

Final Documentation

for

Task Roulette

Group 1

May 8, 2014

Prepared by:

**Natalia Alonso
Jared Fowler**

**Ilia Benson
Daniel Smith**

Cory Carvalho

Table of Contents

- [1. Introduction](#)
 - [1.1 Purpose](#)
 - [1.2 Scope](#)
- [2. User Cases](#)
 - [2.1 User Case Diagram](#)
 - [2.2 User Case Descriptions](#)
 - [2.2.1 Create Profile](#)
 - [2.2.2 View Profile](#)
 - [2.2.3 View Rewards](#)
 - [2.2.4 View Achievements](#)
 - [2.2.5 Edit Profile](#)
 - [2.2.6 Delete Task](#)
 - [2.2.7 Task](#)
 - [2.2.8 Achievements](#)
 - [2.2.9 Session](#)
 - [2.2.10 Finish Task](#)
 - [2.2.11 Purchase Rewards](#)
 - [2.2.12 Edit Task](#)
 - [2.2.13 Spin Wheel](#)
 - [2.2.14 Add Task](#)
 - [2.2.15 Rewards](#)
- [3. System Requirements Specifications](#)
 - [3.1 Preface](#)
 - [3.1.1 Revisions](#)
 - [3.2 Introduction](#)
 - [3.2.1 Overview of the Document](#)
 - [3.3 Glossary](#)
 - [3.4 User Requirements Definitions](#)
 - [3.4.1 Mandatory Requirements](#)
 - [3.4.2 Desirable Requirements](#)
 - [3.4.3 Optional Requirements](#)
 - [3.5 System Architecture](#)
 - [3.5.1 Model View Controller](#)
 - [3.6 System Requirements Specification](#)
 - [3.6.1 External Interfaces](#)
 - [3.6.2 Functional Requirements](#)
 - [3.6.3 Non-Functional Requirements](#)
 - [3.7 System Models](#)
 - [3.7.1 Activity Model](#)
 - [3.7.2 Context Model](#)
 - [3.7.3 Use Case Model](#)
 - [3.7.4 CRC Cards](#)
 - [3.7.5 Analysis Level Class Diagram](#)
 - [3.8 System Evolution](#)
 - [3.8.1 Fundamental Assumptions](#)
 - [3.8.2 Effect of Product](#)

[3.8.3 Anticipated Changes](#)

[3.8.4 Future Additions](#)

[3.8.5 Future of the Product](#)

[3.9 Appendices](#)

[Appendix A: Hardware](#)

[Appendix B: Software](#)

[Appendix C: Database](#)

[Appendix D: Third Party Software](#)

[4.1 Test Plan Identifier](#)

[4.2 References](#)

[4.3 Introduction](#)

[4.4 Test Items](#)

[4.5 Software Risk Issues](#)

[4.6 Features to be Tested](#)

[4.7 Features not to be Tested](#)

[4.8 Approach](#)

[4.9 Item Pass/Fail Criteria](#)

[4.10 Suspension Criteria and Resumption Requirements](#)

[4.11 Test Deliverables](#)

[4.12 Remaining Test Tasks](#)

[4.13 Environmental Needs](#)

[4.14 Staffing and Training Needs](#)

[4.15 Responsibilities](#)

[4.16 Schedule](#)

[4.17 Planning Risks and Contingencies](#)

[4.18 Approvals](#)

[4.19 Glossary](#)

[5. Activity Diagrams](#)

[6. Sequence Diagrams](#)

[7. UML Class Diagrams](#)

1. Introduction

1.1 Purpose

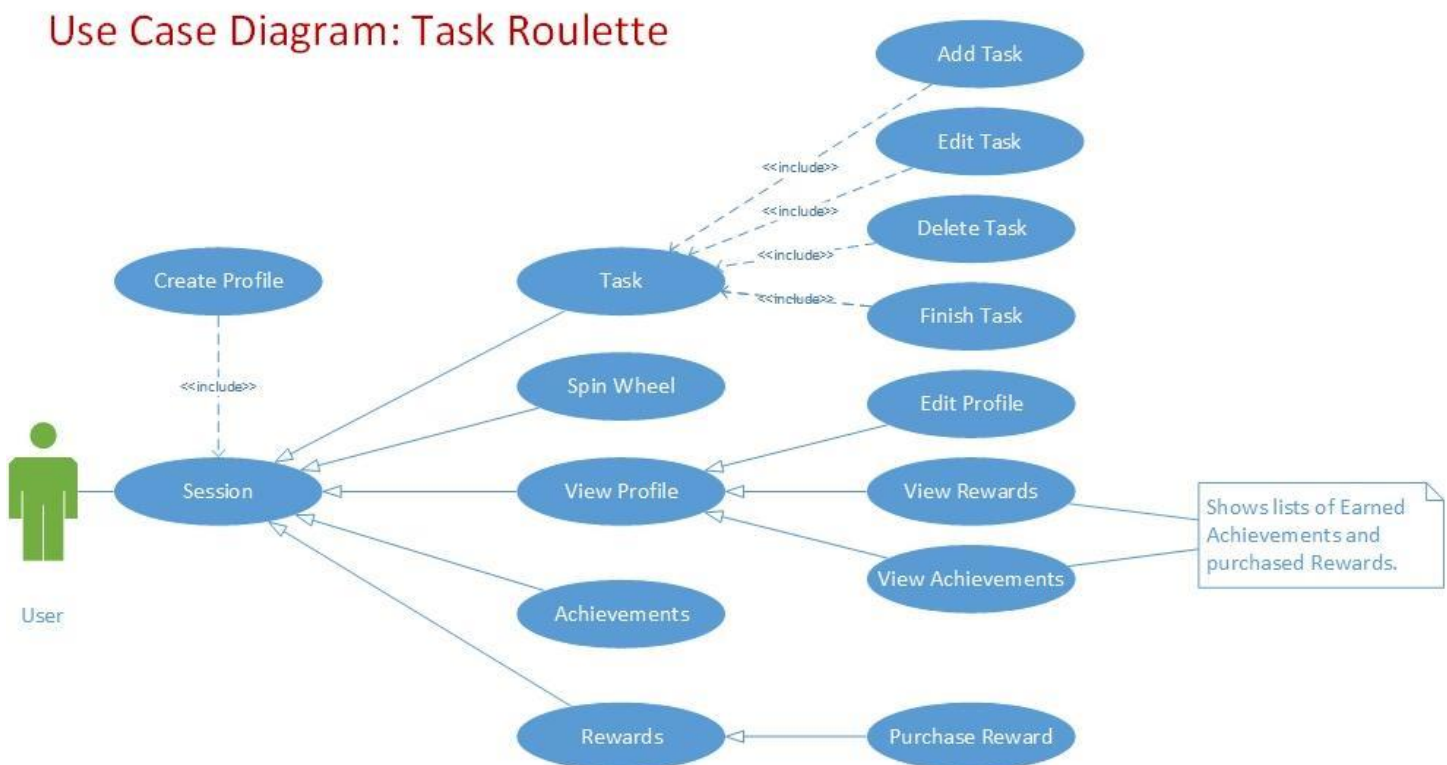
The purpose of this document is present the documentation for the Task Roulette application for Android.

1.2 Scope

Task Roulette is an application that motivates its users to accomplish their goals via an exciting random selection goal system topped with the possibility of achievements and rewards. The application is intended for those who want to accomplish goals but lack motivation. Goals entered by the user shall be selected at random and given back to the user as a challenge. Upon completing this goal, the user can check it off and receive “chips” which can then be used for rewards. Different achievements can be unlocked by completing a certain number of goals. Application shall be mainly stand alone, though, it shall have the capability to connect to the internet. The application is not dependent upon this connection and should function correctly without it.

2. User Cases

2.1 User Case Diagram



2.2 User Case Descriptions

2.2.1 Create Profile

Description: Upon first start up, user creates a profile

Scenario: When the app is first started, the user is asked to create a profile in order to use the app.

Actors: User

Related use case: Session, View Profile

Stakeholders: The user needs a profile in order to use the app.

Pre-condition: The app must be installed and initialize. The create profile use case will appear only until completed once.

Post-condition: A user profile will added to the app with user information. The profile is viewable/editable from the menu.

Flow of Events: Actor

1. First start up (or each start up until profile is created)
3. App main menu

System

2. Prompt user to input profile information
4. Initialize app with user profile information and store it until app is uninstalled

Exception:	Step	Condition	Action Description
	2a	if step 2 fails	Close app
	4a	if step 4 fails	Return to create profile

Edit Profile

2.2.2 View Profile

Description: From a menu, user can view and edit profile information

Scenario: User selects View Profile from menu in order to view or edit profile information

Actors: User

Related use case: Create Profile, Achievements, Rewards

Stakeholders: The user can update their profile information as needed

Pre-condition: User must have existing profile

Post-condition: View: No change, Edit: App will have updated user profile information

Flow of Events: Actor

1. Select View Profile from menu
3. Edit and save profile

System

2. Display profile
4. Update profile information and displayed updated profile

Exception:	Step	Condition	Action Description
	2a	if step 2 fails	Return to menu
	4a	if step 4 fails	Display unchanged profile

2.2.3 View Rewards

Description: From the profile, user can view purchased rewards

Scenario: User selects View Rewards from menu in order to view or edit profile information

Actors: User

Related use case: Rewards, View Profile

Stakeholders: The user can view their purchased rewards

Pre-condition: User must be in profile page

Post-condition: No change

Flow of Events:

Actor
1. Select View Rewards from menu

System
2. Display displays purchased rewards

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Return to menu

2.2.4 View Achievements

Description: From the profile, user can view earned achievements

Scenario: User selects View Achievements from the profile

Actors: User

Related use case: Rewards, View Profile

Stakeholders: The user can view their earned achievements

Pre-condition: User must be in profile page

Post-condition: No change

Flow of Events:

Actor
1. Select View Achievements from menu

System
2. Display displays earned achievements

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Return to menu

2.2.5 Edit Profile

Description: From the profile, the user can choose to edit their profile

Scenario: User selects Edit Profile from the profile page

Actors: User

Related use case: View Profile

Stakeholders: The user can update their profile information as needed

Pre-condition: User must have existing profile

Post-condition: App will have updated user profile information

Flow of Events:

Actor
1. Select Edit Profile from profile page
3. Edit and save profile

System
2. Display edit profile fields
4. Update profile information and displayed updated profile

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Return to menu
	4a	if step 4 fails	Display unchanged profile

2.2.6 Delete Task

Description: If a task is no longer wanted, user can choose to delete from to-do list

Scenario: User selects Delete Task from Task Menu in order to delete an unwanted task

Actors: User

Related use case: Task, Add Task, Edit Task

Stakeholders: The user can delete a task from the task list so it will no longer be a contender for roulette

Pre-condition: The user must have at least one task in the task list

Post-condition: The task list will be updated for the roulette

<i>Flow of Events:</i>	Actor
	1. Select Delete Task from the task list
	3. User selects task to deletes and initiates deletion
	System
	2. Display list of tasks with delete option
	4. Update task list and display updated list

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Return to menu
	4a	if step 4 fails	Display unchanged list

2.2.7 Task

Description: A parental case to all other cases that access, update, or change things pertaining to the task list.

Scenario: Actor selects button Tasks

Actors: User

Related use case: Parent: Session, Children: Add Task, Edit Task, Delete Task, Finish Task.

Stakeholders: User

Pre-condition: Session Activity needs to be started and current

Post-condition: Possible change in the task. Returns to activity Session.

<i>Flow of Events:</i>	Actor
	1. User presses the "Task" button
	3. User selects one of six options (Add Task, Edit Task, Delete Task, Finish Task, Go Back)
	System
	2. System loads list activity
	4. Loads appropriate Activity or Ends current Activity

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Return to Session
	4a	if step 4 fails	Return to Menu

2.2.8 Achievements

Description: User is able to view achievements. Achievements can be earned with completed tasks.

Scenario: From the “View Profile” use case, the user can access this use case. User can then view previously earned achievements.

Actors: Task the primary actors – User.

Related use case: Parents: View Profile

Stakeholders: User

Pre-condition: Activity “View Profile” must be current.

Post-condition: Possible update of user achievements.

Flow of Events: Actor

1. User presses the “Achievements” button.
3. User views achievements.
5. User pushes the “Go Back” button

System

2. System initiates the Achievements Activity.
4. System displays achievements.
6. System returns to either “View Profile”

Exception: Step Condition Action Description

- | | | |
|----|-----------------|---|
| 2a | if step 2 fails | Return to “View Profile” |
| 4a | if step 4 fails | No change |
| 6a | if step 6 fails | Re-attempt to return to either “View Profile” |

2.2.9 Session

Description: This is the initial start of the app, assuming profile exists, and shows main menu with all its options

Scenario: If no profile exists, then create a profile. Else, list 5 following menu options: 1) Spin Wheel 2) Profile 3) Rewards 4) Achievements

Actors: User

Related use case: Create Profile, View Profile

Stakeholders: User uses this case to navigate through the apps functions, system knows what to display next based on user input

Pre-condition: App needs to be installed, user must have a profile

Post-condition: Navigates to subroutine of program based on user’s selection in menu

Flow of Events: Actor

1. Startup of app, after initial profile creation
3. User selects item from menu

System

2. System displays menu selection
4. System runs subroutine of item selection in the menu

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Close app

2.2.10 Finish Task

Description: This is an option from the Task of tasks menu. Allows user to check off a previously created task, marking it as completed.

Scenario: Asks user to select a task and then provides option to mark that task as completed

Actors: User

Related use case: Task, Edit Task, Delete Task, Display Task, Add Task

Stakeholders: User uses this case to mark off completed tasks and claim their reward in the form of chips, which are used to spin the roulette wheel.

Pre-condition: user must have a profile, app needs to be installed, and task needs to be created.

Post-condition: Receive chips for completing task

Flow of Events: Actor

1. Select task from list
3. User selects phonebook or a social network

System

2. Displays option for selected task

Exception	Step	Condition	Action Description
	2a	if step 2 fails	Displays error message that says no current tasks exist
	4a	if step 4 fails	Returns back to Task menu

2.2.11 Purchase Rewards

Description: Provides a way for user to spend “chips” on rewards

Scenario: Tasks available rewards, each with a cost associated with buying them. upon purchasing the reward, the option to buy will disappear, the reward from the list will have a visual tell to show that it is now “earned”.

The appropriate cost will be subtracted from the user’s total chips. if the user does not have enough chips, the “buy” option will not be available to initiate. rewards will have different visual tells to show a difference. upon unlocking either, they will report to the profile for an update.

Actors: User

Related use case: Rewards, View Profile

Stakeholders: User uses this case to spend game currency on titles and other rewards in order to drive them to keep using the application and show progress.

Pre-condition: Session has started, profile has been created.

Post-condition: Profile will be updated, currency will be (potentially) decreased

Flow of Events: Actor

1. User chooses reward
2. User chooses “buy” option
7. Reward is purchased

System

3. Appropriate number of chips are subtracted
4. Reward is visually changed
5. Reward element loses “buy” listener
6. User profile is updated

<i>Exception:</i>	Step	Condition	Action Description
	1a	if step 1 fails	if user does not have enough chips, buy listener is killed.

2.2.12 Edit Task

Description: Allows user to change the specifics of a task. (name / category / difficulty / time frame)

Scenario: actor brings up a list of information about task and displays them to the user. The user can choose task traits and has the option to change them. Once, done, the user has a “save changes” option and the task is updated.

Actors: User

Related use case: Tasks

Stakeholders: Task is updated

Pre-condition: A task must be created so that it can be edited.

