Actions on variables in the Domain Model are difficult, which is why we don't ask for them. We ask for (1) states (i.e., your UI screens), (2) transitions (events and conditions that cause a transition to a new UI screen) and (3) **state variables (that track information about the current state of execution and conditions on transitions) including actions on state variables.**
MODERN FAMILY did a great job on states and transitions. Ignore most actions on transitions, and ignore activities within states.

A key state variable in most systems is the "current user" -- indicating the user account that is logged into the system app. This "current user" variable needs to be declared.
VARIABLES
account: FamilyMemberAccount

RED HIGHLIGHTS:
input from the Login Screen when creating an account
should be the name, email, password of the new account -- not a whole account
createAccount(name, email, password)
 account = new FamilyMemberAccount (name, email, password)

ORANGE HIGHLIGHTS:
authentication means comparing the input email, password
against the accounts recorded in the Domain Model
login(email, password)
 [exists act:FamiilyMemberAccount
   act.email == email
   act.password == password]
 account = act

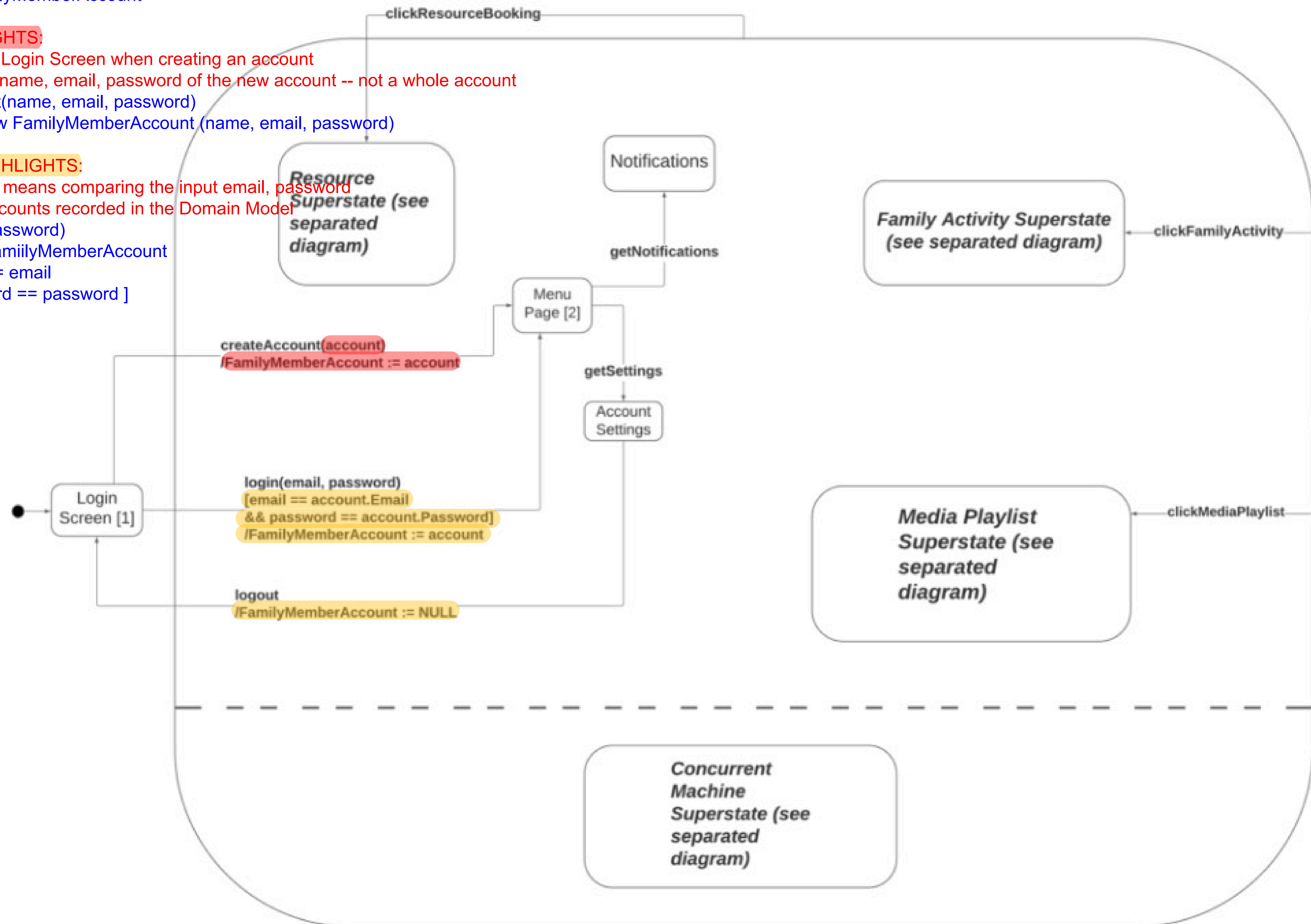

Figure 21: Overarching Navigation Diagram

Ignore transition actions. Ignore state activities.

