged into the system. The administrator chooses to add an IP to the system.
Overview/Postcondition: The new IP has been added to the system.
References: Press Conference #3: 12
Related Use Cases: None

Typical process description:

<table>
<thead>
<tr>
<th>Administrator Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator chooses to add an IP to the system</td>
<td></td>
</tr>
<tr>
<td>2. System prompts administrator to enter an IP to add</td>
<td></td>
</tr>
<tr>
<td>3. The administrator enters an IP</td>
<td></td>
</tr>
<tr>
<td>4. System validates(^1) the IP and makes sure the IP isn’t already in the system</td>
<td></td>
</tr>
<tr>
<td>5. System adds the IP, setting its default status to “disabled”</td>
<td></td>
</tr>
</tbody>
</table>

Notes: \(^1\) A valid IP is in IPv4 format: x.y.z.w, where x, y, z, w are integers between 0 and 255, inclusive.

Name: Running a Hardware Test
Number: UC14
Authors: SE1_06
System: A VoIP telephone system
Actors: Time (initiator)
Event/Precondition: 30 seconds have elapsed since the previous test on the phone has completed
Overview/Postcondition: The next test on the phone has completed.
References:
Course Project Overview: 27, 28, 35
Minutes #2: 5.6, 5.7
Minutes #3: 5.2
Related Use Cases: None

Typical process description:

<table>
<thead>
<tr>
<th>Time Actions</th>
<th>System Response</th>
<th>Phone Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. 30 seconds have elapsed since the previous test on</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

41
Name:

Starting the System

Number: UC15
Authors: SE1_06
System: A VoIP telephone system
Actors: Administrator (Initiator)
Event/Precondition: The system is off.
Overview/Postcondition: The system is back online and ready to process calls.
References: Meeting #2: 5.15, 8.2
Related Use Cases: UC14

Typical process description:

<table>
<thead>
<tr>
<th>Administrator’s Actions</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrator starts the system</td>
<td>2. System turns back on, and becomes ready to accept hardware events from the phones</td>
</tr>
<tr>
<td></td>
<td>3. System charges all phone accounts no service fee for the duration that it was down</td>
</tr>
<tr>
<td></td>
<td>4. System begins an automated test for each of the phones</td>
</tr>
</tbody>
</table>
3.2.2 Domain Model

Figure 4 - Domain class diagram
3.2.3 Functional Specifications

Naming Conventions for Objects and Links
Our group used a general naming convention to simplify the naming of sets and links. In all cases, the set of entities named "A" is called "As". For example, the set of "PhoneAccount" entities is called "PhoneAccounts". For relations (links) we used the name of the association on the class diagram. For example, the links between the "Extension" entities and "PhoneAccount" entities is called "blockedExtensions," as is labeled on the diagram. Our group followed this convention to simplify the task of reading the document and also to alleviate the needs to define sets in the class diagram or in each function table. All sets that do not follow this convention are specifically mentioned and defined in the function tables. There is an extra set of Extension objects, called “Blocked”, used to store the list of blocked extensions.

<table>
<thead>
<tr>
<th>Making a Call</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong></td>
</tr>
<tr>
<td>Making a call to another phone</td>
</tr>
</tbody>
</table>

| **Inputs:** |
| myExt, otherExt : Integer |

| **Preconditions:** |
| 0 ≤ myExt < 10000, |
| 0 ≤ otherExt < 10000, |
| |PhoneCalls| ≤ PhoneCall.maxCalls, |
| ∃ p ∈ Phones : (p.Extension.extension = myExt AND p.onHook = false), |
| ∃ p ∈ Phones : (p.Extension.extension = otherExt AND p.onHook = true), |
| ∃ acct ∈ PhoneAccounts : ( |
| acct.NormalExtension.extension = myExt AND |
| acct.send = true AND |
| acct.isSuspended = false), |
| ∃ acct ∈ PhoneAccounts : ( |
| acct.NormalExtension.extension = otherExt AND |
| acct.receive = true AND |
| acct.isSuspended = false), |
| NOT ∃ c ∈ PhoneCalls : ( |
| c.caller.Extension.extension = myExt OR |
| c.callee.Extension.extension = myExt OR |
| c.caller.Extension.extension = otherExt OR |
| c.callee.Extension.extension = otherExt), |
| NOT ∃ b ∈ blockedExtensions : ( |
| b.Extension.extension = myExt AND |
| b.BlockList.PhoneAccount.NormalExtension.extension = otherExt), |
| NOT ∃ f ∈ filteredExpressions : ( |
| f.FilterList.PhoneAccount.NormalExtension.extension = myExt AND |
Ending a Call

Overview:
Hanging up the phone during a call. Note that this function does not charge for the final portion of the call. There is another function that charges, and that function will be called to charge the caller when the call ends.

Inputs:
myExt, otherExt : Integer

Preconditions:
\[ \exists c \in \text{PhoneCalls} : ( \\
\quad (\text{c.caller.Extension.extension} = \text{myExt AND} \\
\quad \text{c.callee.Extension.extension} = \text{otherExt}) \\
\quad \text{OR} (\text{c.callee.Extension.extension} = \text{myExt AND} \\
\quad \text{c.caller.Extension.extension} = \text{otherExt}) \\
\quad ) \\
\) \\

**Modifies:**

PhoneCall

**Postconditions:**

\[
\text{let me} = p \in \text{Phones} : p.\text{Extension.extension} = \text{otherExt} \\
\text{let other} = p \in \text{Phones} : p.\text{Extension.extension} = \text{otherExt} \\
\text{if} \exists c \in \text{PhoneCalls} : (\text{c.caller} = \text{me AND c.callee} = \text{other}) \\
\quad \text{let endedCall} = c \in \text{PhoneCalls} : (\text{c.caller} = \text{me AND c.callee} = \text{other}) \\
\quad \text{else} \\
\quad \text{let endedCall} = c \in \text{PhoneCalls} : (\text{c.caller} = \text{other AND c.callee} = \text{me}) \\
\text{PhoneCalls'} = \text{PhoneCalls - \{endedCall\}}
\]

**Exceptions:**

If any precondition does not hold

PhoneCalls’ = PhoneCalls

**References:**

Course Project Overview: 9, 33

**Notes:**

1. A date CONTAINS a time if date.start \leq time and time \leq date.end

### Charging For Part of a Call

**Overview:**

The system charges a single call in a single discount period. This function should be called multiple times if a call spans multiple discount periods. This function should also be called when a phone call terminates, before disconnecting the call.

**Inputs:**

callerExt, calleeExt : Integer

**Preconditions:**

\[ \exists \text{call} \in \text{PhoneCalls} : ( \\
\quad \text{call.caller.Extension.extension} = \text{callerExt AND} \\
\quad \text{call.callee.Extension.extension} = \text{calleeExt} \\
\quad ) \\
\]
Modifies:
new charge’ : CallCharge, accountCharges, PhoneAccounts, call, Charges, CallCharges

Postconditions:
let c = call∈ PhoneCalls :
    call.caller.Extension.extension = callerExt AND
    call.callee.Extension.extension = calleeExt
let startTime = c.startTime
let billingPlan = c.BillingPlan
let phoneAccount = billingPlan.PhoneAccount
if ∃ p ∈ billingPlan.discountPeriod :
    p.startTime < call.startTime AND
    p.endTime > call.startTime
) then
    let rate = billingPlan.baseRate * p.discountPercentage
    if call.endTime > p.endTime then
        let duration = p.endTime - call.startTime
        call’.startTime = p.endTime
    else
        let duration = call.endTime - call.startTime
        call’.startTime = call.endTime
      else
        let rate = billingPlan.baseRate
      if ∃ p ∈ billingPlan.discountPeriod :
          p.startTime < call.startTime OR
          p.endTime < call.startTime
      ) then
        let duration = call.endTime - call.startTime
        call’.startTime = call.endTime
      else
        let nextDP ∈ billingPlan.discountPeriod :
            nextDP.startTime > call.startTime AND
            ∀ dp ∈ billingPlan.discountPeriod :
                (dp.startTime > call.startTime AND dp ≠ nextDP) ⇒
                nextDP.startTime < dp.startTime
        )
        let duration = nextDP.startTime - call.startTime
        call’.startTime = nextDP.startTime
      if phoneAccount.outstandingBalance < 0 then
        if |phoneAccount.outstandingBalance| > rate * duration then
            new charge’ = (currentTime, rate * duration, rate * duration, callerExt, calleeExt, duration, rate)
        else
            new charge’ = (currentTime, rate * duration,
                            phoneAccount.outstandingBalance, callerExt, calleeExt, duration, rate)
else
  \text{new charge}' = (\text{currentTime}, \text{rate} \ast \text{duration}, 0, \text{callerExt}, \text{calleeExt}, \text{duration}, \text{rate})
\text{Charges}' = \text{Charges} \cup \{\text{charge}'\}
\text{CallCharges}' = \text{CallCharges} \cup \{\text{charge}'\}
\text{phoneAccount'.outstandingBalance} = \text{phoneAccount.outstandingBalance} + \text{rate} \ast \text{duration}
\text{accountCharges}' = \text{accountCharges} \cup \{(\text{phoneAccount'}, \text{charge}')\}

\textbf{Exceptions:}
If preconditions are not met
  \text{Charges}' = \text{Charges}
  \text{CallCharges}' = \text{CallCharges}
  \text{accountCharges}' = \text{accountCharges}
  \text{PhoneAccounts}' = \text{PhoneAccounts}
  \text{call}' = \text{call}

\textbf{References:}
Course Project Overview: 44
Minutes #1: 7.10
Minutes #2: 6.6
Minutes #3: 6.6, 6.7, 8.1

\begin{tabular}{|l|c|}
\hline
\textbf{Charging Monthly Service Fee} & \textbf{ID: 04 Importance: D} \\
\hline
\textbf{Overview:} & \\
Recording a monthly service fee for an account, and suspend it if a charge hasn’t been paid for over 90 days. & \\
\hline
\textbf{Inputs:} & \\
p : \text{PhoneAccount} & \\
\hline
\textbf{Preconditions:} & \\
30 days have elapsed since the last billing date, & \\
p.isLocked = false & \\
\hline
\textbf{Modifies:} & \\
p, new \text{c'} : \text{Charges, ServiceCharges, accountCharges} & \\
\hline
\textbf{Postconditions:} & \\
let newest = c \in p.Charge | \text{ (} \\
c \in \text{ServiceCharges AND} \\
\forall \text{ch} \in p.Charge : ( \\
  \text{ch} \in \text{ServiceCharges AND ch} \neq \text{newest} \Rightarrow \text{ch.chargeDate} < \text{newest.chargeDate}) \\
\text{) } & \\
\text{let amount} = (\text{currentTime} - \text{newest.chargeDate}) \ast p.BillingPlan.serviceFee / 30 & \\
\text{if p.outstandingBalance < 0 then} & \\
\hline
\end{tabular}
if \(|p.outstandingBalance| > rate \times duration\) then
  \(new c' = (currentTime, rate \times duration, rate \times duration, "Service Fee")\)
else
  \(new c' = (currentTime, rate \times duration, 0, "Service Fee")\)
\(p'.outstandingBalance = p.outstandingBalance + amount\)
let oldest = \(c \in p.Charge | (\)
  \(\text{oldest.amountPaid} < \text{oldest.amount AND} \)
  \(\forall ch \in p.Charge : (\)
    \((ch \neq \text{oldest AND} \text{ch.amountPaid} < \text{ch.amount}) \Rightarrow \)
    \(\text{oldest.chargeDate} < \text{ch.chargeDate}\))\)
\)
let difference = currentTime - oldest.chargeDate
let numBillingPeriods = \((\text{difference} - (\text{difference} \mod 30)) / 30\)
if (numBillingPeriods >= 3) then
  \(p'.isSuspended = \text{true}\)
accountCharges' = accountCharges \cup \{(\text{phoneAccount'}, c')\}
ServiceCharges' = ServiceCharges \cup \{c'\}
Charges' = Charges \cup \{c'\}