Post-condition: Task will be updated

Flow of Events:

Actor	
1. User chooses to edit task	
2. Task traits are displayed to user	
3. User chooses trait	
6. User chooses new option from checklist or keyboard	
7. User edits item and saves work	

System

4. Trait listener is activated
5. Options displayed / keyboard displayed for editing
8. Updates task

<i>Exception:</i>	Step	Condition	Action Description
	7a	if step 7 fails	User does not save work, trait/task is not updated

2.2.13 Spin Wheel

Description: When the user clicks Spin Wheel, a spinning wheel animation is shown. A task is selected from the list of tasks to be completed.

Scenario: The user will press a “Spin Wheel” button in the center of the wheel. The app will display a spin animation, and a task will be selected from the list of tasks to be completed.

Actors: User

Related use case: Session

Stakeholders: The user will received a new task to complete

Pre-condition: User needs to have at least one task in the last. If there is no task in the list, user will be prompted to input task

Post-condition: After the wheel spin, the user will see their current task to complete. The system will update the user’s current task.

Flow of Events:

Actor	
1. The user clicks the Spin Wheel button which executes the wheel spin activity	
3. The user clicks the accept button and the chosen task becomes the current task	

System

2. The system displays a wheel spin animation while randomly choosing a task to display.

<i>Exception:</i> Wheel"	Step	Condition	Action Description
	2a	if step 2 fails	If no tasks in list, error message shown. Else, return user to "Spin
	4a	if step 4 fails	Display error message and ask user to spin again. Return user to spin wheel.

2.2.14 Add Task

Description: When the actor wants to create a new task to add to his/her to-do list, a button is clicked and a new task is created and added to the list. Then the actor will be prompted to fill out the data for this new task such as task description, category, and time limit.

Scenario: The user chooses "Add Task" from the list menu, an input window/screen appears for them in input new task

Actors: User

Related use case: Task

Stakeholders: The user intends to ask one or more tasks to the list of tasks to be completed

Pre-condition: User must have a profile.

Post-condition: The user's task list is updated with each added task

Flow of Events:

Actor	1. The user selects "Add Task" from the Task menu
	3. The user inputs each new task and clicks a confirmation button
System	2. The system displays a window/screen for the user in input their new tasks
	4. The system displays a success message. The system then updates the user's task list which the additions.
	5. The system displays the updated task list

<i>Exception:</i> task screen	Step	Condition	Action Description
	2a	if step 2 fails	Return to list menu
	4a	if step 4 fails	The system displays an error message and returns to the new task screen

2.2.15 Rewards

Description: A screen displays currently earned and all available rewards. The actor is able to spend chips to purchase rewards or view their rewards through their profile.

Scenario 1: The actor spends chips in order to purchase/unlock a reward.

Scenario 2: The actor clicks on rewards in the profile view and is able to see which rewards have been obtained.

Actors: User

Related use case: Purchase Rewards

Stakeholders: The user can view and purchase rewards that will be displayed on their profile

Pre-condition: The user must have a profile

Post-condition: If the rewards are simply viewed there is no change. If the rewards are purchased then they are added to the profile data

Flow of Events:

Actor	1. The user selects Rewards from the main menu
	3. The user can select a reward to purchase
	5. If user selects rewards to purchase:
	a. The user has enough chips: can click buy

- b. The user does not have enough chips, buy button not available

System

- 2. The system displays a screen with earned and available rewards
- 4. If user selects reward to purchase
 - a. The user has enough chips: a confirmation screen appears to confirm reward purchase
 - b. The user does not have enough chips: an error screen appears notifying the user they do not have enough chips
- 6. If user selected reward to purchase
 - a. If the user accepts purchase, update Rewards list and subtract chip cost from chips balance. Each purchased reward appears greyed out.

- b. If the user declined, return to Rewards page
- c. When the user clicks OK on the error message, return to Rewards page

<i>Exception:</i>	Step	Condition	Action Description
	2a	if step 2 fails	Return to profile menu
	4a	if step 4 fails	Return to rewards page
	6a	if step 6 fails	Return to rewards page with no change

3. System Requirements Specifications

3.1 Preface

3.1.1 Revisions

Date	Version	Description	Author
February 18, 2014	1	Original Draft	Natalia Alonso
May 4, 2014	2	Revised specifications and diagrams	Natalia Alonso

3.2 Introduction

3.2.1 Overview of the Document

This document provides a guideline for the Task Roulette application implementation and is meant to be easily read and understood by those with background in Computer Science.

3.3 Glossary

Android SDK - Android software development kit; allows for creation of software in the Android OS

Android OS - Android operating system; an operating system meant for touchscreen devices

Facebook - online social networking service

Google Play - distribution platform for Android applications, as well as digital media

GUI - graphical user interface; allows users to interact with electronic devices

Twitter - online microblogging and social networking service

3.4 User Requirements Definitions

3.4.1 Mandatory Requirements

The Task Roulette application will take a list of tasks input by the user and randomly select a task for the user to complete. The user must be able to create a profile with their personal information. The user must be able to input tasks, edit tasks, finish completed tasks, and delete tasks from their list. The user must be able to purchase rewards with chips and view their achievements.

3.4.2 Desirable Requirements

The user may be able to share their completed goals, earned achievements, and purchased rewards on social networking. This may include but is not limited to Facebook and Twitter. The user may be able to make these sharing events manual or automatic.

3.4.3 Optional Requirements

A friends list will be added to the application. The user would be able to search for and add friends who are already using the application. They may also invite friends to install and use the application. The friends list would include a friends timeline, allowing friends to share their completed goals, earned achievements, and purchased rewards amongst themselves.

3.5 System Architecture

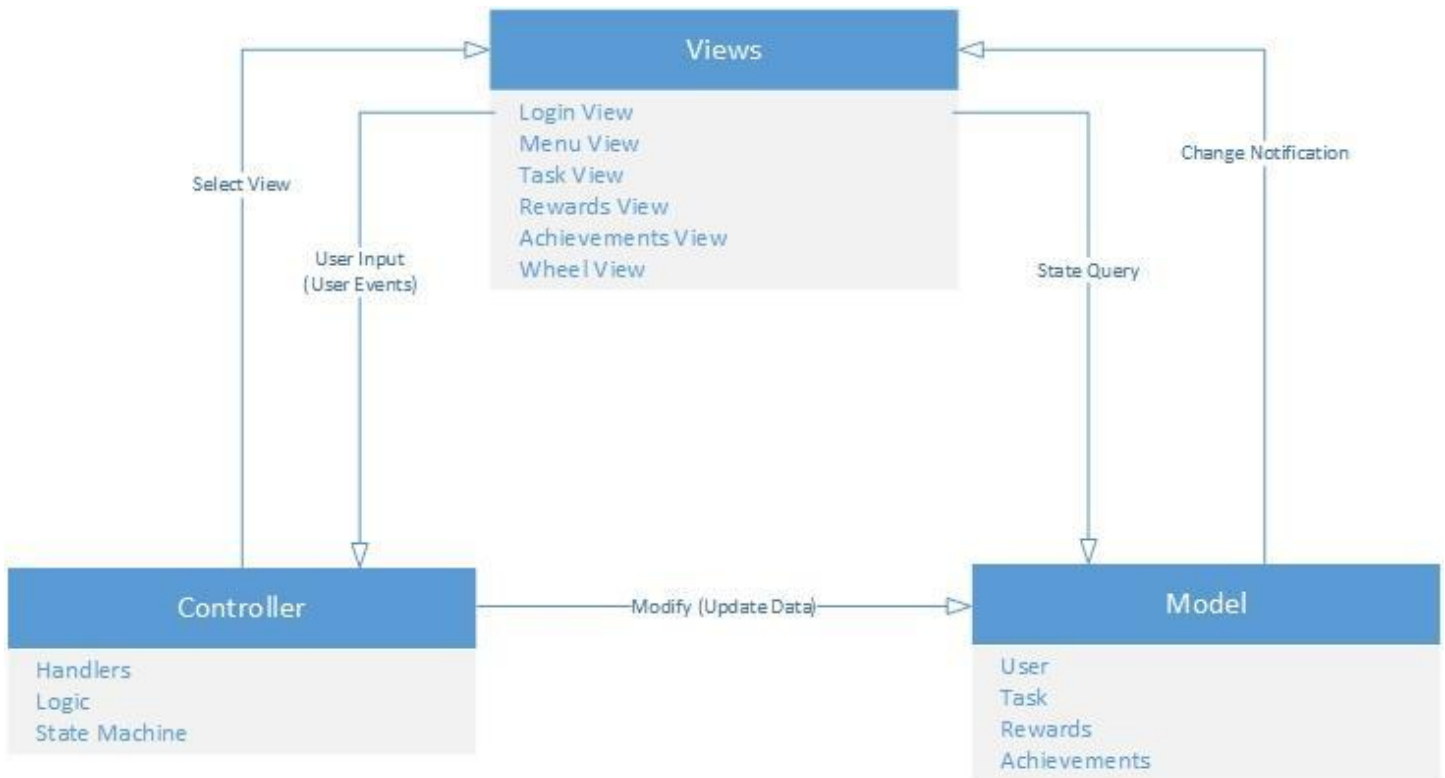
3.5.1 Model View Controller

Our system will use a MVC system architecture, which is broken up into three main parts: The model, the view, and the controller.

The Model will consist of all of the data structures that we will need to build in order to keep our data organized and available. These data structures include User, Tasks, Rewards, and Achievements. The model will notify the view of changes in the model so that those changes will be reflected on screen.

The View will be the different displays that the user will see on the screen and interact with. This includes the login view, menu view, task view, wheel view, rewards view, and achievements view. Also included in the view are any user input event listeners. The view will make state queries to the model to be sure it has current data. It will also propagate user input and user events to the controller.

The Controller will consist of all the handlers, logic and the state machine. This includes the task handler, event handlers, the animation thread for the spinning wheel, and the application state machine. The controller will select the view and modify or update data in the model.



3.6 System Requirements Specification

3.6.1 External Interfaces

The user interface is one GUI that will be displayed to the user containing menus and options. Input will be entered through integrated OS keyboard directly to the Task Roulette application and saved in local storage. The hardware interfaces is Android Virtual Machine and user cell phone running Android OS. The software interface is the AVM used for development and testing. The user's cell phone will host the application.

3.6.2 Functional Requirements

3.6.2.1 The user must be able to create a profile. When the application is first started, the user is asked to create a profile in order to use the application. The profile stores the users information, and when viewed, displays awards and achievements in addition to personal information. Upon first start up, or on each start up until a profile is created, the user is prompted to enter their information to create a new profile on the application. The application is initialized with the user's profile information and stored until the application is uninstalled. If there is missing information, the profile will not be created and the user will be notified. The user will continue to be prompted to create a profile and enter personal information until the profile is completely filled out.

The exact sequence of operation is:

1. Application is started
2. User is prompted to input profile information
3. Once all information is entered, user clicks "Create Profile"
4. The application is initialized with the user's profile information
5. The profile is stored until the application is uninstalled

If the user does not complete all fields, the application will prompt the user for the missing fields. If the application fails to create a profile, the user will be returned to the "Create Profile" state. In summary, the user will input their personal information, the information will be saved and used to initialize the application,

information such as name may be used throughout the application, and the information will be displayed on the user's profile page

3.6.2.2 The task list in the application is composed of a series of tasks that are input by the user. There are several options for tasks: New Task, Edit Task, Delete Task, and Finish Task.

3.6.2.3 When the user wants to create a new task to add to his/her task list, a button is clicked and a new task is created and added to the list. Then the user will be prompted to fill out the data for this new task such as task description and time limit. The exact sequence of operation is:

1. The user selects "New Task" from the Task menu
2. The system displays a window/screen for the user to input their new tasks
3. The user inputs each new task and clicks a confirmation button
4. The system then updates the user's task list with the additions.
5. The system displays the updated task list

3.6.2.4 If a task is no longer wanted, user can choose to delete it from task list. If deletion fails, list remains unchanged. The user is returned to unchanged list view. When user deletes a task, the task is removed from the task list. The task is no longer available to be chosen by Spin Wheel as new current task. The exact sequence of operation is:

1. User selects Task Menu
2. User selects task to delete and initiates deletion
3. Update task list and display updated list

3.6.2.5 The application allows user to finish a previously created task, marking it as completed. A completed task is removed from task list. If task update fails, task is not marked as completed. User is returned to unchanged task list. The exact sequence of operation is:

1. User selects task from list
2. System displays option for selected task
3. User selects task to be marked as completed
4. System checks off task
 - i. Adds it to completed tasks list
 - ii. Removes task from task list

3.6.2.6 When the user clicks Spin Wheel, a spinning wheel animation is shown. A task is selected from the list of tasks to be completed. It becomes their current task. The user will press a "Spin Wheel" button. The application will display a spin animation, and a task will be selected from the list of tasks to be completed. If no tasks in list, error message shown and user is redirected to Create Task. If wheel spin fails, display error message and ask user to spin again. The user is returned to Spin Wheel. The wheel spin chooses a new current task. The exact sequence of operation is:

1. The user clicks the Spin Wheel button which executes the wheel spin activity
2. The system displays a wheel spin animation while randomly choosing a task to display.
 - i. Displays the task as a message box with an accept button
3. The user clicks the accept button and the chosen task becomes the current task

3.6.2.7 The user is able to spend chips to purchase rewards or view their rewards through their profile. If rewards purchase fails, no reward is acquired and chips balance is unaffected. The user is returned to Rewards page. The exact sequence of operation is:

1. The user selects Rewards from the main menu

2. The system displays a screen with available rewards
3. The user can select a reward to purchase
4. If user selects reward to purchase
 - i. The user has enough chips: the buy button is available
 - ii. The user does not have enough chips: the buy button is not available

3.6.2.8 The user will be able to view achievements. Achievements can be earned with completed tasks. If achievements cannot be displayed, user is returned to the main menu. The exact sequence of operation is:

1. User presses the “Achievements” button.
2. System initiates the Achievements Activity
3. System displays achievements
4. User views achievements

3.6.3 Non-Functional Requirements

3.6.3.1 Product Requirements

Only one instance of the application is allowed at a time. Only one user profile is supported at a time. The only information handled by the application is the input given by the user.