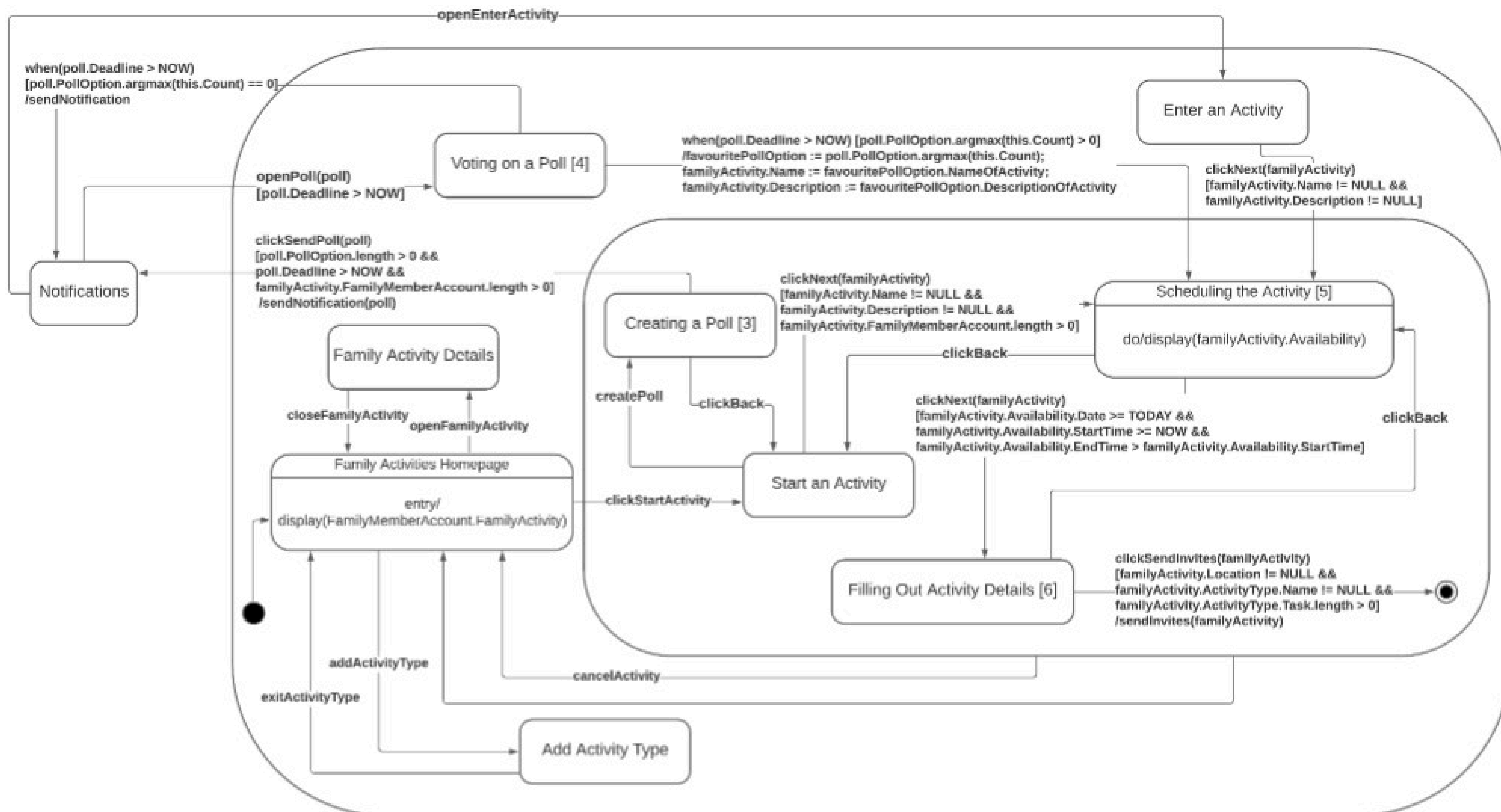

Figure 22: Navigation Diagram for Planning a Family Activity Use Case

Ignore transition actions. Ignore state activities.

Select Resource Page [7]
entry/display(FamilyMemberAccount Resource)
clickDeleteResource(resource)
Add Resource Page
clickAddNewResource
clickAdd(resource)
givenApprovedBookingOverride
clickBook(resource, datePicker)
[resource != NULL &&
datePicker.startDate != NULL && datePicker.startDate > TODAY &&
datePicker.endDate != NULL && datePicker.endDate > TODAY]
givenDeniedBookingOverride
Notifications
clickDone(booking)
[booking.priorityLevel != NULL && booking.reason != NULL]
Resource Booking Page [8]
entry/displaySelectedResource
do/fetchAndCheckAvailabilities
selectUsageTimeSlot(resource, booking, from, to)
[!(resource.Availability.StartTime < from &&
to < resource.Availability.EndTime)]
/booking.isAvailable := false
when(booking.isAvailable == true)
Menu Page [2]
clickSend(booking)
[booking.priorityLevel != NULL &&
booking.reason != NULL]
/sendNotification
Resource Booking Override Request [9]
after(1)
[!(resource.Availability.StartTime < from &&
to < resource.Availability.EndTime)]
/booking.isAvailable := false
clickSend(booking)
[booking.priorityLevel != NULL && booking.reason != NULL]
clickOpenResourceBookingOverrideRequest
clickApprove
Resource Booking Agreement
clickDeny

Figure 23: Navigation Diagram for Coordinate the Usage of Shared Resources Use Case

This submachine has multiple state variables: mode, error, files, isPlaying. These variables should be declared and transitions should include any actions on these variables.