**Exceptions:**
If any precondition does not hold
  \(p' = p\)
  accountCharges' = accountCharges
  Charges' = Charges
  ServiceCharges' = ServiceCharges

**References:**
Minutes #1: 7.13, 7.15

---

**Recording Bill Payment**

**ID:** 05  **Importance:** D

**Overview:**
Record a Bill Payment for a particular phone account

**Inputs:**
requestor: Administrator
\(p: \text{PhoneAccount}\)
amount: \(\text{dollarAmount}\)

**Preconditions:**
requestor initiates a bill payment,
requestor \(\in\) Administrators,
requestor.isLoggedIn = \text{true},
p ∈ PhoneAccounts,
   amount > 0

**Modifies:**
requestor, p, \{c ∈ Charges | c.amountPaid < c.amount\}

**Postconditions:**
requestor'.lastActionTime = currentTime
p'.outstandingBalance = p.outstandingBalance - amount
outstandingCharges = \{c ∈ p.Charge | c.amountPaid < c.amount\}
while (amount > 0 AND |outstandingCharges| > 0)
   ∀ c ∈ outstandingCharges : (∀ c2 ∈ outstandingCharges :
      (c ≠ c2 ⇒ c.chargeDate < c2.chargeDate) ⇒
         if amount ≥ (c.amountPaid - c.amount) then
            amount = amount - (c.amountPaid - c.amount)
            c'.amountPaid = c.amount
            let difference = currentTime - c.chargeDate
            let numBPeriods = (difference – (difference % 30)) / 30
            if (numBPeriods < 3 AND p.isSuspended = true) then
               p'.isSuspended = false
            else
               c'.amountPaid = c.amountPaid + amount
               amount = 0
               outstandingCharges = outstandingCharges - c
         )
   )

**Exceptions:**
If any precondition does not hold
   requestor' = requestor
   p' = p
   ∀ c ∈ Charges : (c' = c)

**References:**
Course Project Overview: 51

---

**Prorating a Service Fee**

**ID:** 06  **Importance:** D

**Overview:**
Prorate the service fee of an account

**Inputs:**
phoneAccount : PhoneAccount
Preconditions:
phoneAccount ∈ PhoneAccounts,
phoneAccount.isDeleted = false

Modifies:
phoneAccount, new c’ : ServiceCharge, accountCharges, Charges, ServiceCharges

Postconditions:
let newest = c ∈ phoneAccount.Charge | (c ∈ ServiceCharges AND
∀ ch ∈ phoneAccount.Charge :
(ch ∈ ServiceCharges AND ch ≠ newest) ⇒ ch.chargeDate < newest.chargeDate)

let baseFee = phoneAccount.BillingPlan.serviceFee
let amount = (currentTime - newest) * baseFee / 30
new c’ = (currentTime, amount, 0, “Prorated Service Fee”)
Charges’ = Charges ∪ {c’}
ServiceCharges’ = ServiceCharges ∪ {c’}
accountCharges’ = accountCharges ∪ {(phoneAccount’, charge’)}

Exceptions:
If any precondition does not hold
phoneAccount’ = phoneAccount
accountCharges’ = accountCharges
Charges’ = Charges
ServiceCharges’ = ServiceCharges

References:
Minutes #1: 7.11
Minutes #3: 8.1

Editing a Phone Account

Overview:
Edit an existing phone account

Inputs:
phoneAccount : PhoneAccount,
isSuspended, send, receive : boolean, 
extension : int, 
phone : Phone, 
billingPlan : BillingPlan, 
requestor : Administrator

Preconditions:
requestor initiates editing a phone account,
requestor ∈ Administrators,
requestor.isLoggedIn = true
phoneAccount.isLocked = false,
phoneAccount.isDeleted = false,
phoneAccount ∈ PhoneAccounts,
phone ∈ Phones,
billingPlan ∈ billingPlans,
0 ≤ extension < 10000,
∀ ae ∈ accountExtensions : (ae.NormalExtension = extension ⇒
ae.PhoneAccount = phoneAccount)

Modifies:
requestor, phoneAccount, new ext': NormalExtension, Extensions, NormalExtensions,
phoneExtensions, accountExtensions, accountBillingPlans, new c' : Charge, Charges,
ServiceCharges

Postconditions:
requestor'.lastActionTime = currentTime
if (phoneAccount.isSuspended = true AND isSuspended = false) then
    new c' = (currentTime, 0, 0, “No service fee while suspended”)
else if (phoneAccount.isSuspended = false AND isSuspended = true) then
    let newest = c ∈ phoneAccount.Charge | (c ∈ ServiceCharges AND
    ∀ ch ∈ phoneAccount.Charge : (ch ∈ ServiceCharges AND ch ≠ newest) ⇒
    ch.chargeDate < newest.chargeDate)
    let baseFee = phoneAccount.BillingPlan.serviceFee
    let amount = (currentTime – newest) * baseFee / 30
    new c' = (currentTime, amount, 0, “Prorated service fee before suspension”)
Charges' = Charges ∪ {c'}
ServiceCharges' = ServiceCharges ∪ {c'}
accountCharges' = accountCharges ∪ {(phoneAccount', c')}
phoneAccount'.isSuspended = isSuspended
phoneAccount'.send = send
phoneAccount'.receive = receive
phoneAccount'.createDate = phoneAccount.createDate
new ext' = (extension)
let oldExt = phoneAccount.NormalExtension
accountExtensions' = accountExtensions - {(phoneAccount, oldExt)} ∪ {(phoneAccount', ext')}
Extensions' = Extensions - {oldExt} ∪ {ext'}
NormalExtensions' = NormalExtensions - {oldExt} ∪ {ext'}
phoneExtensions' = phoneExtensions - {(oldExt, oldExt.Phone)} ∪ {(ext', phone)}
if (phoneAccount.BillingPlan ≠ billingPlan) then
    let newest = c ∈ phoneAccount.Charge | (c ∈ ServiceCharges AND ∀ ch ∈ phoneAccount.Charge : (ch ∈ ServiceCharges AND ch ≠ newest) ⇒ ch.chargeDate < newest.chargeDate)

    let baseFee = phoneAccount.BillingPlan.serviceFee
    let amount = (currentTime - newest) * baseFee / 30
    new c’ = (currentTime, amount, 0, “Prorated service fee before billing plan change”)

    Charges = Charges ∪ {c’}
    ServiceCharges = ServiceCharges ∪ {c’}
    accountCharges’ = accountCharges ∪ {(phoneAccount’, c’)}
    accountBillingPlans’ = accountBillingPlans -
    phoneAccount’.isLocked = false

Exceptions:
If any precondition does not hold
    requestor’ = requestor
    phoneAccount’ = phoneAccount
    Extensions’ = Extensions
    NormalExtensions’ = NormalExtensions
    phoneExtensions’ = phoneExtensions
    accountExtensions’ = accountExtensions
    accountBillingPlans’ = accountBillingPlans
    Charges’ = Charges
    ServiceCharges’ = ServiceCharges

References:
Course Project Overview: 24
Minutes #1: 4.2, 4.3, 4.13, 6.1
Minutes #3: 5.3, 5.6

Deleting a Phone Account

Overview:
Delete a phone account from the system.

Inputs:
requestor : Administrator
phoneAccount : PhoneAccount

Preconditions:
requestor initiates deleting a phone account,
requestor ∈ Administrators,
requestor.isLoggedIn = true,
phoneAccount.isLocked = false,
phoneAccount.outstandingBalance >= 0

**Modifies:**
requestor, phoneAccount, accountCharges, new c’ : ServiceCharge, Charges, ServiceCharges, accountExtensions, incomingFilters, outgoingFilters, FilterLists, filteredExpressions, FilterExpressions

**Postconditions:**
requestor’.lastActionTime = currentTime
phoneAccount’.isDeleted = true
accountExtensions’ = accountExtensions – {(phoneAccount, phoneAccount.NormalExtension)}
outgoingFilters’ = outgoingFilters – {(phoneAccount, phoneAccount.outlist)}
incomingFilters’ = incomingFilters – {(phoneAccount, phoneAccount.inclist)}
FilterLists’ = FilterLists - {phoneAccount.outlist} - {phoneAccount.inclist}
let oldFilteredExpressions = {(f, fe) ∈ filteredExpressions |
    (f ∈ phoneAccount.outlist OR f ∈ phoneAccount.inclist)}
filteredExpressions’ = filteredExpressions – oldFilteredExpressions
let oldFilterExpressions = {fe ∈ FilterExpressions |
    ∃ fde ∈ oldFilteredExpressions : fde.FilterExpression = fe}
FilterExpressions = FilterExpressions - oldFilterExpressions
let newest = c ∈ phoneAccount.Charge | (c ∈ ServiceCharges AND
    ∀ ch ∈ phoneAccount.Charge : (ch ∈ ServiceCharges AND ch ≠ newest) ⇒ ch.chargeDate < newest.chargeDate)
let baseFee = phoneAccount.BillingPlan.serviceFee
let amount = (currentTime – newest) * baseFee / 30
new c’ = (currentTime, amount, 0, “Prorated service fee for final billing period”)
accountCharges’ = accountCharges ⊔ {(phoneAccount’, c’)}
Charges = Charges ⊔ {c’}
ServiceCharges = ServiceCharges ⊔ {c’}

**Exceptions:**
If any precondition does not hold
requestor’ = requestor
phoneAccount’ = phoneAccount
accountCharges’ = accountCharges
ServiceCharges’ = ServiceCharges
Charges’ = Charges
accountExtensions’ = accountExtensions
incomingFilters’ = incomingFilters
outgoingFilters’ = outgoingFilters
FilterLists’ = FilterLists
filteredExpressions’ = filteredExpressions
FilterExpressions’ = FilterExpressions

References:
Course Project Overview: 24
Minutes #1: 4.2, 4.3, 4.13, 6.1
Minutes #3: 5.3, 5.6

Adding a User Account

Overview:
Add a user account to the system.