3.6.3.2 Organizational Requirements

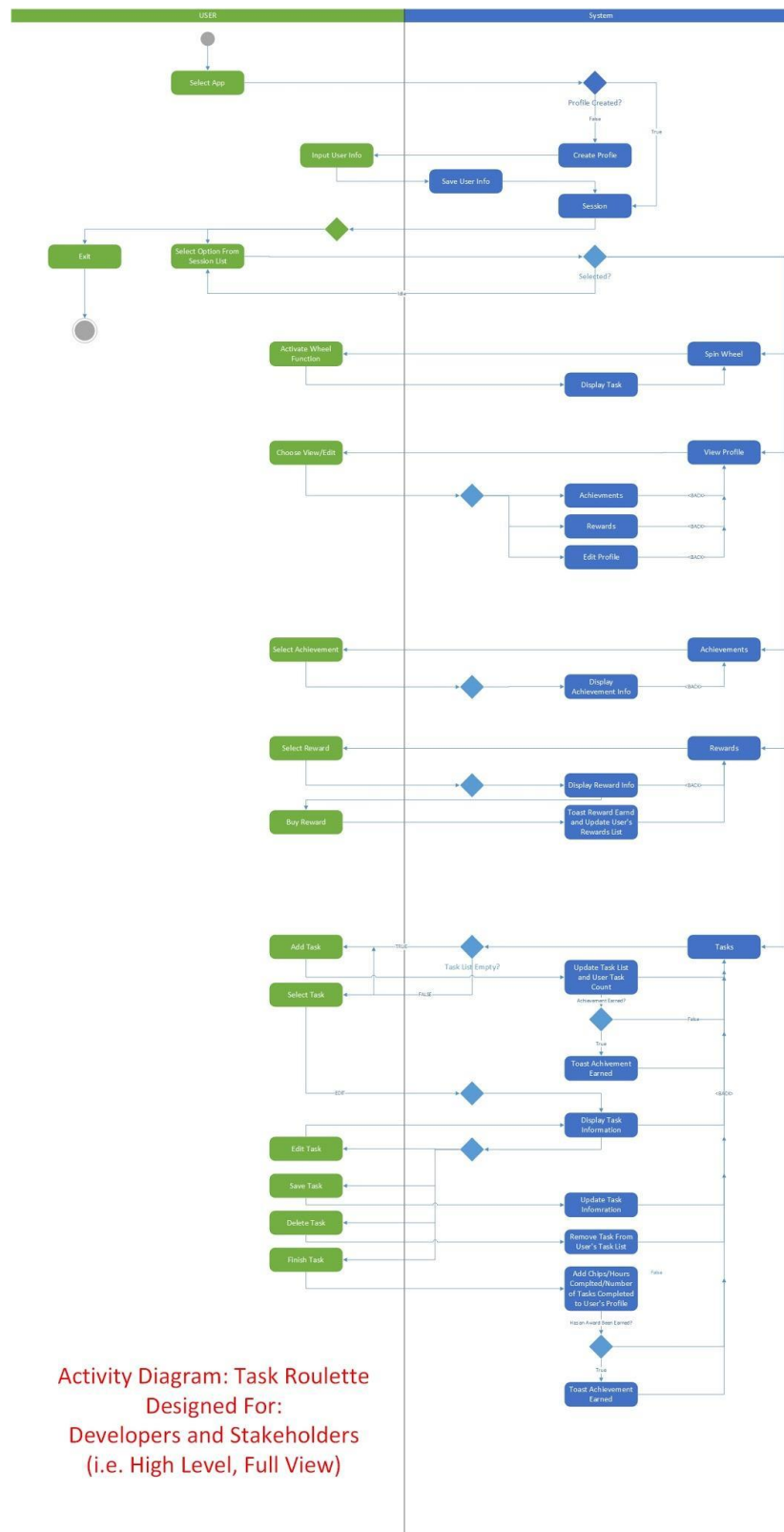
The application is designed and developed to be used on the Android OS. The application will be written in Java using the Android SDK.

3.6.3.3 External Requirements

The application is designed to gather and store user information locally. The user information will not be collected or shared outside of the application.

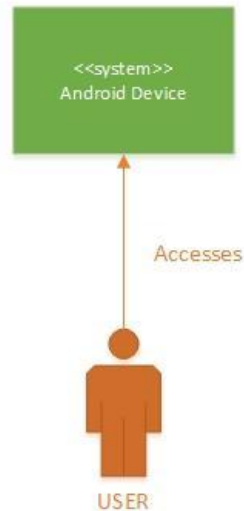
3.7 System Models

3.7.1 Activity Model



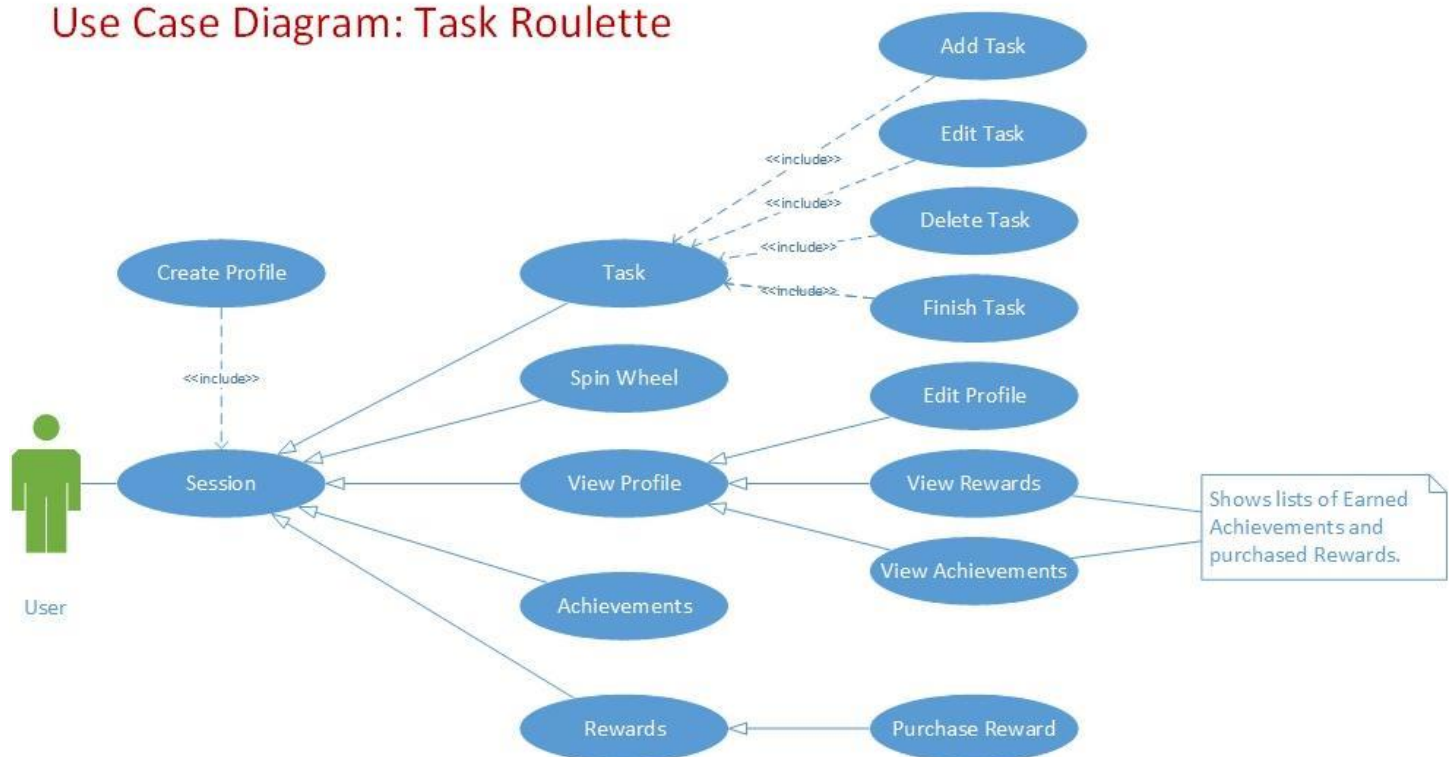
3.7.2 Context Model

Context Model: Task Roulette



3.7.3 Use Case Model

Use Case Diagram: Task Roulette



3.7.4 CRC Cards

User/Profile	
Responsibilities <ul style="list-style-type: none"> • Display profile • Store rewards • Store achievements • Store user information • Store task list 	Collaborators <ul style="list-style-type: none"> • Task • Rewards • Achievements

Task	
Responsibilities <ul style="list-style-type: none"> • New task • Edit task • Delete task • Finish task 	Collaborators <ul style="list-style-type: none"> • Profile • Category

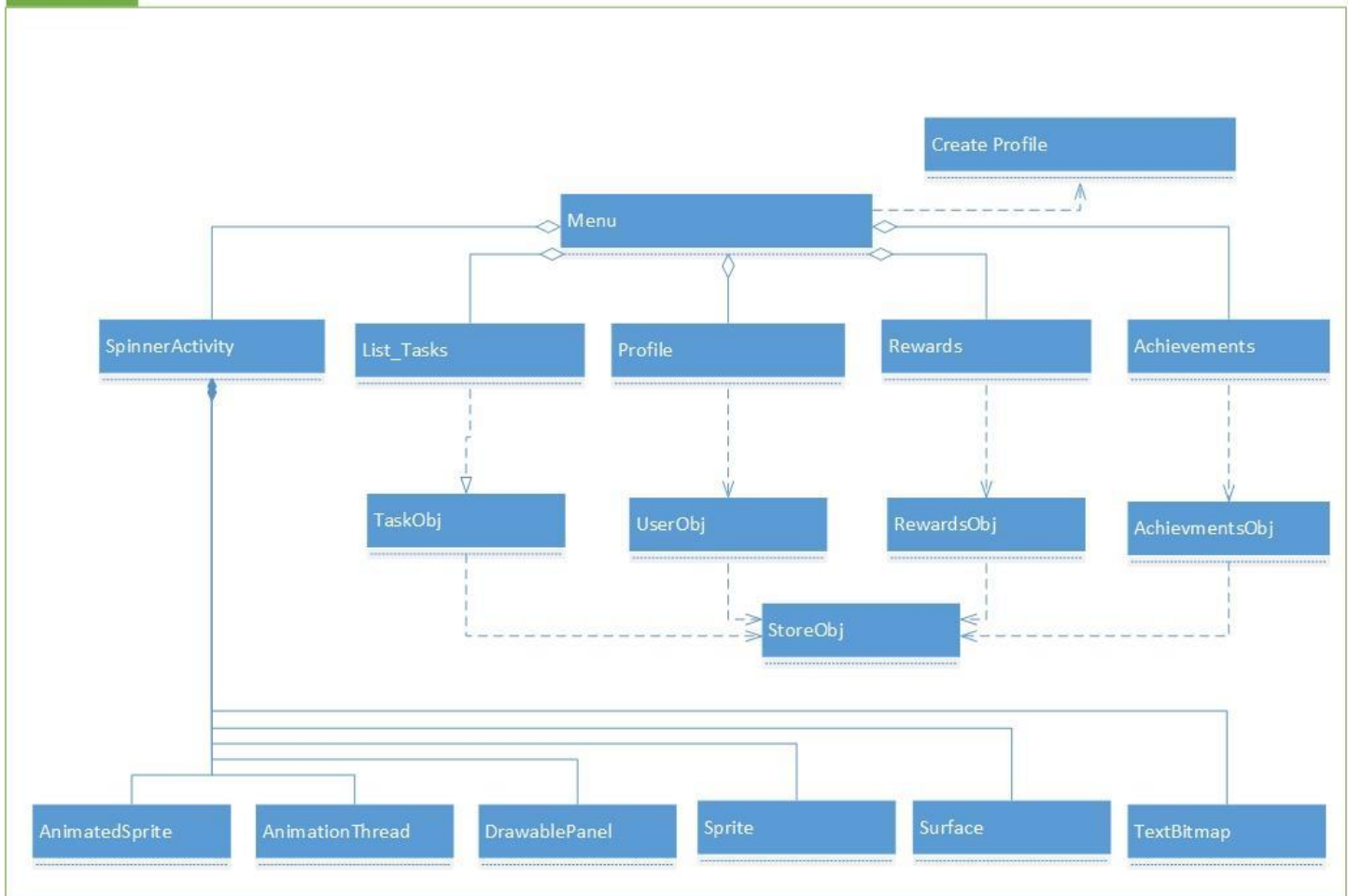
Rewards	
Responsibilities <ul style="list-style-type: none"> • Knows Reward Name • Knows Reward Cost • Knows if Reward is Owned by User • Knows Reward Type 	Collaborators <ul style="list-style-type: none"> • Profile

Achievements	
Responsibilities <ul style="list-style-type: none"> • Achievement name • Achievement ID • Requirement(s) to be met • Flavor text • Gives chips 	Collaborators <ul style="list-style-type: none"> • Profile

Scene	
Responsibilities <ul style="list-style-type: none"> • Read in Layout • Display Layout on screen • Send messages to objects when buttons are clicked 	Collaborators <ul style="list-style-type: none"> • Profile • Task • Rewards • Achievements • Menu • Wheel

3.7.5 Analysis Level Class Diagram

Task Roulette



3.8 System Evolution

3.8.1 Fundamental Assumptions

The Task Roulette application is built to run on Android devices that are running version 3.0 (Honeycomb) and above. The latest version that will be initially supported is 4.4 (Kit Kat). We are hoping that people would enjoy making their chores into a fun activity. We are also assuming that the roulette fantasy will be one that our audience will know and enjoy.

3.8.2 Effect of Product

Initially, we hope to launch a product that will suit our own personal productivity and entertainment needs. Our goal is to get everyone in our class section to download and use the application. This is an integral part of the future of the application because it should yield feedback on the usefulness, engagement, entertainment, and quality of the product. We are designing this product to, firstly, satisfy our own needs. Ultimately, our product is similar to many that are currently available on the market, but with our own unique twist. We hope that our twist on an already flooded market could be just the thing to allow our application to get noticed.

3.8.3 Anticipated Changes

We anticipate that our currency system will become more robust towards the end of development. Initially, we are planning on making “spins” free so that our application can have functionality, and we can get a feel for how it works. Every time a task is completed, “chips” will be earned. We are going to initially only allow spending this currency on our rewards system. Later on in development, we are expecting to take a second look at how those chips could be spent on “spins” and possibly even “gambled” away and lost.

3.8.4 Future Additions

We anticipate that our reward system will evolve over time to include more aesthetic rewards and achievements. These will be integral to maintaining a fun and entertaining environment to keep users intrigued. If we market our product on Google Play in the future, we might add some sort of monetization scheme. We have talked about making the “chips” into a system that can be purchased with real money. This is not something we plan on doing for the initial release, but something that could possibly be done if there was a demand for and reasonable way to do so. Another addition to our application might be to expand the “sharing” capabilities to cover more social networks and be more boastful and fun. In order to support the social side of our application, we might have to build and maintain a database so that we can store information about each user.

3.8.5 Future of the Product

If our product turns out to be useful and engaging, it is possible that we support it past its initial launch date and release it to the Google Play market. At this point, the future of our product will be to obtain a large volume of downloads per month. Our product will need to be maintainable and upgradeable. Maintainability will require that we provide support and follow up to fix any bugs that might pop up. Additionally, we will have to provide version updates so that the application continues to run on the latest devices and android versions. We will also need to provide content updates, such as new rewards. Another way that we can support our product is by providing functionality improvements and additional features. If we choose to monetize our product, we might have to implement some sort of store and a database to keep track of purchases.

3.9 Appendices

Appendix A: Hardware

A.1 Devices running android will be the platform for which the application, Task Roulette, will be run on. All android devices have the necessary hardware that will be needed to facilitate this app. Said hardware must include at the minimum: WiFi, carrier provided data connection (3G or 4G), single core intel or ARM based CPU, at least 256MB on board RAM (system memory), at least 10MB on board memory (storage memory), touch screen of any resolution, and of course android OS. The optimal configurations to facilitate the app are identical to the minimum except for the following changes: dual core intel or ARM based CPU, 512MB on board RAM (system memory), and 20MB on board memory (storage memory).

A.2 The minimum configuration requirements will ensure that the application functions properly on the android device. Having the recommended configuration will make the application not only run faster and smoother but will provide faster response time when switching between the Task Roulette application and other application on the device.

Appendix B: Software

B.1 Aside from being an android OS device, the following software requirements must be present with the OS OEM applications for the minimal configuration: Calendar application and Alarm application. For optimal

configuration, in addition to the minimal configuration the following OS and/or device manufacturer applications are needed: device manufacturer social network integration. See appendix D for more information.

B.2 The minimum configuration requirements will ensure that the application functions properly on the android device. Having the recommended configuration will allow the user to experience all the features that the application has to offer. See appendix D for more information.