Ignore actions on variables in the Domain Model. Ignore state activities.

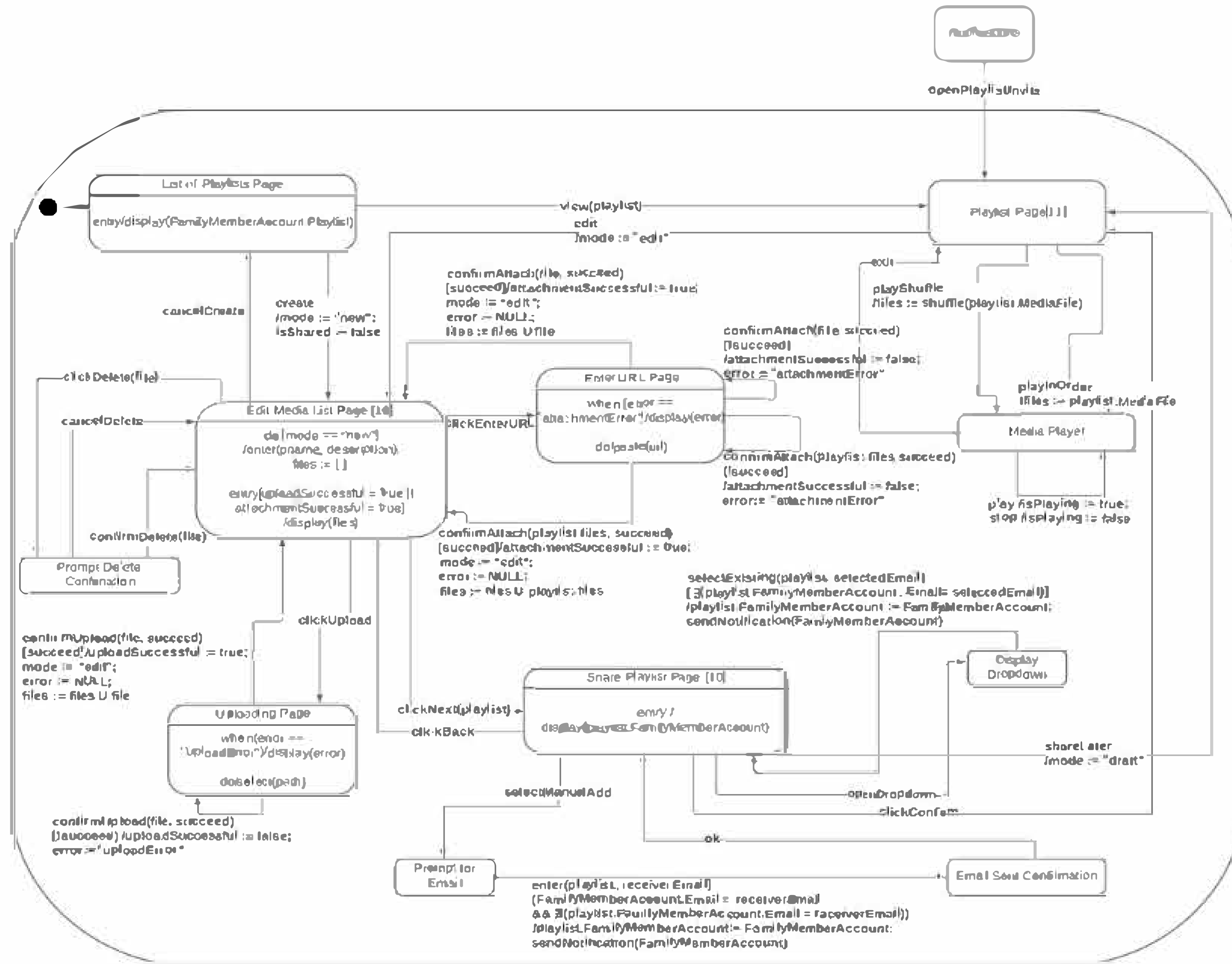

Figure 24: Navigation Diagram for Collaborating on Shared Media Playlists Use Case

This submachine has state machine variables: increment, selectedMediaFile, isPlaying. These should be declared and transitions should include any actions on those variables.

Ignore transition actions on Domain Model variables. Ignore state activities.

Figure 25: Navigation Diagram for the Media Player Concurrent Machine

In the Legend of States, include for each state the state **name** and **number** (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: **trigger event, condition, and action** (if any)
MISSING BELOW is the legend of **transition triggers** and to its corresponding **interface event**

Variables Legend

- NOW = current time as tracked by the system
- TODAY = current day as tracked by the system

General (State Legend)

Login Screen [1]
An interface of the application that provides the input fields for an email and a password for the family members who already have family member accounts on the system. For the family members who do not have an account, it also provides a text link which directs them to create a family member account.