Inputs:
name, address, alternatePhone : String
birthDate : Date
isSuspended, send, receive : Boolean
extension : int
phone : Phone
billingPlan : BillingPlan
requestor : Administrator

Preconditions:
requestor initiates adding a user account,
requestor ∈ Administrators,
requestor.isLoggedln = true,
phone ∈ Phones,
billingPlan ∈ BillingPlans,
0 ≤ extension < 10000,
|Extensions| < 10000,
phone.Extension = ∅,
NOT ∃ ext ∈ Extensions : ext.extension = extension

Modifies:
requestor, new user': UserAccount, UserAccounts, new phoneAccount': PhoneAccount,
PhoneAccounts, new ext': NormalExtension, Extensions, NormalExtensions,
userPhoneAccounts, accountExtensions, phoneExtensions, accountBillingPlans

Postconditions:
requestor’.lastActionTime = currentTime
if |UserAccounts| > 0 then
    let mostRecent = u ∈ UserAccounts |
          ( ∀ u2 ∈ UserAccounts : u ≠ u2 ⇒ u.accountNum > u2.accountNum )
    let id = mostRecent.accountNum + 1
else
    let id = 0
new user' = (id, name, address, alternatePhone, birthDate, false, false)
new phoneAccount' = (isSuspended, 0, send, receive, false, currentTime ,false)
new ext' = (extension)
UserAccounts' = UserAccounts ∪ {user'}
PhoneAccounts' = PhoneAccounts ∪ {phoneAccount'}
NormalExtensions' = NormalExtensions ∪ {ext'}
Extensions' = Extensions ∪ {ext'}
userPhoneAccounts' = userPhoneAccounts ∪ {(user', phoneAccount')}
accountExtensions' = accountExtensions ∪ {(phoneAccount', ext')}
phoneExtensions' = phoneExtensions ∪ {(ext', phone)}
accountBillingPlans' = accountBillingPlans ∪ {(phoneAccount', billingPlan)}

Exceptions:
If any precondition does not hold
  requestor' = requestor
  UserAccounts' = UserAccounts
  PhoneAccounts' = PhoneAccounts
  NormalExtensions' = NormalExtensions
  Extensions' = Extensions
  userPhoneAccounts' = userPhoneAccounts
  accountExtensions' = accountExtensions
  phoneExtensions' = phoneExtensions
  accountBillingPlans' = accountBillingPlans

References:
Course Project Overview: 6, 15, 21, 22, 24
Minutes #1: 4.2, 4.3, 4.4, 4.8, 5.1, 7.6
Minutes #2: 2.4, 2.8, 6.3
Minutes #3: 4.2, 4.3

Deleting a User Account

ID: 10  Importance: D

Overview:
Delete a phone account from the system.

Inputs:
requestor : Administrator
userAccount : UserAccount

Preconditions:
requestor initiates deleting a user account,
requestor ∈ Administrators,
requestor.isLoggedIn = true,
userAccount.isLocked = false,
\forall p \in userAccount.PhoneAccount : (p.isLocked = false AND p.outstandingBalance \leq 0)

**Modifies:**
requestor, userAccount, p’ \in userAccount.PhoneAccount, accountCharges, new c’ :
ServiceCharge, ServiceCharges, Charges, accountExtensions, incomingFilters, outgoingFilters,
FilterLists, filteredExpressions, FilterExpressions

**Postconditions:**
requestor’.lastActionTime = currentTime
userAccount’.isDeleted = true
let phoneAccts = userAccount.PhoneAccount
accountExtensions’ = accountExtensions - \{(p, p.NormalExtension) | p \in phoneAccts\}
outgoingFilters’ = outgoingFilters – \{(p, p.outlist) | p \in phoneAccts\}
incomingFilters’ = incomingFilters – \{(p, p.inclist) | p \in phoneAccts\}
FilterLists’ = FilterLists - \{p.outlist | p \in phoneAccts\} - \{p.inclist | p \in phoneAccts\}
let oldFilters = \{p.outlist | p \in phoneAccts\} \cup \{p.inclist | p \in phoneAccts\}
let oldFilteredExpressions = \{(f, fe) \in filteredExpressions | f \in oldFilters\}
filteredExpressions’ = filteredExpressions – oldFilteredExpressions
let oldFilterExpressions = \{fe \in FilterExpressions |
\exists fde \in oldFilteredExpressions : fde.FilterExpression = fe\}
FilterExpressions = FilterExpressions - oldFilterExpressions
\forall phoneAccount \in phoneAccts : ( phoneAccount’.isDeleted = true
let newest = c \in phoneAccount.Charge | ( c \in ServiceCharges AND
\forall ch \in phoneAccount.Charge : ( ch \in ServiceCharges AND ch \neq newest) \Rightarrow ch.chargeDate < newest.chargeDate ) )

let baseFee = phoneAccount.BillingPlan.serviceFee
let amount = (currentTime – newest) * baseFee / 30
new c’ = (currentTime, amount, 0, "Prorated service fee for final billing period")
accountCharges’ = accountCharges \cup \{(phoneAccount’, c’\)}
ServiceCharges’ = ServiceCharges \cup \{c’\}
Charges’ = Charges \cup \{c’\}

**Exceptions:**
If any precondition does not hold
userAccount’ = userAccount
phoneAccount’ = phoneAccount
accountExtensions’ = accountExtensions
accountCharges’ = accountCharges
ServiceCharges’ = ServiceCharges
Charges’ = Charges
incomingFilters’ = incomingFilters
outgoingFilters’ = outgoingFilters
FilterLists’ = FilterLists
filteredExpressions’ = filteredExpressions
FilterExpressions’ = FilterExpressions

References:
Course Project Overview: 24
Minutes #1: 4.2, 4.3, 4.13, 6.1
Minutes #3: 5.3, 5.6

Editing a Billing Plan

Overview:
A old Billing Plan is modified

Inputs:
requestor : Administrator
bp : BillingPlan
name : String
serviceFee, baseRate : dollarAmount
hasCallBlocking : boolean
discountPeriods : set of DiscountPeriod

Preconditions:
requestor initiates modifying a billing plan,
requestor ∈ Administrators,
requestor.isLoggedIn = true,
bp.isLocked = false
bp isn’t locked by another administrator,
serviceFee ≥ 0,
baseRate ≥ 0,
bp ∈ BillingPlans,
∀ dp ∈ discountPeriods : (
    ∀ dp2 ∈ discountPeriods : (
        dp ≠ dp2 ⇒ (dp.endTime < dp2.startTime OR
                     dp2.endTime < dp.startTime)) AND
        dp.discountPercentage ≥ 0 AND
        dp.startTime < dp.endTime AND
        dp.startTime.Date = dp.endTime.Date)

Modifies:
requestor, bp, DiscountPeriods, planDiscountPeriods
**Postconditions:**
requestor'.lastActionTime = currentTIne
bp'.name = name
bp'.serviceFee = serviceFee
bp'.hasCallBlocking = hasCallBlocking
bp'.baseRate = baseRate,
let oldPDPs = {pdp ∈ planDiscountPeriods | pdp.BillingPlan = bp}
let newPDPs = {(bp', y) | y ∈ DiscountPeriods}
planDiscountPeriods' = planDiscountPeriods - oldPDPs ∪ newPDPs
let oldDPs = {dp ∈ DiscountPeriods | dp.BillingPlan = bp}
DiscountPeriods' = DiscountPeriods - oldDPs ∪ discountPeriods
bp'.DiscountPeriods = discountPeriods
bp'.isLocked = false

**Exceptions:**
If any of the preconditions are not met
requestor' = requestor
bp' = bp
DiscountPeriods' = DiscountPeriods
planDiscountPeriods' = planDiscountPeriods

**References:**
Course Project Overview: 54, 56, 57, 58
Minutes #1: 7.6
Minutes #2: 6.2
Minutes #3: 1.2, 6.2, 6.3, 6.4

**Adding an Administrator Account**

**Overview:**
Adds a new administrator account to the system with the specified username, password and full name

**Inputs:**
requestor : Administrator
username, password, fullName : String

**Preconditions:**
requestor initiates adding a new administrator account,
requestor ∈ Administrators,
requestor isLoggedIn = true,
NOT ∃ admin ∈ Administrators : (admin.username = username),
password is alphanumeric or containing characters in set {!@#$%^&*()} and has length [8, 256] characters,
username is alphanumeric and has length [8, 256] characters
Modifies:
requestor, Administrators, new admin' : Administrator

Postconditions:
requestor'.lastActionTime = currentTime
new admin' = (username, password, fullName, false, currentTime, false)
Administrators' = Administrators ∪ {admin'}

Exceptions:
If any precondition does not hold
    requestor' = requestor
    Administrators' = Administrators

References:
Course Project Overview: 20
Minutes #1: 3.2
Minutes #2: 5.5

Deleting an Administrator Account

Overview:
Deletes an administrator account

Inputs:
requestor, deleted : Administrator

Preconditions:
requestor initiates deleting an administrator account,
requestor ∈ Administrators,
requestor.isLoggedIn = true,
deleted ∈ Administrators,
deleted.isLocked = false

Modifies:
requestor, Administrators, deleted

Postconditions:
requestor'.lastActionTime = currentTime
deleted'.isLoggedIn = false
Administrators = Administrators - deleted

Exceptions:
If any precondition does not hold
    requestor' = requestor
    Administrators' = Administrators
Adding an Incoming Filter Expression

Overview:
A Incoming Filtered Extension is created

Inputs:
requestor : Administrator
p : PhoneAccount
ext : String

Preconditions:
requestor initiates the creation of an incoming filtered expression,
requestor ∈ Administrators,
requestor.isLoggedIn = true,
p.isLocked = false,
p ∈ PhoneAccounts,
ext is a 4 digit extension (allowing wildcards),
NOT ∃ fe ∈ FilterExpressions : (fe.expression = ext AND fe.list.account = p)

Modifies:
requestor, filteredExpressions, new fe' : FilterExpression, FilterExpressions

Postconditions:
requestor'.lastActionTime = currentTime
new fe' = (ext)
filteredExpressions' = filteredExpressions ∈ {(p.incList, fe')}
FilterExpressions' = FilterExpressions U {fe'}
p.isLocked = false

Exceptions:
If any precondition does not hold
  requestor' = requestor
  filteredExpressions' = filteredExpressions
  FilterExpressions' = FilterExpressions

References:
Minutes #1: 4.1, 4.10, 4.11, 4.12
Minutes #3: 1.3, 1.12, 4.1
Adding a Blocked Extension

Overview:
The user adds a blocked extension to the blocked numbers list, using the #701 special number mechanism.