Appendix C: Database

C.1 Although the device does not need a database for its initial release and overall functionality, a possible future feature of a ranking system will require it. The system will be able to collect certain information from all the users and rank them accordingly and then display said ranks in the “Ranking” section of the application. This feature is a planned future feature and is not guaranteed to be released with the applications initial launch.

Appendix D: Third Party Software

D.1 Third party software is described as follows: any software that is not part of the OEM Android OS and is not part of the Task Roulette series of software. Certain applications that are widely used by users of all mobile devices will be targeted by our app but are not necessarily required. The following third party applications are recommended but not required for the functionality of the Task Roulette application: social networking apps Facebook and Twitter, as well as, accounts that link to a profile to the social networking account.

4. Master Test Plan

4.1 Test Plan Identifier

G1-TR01.1

Authors: Natalia Alonso, Ilia Benson, Cory Carvalho, Jared Fowler, Daniel Smith

Date: April 1, 2014

4.2 References

- Task Roulette SRS Document v1.0
- Task Roulette CRC Cards
- Task Roulette Activity Diagrams
- Task Roulette Sequence Diagrams

4.3 Introduction

This is the Master Test Plan for the Task Roulette App for Android. This plan will relate to the basic functions that will be provided by the app, as specified by the user cases. The user cases are defined in the SRS document. The primary focus of this plan is to ensure basic functionality of the use cases.

The project will have three levels of testing: Unit, System, and Acceptance. The description for each of these levels will be addressed in the approach section and level specific plans.

The estimated time for this project is one month. The limited time suggests that any delays in the development process will severely affect the timeliness of the test plan.

4.4 Test Items

The following is a list of items to be tested:

- Main Menu
- Create Profile
- Edit Profile
- Profile Menu
- User Object
- Task
- Task Object
- Rewards
- Rewards Object
- Achievements
- Achievement Object
- Spinner
- Store Object

4.5 Software Risk Issues

The following software issues are critical areas that need to be extensively tested:

- Store Object
 - The ability to store objects and retrieve them between activities has been a major issue in the development of this application. We need to vigorously test our singleton and the ability of Store Object to synchronize between activities correctly. This is an area of risk that could extend the application's release date.
- Achievements
 - Communication of achievements to the profile.
- Task Object
 - Some functionality with Tasks being automatically removed from the list after their time expires will need to be tested to ensure that it is working correctly. We will also need to make sure that Tasks are being updated correctly between activities.

Software risks that need to be identified:

- We should address the user experience to determine the complexity of our application to the user. Specifically, we should address navigating through activities via back button option only.

Unclear requirements or requirements that cannot be tested:

- Rewards
 - Specific rewards have not been documented. Currency usage has not been explicitly discussed in documentation.
- Achievements
 - The Achievements activity is not extensively documented. A specific list of achievements and its communication with our Profile has not yet been fully documented. Currency usage has not been explicitly discussed in documentation.

4.6 Features to be Tested

- Create profile (H)
- Edit profile info (L)
- View achievement info from profile (L)
- View reward info from profile (L)
- View current task (H)
- Input a new task (H)
- Edit existing task (M)
- Delete task (M)
- Mark task as complete (H)
- Purchase rewards (M)
- Spin wheel to choose task (H)
- Change current task (M)

(H) - High level of importance. Feature implementation is required for functionality of app

(M) - Medium level of importance. Feature implementation is desired but not required for functionality of app.

(L) - Low level of importance. Feature is complementary, but not necessary for app.

4.7 Features not to be Tested

- Change backgrounds (reward)
- Change music (reward)
- Change avatar (reward)

4.8 Approach

- We will use Android Virtual Devices and a physical Android device to test the app.
- We will collect metrics for loading times, hiccups, etc.
- We will test several different screen resolutions and combinations of HW and SW.
- We will test on as many different devices that we have access to.
 - List of devices: Samsung Galaxy Player
- This will be tested on the Android OS version 2.3.5 and above.
- Each HW device is limited to a specific version of Android OS.
- Full regression testing will be performed with a major revision. With minor revisions, the scope of the change will be assessed and affected parts will be tested accordingly.
- Regression testing will be based on the scope of changes and will also be performed upon major revisions.
- If an element in the requirements and design does not make sense, it will be removed from the requirements or design. If something is untestable, then it will not be tested.

4.9 Item Pass/Fail Criteria

The completion criteria is that all modules work as intended with no serious unintended malfunctions.

At the unit test level:

- All test cases must pass.

At the Master test plan level:

- All lower level plans completed.

There may be some defects which are never detected. These will have to be left in the application.

4.10 Suspension Criteria and Resumption Requirements

Stoppage for a test or series of tests would constitute of defects so severe as to compromise the basic functionality of the application. In this case, testing would be suspended until the serious errors were resolved. Once basic function is restored, testing can resume.

4.11 Test Deliverables

- Application Output
- Error Logs

4.12 Remaining Test Tasks

None. The application is to be released fully functional and tested.

4.13 Environmental Needs

An android device such as a phone/tablet/emulator will be needed. The android device must support utf-8 or higher.

4.14 Staffing and Training Needs

Tests will be conducted by the Group 1 Development Team. This team is composed of five members which will first individually test a portion of the program to which they are assigned, and then upon successful testing, minor testing may then be continued by the rest of the team.

4.15 Responsibilities

Each member of the Group 1 Development Team will be responsible for testing their individual portion of the program. However, each group member may further test another's portion if such portion relates to their own.

The testing responsibilities are as follows:

- Natalia Alonso - Create Profile, Edit Profile, Profile Menu, User Object
- Ilia Benson - Task, Task Object, Main Menu, Store Object
- Cory Carvalho - Spinner, Ad-Hock, Boundary Testing
- Jared Fowler - Achievements, Achievement Object
- Daniel Smith - Rewards, Rewards Object

In the case of conflict or critical decision, counsel should be taken with other members of the group.

4.16 Schedule

The schedule for this project is largely dictated by the 380 course in which it has been integrated. We are to finish and present the project by the second to last week of the semester: May 15th. Because of this hard deadline, we have set a soft deadline of two weeks prior, resulting in a completion date of May 1st. This should give us two weeks to make any minor changes or aesthetic additions to our application as well as clean up the code and finish documentation regarding our application.

Given these deadlines, we should be testing our application during the last week of April. If this testing goes well, we should have time to complete our application on time. Should our testing bring up any moderate level concerns, we should have enough time to fix such bugs by our deadline.

Our application should have most of its functionality in place by April 17th in order for us to be able to start testing the following week or two after. The testing start date will be relative to the development end date (tentatively: April 17th) and will take no later than 2 weeks to complete. If the development end date is late, the testing phase will try to complete testing within a 1 week time frame; however, it is understood that it is the responsibility of development to complete the project on time.

If any highly volatile bugs are discovered or extreme functionality dysfunction exists during testing, we should have enough time to take action in order to correct our application and release it by the given deadlines. If for any reason our application fails to be fixable by the 8th of May, we will salvage what we can of the application for presentation to our class. In this worst case scenario, we will have one week to cut out any portions of our application that are broken and focus on documentation.. This documentation will be presented in lieu of or in combination with our application.

4.17 Planning Risks and Contingencies

- Group members may be unavailable for testing or cannot dedicate enough time to the testing process.
- There is only one physical android device being used for testing. The rest are emulators.
- The application deadline is soon after testing will begin.
- All features may not be implemented and some may have to be changed.

- If group members are unavailable or have limited time for testing:
 - The number of tests performed may have to be reduced.
 - The remaining group members will have to work overtime.

- If the requirements for the application change:
 - The number of tests performed may have to be reduced.
 - The number of acceptable defects will be increased.
 - The test team will work overtime.

4.18 Approvals

Development will conclude after the application has fulfilled all original documented requirements and is relatively stable. At this point, the approval process will take place to determine what steps need to be taken in order to proceed to the testing phase of development.

The application team (developers) will be responsible for the approval process. There will be a majority vote taken on whether or not to proceed from development to testing; and subsequently to polishing and post production.

It may be necessary to utilize other members of the course section in order to receive user input and help the development team during the approval process.

Our final presentation of the project will conclude with input from the course section as well as the course professor. This feedback will help to determine if the application developers should continue another development cycle and release the product to the public.

4.19 Glossary

Development team (developers, application team, group members, etc) all refer to group 1, consisting of: Natalia Alonso, Ilia Benson, Cory Carvalho, Jared Fowler, and Daniel Smith.

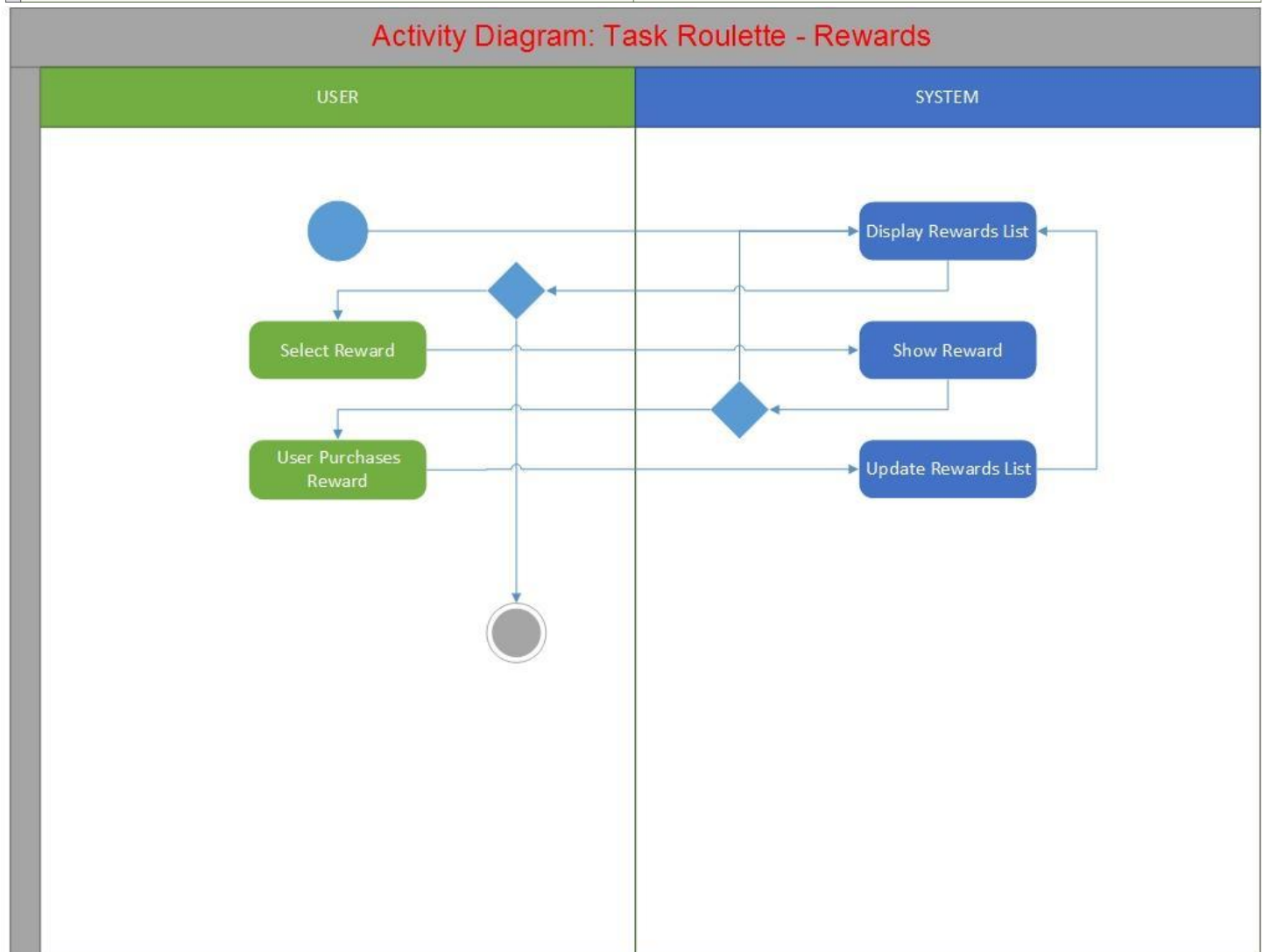
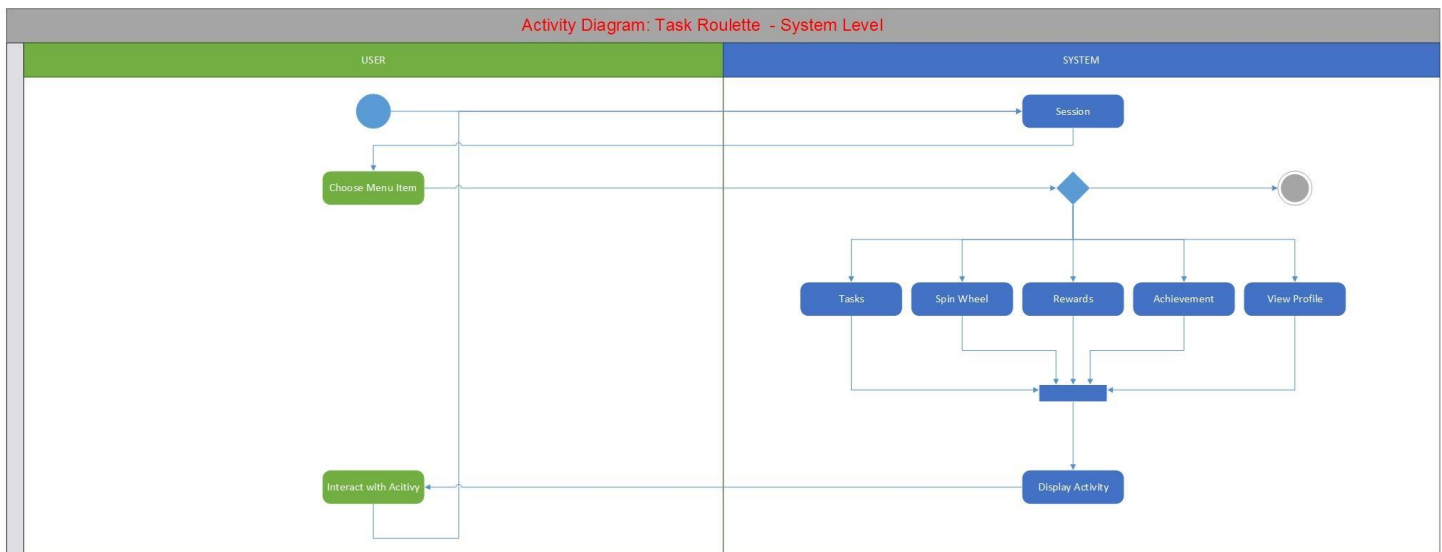
Professor refers to Professor Wang.

Section (course, course section) all refer to course section 15101 of Spring Semester 2014 at California State University, Northridge.

