SEG2105 FINAL PROJECT

Submitted
to:
Miguel Garzon
University of Ottawa
75 Laurier Avenue E
Ottawa, ON

by:
Ryan Ethier: 8606056
Joshua Vinge: 8655939
Walter Wong: 8752209
Jonathan Popowick-Bastien: 8432984

Requirements:

Functional:

- The application must allow a user to view a list of all current tasks
- The application must allow a user to view all their tasks in one location
- The application must allow a user to allocate an unassigned task to themselves
- The application must allow a user to create a communal shopping list that can then be viewed by others
- The application must allow a user to view all other users in their “Home”
- The application must allow a user to release a user from a task and reassign it to another user
- The application must allow a user to switch between user accounts
- The application must allow an adult to create and delete child accounts
- The application must allow a user to mark a task as completed
- The application must allow an adult to edit the name of a task
- The application must allow an adult to delete a task whether it has been completed or not
- The application must allow a user to add an optional time for task completion
- The application must allow a user to view a history of their completed tasks
- The application must allow an adult to create a task
- The application must allow an adult to assign a task to another user
- The application must allow an adult to create a “Home” to associate other users with
- The application must allow a user to add, edit or remove items from a shopping list
- The application must allow a user to mark an item in a shopping list as purchased
- The application must allow a user to leave messages on a communal “Message Board”
- The application must allow an adult to edit or delete existing messages on the “Message Board”

Non-Functional

- The application must be ready for deployment by December 6th
- The application must run on the earliest android SDK version allowed by the API methods used
- The application should follow Google’s standards for Material Design
- The application shall run on Android 4.0 and up
- The application must display the list of assigned tasks straight away from the time the app is opened so that the tasks can be completed with one tap

Use Cases:

Use Case 1: Allocate Task

Actors:

- User

Goals:

- Users use this case to allocate tasks.

Preconditions:

- An unallocated task must exist.
- A user must either be a parent or logged into the account they wish to assign the task to

Summary:

- This use case is for users to allocate unallocated tasks. Any task that is either new or has been released by another user is considered unallocated.

Steps:

Actor Action	System Responses
Choose a task	(If user is a parent) Prompt user for account to allocate task to (otherwise skip to next system response)
Select a user to allocate to	Prompt user for confirmation
Confirm allocation	confirm to user action has been completed

Post-Conditions:

- The unallocated task is now allocated.
- The formerly unallocated task is now allocated to the user.

Use Case 2: Create New Account

Actors:
- Adult

Goals:
- Create a new Adult or Child account

Preconditions:
- If an Adult account exists in the app, the user must be logged into it.
- If no accounts exist, the user will be prompted to create a new Adult account.

Summary:
- This use case allows an Adult to create an account for either another or a child. The adult has elevated privileges that allow them to do so. This ensures control of the app is left in the hands of the adult.

Related Use Cases:
- Set up app (If no accounts exist)

Steps:

Actor Action	System Responses
Opens app for first time	If no accounts exist, prompt user to set up app
Choose "Create Account" from settings menu	Prompts user for new account information
User enters Name and chooses privilege level (Adult/Child)	
User confirms information	Confirm to user account has been created

Post-Conditions:
- The account will now be available to choose when assigning tasks
- The account will now appear in the list of users.

Use Case 3: Create A Communal Shopping List

Actors:
- Adult

Goals:
- Create a new communal shopping list and associate it with a Home.

Preconditions:
- The user must be logged in as an adult
- A home must have been created prior.

Summary:
- Create a shopping list for a home.

Related Use Cases:
- Add an item to the Shopping List
- Edit an item on the shopping list
- Remove an item on the shopping list

Steps:

Actor Action	System Responses
Select "Create a shopping list"	Open a dialog prompting for shopping list name, Home it is associated with, and (optionally) list of items
User enters shopping list name, home it is associated with, and (optionally) a list of items	
User confirms information	Save the shopping list
	Confirm to user that the shopping list is saved.

Post-Conditions:
- The shopping list will now be available on a list of shopping lists for a given Home.
- The shopping list can be modified by of the members of the Home.