Menu Page [2]
The first screen that the family member accounts land on when they sign in to the application. It displays a centralized management center where the family members can access the use cases such as planning family activities, collaborating on shared playlists, and managing family resources that the application provides.

Account Settings
An interface that displays the log out option for the family member accounts.

Notifications
A state that represents the situation where in-app notifications are sent to family member accounts. The notifications alert about resource booking conflicts, family activity polls, and invitations for collaborating on shared media playlists, among others.

Use Case 1: Plan a Family Activity (State Legend)

Family Activities Homepage
This screen lists all the existing family activities that the family member account is invited to. This is characterized by the display(FamilyMemberAccount.FamilyActivity) action. It also gives family members the option of starting a new family activity.

Family Activity Details
This screen lists the details of a particular family activity that the family member account is invited to. It can be accessed from the Family Activities Homepage.

Add Activity Type
This screen allows the family member to add a family activity type and associated tasks related to that type. This allows the family member to add the preset family activity types that will be used later on in the planning of the family activity.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

Start an Activity
This screen is reached when the family member decides to host a family activity. On this screen, the family member would specify the other family member accounts who are invited to the activity and select a method of deciding what the family activity will be - the latter can be either through a poll or manually entered by the host.

Creating a Poll [3]
After the host decides to use a poll to determine the family activity, a poll is created. The host would specify the deadline for voting and a list of poll options (each one includes the name of a potential family activity and a description of the activity), and then sends the poll to the invited family member accounts.

Scheduling the Activity [5]
After the family activity is determined, the family member who hosts the activity selects a date range, an intended duration and at least one time of day for the activity. The system constantly fetches the up-to-date common availabilities of the participating family members, based on the specified criteria, and through the integration with the calendar service (this is done through the display(familyActivity.Availability) activity).

Filling Out Activity Details [6]
After the date and time of the family activity is selected, the host fills out the remaining details (i.e. location, activity type, tasks to be assigned) of the upcoming family activity.

Voting on a Poll [4]
After the family member who hosts the family activity decides to open a poll for the potential family activities, the family members who are invited to participate get a notification for the poll in which they can choose their desired family activities among all the options provided on the poll. The poll is closed when the deadline to vote has passed.

Enter an Activity
In the event that no family members submit their votes for the family activity through the poll, the host of the family activity will receive a notification and the option to manually enter the family activity name and description. This screen allows the host to do that.

Use Case 2: Coordinate the Usage of Shared Resources (State Legend)

Add Resource Page
A form interface that allows a family member to enter information about a shared resource to the family's shared resource inventory.

Select Resource Page [7]
An interface that shows an inventory of shared resources of the family. Here, the family member can select a resource and date range for booking.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

Resource Booking Page [8]
An interface that requires the necessary information about the booking of a shared resource. The family member needs to choose a priority level, a time slot for the usage, as well as specifying the reason for using the resource. Once the family member chooses the usage time slot, the system checks if the availabilities of the resource contains the chosen usage time slot, and assigns corresponding value to isAvailable.

Resource Booking Override Request [9]
An interface that is a variation of Resource Booking Page [8] that only shows when a conflict for using any shared resource occurs. However, the usage time slot on resource override request is dynamic, and every time the family member modifies the usage time slot, the system checks if the availabilities of the resource contains the chosen usage time slot, and assigns the corresponding value to isAvailable.

Resource Booking Agreement
Family members who book a shared resource prior to other members who want to book the same resource on overlapping time slots receive the notifications for conflict. The notifications direct to an agreement screen that allows the family members to view the priority level and reason from other members, and to decide to approve or deny overriding their earlier booking of the resource.

Use Case 3: Collaborate on Shared Media Playlists (State Legend)

List of Playlists Page
An interface that displays a list of playlists that the family member account has access to.

Enter URL Page
A page where the family member needs to paste a valid URL to the media file.

Edit Media List Page [10]
The first part of the playlist creation screen which requires the family member to enter the name and the description of the newly created playlist. The family member can continuously add media files to the playlist until they are done creating the playlist.

Share Playlist Page [10]
The second part of the playlist creation screen which allows the family member to share the playlist with other family members. This step checks if the playlist is shared, and this status will be reflected on the page.

Playlist Page [11]
An interface of a single playlist that displays the name, description, list of media files, and if the list is shared.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

Uploading Page
A transition page that shows when family members upload a local media file to the application. It checks if any error occurs during uploading, and displays error messages whenever there are any problems with uploading.

Prompt Delete Confirmation
This is a confirmation prompt that appears when a family member wants to delete a media file.

Prompt for Email
Preview of the invitation email to invite family members to collaborate on the media playlists, and the confirmation button to send the email.

Email Sent Confirmation
A confirmation screen that communicates that invitations for collaborating on the playlist are sent.

Display Dropdown
A dropdown that consists of a list of family members who have an account registered with the system with whom the playlist can be shared.

Media Player
A service that plays media files, where the state can be paused or continued for playing.

Concurrent Machine (State Legend)

Idle
A state in which no interaction with the media player has been made yet.