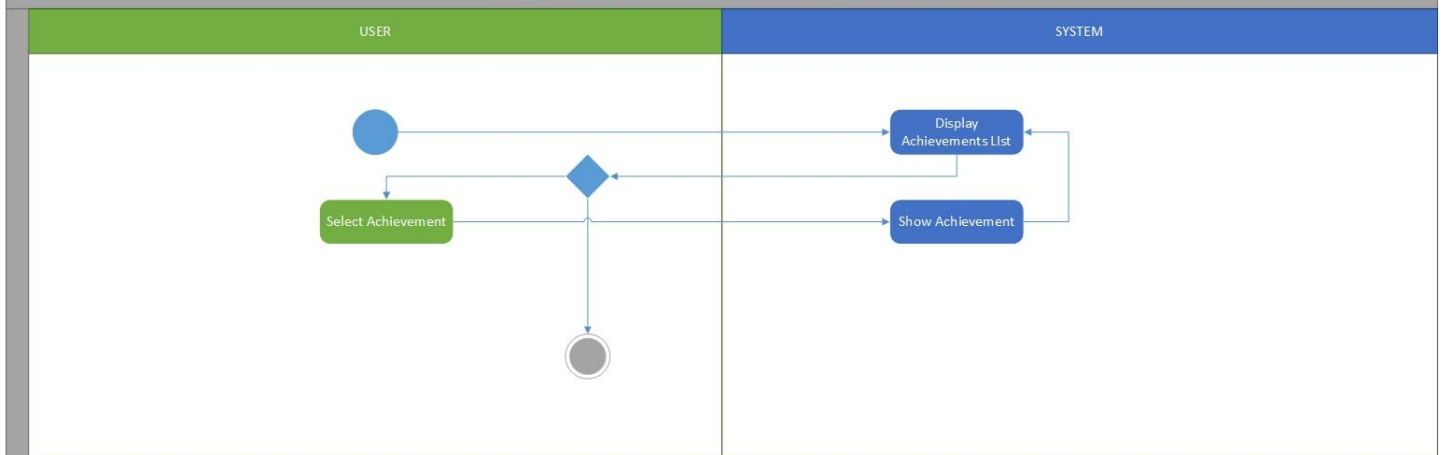
Spinner refers to the “spin” activity that is used to display a roulette wheel animation and facilitate choosing a new task to complete.

Task refers to a task object which is used to describe a user defined “to-do” item for our application.

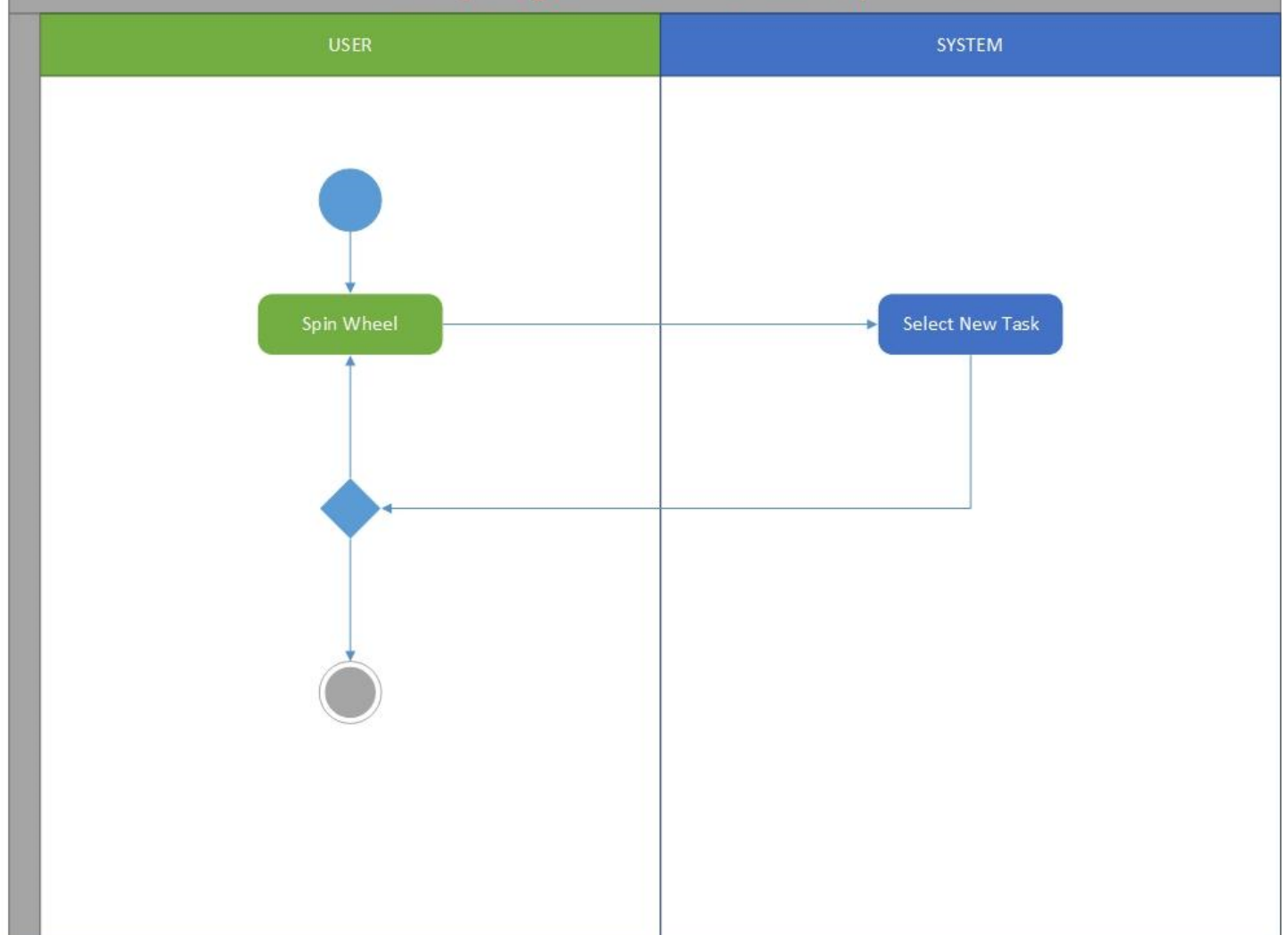
5. Activity Diagrams



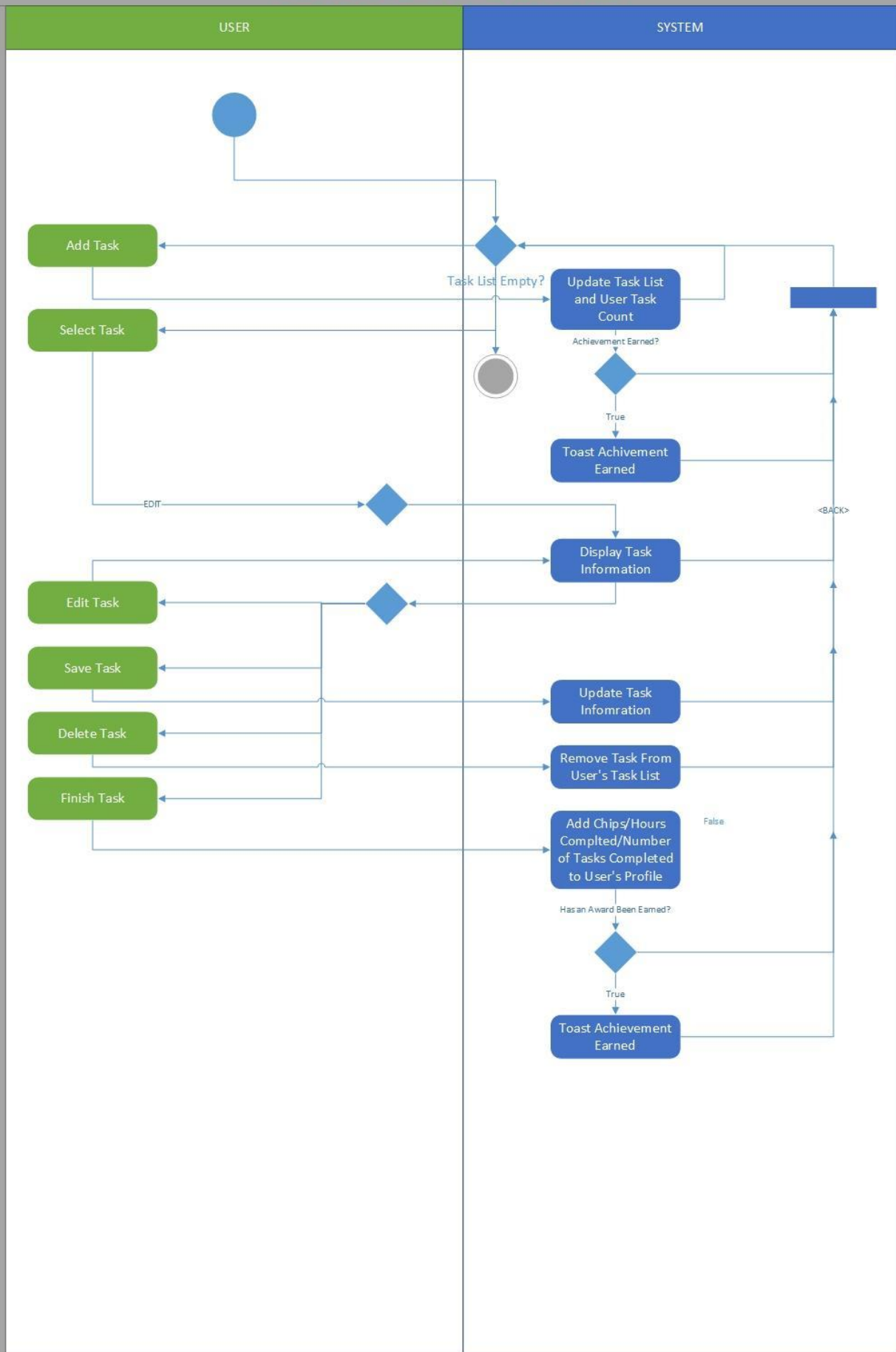
Activity Diagram: Task Roulette - Achievements



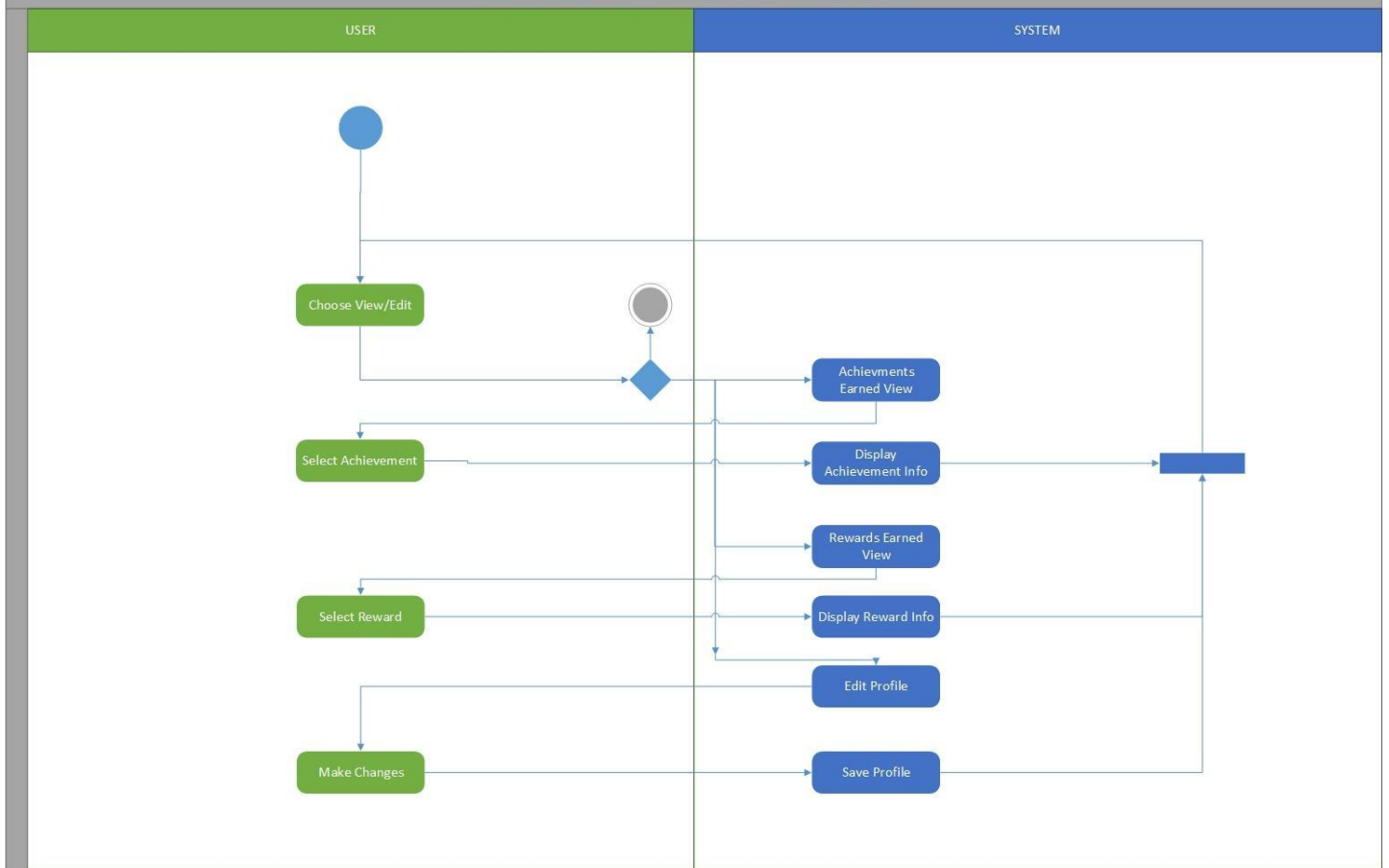
Activity Diagram: Task Roulette - Spin Wheel