UML Class Diagram:

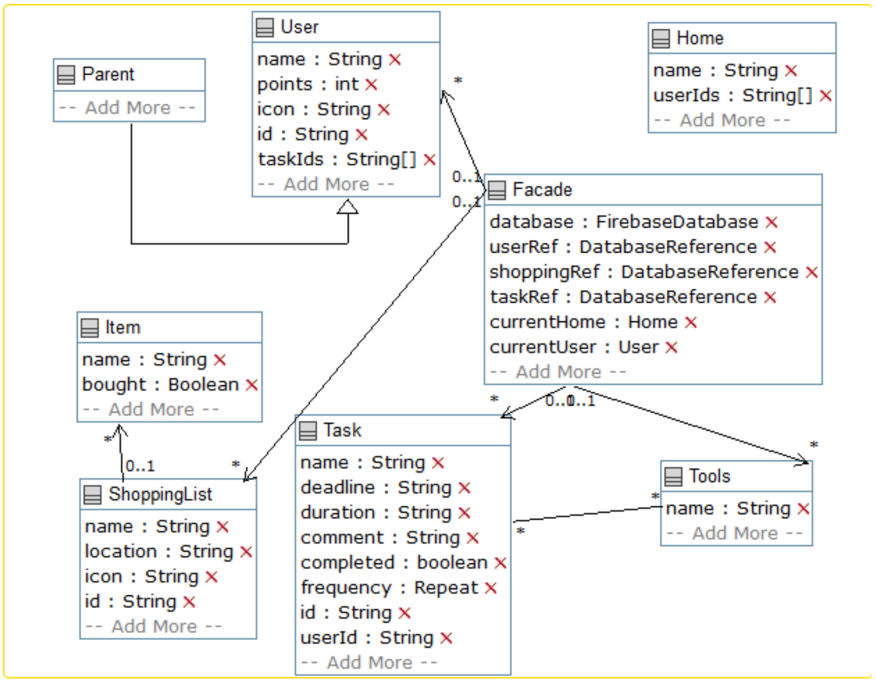


Figure 1: A UML Class diagram of the Chore Chart application.

UML Sequence Diagram 1:

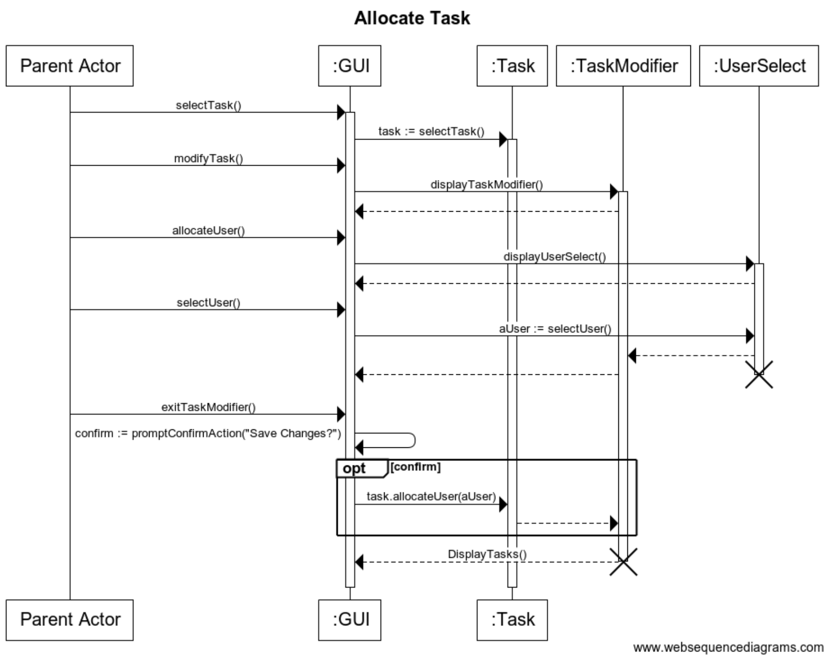


Figure 2: A UML Sequence Diagram for the "Allocate Task" use case. (Use Case 1)