Media Selection
A state that records the selected media file that is being played, and determines if the application should play the next file based on if there is an increment in the files or if the family member has reached the end of the playlist.

Media Player Plays
A state that shows if the media player is playing a media file.

General (Transition Legend)

login(email, password)
[email == account.Email && password == account.Password]
/FamilyMemberAccount := account
The family member logs into their family member account with an email and a password on the Login Screen [1]. This action checks if the input email and password pair matches what is stored in the family member account object and brings the family member to the Menu Page [2].

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

logout
/FamilyMemberAccount := NULL
The family member logs out of their family member account from the Account Settings page. This action sets the FamilyMemberAccount to NULL and brings the family member back to the Login Screen [1].

createAccount(account)
/FamilyMemberAccount := account
The family member creates a new account from the Login Screen [1]. This action sets the FamilyMemberAccount to the newly created account and brings the family member to the main Menu Page [2].

getSettings
From the Menu Page [2], the family member can access their account settings from the downwards arrow symbol beside the "Family Centre" headline. They are able to logout of their family member account from here as well.

getNotifications
From the bell icon on the Menu Page [2], the family member is able to view a list of all their notifications on the Notifications screen.

clickFamilyActivity
Family members are able to click the "Family Activity" option to access the family activity superstate through the hamburger menu from anywhere in the application or directly from the Menu Page [2]. The use case's start state is the Family Activities Homepage screen.

clickResourceBooking
Family members are able to access this use case superstate through the hamburger menu from anywhere in the application or directly from the Menu Page [2]. The use case's start state is the Select Resource Page [7] screen which provides a list view of the resource inventory a family owns as well as a Date Picker for the family member to select a date range.

clickMediaPlaylist
Family members are able to access this use case's superstate through the hamburger menu from anywhere in the application or directly from the Menu Page [2]. The use case's start state is the List of Playlists Page which provides a list view of all the playlists a family member owns or has been invited to collaborate on.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

Use Case 1: Plan a Family Activity (Transition Legend)

openFamilyActivity
Family members are able to view the details of a particular family activity by clicking on "Open" listed next to that family activity on the Family Activities Homepage.

closeFamilyActivity
To close the details of this family activity, family members are able to click "Close" on the page, and then they will navigate back to the Family Activities Homepage.

addActivityType
If a family member wants to add a preset family activity type, they can do so by clicking "Add Activity Type" on the Family Activities Homepage. They will be navigated to the Add Activity Type screen to add the name and associate tasks of an activity type.

exitActivityType
Once the family member has finished adding their activity type, then they can exit the page by clicking on "Exit" and returning back to the Family Activities Homepage.

clickStartActivity
If the family member would like to start a new activity, this transition will lead them to the Start an Activity screen where they can specify which family members to invite and select a method to determine what activity the family can participate in.

cancelActivity
At any point during the planning of the family activity, the family member can choose to exit the activity creation superstate, cancel the family activity and return back to the Family Activities Homepage. This cancelActivity transition allows them to do this.

clickNext(familyActivity)
[familyActivity.Name != NULL && familyActivity.Description != NULL && familyActivity.FamilyMemberAccount.length > 0]
From the Start an Activity screen, the family member can click "Next" to move on to the next step in this workflow, which is activity scheduling on the Scheduling the Activity [5] screen. During this transition, there is an additional check to ensure the familyActivity.Name, familyActivity.Description and list of invitees (familyActivity.FamilyMemberAccount) are not empty. Additionally, the instance of the activity is passed to the next screen to allow the family member to figure out the scheduling for this particular instance of the activity.

clickNext(familyActivity)
[familyActivity.Availability.Date >= TODAY && familyActivity.Availability.StartTime >= NOW && familyActivity.Availability.EndTime > familyActivity.Availability.StartTime]
From the Scheduling the Activity [5] screen, the family member can click "Next" to move on to the Filling Out Activity Details [6] screen to continue filling out the details of the activity. During

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

this transition, there is a check to make sure that the familyActivity.Availability.Date, familyActivity.Availability.StartTime and familyActivity.Availability.EndTime are all valid values. Additionally, the instance of the activity is passed to the next screen to allow the family member to fill in the rest of the details for this particular instance of the activity.

clickBack
From the Scheduling the Activity [5] screen, the family member may wish to return to the Start an Activity screen to modify previously entered details. This transition enables family members to return back to that screen.

From the Filling Out Activity Details [6] screen, the family member may wish to return to the Scheduling the Activity [5] screen to modify previously entered details. This transition enables family members to return back to that screen.