Activity Diagram: Task Roulette - Tasks

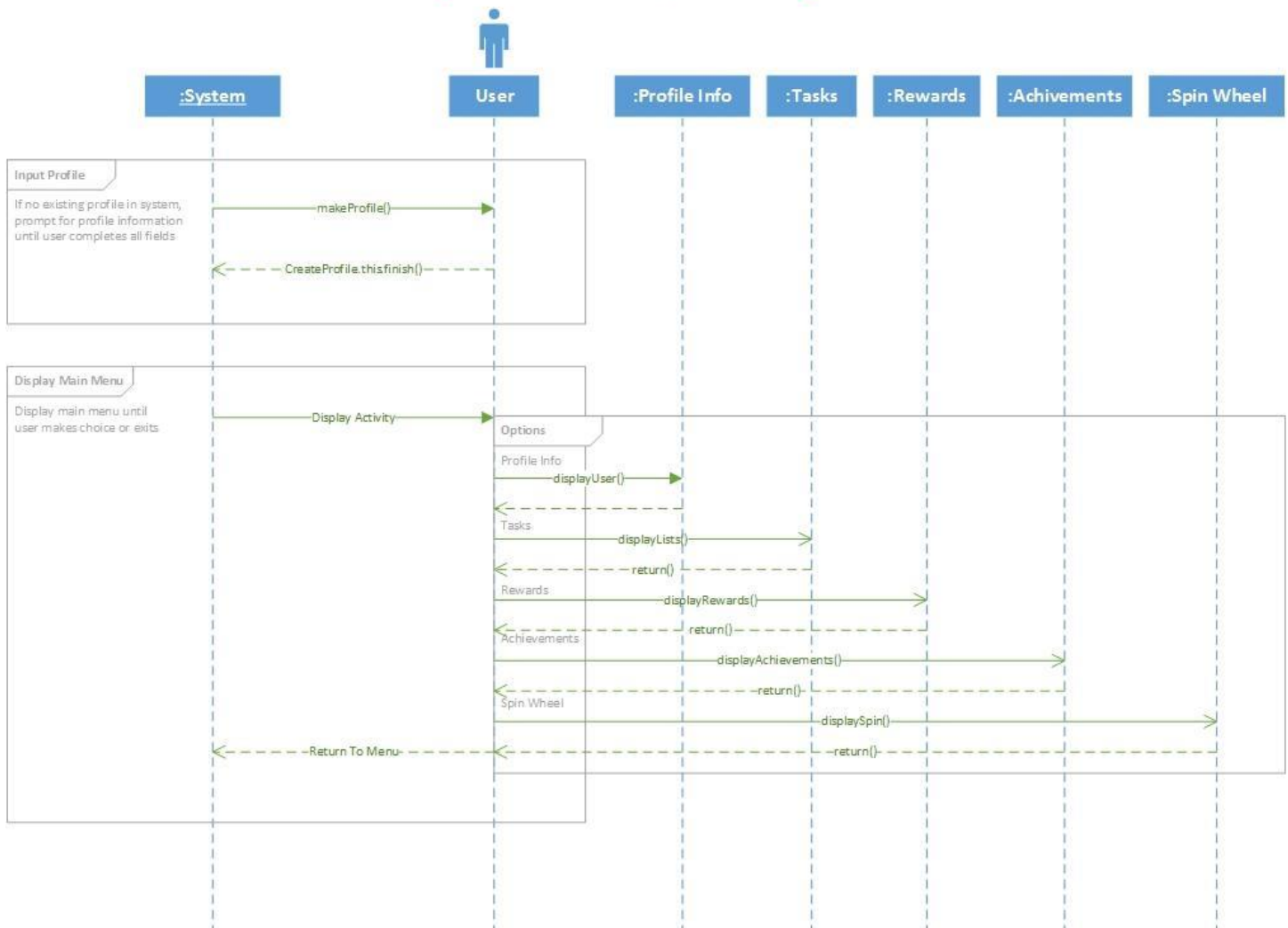


Activity Diagram: Task Roulette - View Profile

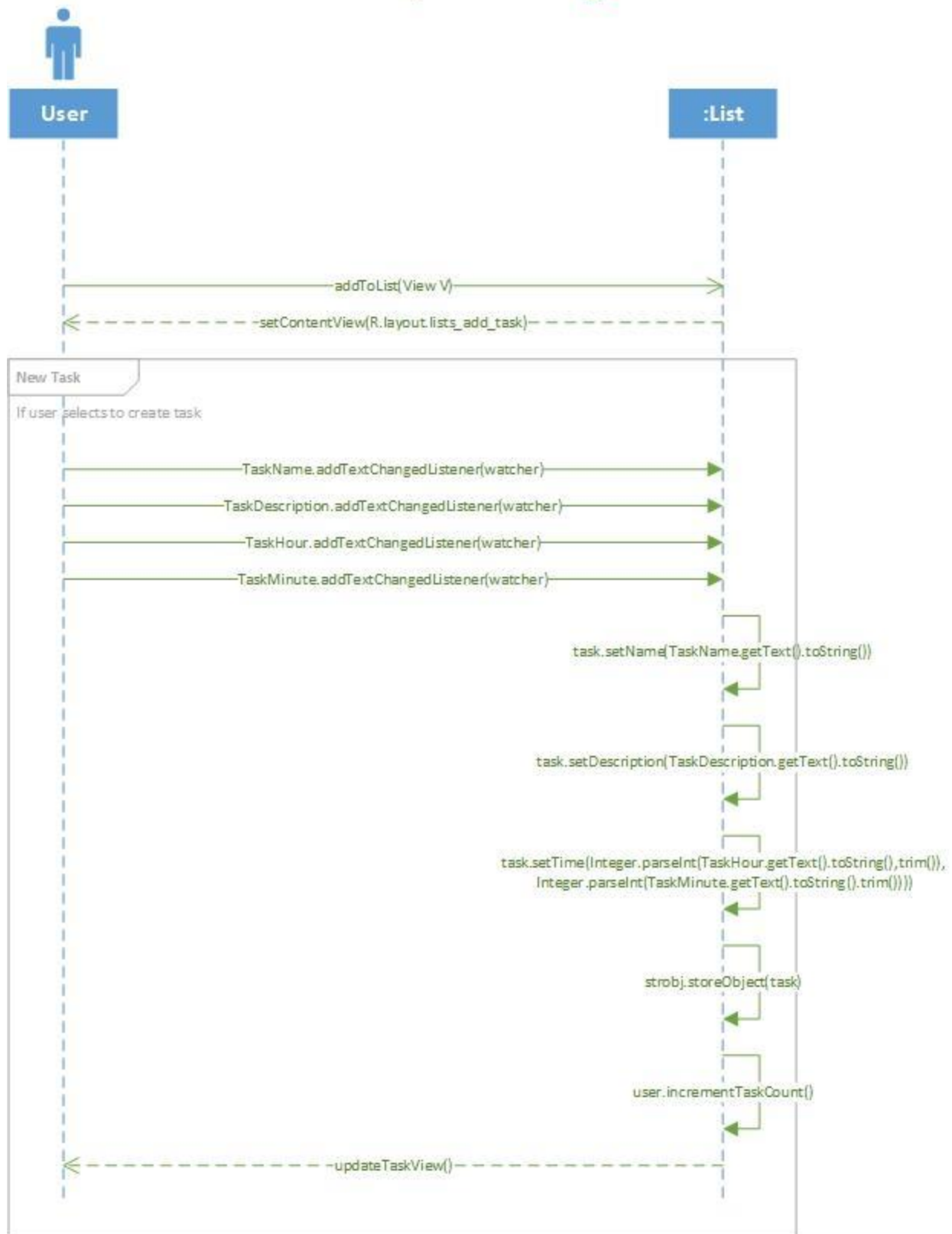


6. Sequence Diagrams

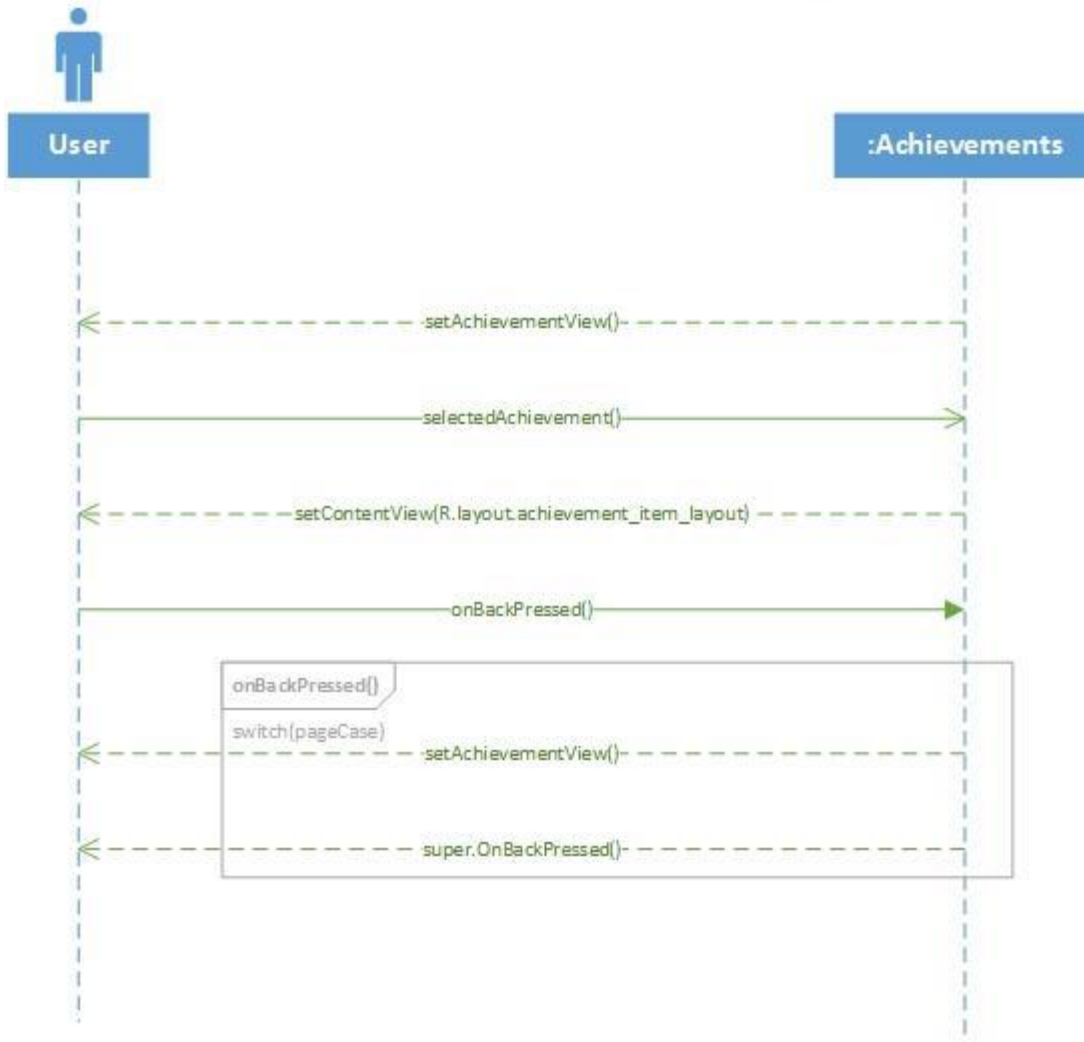
System Level Sequence Diagram



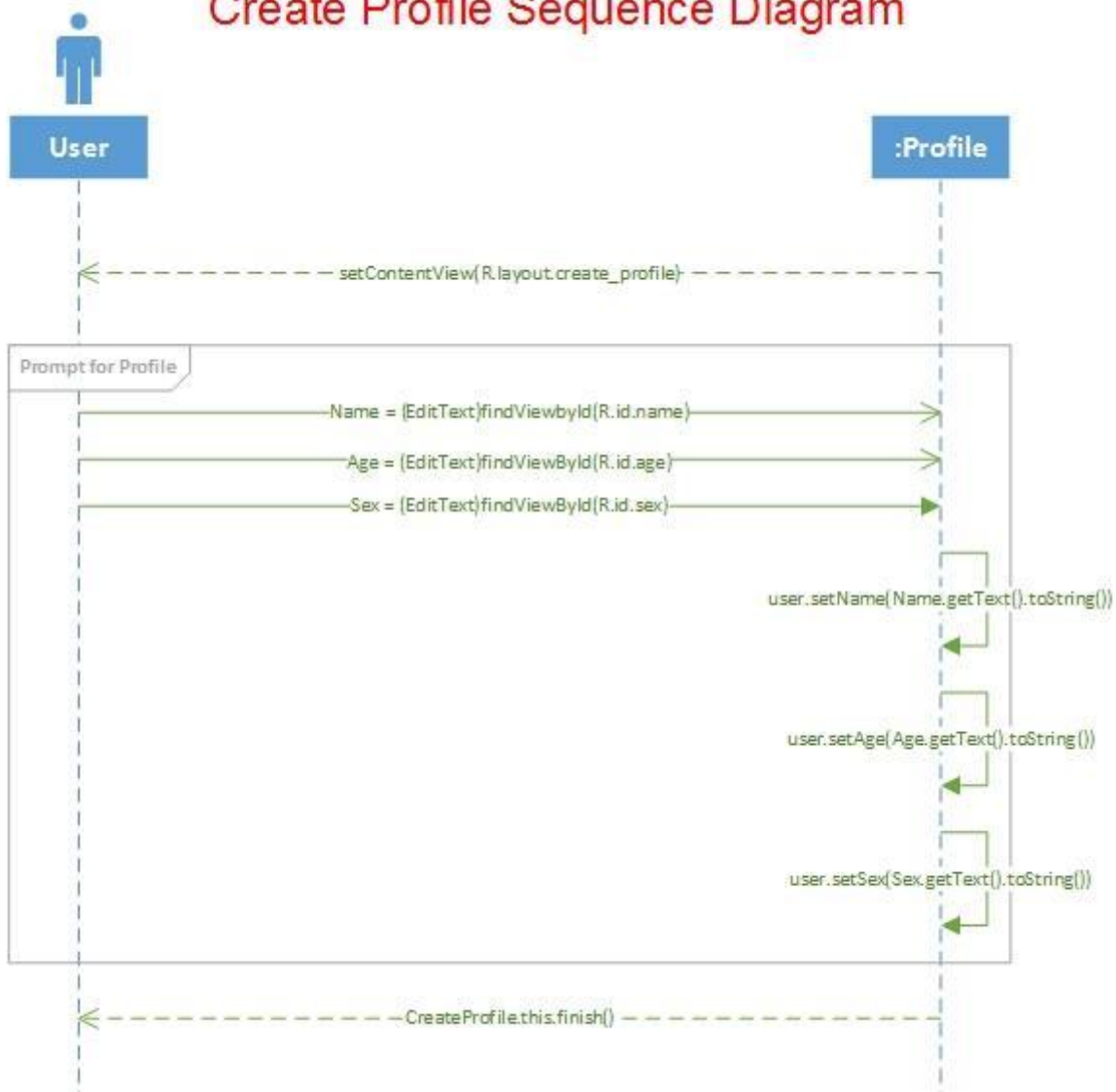
Add Task Sequence Diagram



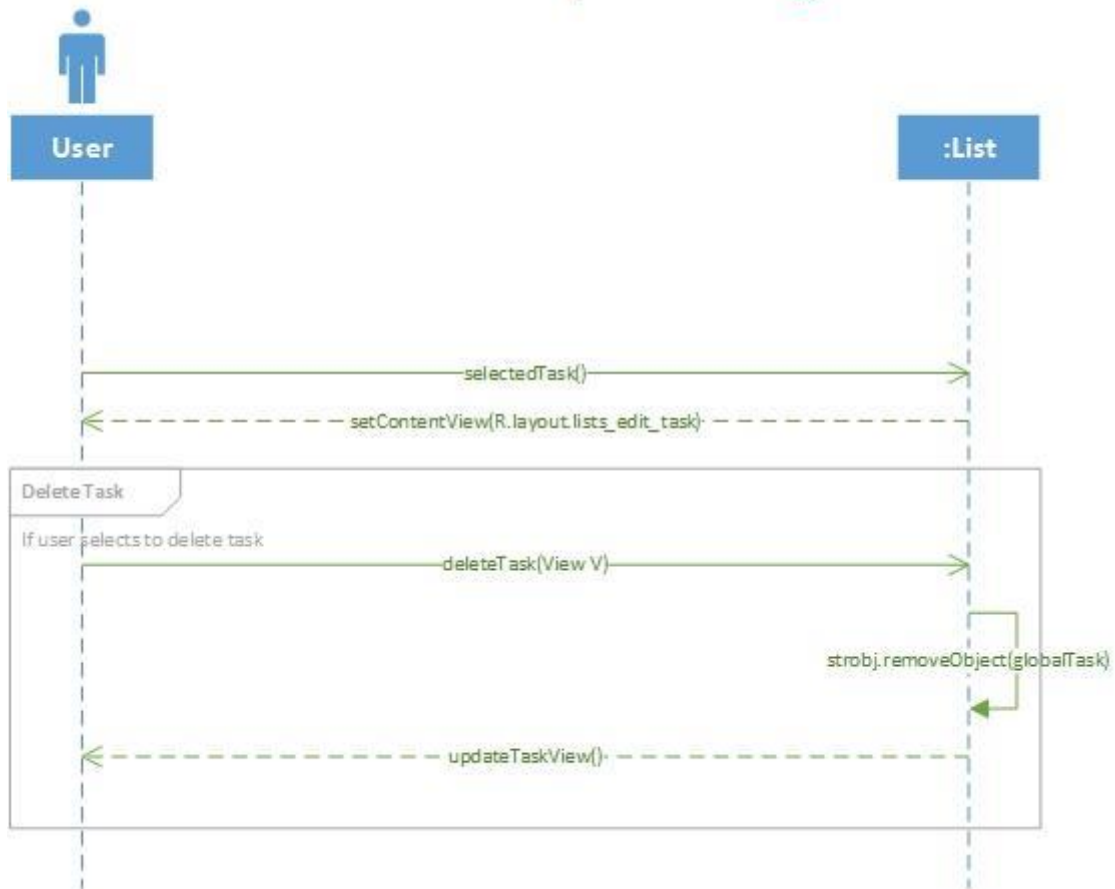