Inputs:
blockingAccount : PhoneAccount
blockedExtension : integer

Preconditions:
phone user initiates adding a blocked number,
blockingAccount ∈ PhoneAccounts,
blockingAccount.isSuspended = false,
blockingAccount.accountBillingPlans.hasCallBlocking = true
0 ≤ blockedExtension < 10000
NOT ∃ b ∈ blockedExtensions : (b.Extension.extension = blockedExtension AND b.PhoneAccount = blockingAccount)

Modifies:
Blocked, blockedExtensions, new ext’ : Extension

Postconditions:
if ∃ e ∈ Blocked : e.extension = blockedExtension
   blockedExtensions' = blockedExtensions ∪ {(blockingAccount, e)}
else
   new ext’ = (blockedExtension)
   Blocked’ = Blocked ∪ {ext’}
   blockedExtensions' = blockedExtensions ∪ {(blockingAccount, ext’)}

Exceptions:
If preconditions are not met
   Blocked’= Blocked
   blockedExtensions' = blockedExtensions

References:
Minutes #1: 8.1, 8.4
Minutes #2: 7.1, 7.3, 7.4, 7.5, 7.6
Minutes #3: 2.1, 2.2
### Adding an IP Address

**Overview:**
An administrator informs the system that there a phone on the network at the given IP address.

**Inputs:**
- `ip`: String
- `requestor`: Administrator

**Preconditions:**
- `requestor` requests to add an IP address,
- `requestor ∈ Administrators`,
- `requestorisLoggedIn = true`,
- `∀ p ∈ Phones : p.ip ≠ ip`

**Modifies:**
- `requestor`, `Phones`

**Postconditions:**
- `requestor'.lastActionTime = currentTime`
- `Phones' = Phones ∪ (false, false, ip, true)`

**Exceptions:**
If preconditions are not met
- `requestor' = requestor`
- `Phones' = Phones`

**References:**
Minutes #3: 12

### Running a Hardware Test

**Overview:**
Either an administrator requests a hardware test for a particular phone, or the system's periodic testing triggers a hardware test for a particular phone.

**Inputs:**
- `ip`: String

**Preconditions:**
- Administrator or system initiates a hardware test,
- `∃ p ∈ phone : p.ip = ip`

**Modifies:**
Errors, p

**Postconditions:**
if hardware failure was detected on phone p
   Errors' = Errors ∪ \{("Hardware failure at ip address " + ip)\}
   p’.isInService = false
else
   Errors' = Errors
   p’.isInService = true

**Exceptions:**
If preconditions are not met
   Errors' = Errors
   p’ = p

**References:**
Course Project Overview: 27, 28, 35
Minutes #1: 1.7, 5.2
Minutes #2: 5.6, 5.7, 5.10, 5.14
Minutes #3: 5.1, 5.2
3.2.4 State Machine Models

VAR startTime : int = null

Phone System

When the phone system is turned off, the administrator interface remains on, so that the administrator can turn the system back on. When the system is on, call processing is active (Phones state), testing of phone hardware (Testing state) and the automatic billing (Waiting for Spm state).

Phone System On

Phones

Testing

Waiting for Spm

at(Spm) / addServiceCharges, suspendDelinquentAccounts

Figure 5 - Phone System SSD

The highest level SSD that contains all functionality of the system.
Figure 6 - Administration SSD
Models the administrator’s GUI at the highest level.
Figure 7 - Administrators Account SSD
Navigation screens for managing the administrators of the system.
Figure 8 - Editing an Admin Account SSD
Models the process of editing an administrator’s account.
Figure 9 - Remove Admin Account SSD
Modes the process of removing an administrator or set of administrators from the system.
Figure 10 - Add New Administrator SSD
Models the process of adding an administrator to the system.
Figure 11 - Configure Billings Plans SSD
Navigation screens for the managing of the billing plans.
Figure 12 - Enter Billing Plan Details SSD
Models the process of entering billing plan details for a new or existing administrator account.
Figure 13 - Configuring System SSD
Navigation screens for configuring the system.
Figure 14 - Request System Test SSD
Models the process of requesting a hardware test on one or more phones.

The testFinished signal comes from the Testing state machine.
Figure 15 - Calls In System SSD
Models the process of viewing and terminating active calls in the system.
Figure 16 - Configuring Maximum Number of Calls SSD
Models the process of viewing and modifying the maximum number of calls in the system.
Figure 17 - IPs SSD
Models the process of adding IPs to the system, removing IPs from the system and changing the state of IPs.
Figure 18 - Configure Emergency Extension SSD
Models the process of configuring the emergency extensions in the system.

Note: We lock both emergency extensions together because there are only two of them and they will not be changed frequently in typical system usage.
Figure 19 - Modify User Account SSD

Models the process of modifying the user accounts in the system.
Figure 20 - Add User Account
Models the process of adding a user account to the system. This includes adding a phone account for the new user account.

Locking is not necessary for adding a new user account because no other administrator can see the account until it has been saved, at which point it can only be changed by being "edited" which is depicted in a different section of the SSD and does not lock the account.
Figure 21 - Modify Phone Account SSD
Models the process of modifying and deleting phone accounts. This includes paying bills for that phone account.
Figure 22 - Add Phone Account SSD
Models the process of adding a phone account to the system.
Figure 23 - Configure Filtered Extensions SSD
Models the process of configuring extensions in the system.
Figure 24 - Phones SSD
Models the phone hardware at a high level.

The testFinished signals are sent from the Testing state machine. The Enable and Disable signals are sent from the IPs state machine.
Figure 25 - Phone_inService SSD
Models call processing.
Figure 26 - Call Blocking SSD
Models the process of a user modifying their call blocking settings.
Figure 27 - Testing SSD
Models the automated and manual testing of phone hardware.
3.3 Performance

Performance is a critical aspect of the system (Minutes #5: 3.50); in situations where tradeoffs are to be considered, the first preference is given to its optimization.

Note: Performance is measured with respect to a typical system load, which consists of 500 calls. The benchmark used will be a Pentium dual core Xeon with 3 GB of memory and 1 TB of storage.

### Recording Call's Charge

<table>
<thead>
<tr>
<th>ID: NF01</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Time required for a call to be recorded upon completion.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
</tr>
<tr>
<td>0.1 s</td>
<td>0.5 s</td>
</tr>
<tr>
<td>References: Minutes #5: 3.2</td>
<td></td>
</tr>
</tbody>
</table>

### Phone Ringing

<table>
<thead>
<tr>
<th>ID: NF02</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Time required for callee’s phone to ring after the extension is dialled.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
</tr>
<tr>
<td>0.1 s</td>
<td>0.5 s</td>
</tr>
<tr>
<td>References: Minutes #5: 3.3</td>
<td></td>
</tr>
</tbody>
</table>

### Establish Audio Connections

<table>
<thead>
<tr>
<th>ID: NF03</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Time required to connect a call (audio connection) when the callee’s receiver is picked up.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
</tr>
<tr>
<td>0.1 s</td>
<td>0.5 s</td>
</tr>
<tr>
<td>References: Minutes #5: 3.4</td>
<td></td>
</tr>
</tbody>
</table>

### Responsiveness of the Administrator UI

<table>
<thead>
<tr>
<th>ID: NF04</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Time required for a UI screen update after an action has been taken (includes requesting hardware tests).</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
</tr>
<tr>
<td>0.5 s</td>
<td>5 s</td>
</tr>
<tr>
<td>References: Minutes #5: 3.5, 3.6</td>
<td></td>
</tr>
<tr>
<td><strong>Hardware Failure Detection</strong></td>
<td>ID: NF05</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Overview: Time required for a phone hardware failure to be detected.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
</tr>
<tr>
<td></td>
<td>1 min</td>
</tr>
<tr>
<td>References: Meeting #2: 5.6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Notification of Errors</strong></th>
<th>ID: NF06</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Time required for administrators to be notified of detected errors.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>10 s</td>
<td></td>
</tr>
<tr>
<td>References: Minutes #5: 3.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Modification of Billing Plans</strong></th>
<th>ID: NF07</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Time required to save modification of a phone account's billing plan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>0.1 s</td>
<td>0.5 s</td>
</tr>
<tr>
<td>References: Minutes #5: 3.8</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Maximum Simultaneous Calls</strong></th>
<th>ID: NF08</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Maximum simultaneous calls supported by the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>20 000</td>
<td>10 000</td>
</tr>
<tr>
<td>References: Minutes #5: 3.10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Maximum Simultaneous Administrators</strong></th>
<th>ID: NF09</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Maximum simultaneous logged in administrators supported by the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>References: Minutes #5: 3.11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Maximum Administrator Accounts</strong></th>
<th>ID: NF010</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Maximum number of total administrator accounts supported by the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Maximum Administrator Accounts

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2000</td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.11

### User Data

Overview: The initial number of user accounts and expected growth of user accounts supported by the system.

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10,000 + 200/day</td>
<td>5,000 + 100/day</td>
<td>5,000 + 5/day</td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.37

### 3.4 Design Constraints

Portability, modifiability, and reuse are important aspects of the design constraint. Security is not a factor and will not be considered. (Minutes #5: 3.50)

The system will be completely isolated from external factors. External factors will not constrain the system.

The system will run on a dedicated server, so computer usage is not an issue.

### Upgrade Constraints

Overview: Time that the system can be down during upgrades.

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Zero-downtime upgrades</td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.28

### Time to Add Features

Overview: Time required to add a new requested feature to the system.

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2 months</td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.31
<table>
<thead>
<tr>
<th><strong>Security Standards</strong></th>
<th>ID: NF14</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Security standards to which the system should adhere.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>SHA-256</td>
<td>MD5</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #2: 5.11, Minutes #5: 3.34</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Supported Platform</strong></th>
<th>ID: NF15</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Platforms with which the system must be compatible.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>BSD, Linux 2.6+, Solaris, Windows</td>
<td></td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.36</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Cost of the System</strong></th>
<th>ID: NF16</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Expected cost, in Canadian dollars, to develop the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>$0</td>
<td>$3600</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.42</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Rounding Precision</strong></th>
<th>ID: NF17</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> The rounding methods required to achieve desired precision.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>Banker's rounding, 1/100 cent</td>
<td></td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.40</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Delivery Time Line</strong></th>
<th>ID: NF18</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Time required to develop the software.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>4 months</td>
<td></td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.41</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Documentation</strong></th>
<th>ID: NF19</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Required documentation for system, phone users and administrators.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Documentation

**ID:** NF19  
**Importance:** D

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
</table>
| Manual for administrators  
Manual for phones  
Full system design document  
Preconditions and postconditions for all functions |

References: Minutes #5: 3.43

## System Architecture Design

**ID:** NF20  
**Importance:** D

**Overview:** The constraints imposed on the architecture of the software system.

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular, plug-in style architecture.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.46

## Programming Language Constraints

**ID:** NF21  
**Importance:** E

**Overview:** The constraints put on the programming language used to develop the system.

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Must be written in C.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.47

## External Dependency

**ID:** NF22  
**Importance:** D

**Overview:** The restrictions placed on the system's dependency on external sources (i.e. libraries)

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only have dependency on open source libraries.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.49

## Memory Restrictions

**ID:** NF23  
**Importance:** D

**Overview:** The restriction on the memory footprint of the system.

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>50 MB</td>
<td>100 MB</td>
<td>3 GB</td>
<td></td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.49
3.5 Quality Attributes

The critical quality measures are phone user usability, maintainability, and testability. Readability is less important, and administration interface usability is the least important. (Minutes #5: 3.50)