UML Sequence Diagram 2:

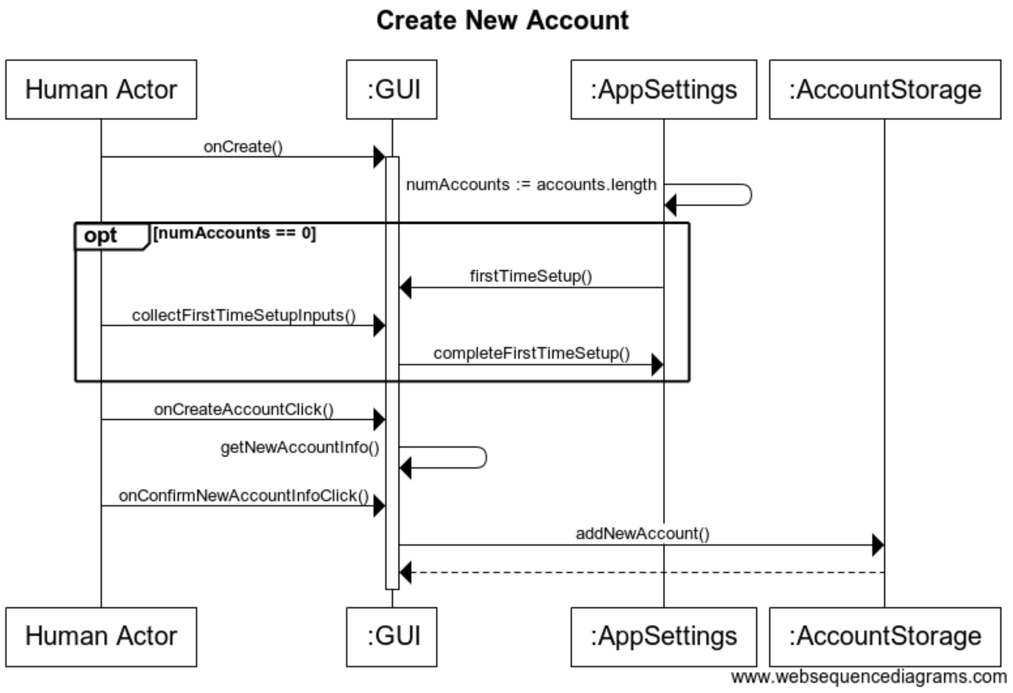


Figure 3: A UML Sequence Diagram for the "Create New Account" use case. (Use Case 2)

UML Sequence Diagram 3:

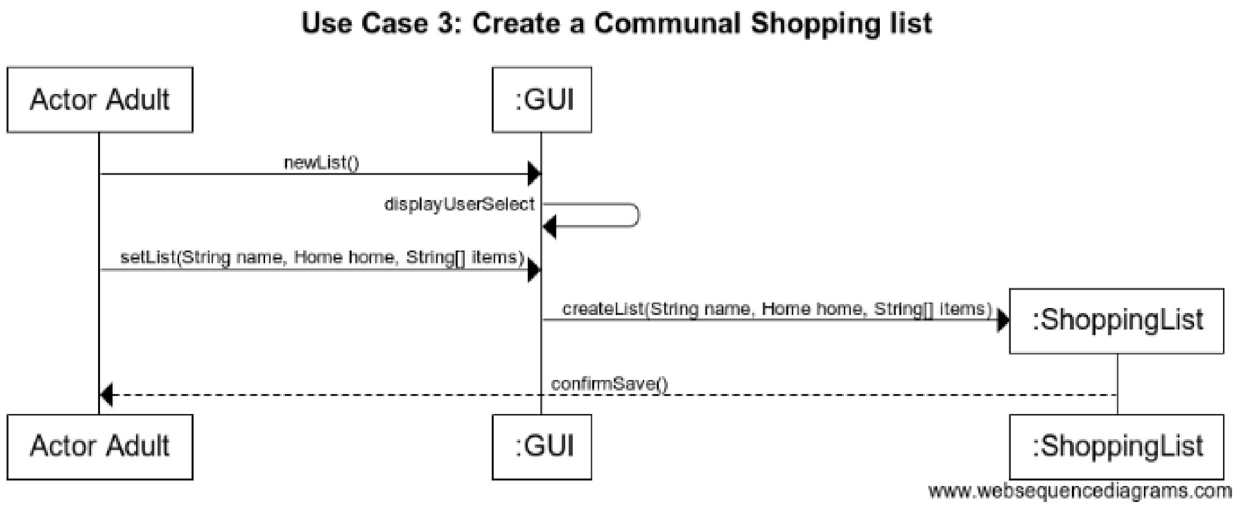

Use Case 3: Create a Communal Shopping list
Actor Adult
:GUI
newList()
displayUserSelect
setList(String name, Home home, String[] items)
createList(String name, Home home, String[] items)
:ShoppingList
confirmSave()
Actor Adult
:GUI
:ShoppingList
www.websequencediagrams.com

Figure 4: A UML Sequence Diagram for the "Create a Communal Shopping List" use case. (Use Case 3)

UML State Machine Diagram:

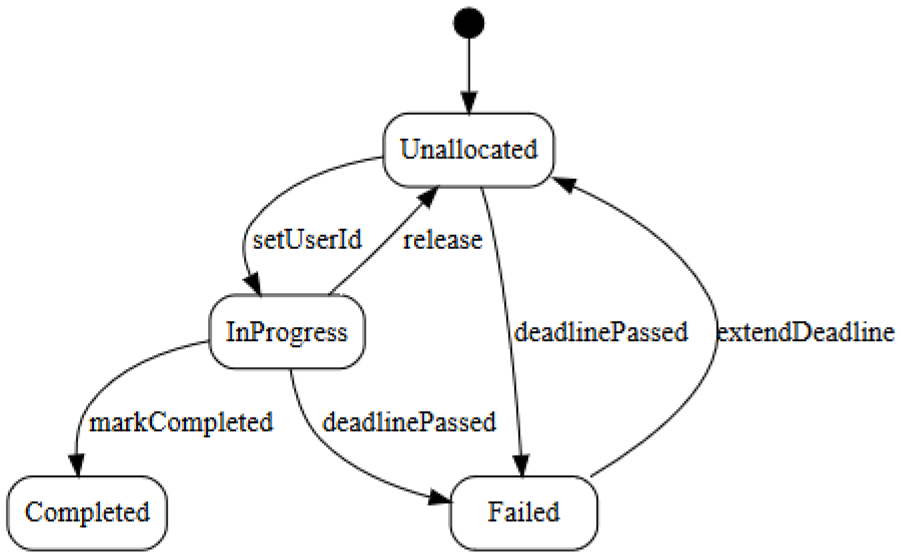
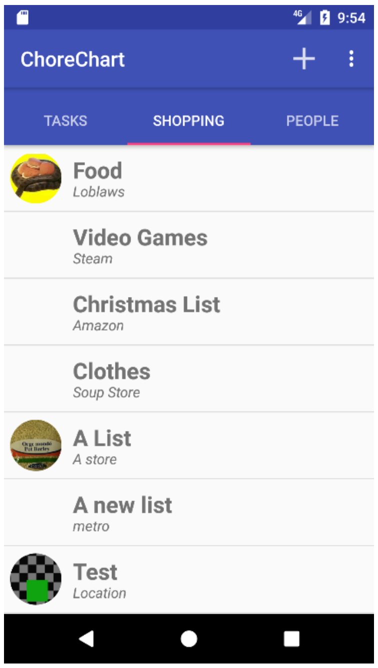


Figure 5: A State Machine Diagram for Tasks

UI Screenshots:

The Shopping List tab allows the user to view and add shopping lists. Once a shopping list is clicked on, they are taken to another activity, where they are able to view and add items to the shopping list. They are also able to mark each item as purchased or not.