From the Creating a Poll [3] screen, the family member may wish to return to the Start an Activity screen to modify previously entered details. This transition enables family members to return back to that screen.

clickSendInvites(familyActivity)
[familyActivity.Location != NULL && familyActivity.ActivityType.Name != NULL && familyActivity.ActivityType.Task.length > 0]
/sendInvites(familyActivity)
Once all activity details have been completed on the Filling Out Activity Details [6] screen, invitations can be sent to all invitees by clicking “Send Invites” on that screen. There is an additional check to make sure that the familyActivity.Location, familyActivity.ActivityType.Name and familyActivity.ActivityType.Task fields are valid. In the sendInvites action, the instance of the activity is passed so the data within the activity can be sent to each of the family members in the familyActivity.FamilyMemberAccount list, as specified in the activity. This transition leads to the end state as the use case has now been completed.

createPoll
From the Start an Activity page, the family member may choose to initiate a poll to determine what type of family activity their family members may be interested in. The createPoll transition brings the family member to the Creating a Poll [3] screen where the poll details can be set up.

clickSendPoll(poll)
[poll.PollOption.length > 0 && poll.Deadline > NOW && familyActivity.FamilyMemberAccount.length > 0]
/sendNotification(poll)
Once the poll has been set up, a notification must be sent to the list of invitees asking them to submit their votes to the family activity poll. The poll is passed as a parameter in this transition. Furthermore, the poll.PollOption.length, poll.Deadline and familyActivity.FamilyMemberAccount fields are checked for validity before sending the notification through the sendNotification action.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

openPoll(poll)
[poll.Deadline > NOW]
If invited to partake in a family activity poll, the family member will be notified via the in-app notification service. All notifications can be viewed from the Notifications screen. Clicking on the poll notification will open the poll for family members to make their voting selections on the Voting on a Poll [4] screen. Family members can only open this poll if the deadline has not yet passed.

when(poll.Deadline > NOW)
[poll.PollOption.argmax(this.Count) > 0]
/favouritePollOption := poll.PollOption.argmax(this.Count);
familyActivity.Name := favouritePollOption.NameOfActivity;
familyActivity.Description := favouritePollOption.DescriptionOfActivity
Once the deadline has passed, and at least one vote has been submitted for a pollOption, then the system will calculate the pollOption with the highest Count value. That option will be the family activity that the family members will partake in (favouritePollOption). Additionally, during this transition, the familyActivity.Name is set to the favouritePollOption.NameOfActivity and the familyActivity.Description is set to the favouritePollOption.DescriptionOfActivity. After this, the family member moves on to the Scheduling the Activity [5] screen.

when(poll.Deadline > NOW)
[poll.PollOption.argmax(this.Count) == 0]
/sendNotification
Once the deadline has passed, and there are no votes submitted for a pollOption, then the system will send a notification to the host of the family activity. This notification will allow the host family member to manually input a name and description for the family activity.

openEnterActivity
The host family member can open the notification that tells them that no votes were submitted for the poll. They will be redirected to the Enter an Activity screen where they can manually enter values for familyActivity.Name and familyActivity.Description.

clickNext(familyActivity)
[familyActivity.Name != NULL && familyActivity.Description != NULL]
Once the host family member has entered the familyActivity.Name and familyActivity.Description, they can click "Next" to proceed to the scheduling step on the Scheduling the Activity [5] screen.

Empty transition from the superstate for planning the family activity back to the Family Activities Homepage
Once the family member has finished planning the family activity (they have reached the termination state of the superstate for planning the activity), then they will be automatically redirected back to the Family Activities Homepage.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

Use Case 2: Coordinate the Usage of Shared Resources (Transition Legend)

clickAddNewResource
From the Select a Resource Page [7], family members may wish to add a new family resource (i.e. a new vehicle, an additional TV, etc.). This will bring the family member to the Add Resource Page where they are able to input the new resource's details such as the resource name, description, and more.

clickAdd(resource)
Once the resource details have been filled out on the Add Resource Page, the "Add" button can be selected. This transition will lead the family member back to the Select a Resource Page [7] with the newly added resource included.

clickDeleteResource(resource)
From the Select a Resource Page [7], the family member may wish to delete an existing resource. This transition will lead the family member back to the Select a Resource Page [7], this time with the deleted resource removed.

clickBook(resource, datePicker)
[resource != NULL && datePicker.startDate != NULL && datePicker.startDate > TODAY && datePicker.endDate != NULL && datePicker.endDate > TODAY]
Once a resource has been selected from the Select a Resource Page [7], the family member will need to input a date range for which they want to borrow the resource for. When they click on the "Book" button, this transition will bring the family member to the Resource Booking Page [8] where booking information details can be filled out. The selected startDate and endDate must not be NULL and must be a future date.

selectUsageTimeSlot(resource, from, to)
[!(resource.Availability.StartTime < from && to < resource.Availability.EndTime)]
/booking.isAvailable := false
When a family member makes a change to the start time, from, and end time, to, that they want to book a resource for, on Resource Booking Page [8], and the selected time range is not contained in the Resource's Availability time range, the Booking's isAvailable attribute is set to false and the system brings the family member to the Resource Booking Override Request [9].

clickDone(booking)
[booking.priorityLevel != NULL && booking.reason != NULL]
On the Resource Booking Page [8], details such as the date and time, priority level and reason for booking are filled by the family member prior to finalising the booking. Once the "Done" button is clicked, the system updates the calendars by making a request to the calendar service, which means a family member has successfully booked the resource for their selected date and time range This leads to the end state in this superstate.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

when(booking.isAvailable == true)
On the Resource Booking Override Request [9], a family member can change the start time and end time that they want to book for, which updates the Booking's isAvailable attribute. Once the selected time range is available, the family member is shown the Resource Booking Page [8].