<table>
<thead>
<tr>
<th>Administrator Training Time</th>
<th>ID: NF24</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: The median time required to train new administrators.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>2 days</td>
<td>1 week</td>
<td>2 weeks</td>
</tr>
<tr>
<td>References: Minutes #5: 3.12</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intuitive Use</th>
<th>ID: NF25</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: The percentage of tasks that can be completed by the administrator without using a manual.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>90%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>References: Minutes #5: 3.14a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Help Pages</th>
<th>ID: NF26</th>
<th>Importance: O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Help pages per UI screen.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>References: Minutes #5: 3.14b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Help Quality</th>
<th>ID: NF27</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: Percentage of questions resolved through system help without assistance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>100%</td>
<td>100%</td>
<td>95%</td>
</tr>
<tr>
<td>References: Minutes #5: 3.14c</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Time to Add Customer Accounts</strong></td>
<td>ID: NF28</td>
<td>Importance: D</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------</td>
<td>---------------</td>
</tr>
<tr>
<td><strong>Overview:</strong> Time required to modify/add customer and phone accounts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>5 min (avg)</td>
<td>10 min (avg)</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Administrator Errors</strong></th>
<th>ID: NF29</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Permissible frequency and severity of administrator errors</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>99.9% free, minor errors</td>
<td>98% free, minor errors</td>
</tr>
<tr>
<td></td>
<td>100% free, major errors</td>
<td>100% free, major errors</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Customer Satisfaction</strong></th>
<th>ID: NF30</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Ratio of customers that respond with “Satisfied” or better when asked about their happiness with the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>10/10 users</td>
<td>9/10 users</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>System Availability</strong></th>
<th>ID: NF31</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> The required uptime of the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>100%</td>
<td>99.9%</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.19</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Mean Time to Failure</strong></th>
<th>ID: NF32</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overview:</strong> Mean time to failure of the system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fit Criteria:</strong></td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td></td>
<td>20 000 days</td>
<td>10 000 days</td>
</tr>
<tr>
<td><strong>References:</strong> Minutes #5: 3.20a</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Dropped Calls</strong></th>
<th>ID: NF33</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>ID: NF33</td>
<td>Importance: E</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td>--------------</td>
</tr>
<tr>
<td><strong>Dropped Calls</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview:</td>
<td>Number of calls dropped by the system permissible.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>1/20,000 calls</td>
<td>1/10,000 calls</td>
<td>1/8,000 calls</td>
</tr>
<tr>
<td>References:</td>
<td>Minutes #5: 3.20b</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>ID: NF34</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GUI Uptime</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview:</td>
<td>Mean time before GUI becomes completely unresponsive (goes down).</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>3 years</td>
<td></td>
<td></td>
</tr>
<tr>
<td>References:</td>
<td>Minutes #5: 3.20c</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>ID: NF35</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record Lifetime</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview:</td>
<td>Length of time the system should keep records.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>Indefinitely</td>
<td></td>
<td></td>
</tr>
<tr>
<td>References:</td>
<td>Minutes #5: 3.20c</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>ID: NF36</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Failure Recovery</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview:</td>
<td>Time it takes for the system to be restored (operationally) after a failure.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>5 min</td>
<td>10 min</td>
<td>20 min</td>
</tr>
<tr>
<td>References:</td>
<td>Minutes #5: 3.24</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>ID: NF37</th>
<th>Importance: D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bugs Per Lines</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview:</td>
<td>Number of bugs per lines of code written.</td>
<td></td>
</tr>
<tr>
<td>Fit Criteria:</td>
<td>Outstanding</td>
<td>Target</td>
</tr>
<tr>
<td>1 bug/200 lines</td>
<td></td>
<td></td>
</tr>
<tr>
<td>References:</td>
<td>Minutes #5: 3.25</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>ID: NF38</th>
<th>Importance: E</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fault Tolerance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overview:</td>
<td>The percentage of faults handled by the system</td>
<td></td>
</tr>
</tbody>
</table>

95
## Fault Tolerance

**ID:** NF38  
**Importance:** E

<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>The system must tolerate 100% of all non-critical faults. It should never have a segmentation fault.</td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.30

## Effectiveness of Hardware Tests

**ID:** NF39  
**Importance:** E

<p>| Overview: The percentage of phone failures which should be detected by automated testing. |</p>
<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>99.9%</td>
<td>99%</td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.6

## Efficacy of Maintenance

**ID:** NF40  
**Importance:** E

<p>| Overview: What percentage of bug reports should be reviewed within one working day? What percentage of high priority bugs will be resolved within one day? |</p>
<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>95% of bug reports will be reviewed within 8 working hours. High priority defects will be resolved within 24h in 95% of cases.</td>
<td>-</td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.27

## Duration of Maintenance

**ID:** NF41  
**Importance:** E

<p>| Overview: How long will the software system be maintained and supported? |</p>
<table>
<thead>
<tr>
<th>Fit Criteria:</th>
<th>Outstanding</th>
<th>Target</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10 years</td>
<td>10 years</td>
</tr>
</tbody>
</table>

References: Minutes #5: 3.26
**Appendix A - Glossary**

**Basic Terms**

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator account</td>
<td>a username and password that an administrator can use to log in to the system</td>
</tr>
<tr>
<td>Alphanumeric</td>
<td>Containing either Roman letters or numerals</td>
</tr>
<tr>
<td>Billing plan</td>
<td>a monthly fee and cost per minute to call during periods of the day, that also specifies if a telephone account has call blocking or not</td>
</tr>
<tr>
<td>Bill</td>
<td>A document showing the amount owed by a phone system customer (along with a list of all applicable charges) for a specific billing period.</td>
</tr>
<tr>
<td>Billing Plan</td>
<td>A basic plan that is required by every phone account. Contains the base rate per minute for charging phone calls as well as the monthly service charge to be applied to a phone</td>
</tr>
<tr>
<td>Call (Phone call)</td>
<td>the process of one phone user using one of his/her phones to place a telephone call to valid extension within the system</td>
</tr>
<tr>
<td>Cancel</td>
<td>The action of aborting a particular User Interface operation</td>
</tr>
<tr>
<td>Charge</td>
<td>A single amount representing the amount owed by a user for a single phone call or service fee.</td>
</tr>
<tr>
<td>Disabled</td>
<td>a phone is disabled if it is not part of the automatic hardware testing</td>
</tr>
<tr>
<td>Discount Period</td>
<td>a period during which calls a charged only a percentage of the base rate of the current billing plan associated with a phone account</td>
</tr>
<tr>
<td>Enabled</td>
<td>a phone is enabled if it is part of the automatic hardware testing</td>
</tr>
<tr>
<td>Extension</td>
<td>a four-digit number used to identify a telephone account. Some extensions are reserved for special numbers (such as emergency numbers)</td>
</tr>
<tr>
<td>Error</td>
<td>An indication that a certain operation has failed to complete successfully</td>
</tr>
</tbody>
</table>
Filter List: a list of regular expressions and/or exact extension that apply to a particular incoming or outgoing filter

Filter Expression: a valid expression that can be added to a particular filter list

Hardware tests: remotely initiated tests to verify whether or not a phone is in working order

In-service: a phone is in-service if the most recent hardware test for it passed

Input: data which is provided by an entity outside the system to the system in order for the system to perform a certain action

Mechanism: a technical manifestation that performs a certain process

Out-of-service: a phone is out-of-service if the most recent hardware test for it failed

Phone (telephone): a telecommunications device that allows for voice messages to be transmitted between two or more distinct units over any distance.

Phone account: an entity used to represent one phone that is assigned to the user. Contains all billing plans associated with the phone, the block list, the incoming/outgoing filter lists and all the charges that have been applied to the particular phone.

Prorate: divide proportionally according to the time used

Password: A sequence of keyboard characters used to secure access to the system

Precondition: A statement (usually expressed with formal logic) that has to be true for a certain function or procedure to be initiated

Postcondition: A statement (usually expressed with formal logic) that expresses the state of the system upon the completion of a certain function or procedure

Ringback Tone: the tone heard by the caller while the callee's phone is ringing

Suspended: a suspended telephone account is unable to send and receive calls, is not billed, but still has an active extension

Screen: A visual collection of widgets allowing the user to perform some desired functionality
String: An ordered sequence of alphanumeric characters. The sequence can be of varying length

Telephone Account: a collection of a telephone IP, extension, billing plan, filters, call blocking lists, and send/receive permissions

Telephone IP: an IP address that is statically associated to a phone

Telephone: see phone

User account: a user's account number, name, address, date of birth, and alternate phone number

Username: A word uniquely identifying a user of the system

User Interface (UI): A visual interface allowing a user to invoke software functionality

Wildcard: A character that can be substituted for any other valid character in a string

**Domain Model**

Phone: (class) A class modeling a phone

isInService: (attribute of Phone) Whether the phone is passing automated tests

isEnabled: (attribute of Phone) Whether the phone is part of automated testing

ip: (attribute of Phone) The ip of the phone

onHook: (attribute of Phone) Whether the receiver is on the hook

Extension: (class) A class modeling an extension

extension: (attribute of Extension) A 4 digit number

EmergencyExtension: (class, subclass of Extension) A class modeling an emergency extension

NormalExtension: (class, subclass of Extension) A class modeling a normal extension

PhoneCall: (class) A class modeling a call in progress

startDate: (attribute of PhoneCall) The starting date/time of the call
**maxCalls:** (attribute of PhoneCall) The maximum number of calls in the system

**DiscountPeriod:** (class) A class modeling a discount period, part of a billing plan

**startTime:** (attribute of DiscountPeriod) The date/time that the discount period starts

**endTime:** (attribute of DiscountPeriod) The date/time that the discount period ends

**discountPercentage:** (attribute of DiscountPeriod) The percentage of the base rate charged in this discount period

**BillingPlan:** (class) A class modeling a billing plan

**name:** (attribute of BillingPlan) The name of the billing plan

**serviceFee:** (attribute of BillingPlan) The service fee applied every 30 days

**hasCallBlocking:** (attribute of BillingPlan) Whether this billing plan contains the call blocking feature

**baseRate:** (attribute of BillingPlan) The rate per minute of this billing plan

**isDeleted:** (attribute of BillingPlan) Whether this billing plan has been deleted

**isLocked:** (attribute of BillingPlan) Whether the billing plan has been locked by an administrator for editing

**UserAccount:** (class) A class modeling a user account

**accountNum:** (attribute of UserAccount) The numerical identifier for the user account

**name:** (attribute of UserAccount) The name of the user

**address:** (attribute of UserAccount) The address of the user

**alternatePhone:** (attribute of UserAccount) The alternate phone number of the user

**birthDate:** (attribute of UserAccount) The user's birth date

**isDeleted:** (attribute of UserAccount) Whether this user account has been deleted
isLocked: (attribute of UserAccount) Whether this user account has been locked by an administrator for editing

PhoneAccount: (class) A class modeling a phone account

isSuspended: (attribute of PhoneAccount) Whether or not the phone account has been suspended

outstandingBalance: (attribute of PhoneAccount) The amount owed by the owner of the phone account

send: (attribute of PhoneAccount) Whether or not this phone account can make calls

receive: (attribute of PhoneAccount) Whether or not this phone account can receive calls

isDeleted: (attribute of PhoneAccount) Whether or not this phone account can make calls

createDate: (attribute of PhoneAccount) The date/time that this phone account has been created

isLocked: (attribute of PhoneAccount) this phone account has been locked by an administrator for editing