9:54

ChoreChart

TASKS SHOPPING PEOPLE

Food
Loblaws

Video Games
Steam

Christmas List
Amazon

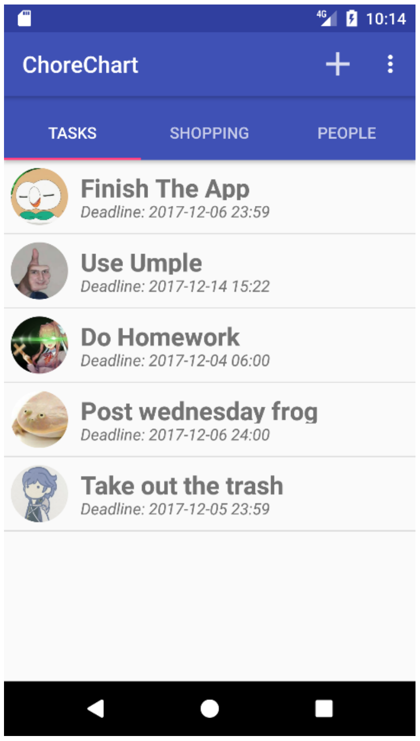
Clothes
Soup Store

A List
A store

A new list
metro

Test
Location

Figure 6: A Screenshot of the Shopping List tab


10:14

ChoreChart

TASKS SHOPPING PEOPLE

Finish The App
Deadline: 2017-12-06 23:59

Use Umple
Deadline: 2017-12-14 15:22

Do Homework
Deadline: 2017-12-04 06:00

Post wednesday frog
Deadline: 2017-12-06 24:00

Take out the trash
Deadline: 2017-12-05 23:59

The Tasks tab allows a user to view or add tasks. The icon on the left side of the task name refers to the user to whom the task is assigned. Users can tap on a task to edit its name, whether it repeats, user it's assigned to, items needed for the job, as well as if the task is complete.

Figure 7: Screenshot of the Tasks Tab

The users tab allows a user to see what users are associated to the "Home". From this screen, you can add or edit a user. Once a user is tapped, you can change the user's name and icon, as well as delete the user.

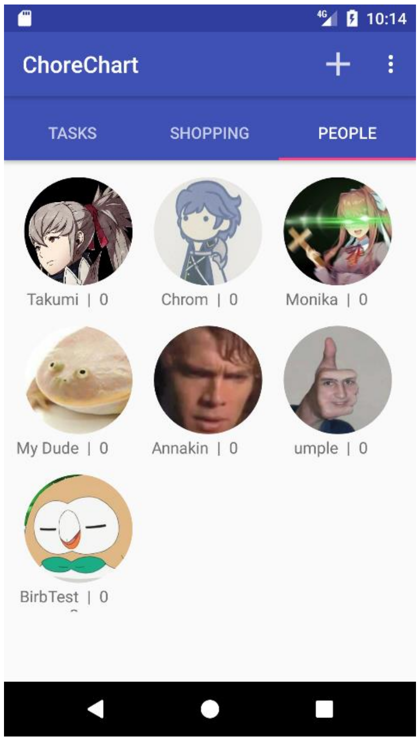
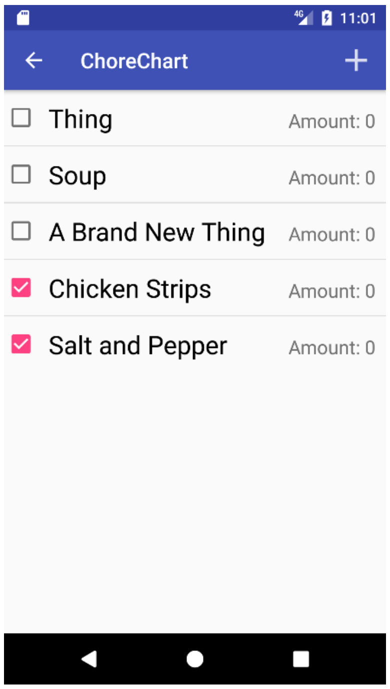