Achievements Sequence Diagram



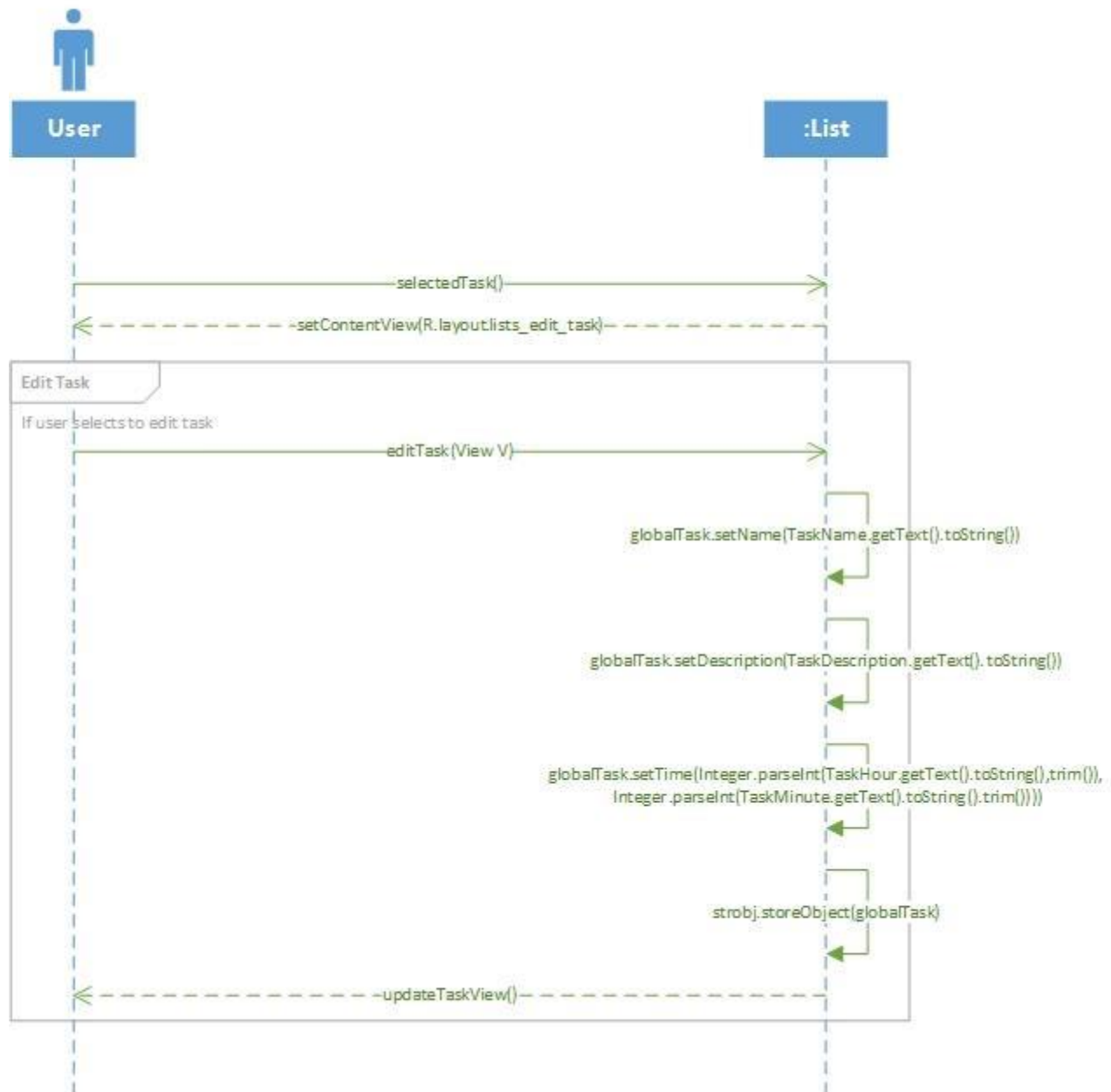
Create Profile Sequence Diagram



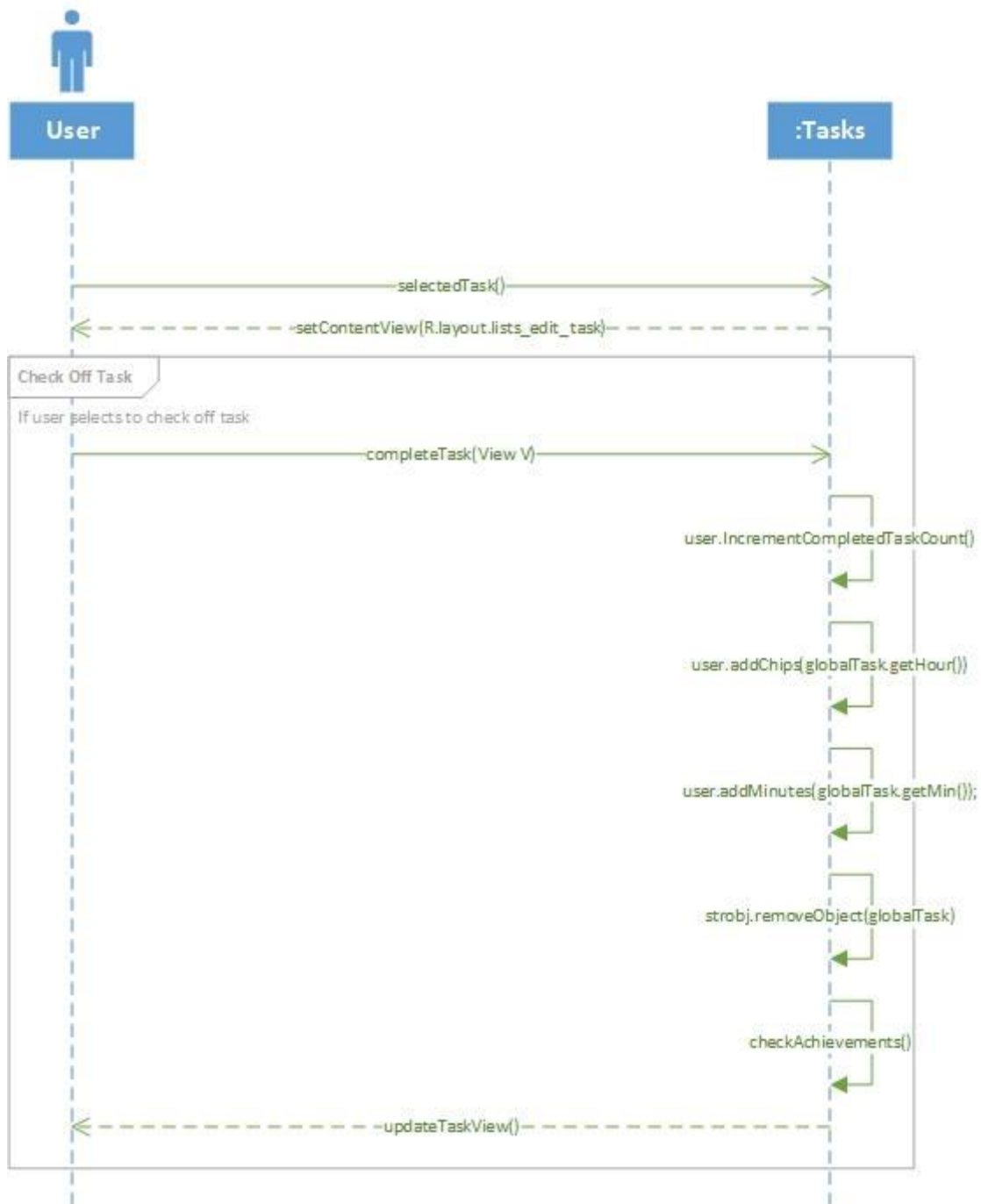
Delete Task Sequence Diagram



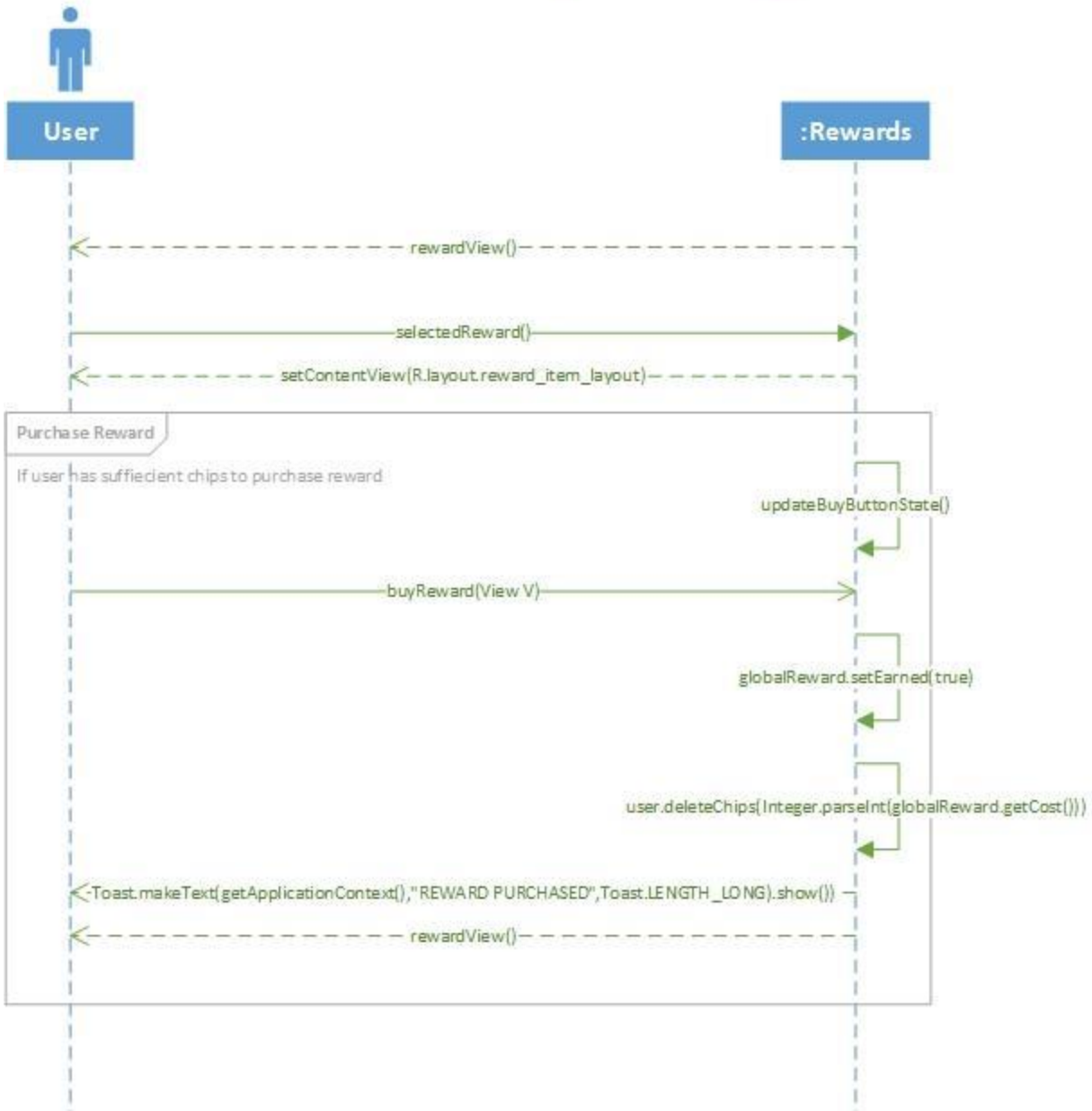
Edit Task Sequence Diagram



Finish Task Sequence Diagram



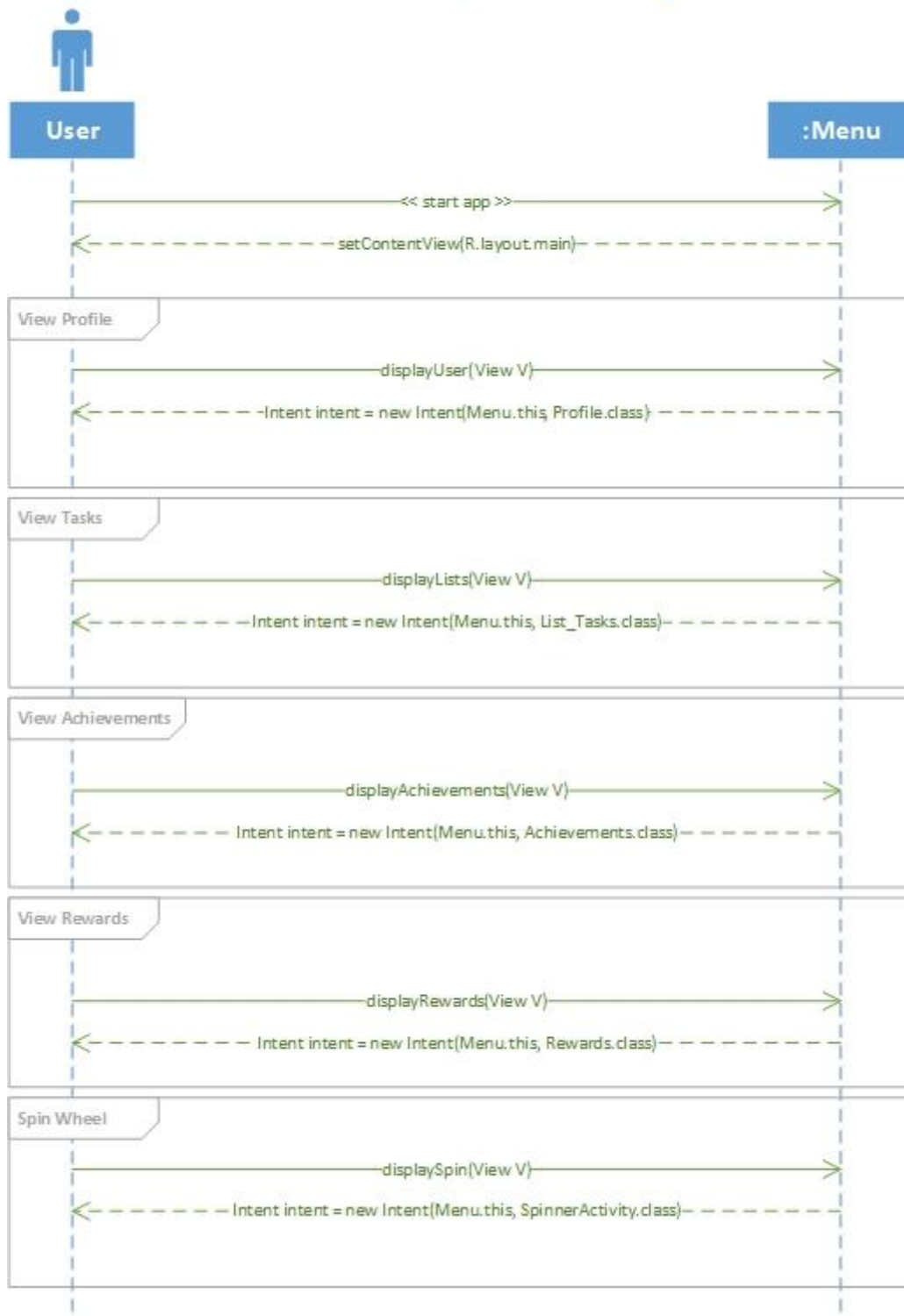
Purchase Reward Sequence Diagram