FilterList: (class) A class modeling a list of filter expressions

isLocked: (attribute of FilterList) Whether or not the list has been locked by an administrator for editing

FilterExpression: (class) A class modeling an expression used for a filter

equation: (attribute of FilterExpression) The expression (string of length 4, composed of digits and *)

Charge: (class) A class modeling a charge on a user's bill

chargeDate: (attribute of Charge) The date/time when the charge was applied

amount: (attribute of Charge) The total amount of the charge

amountPaid: (attribute of Charge) The amount of the charge that has been paid

CallCharge: (class, subclass of Charge) A class modeling a charge for making a call
callerExt: (attribute of CallCharge) The extension of the caller

calleeExt: (attribute of CallCharge) The extension of the callee

duration: (attribute of CallCharge) The duration of the call

baseRate: (attribute of CallCharge) The rate under which the charge is charged

ServiceCharge: (class, subclass of Charge) A class modeling a charge for reasons other than making a call

reason: (attribute of ServiceCharge) The reason the charge was applied

Administrator: (class) A class modeling an administrator

username: (attribute of Administrator) The username of the administrator account

password: (attribute of Administrator) The password of the administrator account

fullName: (attribute of Administrator) The administrator's full name

isLoggedIn: (attribute of Administrator) Whether or not the administrator is logged in

lastActionTime: (attribute of Administrator) The last time the administrator took an action

isLocked: (attribute of Administrator) Whether the administrator account is locked by an administrator for editing

Errors: (class) A class modeling an error

errorMessage: (attribute of Error) The body of the error message

Condition Functions

isLocked(entity): (condition function) This tests if the boolean variable uniquely associated with the given entity is true.

isUserAndPhonesLocked(userAccount):
This tests if the boolean variable uniquely associated with the given user account is true. It also tests to ensure that the boolean variable uniquely associated with each associated phone accounts is true.

**validateLogin(loginInfo):**
(condition function) Login information is considered valid if the username/password pair in the loginInfo matches the username and password for a valid admin in the system. This function returns true if the supplied loginInfo is valid.

**validateAdmin(adminDetails):**
(condition function) This function checks if an existing set of admin details is valid. Admin details are considered valid if the username is alphanumeric and is between 8 and 256 characters long, the password consists of characters in the set [A-Za-z0-9!@#$%^&*()], and the full name consists of printable characters. The username must also not already exist in the system.

**validateNewAdmin(adminData):**
(condition function) This function checks if a new set of admin details is valid. A new set of admin details is considered valid if validateAdmin(adminDetails) passes, and additionally the username does not already exist in the system.

**validateBillingPlan(billingPlan):**
(condition function) Billing plan details are a the regular charge rate for calls, a list of periods and their discount rates, the monthly service charge, and whether or not you can do call blocking. This action will ensure that the regular charge rate is a valid positive dollar amount, the monthly service charge is a non-negative dollar amount.

**validateDiscountPeriod(period, billingPlan):**
(condition function) This function checks that the discount rates are non-negative percentages, and that none of the discount periods overlap.

**validateMaxCalls(num):**
(condition function) This function checks that num is a positive number.

**validateEmergencyExtension(extension, IP):**
(condition function) This function verifies that the extension is a 4-digit number, the extension is not in use, and the IP is not already assigned an extension.

**validateUserAccount:**
(condition function) This function checks that the name, address, alternate phone number, account number, and date of birth are specified. It also checks that the date of birth is some date in the past.

**validatePhoneAccount(phoneAccount):**
(condition function) This function checks that the telephone IP, billing plan, and extension are specified.
validateFiltered(ext):
   (condition function) This function verifies that the ext is 4 characters long, and each of
   the characters is from the set [0-9,*].

validatePaymentAmount(amount):
   (condition function) This function checks that the amount is a positive dollar amount.

okToDeleteIP(IP):
   (condition function) This function will return true if and only if the phone designated by
   the IP is not associated with a phone account.

validateIP(IP):
   (condition function) This function checks if the IP is a string consisting of 4 numbers in
   the range 0-255, separated by dots (as in 192.168.1.55). Also check that the IP does not
   exist in the system.

okayToDeletePhoneAccount(phoneAccount):
   (condition function) Returns true if phone account has no outstanding balances.

okayToDeleteUser(userAccount):
   (condition function) This function checks will return true if none of user account's
   associated phone accounts have outstanding balances.

extAndIPsAvailable:
   (condition function) Returns true if there is at least one unused extension and at least one
   unused IP available in the system.

extAvailable:
   (condition function) Returns true if there is at least one unused extension available in the
   system.

billingPlanAvailable:
   (condition function) Returns true if a non-deleted billing plan is available to be used.

phoneAccountCreated(userAccount):
   (condition function) Returns true if a phone account was created for the defined user
   account.

loggedIn(admin):
   (condition function) Returns true if the specified admin is logged in.

**Action**

disconnectCalls:
(action) This action causes the system to send a callDisconnected signal to every phone's state machine (rather than just an individual phone).

**startSystemTests(adminID, IPs[]):**
(action) This action causes the system to send a startTest(adminID, false) signal to the testing state machines for each IP.

**recordServiceCharges(elapsedTimeRunning):**
(action) This action prorates the service charges of all of the bills to only include time where the system was running.

**saveAdminDetails(adminDetails):**
(action) This action is used to save the username, password, and full name of an administrator in the system.

**deleteAdminAccounts(adminRemoveArray[]):**
(action) For each username specified in the argument array, this action removes the corresponding admin details from the system.

**addAdmin(adminData):**
(action) This action adds the given username, password and full name to the list of valid administrators in the system.

**deleteBillingPlan(billingPlan):**
(action) This action removes a billing plan from the list of available billing plans in the system.

**saveBillingPlan(billingPlan):**
(action) This action saves a billing plan to the list of available billing plans in the system. If the billing plan is already in the system, the previous values are overwritten.

**addDiscountPeriod(period, billingPlan):**
(action) This action adds a discount period to a billing plan, but does not save the result in the system.

**deleteDiscountPeriod(period, billingPlan):**
(action) This action removes a discount period from a billing plan, but does not save the result in the system.

**setMaximumCalls(maxCalls):**
(action) This action sets the maximum number of allowed calls in the system to the given value.

**terminateCall(call):**
The call parameter contains two IPs. This action sends a callDisconnected signal to the phone state machine that handles each of these two IPs.

**saveEmergencyExtension(index, extension, IP):**
(action) This action saves the given extension and IP as emergency extensions in the system.

**addUserAccount(userAccount):**
(action) This action saves a new user to the set stored by the system.

**saveUserAccount(userAccount):**
(action) This action updates the user information stored by the system. If the user is suspended, then all phone accounts are suspended.

**deleteUser(userAccount):**
(action) This action deletes all of the users phone accounts as would happen using the deletePhoneAccount action. If all of the phone accounts are removed (i.e. the user has no outstanding balances), the user is also removed from the system. Deletion of a user account can only commence if the adminstrator has a lock on the user and all associated phone accounts.

**deletePhoneAccount(phoneAccount):**
(action) If the phone account has a non-negative balance, then this action will remove the phone account from the system. If the phone account has a negative outstanding balance, then this action will generate a bill for the phone account, and will not remove the account from the system. This action also charges the last service fee for the account.

**killCalls(phoneAccount):**
(action) If the phone associated with this phone account is in a call, then this action will use the terminateCall action to disconnect the two calls.

**killCalls(userAccount):**
(action) If any of the phone accounts associated to the user have calls in progress, then this action will use the terminateCall action to disconnect the calls.

**addPhoneAccount(phoneAccount):**
(action) Adds a new phone account associated to a user account to the system.

**savePhoneAccount(phoneAccount):**
(action) This action updates the information stored for a phone account in the system.

**loadNewBill(period):**
(action) This action retrieves the bill for a particular historical billing period.

**addIncomingFiltered(ext, phoneAccount):**
(action) Add filters to a phone account's incoming filter list. Changes take effect immediately in the system (no save of the phone account is necessary).

**addOutgoingFiltered(ext, phoneAccount):**
(action) Add filters to a phone account's outgoing filter list. Changes take effect immediately in the system (no save of the phone account is necessary).

**deleteIncomingFiltered/incExt, phoneAccount):**
(action) Remove filters from a phone account's incoming filter list. Changes take effect immediately in the system (no save of the phone account is necessary).

**deleteOutgoingFiltered(outExt, phoneAccount):**
(action) Remove filters from a phone account's outgoing filter list. Changes take effect immediately in the system (no save of the phone account is necessary).

**registerBillPayment(amount, phoneAccount):**
(action) This action atomically increments the given phone account's balance by the given amount.

**addIP(IP):**
(action) This action adds an IP to the list of IPs managed by the system.

**deleteIP(IP):**
(action) This action removes an IP from the list of IPs managed by the system.

**changeState(IP, state):**
(action) This action will set whether or not a phone is enabled or disabled for automatic hardware tests. If state is disabled, it will send a Disable event to the associated phone. If state is enabled, it will send an Enable event to the associated phone.

**addBlockedExtension(extension):**
(action) This action is used by a phone's state machine in order to add an extension to the phone's list of blocked extensions. If the extension was blocked before, this action does nothing.

**removeBlockedExtension(extension):**
(action) This action is used by a phone's state machine in order to remove an extension from the phone's list of blocked extensions. If the extension was not blocked before, this action does nothing.

**clearBlockedExtensions(IP):**
(action) This action is used by a phone's state machine in order to clear all extensions from its blocked number list.

**testFinished(IP, adminID, passed):**
(action) This action fires a `testFinished(IP, adminID, passed)` event. If the passed variable is true, it sets the phone defined by IP to be in service. If passed is false, it sets the phone to be out of service, logs the error, and sends a “phoneHardwareFails(IP)” event to all logged in administrators. AdminId is null if it is not an administrator requested test.

**addErrorToList(IP):**

(action) This action adds an error message denoting the hardware failure of a phone with the given IP to the hardware failiure screen.

**genRandomExt:**

(action) Generates a unused random extension and populates it in the user interface element that initiated the action.

**addServiceCharges:**

(action) Adds service charges to every phone account whose billing period ends today at 5pm. The system will prorate service charges (defined in the billing plan) starting from the date of the last service charge in the current billing period of each qualifying phone account.

**suspendDelinquentAccounts:**

(action) Suspends the account of any user who has a bill that has had an outstanding balance for at least 3 months.

**call(callerIP, calleeIP):**

(action) Used to initiate a call with another phone. Sends a `call(callerIP, calleeIP)` event to the callee's phone.

**adminDeleted(adminRemoveArray[]):**

(action) Sends `adminDeleted(adminID)` event to each admin in the adminIDs array.

**calcCharge(caller, length, billingPlan):**

(action) Adds a charge the the specified caller's phone account, billing for a portion of a call of the specified length made under the specified billingPlan and its discount periods.

**lock(entity):**

(action) This sets the boolean variable uniquely associated with the given entity to true.

**unlock(entity):**

(action) This sets the boolean variable uniquely associated with the given entity to false.

**lockUserAndPhones(userAccount):**

(action) This sets the boolean variable uniquely associated with the given user account to true. It also sets the boolean variable for all associated phone accounts to true.