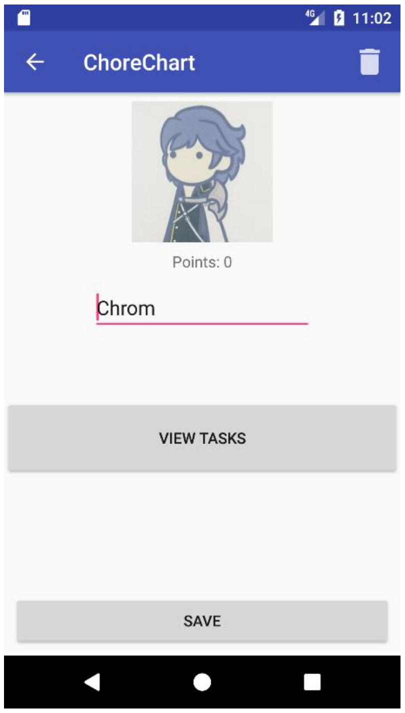
Figure 8: Screenshot of the Users Tab



- [ ] Thing Amount: 0
- [ ] Soup Amount: 0
- [ ] A Brand New Thing Amount: 0
- [x] Chicken Strips Amount: 0
- [x] Salt and Pepper Amount: 0

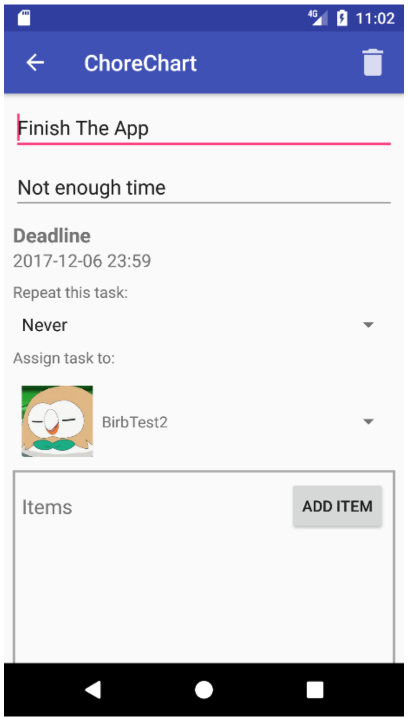
The Item tab allows the user to view items on a shopping list, mark them as bought, as well as add items.

Figure 9: Screenshot of the Items Tab



The User tab allows the user to modify a user's icon and name.

Figure 10: Screenshot of the User Tab



The Edit Task tab allows a user to edit a task's name, deadline, repetition, user, as well as add items to a list. You are also able to mark a task as complete.

Figure 12: Screenshot of the Edit Task Tab

Work Division:

Person	**Work Percentage**	**Work Done**	**Challenges Faced**
Ryan	19%	- Shopping List Implementation - Report	I am unexperienced with Java, so this project was a challenge for me. As well, having to work with completely new forms of android development such as Fragments and Adapters frightened me a bit.
Walter	27%	- General UI - Task Implementation	One challenge I have overcome throughout this project was the Edit Task tab's UI. Having to program separate fragments for various functions on this screen was quite complicated, however I was able to overcome this burden.
Josh	27%	- Database - User Implementation - Shopping List Implementation - Live Database Updates	I had a difficult time getting the app to automatically update with the database, however, this was solved after several hours of work. As well, I had a difficult time figuring out why we were having recursive storage problems. This

			was fixed by re-writing the way objects were associated to each other.
Jonathan	27%	- Umple implementation - Main Activity Framework	I’ve mostly had trouble with the Umple integration, as one problem I’ve encountered was that the Umple file developed would sometimes have compilation problems. With that said, these problems were fixed, but not without some remaining problems. These were taken care with manual file manipulation, and a note was left on the file itself to notify the other team mates.

Conclusion:

Over the last few weeks, we as a group have learned how to work cooperatively, as well as efficiently through various methods. We were able to produce a functioning app, but not without problems along the way that were eventually solved. As well, through GitHub, we were able to work cooperatively without any problems, as GitHub took care of any conflicts that would arise from our code that we submit. Overall, we found that this project was an overwhelming success.