after(1)
[!(resource.Availability.StartTime < from && to < resource.Availability.EndTime)]
/booking.isAvailable := false
While the family member is on the Resource Booking Override Request [9], the system will constantly check if the selected time range is available, and will set Booking.isAvailable to false so that the screen will remain the same.

clickSend(booking)
[booking.priorityLevel != NULL && booking.reason != NULL]
Should a family member wish to override another family member's resource booking, they will be led to the Resource Booking Override Request [9]. Once the request is sent, the priority level and reason of the requestor is sent to the Resource Booking Agreement, which the original booker of the resource can view and subsequently approve or deny the request.

clickSend(booking)
[booking.priorityLevel != NULL && booking.reason != NULL]
/sendNotification
When a family member clicks 'Send' for an override request, it sends a notification to the existing booking owner through Notifications.

clickOpenResourceBookingOverrideRequest
When the family member opens their notifications and sees that they have a request from another family member to override their existing booking, they can click a button to view the override request's details in the Resource Booking Agreement.

clickApprove
When a family member who has been sent an override request clicks on "Approve" for the override request, this cancels their existing booking, and it brings them to the Resource Booking Page [8] so they can schedule another booking for the resource.

clickDeny
When a family member who has been sent an override request clicks on "Deny" for the override request, nothing happens to their existing booking and they are simply brought to the Menu Page [2].

givenDeniedBookingOverride
When a family member who sent an override request has been denied, they will see this in their notifications, and will be brought to the Resource Booking Page [8] to schedule another booking.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

givenApprovedBookingOverride
When a family member who sent an override request has been approved, they will see this in their notifications, and this brings them to successfully book a resource as they require.

Empty transition from the superstate for booking a resource back to Menu Page [2]
Once a family member is finished booking a resource, then they will be automatically redirected back to the Menu Page [2].

Use Case 3: Collaborate on Shared Media Playlists (Transition Legend)

view(playlist)
From the List of Playlists Page, the family member may view an individual playlist and be brought to the Playlist Page [11] where they can view the selected playlist's name, description and files.

edit
/mode := "edit"
From the Playlist Page [11], the family member may edit the playlist. This transition brings the family member to the Edit Media List [10] page and sets the mode to the string edit.

openDropdown
From the Share Playlist Page [10], the family member can view a dropdown of family members that the playlist can be shared with.

playShuffle
/files := shuffle(playlist.MediaFile)
On the Playlist Page [11], a family member may wish to shuffle the order of the media files in their playlist when they play it. This transition brings the family member to the Media Player screen and sets all media files to shuffle.

playInOrder
/files := playlist.MediaFile
Instead of shuffle play, a family member may wish to play their media files in order. From the Playlist Page [11], a family member can play their media in order. This transition brings family members to the Media Player and sets files to play in order.

exit
While using the Media Player, the family member may wish to leave the Media Player and return to the Playlist Page again. This is completed by the exit transition which brings the family member back to the Playlist Page [11].

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

create
/mode := "new";
isShared := false
From the List of Playlists Page, the family member may wish to create a new playlist. This transition will bring the family member to the Edit Media List Page [10] where the details of their new playlist can be inputted. This transition also sets the mode to the string new.

cancelCreate
If the family member no longer wants to create a new playlist, they may click cancel at any time and be brought back to the List of Playlists Page.

clickEnterURL
From the Edit Media List Page [10], the family member may upload a media file via a URL link. Selecting this option brings the family member to the Enter URL Page.

confirmAttach(file, succeed)
[!succeed]
/attachmentSuccessful := false;
error := "attachmentError"
The URL attachment of a media file is unsuccessful, the family member is brought back to the Enter URL Page and is presented with an error message.

confirmAttach(file, succeed)
[succeed]
/attachmentSuccessful := true;
mode := "edit";
error := NULL;
files := files U file
Once the appropriate URL for the media file has been successfully attached, the attachmentSuccessful boolean is set to true and the file is added to the playlist.

confirmAttach(playlist.files, succeed)
[!succeed]
/attachmentSuccessful := false;
error:= "attachmentError"
The URL attachment of all the media files in an existing playlist from the adjacent media platform is unsuccessful, the family member is brought back to the Enter URL Page and is presented with an error message.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

confirmAttach(playlist.files, succeed)
[succeed]
/attachmentSuccessful := true;
mode := "edit";
error := NULL;
files := files U playlist.files
Once the appropriate URL for all the media files in an existing playlist from the adjacent media platform have been successfully attached, the attachmentSuccessful boolean is set to true and the file is added to the playlist.

clickUpload
From the Edit Media List Page [10], the family member may choose to upload a media file from their own device. Selecting this option brings the family member to the Uploading Page.

confirmUpload(file, succeed)
[!succeed]
/uploadSuccessful := false;
error := "uploadError"
The file upload is unsuccessful, the family member is brought back to the Uploading Page and is presented with an error message.

confirmUpload(file, succeed)
[succeed]
/uploadSuccessful := true;
mode := "edit";
error := NULL;
files := files U file
Once the appropriate media file has been successfully uploaded, the uploadSuccessful boolean is set to true and the file is added to the playlist.

clickDelete(file)
From the Edit Media List Page [10], the family member can delete a media file by clicking on the delete control. This leads the family member to the Prompt Delete Confirmation pop-up where the family member can confirm or cancel the media file deletion.

cancelDelete
If the family member no longer wants to delete the media file, they may click "Cancel" at any time and be brought back to the Edit Media List Page [10].

confirmDelete(file)
Confirming the deletion will delete the media file and bring the family member back to the Edit Media List Page [10].