**unlockUserAndPhones(userAccount):**
(action) This sets the boolean variable uniquely associated with the given user account to false. It also sets the boolean variable for all associated phone accounts to false.

adminDeleted(admin):
   (action) Triggered when the specified administrator's account is deleted.

runPhoneHardwareTest(IP):
   (activity) This activity tests whether a phone, which is specified by the IP parameter, is responsive. The activity sends a test signal to a phone. If the phone responds, then the test passed and the activity concludes. If the phone does not respond within a certain amount of time, then the test fails and the activity concludes. The passed variable of the testFinished action is set accordingly.

Events

call(callerIP, calleeIP):
   (event) Sent by call(callerIP, calleeIP) action from caller to callee to start the callee phone ringing.

pickUpFailed:
   (event) Sent from callee to caller to indicate that the ten rings expired and a busy signal should be played to the caller.

doNotRingAnymore:
   (event) Sent from caller to callee to indicate that the call was canceled and ringing should stop.

callDisconnected:
   (event) Sent to a phone's state machine to indicate that the call has ended, either because an admin terminated the call or the other party hung up.

startTest(adminID, isAutomated):
   (event) Sent to a testing state machine to initiate a test. The initiating administrator's user ID is passed as the first parameter, and the second parameter is used to indicate whether or not the state machine should either a) repeatedly perform automated tests, or b) send back feedback to the given admin.

testFinished(IP, adminID, passed):
   (event) Event sent by the testFinished(IP, adminID, passed) action to the admin that initiated the test and the phones state diagram. The passed parameter is a boolean value indicating whether or not the test passed. AdminId is null if it is not an administrator requested test.

adminDeleted(adminID):
(event) Event sent by adminDeleted(adminIDs[]) action to an admin whenever that admin is deleted.

**generateBills:**
(event) Compiles the statistics necessary to display bills.

**Enable:**
(event) Sent by the changeState(IP, state) action to a phone to enable the phone.

**Disable:**
(event) Sent by the changeState(IP, state) action to a phone to disable the phone.

**call(callerIP, calleeIP):**
(event) Sent by call(callerIP, calleeIP) action from caller to callee to start the callee phone ringing.

**pickUpFailed:**
(event) Sent from callee to caller to indicate that the ten rings expired and a busy signal should be played to the caller.

**doNotRingAnymore:**
(event) Sent from caller to callee to indicate that the call was canceled and ringing should stop.

**callDisconnected:**
(event) Sent to a phone's state machine to indicate that the call has ended, either because an admin terminated the call or the other party hung up.

**startTest(adminID, isAutomated):**
(event) Sent to a testing state machine to initiate a test. The initiating administrator's user ID is passed as the first parameter, and the second parameter is used to indicate whether or not the state machine should either a) repeatedly perform automated tests, or b) send back feedback to the given admin.

**testFinished(IP, adminID, passed):**
(event) Event sent by the testFinished(IP, adminID, passed) action to the admin that initiated the test and the phones state diagram. The passed parameter is a boolean value indicating whether or not the test passed. AdminId is null if it is not an administrator requested test.

**adminDeleted(adminID):**
(event) Event sent by adminDeleted(adminIDs[]) action to an admin whenever that admin is deleted.

**generateBills:**
(event) Compiles the statistics necessary to display bills.

Enable:
(event) Sent by the changeState(IP, state) action to a phone to enable the phone.

Disable:
(event) Sent by the changeState(IP, state) action to a phone to disable the phone.

GUI Events:
Almost of the GUI events correspond to clicking a button and are named "clickX" (e.g. clickCancel or clickErrorOK) where X is the name or behaviour of the button (e.g. cancel or errorOK). For brevity, they are omitted from the glossary. The exceptions to this rule are listed below:

turnSystemOn:
(GUI event) The event triggered by clicking the "Turn System ON" button on GUI screen 1.

turnSystemOff:
(GUI event) The event triggered by clicking the "Turn System OFF" button on GUI screen 4.

GUI Screenshots

1: System on:
(GUI screenshot) The GUI screen for turning the system on.

2: Login administrator:
(GUI screenshot) The GUI screen for logging into the system.

3: Generic Error/Warning Message:
(GUI screenshot) The GUI screen for displaying and confirming error messages.

4: Main screen:
(GUI screenshot) The GUI screen for navigating to other menu subsections.

5: Administrator Management Screen:
(GUI screenshot) The GUI screen for navigating to administrator management operations.

6: Edit administrator list:
(GUI screenshot) The GUI screen for choosing an administrator to modify.

7: Edit administrator details:
8: **Remove administrator list:**
   (GUI screenshot) The GUI screen for removing an administrator from the system.

9: **Remove Administrator Confirmation:**
   (GUI screenshot) The GUI screen for confirming the removal of an administrator.

10: **Add new administrator:**
    (GUI screenshot) The GUI screen for entering information about a new administrator.

11: **Add New Administrator Confirmation:**
    (GUI screenshot) The GUI screen for confirming the addition of an administrator to the system.

12: **Configure Billing Plans:**
    (GUI screenshot) The GUI screen for selecting which billing plan to modify and for selecting to add a billing plan.

13: **Edit Billing Plan:**
    (GUI screenshot) The GUI screen for entering information about a billing plan.

14: **System Management:**
    (GUI screenshot) The GUI screen for navigating to system management operations.

15: **View System Error Logs:**
    (GUI screenshot) The GUI screen for viewing the system's error logs.

16: **Select Telephone for Hardware Testing:**
    (GUI screenshot) The GUI screen for choosing which phones should be manually hardware tested.

17: **View Max Calls in System:**
    (GUI screenshot) The GUI screen for viewing and selecting to modify the maximum calls in the system.

18: **Set max calls in system:**
    (GUI screenshot) The GUI screen for entering new maximum calls in the system.

19: **Terminating an Ongoing Call:**
    (GUI screenshot) The GUI screen for selecting to terminate a established call in the system.

20: **Configure Emergency Extensions:**
    (GUI screenshot) The GUI screen for selecting an emergency extension to modify.
21: Modify Emergency Extension:
   (GUI screenshot) The GUI screen for entering a new IP or extension for an emergency extension.

22: Change Emergency Extension Confirmation Screen:
   (GUI screenshot) The GUI screen for confirming the modification of an emergency extension.

23: List User Accounts:
   (GUI screenshot) The GUI screen for selecting to modify, add, or delete a user.

24: Remove User Account Prompt:
   (GUI screenshot) The GUI screen for deciding to delete a user (saying Yes or No).

25: User Deletion Success:
   (GUI screenshot) The GUI screen for confirming the removal of a user account from the system.

26: Edit User Account:
   (GUI screenshot) The GUI screen for entering information about a user account.

27: Editing User Account Success:
   (GUI screenshot) The GUI screen for confirming the modification of a user account's data.

28: User Account's Phone Account List:
   (GUI screenshot) The GUI screen for selecting to add, edit, record a payment, or delete a phone account.

29: Delete Phone Account Prompt:
   (GUI screenshot) The GUI screen for deciding to delete a phone account (saying Yes or No).

30: Delete Phone Account Success:
   (GUI screenshot) The GUI screen for confirming the deletion of a phone account.

31: Edit Phone Account Detail:
   (GUI screenshot) The GUI screen for entering (modifying) the information of a phone account.

32: Phone Account Save Success:
   (GUI screenshot) The GUI screen for confirming the modification of a phone account's data.

33: View Filtered Extensions:
(GUI screenshot) The GUI screen for adding and deleting the filtered extensions of a phone account.

34: Add incoming filtered extensions:
   (GUI screenshot) The GUI screen for entering an incoming filter to a phone account.

35: Add outgoing filtered extensions:
   (GUI screenshot) The GUI screen for entering an outgoing filter to a phone account.

36: Create Phone Account Success:
   (GUI screenshot) The GUI screen for confirming the successful addition of a phone account.

37: New User Account:
   (GUI screenshot) The GUI screen for entering information for a new user account.

38: New User Account Creation Successful:
   (GUI screenshot) The GUI screen for confirming the addition of a new user account.

39: Display IP List:
   (GUI screenshot) The GUI screen for adding, deleting, or changing the state of a IP, or for running a hardware test.

40: Change IP State:
   (GUI screenshot) The GUI screen for entering a new state for an IP.

41: Add New IP Address:
   (GUI screenshot) The GUI screen for entering information for a new IP.

Functions

Making a Call:
   (function) Checks if two parties can call each other and creates temporary call tracking object, called before call connects

Ending a Call:
   (function) Removes temporary call tracking object, called when call terminates

Charging for Part of a Call:
   (function) Charges the caller for part of a split call, called when a discount period ends, when billing period ends, or before a call terminates

Charging Monthly Service Fee:
(function) Charges the prorated service fee for an account at the end of a billing period and suspends the phone account if a bill over 3 billing periods old has not been fully paid off, called when a billing period ends

**Recording Bill Payment:**
(function) Recording a user's bill payment, called when an administrator enters a bill payment

**Prorating a Service Fee:**
(function) Charges the prorated service fee for an account during a billing period, called when system turns back on, when a billing plan changes, and when a phone account is suspended

**Editing a Phone Account:**
(function) Changes the details of a phone account, such as extension, IP, or billing plan, and charges a prorated service fee if the billing plan changes, called when an administrator edits a phone account and saves changes

**Deleting a Phone Account:**
(function) Marks a phone account as deleted, called when the administrator deletes a phone account

**Adding a User Account:**
(function) Adds a user account and an associated phone account, called when the administrator adds a user account

**Deleting a User Account:**
(function) Marks a user account and its associated phone accounts as deleted, called when the administrator deletes a user account

**Editing a Billing Plan:**
(function) Saves the changes to a billing plan, called when the administrator edits a billing plan and saves changes

**Adding an Administrator Account:**
(function) Adds an administrator account, called when an administrator creates another administrator account

**Deleting an Administrator Account:**
(function) Deletes an administrator account and logs out the deleted administrator, called when an administrator deletes an administrator account

**Adding an Incoming Filter Expression:**
(function) Adds a filter expression to the incoming filter list of a telephone account, called when an administrator adds an incoming filter expression to a phone account
Adding a Blocked Extension:
(function) Adds a blocked extension to a phone account, called when a user adds a
blocked extension to one of his/her phone account's blocked lists

Adding an IP Address:
(function) Adds an IP address to the system, called when an administrator adds an IP
address to the system

Running a Hardware Test:
(function) Updates phone status the results of a hardware test and reports any errors,
called when a hardware test on a phone has finished

States

Add New Admin:
(state) Prompts an administrator for details about a new administrator to be added

Add New Administrator:
(state) Allows an administrator to add a new administrator account

Add New IP:
(state) The state where an administrator is adding a new IP address to the system.

Add Phone Account:
(state) The state where a new phone account is added to the system.

Add user account:
(state) Composite state allowing for the addition of users to the system

Admin Account Locked Error:
(state) Informs an administrator that the the selected administrator account is locked and
cannot be edited

Admin List:
(state) Allows an administrator to select an administrator from a list

Administrators Account:
(state) Allows administrators to add, edit, and remove other administrators.

Billing Plan Locked Error:
(state) Informs an administrator that the billing plan they requested to edit or delete was
locked and could not be accessed

Busy:
(state) The state where a phone is playing a busy tone.
Call Blocking:
(state) The state where a user is modifying their call blocking settings.

Call Blocking Menu:
(state) The state where a phone is waiting for call blocking commands.

Callee Ringing:
(state) The state where a phone is causing another phone's ringer to sound.

Calls in system:
(state) Composite state that informs the administrator of the current calls in the system

Can not delete user error:
(state) State occurs when it is impossible to select a current user

Changing IP State:
(state) The state where an administrator is changing the state of an IP.

Clearing Blocked Numbers:
(state) The state where a phone is clearing all of its phone account's blocked extensions.

Configure Billing Plans:
(state) Prompts an administrator whether they would like to add, edit, or remove a billing plan

Configure Emergency Extension:
(state) The state where an administrator is configuring the emergency extensions in the system.

Configure Emergency Extensions:
(state) The state where an administrator is viewing the two emergency extensions in the system.

Configuring Filtered Extensions:
(state) The state which allows an administrator to modify the filtered extensions of a phone account.

Configuring Maximum Number of Calls:
(state) The state where an administrator views or defines the maximum calls in the system.

Configuring System:
(state) The state where an administrator is configuring the system (or navigating to configure the system).

Confirm Deletion:
(state) The state where the administrator is asked to confirm deleting an account

**Confirm Add Phone Account:**
(state) The state which allows an administrator to confirm the addition of a phone account.

**Confirm Emergency Extension Save:**
(state) The state where an administrator is viewing a message about the successful add of an emergency extension.

**Confirm Save Phone Account:**
(state) The state where the administrator is asked to confirm saving changes to a phone account

**Confirmation screen:**
(state) State informs the administrator that a new user has been successfully added

**Connected:**
(state) The state where a phone is transmitting audio with another phone.

**Delete Phone Account Prompt:**
(state) The state where the administrator is being prompted on whether or not to delete a phone account

**Delete user prompt:**
(state) A state which prompts the administrator which user to delete

**Dialing:**
(state) The state where a user is dialing a number on a phone.

**Digit:**
(state) The state where a user has just entered a digit.

**Disabled:**
(state) The state which denotes a phone (ip) is disabled.

**Disconnected:**
(state) The state where a phone is not connected to another phone.

**Display list of active calls:**
(state) A state which displays a succinct list of all currently ongoing calls in the system

**Display list of all other administrators:**
(state) Prompts an administrator with a list of all administrator accounts, not including the current administrator

**Display tests results:**
Displaying Administrators Account Menu:
(state) Allows an administrator to choose if they would like to add, edit, or delete an other administrator

Displaying Currently Filtered Extensions:
(state) The state which displays the incoming and outgoing extensions of a phone account.

Displaying List of IPs:
(state) The state where an administrator is viewing a list of IPs in the system.

Displaying Maximum Number of Calls in System:
(state) The state where the administrator can view the maximum number of calls in the system.

Displaying System Menu:
(state) The state where an administrator is able to navigate to certain system configuration options.

Edit Admin Account:
(state) Allows an administrator to edit the details of an administrator account

Edit Admin Details:
(state) Prompts an administrator for new administrator account details

Edit user account:
(state) State which awaits input of the new information for a specific user account

Editing Admin Details:
(state) Allows an administrator to edit the details of an administrator account

Editing user account:
(state) A composite state within which a user account is being edited

Enabled:
(state) The state denoting that a particular phone (ip) is enabled.

Enter Billing Plan Details:
(state) Prompts an administrator for details about the new or edited billing plan

Enter Phone Account Details:
(state) The state which allows the administrator to enter phone account information.

Entering Phone Account Details:
(state) The state where the administrator has a phone account locked and is ready to modify it

**Error:**
(state) The state where a phone is playing an error tone.

**Error Adding Discount Period:**
(state) Informs an administrator that there was an error adding the specified discount period

**Error Adding IP:**
(state) The state where an administrator is viewing an error about a failure to add an IP.

**Error Deleting Admin:**
(state) Informs an administrator that the requested delete of an administrator account could not be completed, due to a locking problem

**Error Deleting IP:**
(state) The state where an administrator is viewing an error about a failed deletion of an IP.

**Error Deleting Phone Account:**
(state) The state where the administrator is viewing an error due to a phone account being unable to be deleted

**Error With Admin Details:**
(state) Indicates to an administrator that the supplied administrator account details were not valid

**Error with Bill Payment:**
(state) The state where the administrator is viewing an error due to an invalid bill payment amount being entered

**Error With Billing Plan Details:**
(state) Informs an administrator that the given billing plan was not valid, so it could not be saved

**Error With Extensions:**
(state) The state which allows the administrator to view error information about an entered extension.

**Error With Number:**
(state) The state where an error pertaining to the entry of maximum number of calls is displayed.

**Extension Error:**
(state) The state where an administrator is viewing an error about a failed edit to an emergency extension.

**Fast Busy:**
(state) The state where a phone is playing a fast busy tone.

**Hung Up:**
(state) The state where a phone is on the hook.

**Idle:**
(state) The state where a phone is playing an idle tone.

**Inputting Digit:**
(state) The state where a user is entering non-first digit of an extension to add to or remove from their blocked number list.

**Inputting First Digit:**
(state) The state where a user is entering the first digit of an extension to add to or remove from their blocked number list.

**Inputting Number List:**
(state) The state where a user is entering a number to be added to or removed from their blocked number list.

**Invalid Admin Data:**
(state) Informs an administrator that the to-be-added administrator's data are not valid

**IP Locked Error:**
(state) The state where an administrator is viewing an error about an IP being locked.

**IPs:**
(state) The state where an administrator is managing the IPs in the system.

**Invalid Add Phone Data Page:**
(state) The state where input errors are displayed to the user.

**Invalid Phone Data:**
(state) The state where the administrator is viewing an error about invalid phone data

**Logged In:**
(state) Indicates that a particular administrator is logged in and can access system functionality

**Logged Out:**
(state) Indicates that a particular administrator is logged out and the system is prompting them for a username and password
Login Error:  
(state) Indicates that a particular administrator has entered invalid login credentials.

Lookup:  
(state) The state where a phone is looking up an extension to see if it exists.

Main Screen:  
(state) The top-level navigation screen that allows an administrator to access each of the main components of the administrator UI.

Modify Particular Emergency Extension:  
(state) The state where an administrator is editing the details of a single emergency extension.

Modify user account:  
(state) State allows an administrator to modify user accounts.

No available IP or extension error:  
(state) State shows an error if no IPs or extensions are available in the system.

No Extensions or No IPs Warning:  
(state) Administrator is viewing an error message indicating that there are no extensions or no IPs in the system.

No phones in system error:  
(state) A state informing the administrator that no phones are available in the system for testing.

Not Viewing Hardware Error:  
(state) When an administrator dismisses the notifications of hardware errors, they are returned to this state.

Off Hook:  
(state) The state where a phone is off the hook.

Out of Service:  
(state) The state which denotes a phone is out of service.

Phone Account Details:  
(state) The state where the administrator is entering phone account details to modify the phone account.

Phone Account Locked:  
(state) The state where the administrator is viewing an error about a phone account being locked by another administrator.
Phone System:
(state) Main container state that holds the entire system

Phone System Off:
(state) Indicates that the call processing, hardware testing, and automatic billing systems are not active

Phone System On:
(state) Indicates that the call processing, hardware testing, and automatic billing systems are not active

Phone_InService:
(state) The state which denotes that a phone is in service.

Phones:
(state) The state which defines the state a phone is in.

Phones:
(state) Container state for the call processing system

Prompt For New Maximum Number of Calls in System:
(state) The state where the administrator enters the maximum number of calls (for modification)

Prompt For New Incoming Extension:
(state) The state which allows the administrator to add a new incoming extension.

Prompt For New Outgoing Extension:
(state) The state which allows the administrator to add a new outgoing extension.

Receiving:
(state) The state where a phone is connected to an initiating phone.

Remove Admin Account:
(state) Allows an administrator to remove another administrator's account

Request system test:
(state) Composite state allowing an administrator to request that a system test be run

Ringer On:
(state) The state where the ringer of a phone is sounding.

Ringer Off:
(state) The state where the ringer of a phone is about to sound.
Ringing:
(state) The state where the ringer of a phone is sounding (or about to sound).

Running Hardware Tests:
(state) The state where an administrator is waiting for a requested hardware test to complete.

Save new administrator:
(state) Confirms to an administrator that they have successfully added a new administrator account to the system

Show telephone accounts list:
(state) A state providing the administrator with a list of all applicable phone accounts in the system

Shows confirmation message:
(state) Informs an administrator that the action that they most recently requested (deleting an administrator account) has been completed successfully

Silence:
(state) The state where a phone is not playing any sound.

Silent:
(state) The state where a phone is on the hook and is silent.

System Off:
(state) Administrators are in this state when they are properly logged in, but the system is off. This state allows an administrator to turn the system on

System On:
(state) Administrators are in this state when they are logged in, and the system is on

Testing (substate):
(state) The state where a phone is being tested.

Testing (superstate):
(state) Container state for the automatic hardware testing system

Transmitting:
(state) The state where a phone is connected to a phone and has initiated the connection.

User information screen:
(state) State displays where all necessary user information is entered

User locked error:
(state) State occurs when the selected user is currently locked by another administrator
Using Administrator Interface:
(state) Administrators are in this state when they are viewing any part of the administrator UI

View Error Log:
(state) The state where an administrator is viewing the error logs of the system.

View List of Phone Accounts:
(state) The state where the administrator is viewing a list of phone accounts

View list of users:
(state) State presents a list of all users in the system

Viewing List of Hardware Errors:
(state) When a hardware error occurs, an administrator is notified by being placed in this state and viewing the associated UI

Waiting:
(state) The state where a phone is waiting for a test to begin.

Waiting for test:
(state) A state which waits until the requested system phones test is run

Waiting for 0:
(state) The state where a phone is waiting for the user to enter a 0 key.

Waiting for 5pm:
(state) Waits for the time to become 5pm, so that automatic billing can take place

Waiting for 7:
(state) The state where a phone is waiting for the user to enter a 7 key.

Variables

startTime:
(variable) A variable storing the time that the system was turned on.

timesRung:
(variable) A variable storing the number of times that a phone has rung.

extensionDialed:
(variable) A variable storing a partial extension as it is dialed.

maxCalls:
(variable) A variable storing the maximum number of allowed calls in the system.

**requestedAdminID:**
(variable) A variable storing the ID of the administrator that requested a hardware test.

**automated:**
(variable) A variable indicating whether a test was automatically or manually initiated.
Appendix B – Meeting Minutes

Minutes #1 – Press conference 1 (Sept 22nd)
Minutes #2 – Press conference 2 (Sept 29th)
Minutes #3 – Customer/TA meeting 1 (Oct 6th)
Minutes #4 – Press conference 3 (Oct 20th)
Minutes #5 – Customer/TA meeting 2 (Nov 23rd)

*Please find the minutes attached with this document*