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

clickNext(playlist)
From the Edit Media List Page [10], if the family member selects the "Next" button, they will be brought to the Share Playlist Page [10] where they can share the playlist with other family members.

clickBack
From the Share Playlist Page [10], if the family member selects the "Back" control, they will be brought back to the Edit Media List Page [10] where they can make further modifications to their playlist before sharing. If any changes were made to the Share Playlist Page [10] before clickBack, the changes are saved.

selectManualAdd
On the Share Playlist Page [10] the family member may select a family member who does not yet have an account registered in the system. If this is the case, the family member will be prompted for the family member's email via the Prompt for Email pop-up.

enter(playlist, recieverEmail)
[FamilyMemberAccount.Email = receiverEmail &&
$\nexists$ (playlist.FamilyMemberAccount.Email = receiverEmail)]
/playlist.FamilyMemberAccount := FamilyMemberAccount;
sendNotification(FamilyMemberAccount)
Once prompted for the new family member's email, the family member must enter the receiver's email. This will display the Email Sent Confirmation pop-up.

ok
From the Email Sent Confirmation pop-up, the family member selects the "Ok" control and is brought back to the Share Playlist Page [10] to send more invites.

selectExisting(playlist, selectedEmail)
[$\exists$ (playlist.FamilyAccount.Email = selectedEmail)]
/playlist.FamilyMemberAccount := FamilyMemberAccount;
sendNotification(FamilyMemberAccount)
On the Share Playlist Page [10] the family member may select an existing family member to share the playlist with. If this is the case, the family member will select the existing family member account(s) from the list on this page.

openPlaylistInvite
Once a playlist has been shared with another family member, they will receive a notification via our in-app notification service. Clicking on this playlist invite notification will bring the family member to the Playlist Page [11].

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

shareLater
/mode := “draft”
At any point on the Share Playlist Page [10], the family member may choose to share the playlist later. This transition will save the playlist as a draft and bring the family member back to the Playlist Page [11].

clickConfirm
From the Share Playlist Page, once the “Confirm” control is selected, the playlist is officially created and the family member is brought back to the Playlist Page [11].

play
/isPlaying := true
A media file is being played by the media player.

stop
/isPlaying := false
There are no media files being played by the media player.

Concurrent Machine (Transition Legend)

files := [ ],
isPlaying:= false,
selectedMediaFile := 0,
increment := false
Once the concurrent machine state is entered, there are no media files in the playlist, hence, the files[ ] array is initialized to empty. Moreover, there is no file playing currently so this variable is set to false and there is no media file selected, hence, the selectedMediaFile variable is initialized to 0. Increment is initialized to false.

when(files.empty())
If there are no media files in the playlist, that is, files[ ] is empty, then the family member must make a media file selection. Therefore, this transition moves to the Media Selection state. Moreover, if there are no files left in the selected playlist, the system transitions back to the Idle state.

when(selectedMediaFile > files.length)
/selectedMediaFile := 0
The selectedMediaFile variable indicates the index in the files[ ]. Therefore, when the selectedMediaFile variable is greater than the files[ ] array, this is invalid behavior and the variable is reset to zero. The family member must then select another media file to play, hence the transition returns to the Media Selection state.

In the Legend of States, include for each state the state name and number (if it corresponds to a provided UI screen)
In the Legend of Transitions, include for each transition the full expression: trigger event, condition, and action (if any)
MISSING BELOW is the legend of transition triggers and to its corresponding interface event

after(files[selectedMediaFile].Duration)
/increment := true
Once a file that was playing finishes playing, the media player needs to move on to the next file as indicated by setting the increment variable to true.

when(isPlaying)
Once a media file starts to be played by the media player (when the isPlaying variable is true), the media player starts to play. As long as isPlaying is still true, the system will continue to switch between the Media Selection and Media Player Plays states to continuously queue up the next file to be played, and then play that file.

when(files)
/selectedMediaFile := 0
If there is at least one file in the playlist, that is, files are non-empty, then the selectedMediaFile variable is set to zero to point to the first file of the files[ ] array.

after(1)
[isPlaying == true]
The concurrent machine periodically assesses if a media file has been selected to play. If this is the case, then the isPlaying variable must be set to true. This transition then leads to the Media Player Plays state.

after(0)
[increment]
/selectedMediaFile := selectedMediaFile + 1,
increment := false
When the increment variable is true, the selectMediaFile variable increases by one to indicate that the media player is moving onto the next file in the list of files. This also sets the increment variable to false as the next file being played is not finished playing.