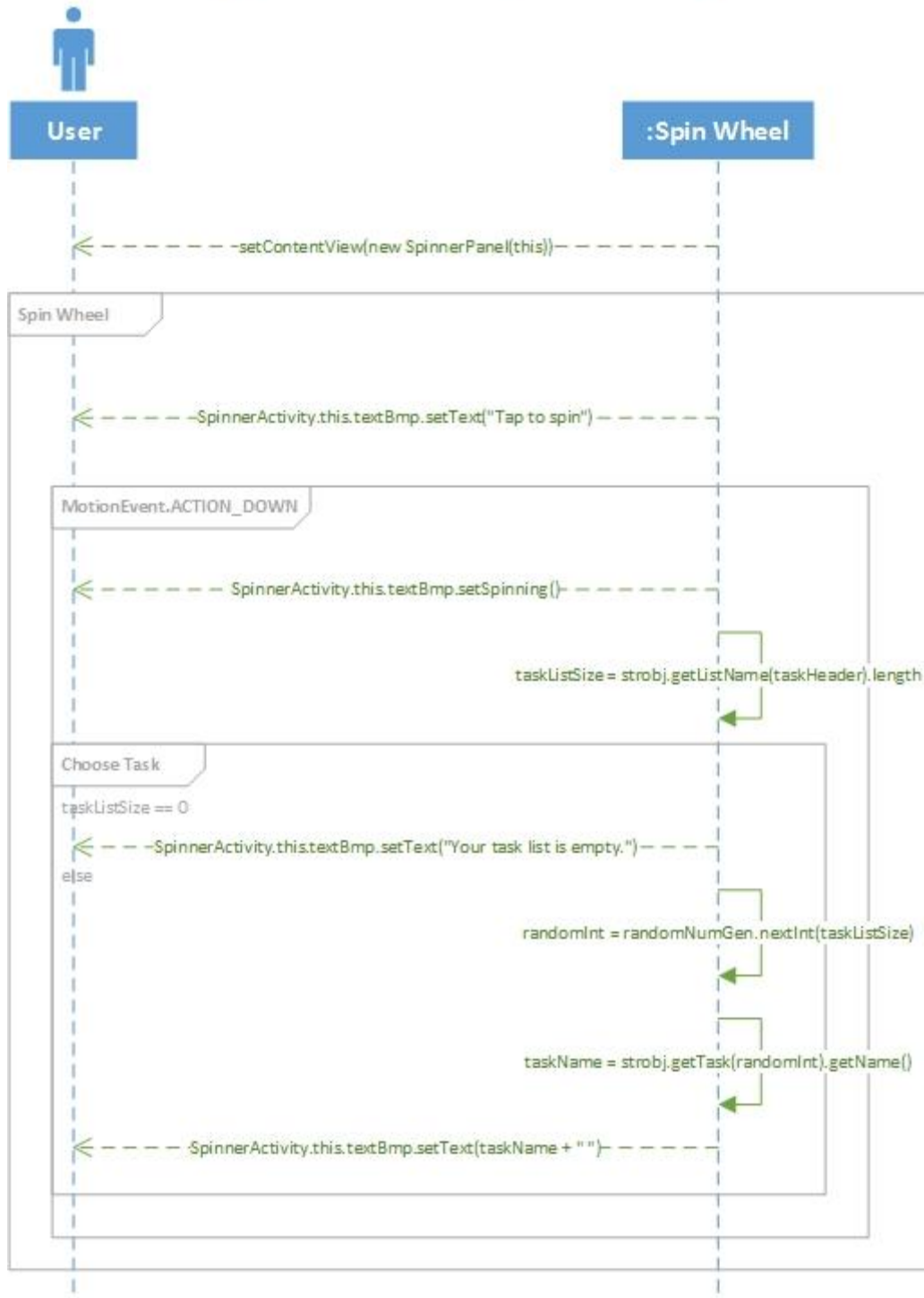
Rewards Sequence Diagram



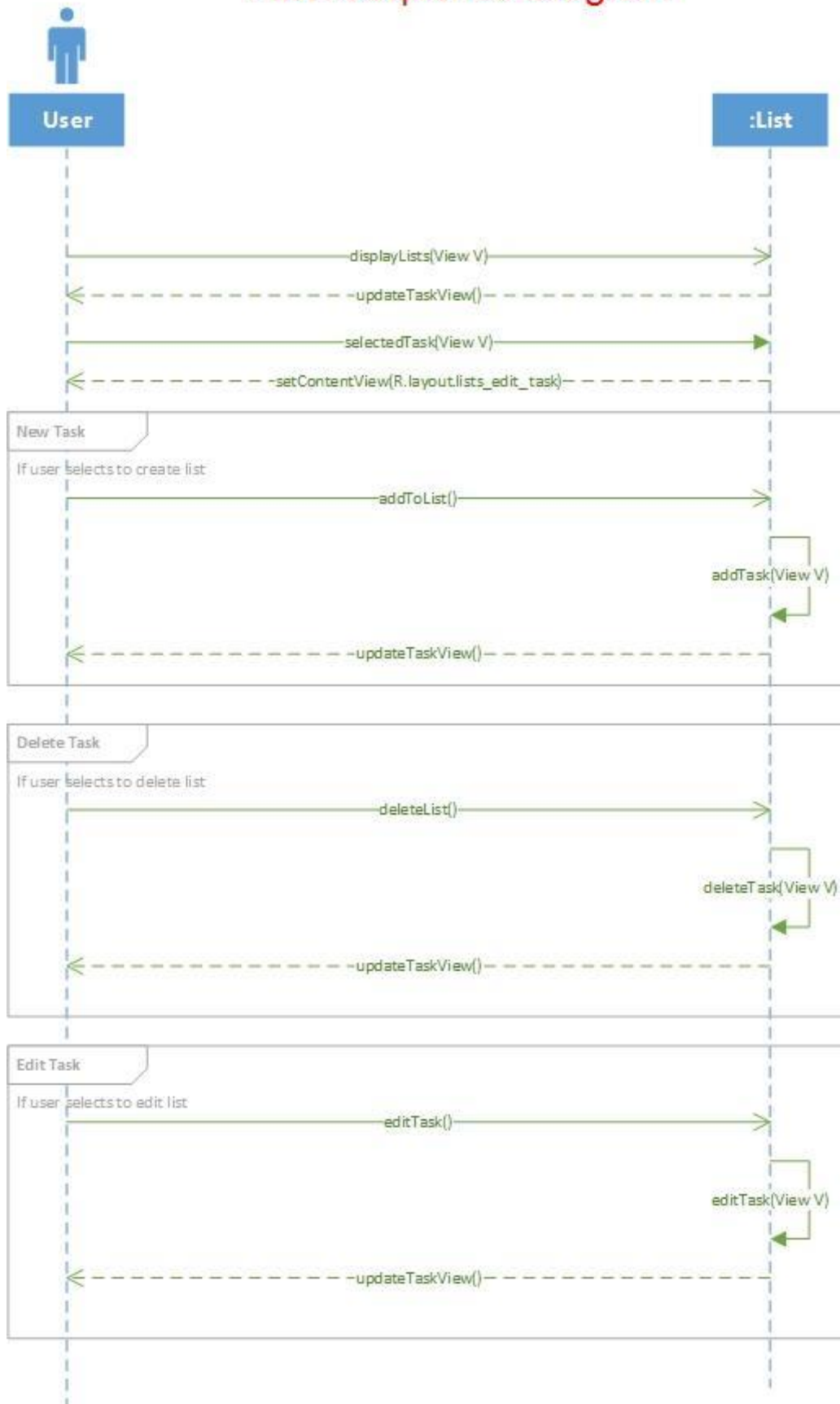
Session Sequence Diagram



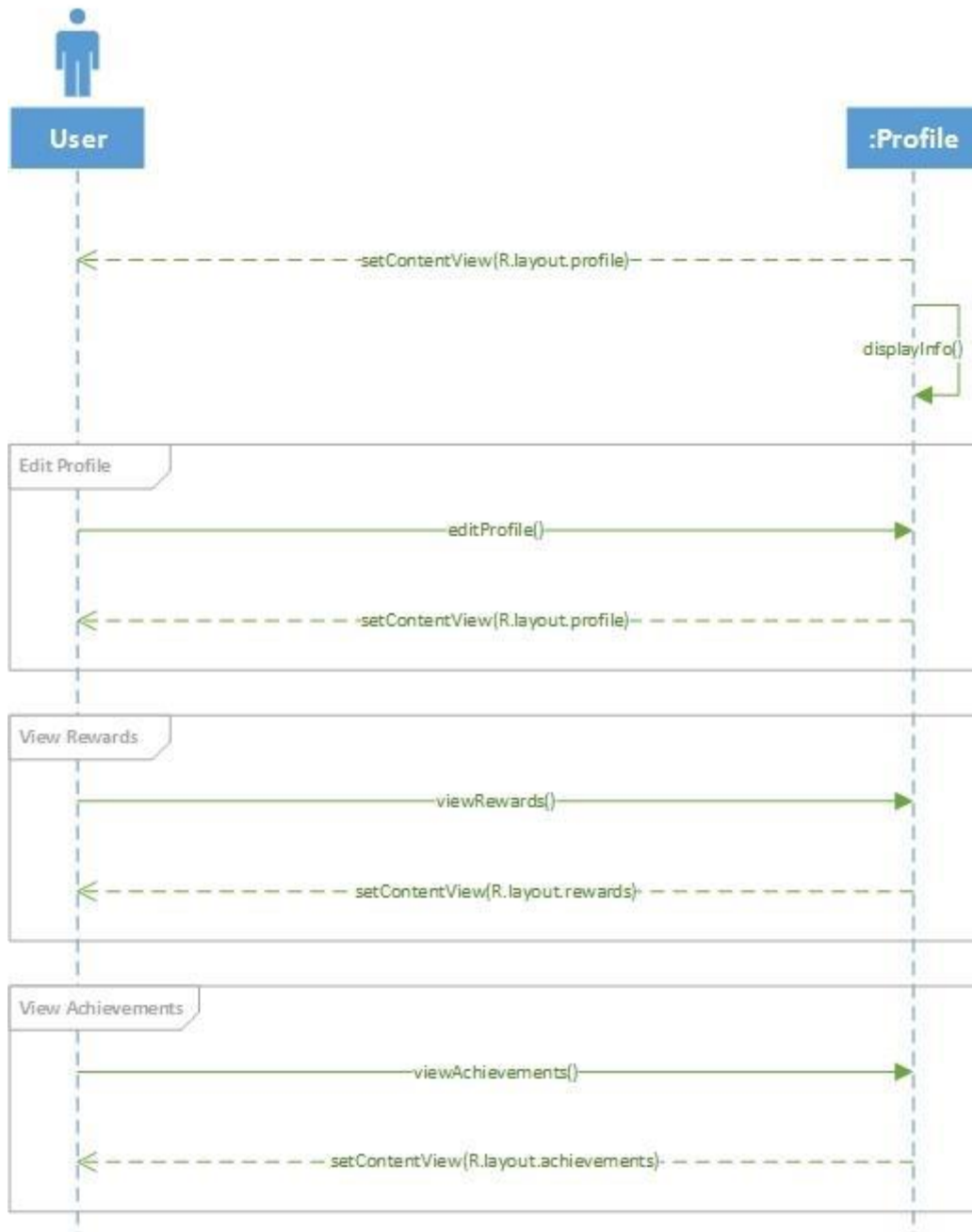
Spin Wheel Sequence Diagram



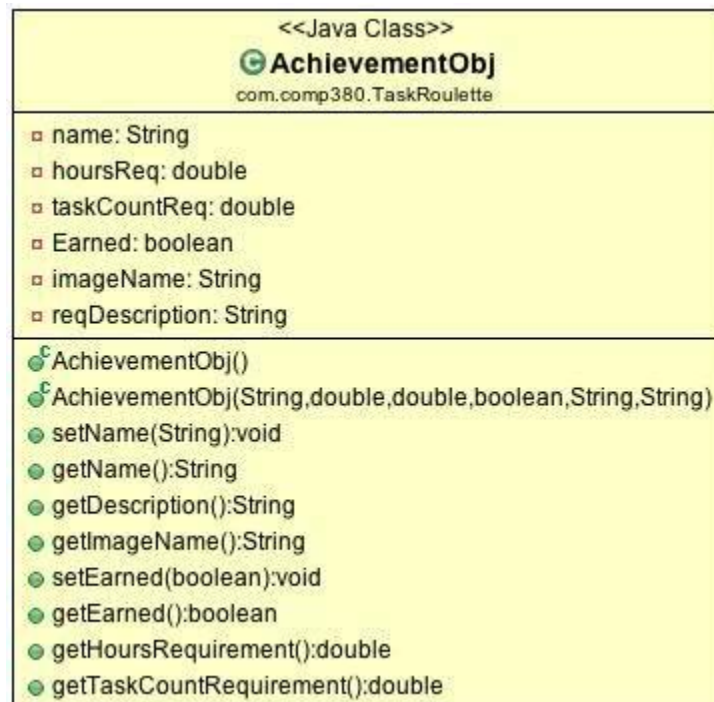
Task Sequence Diagram

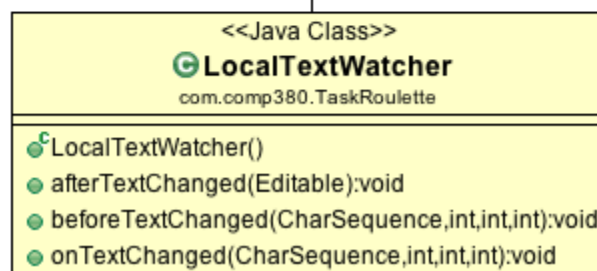
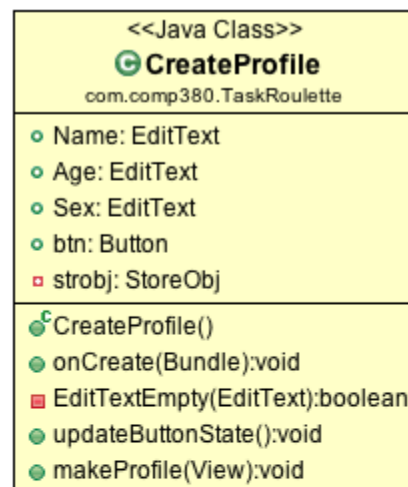
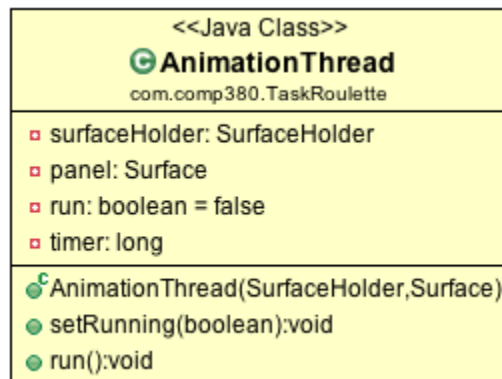
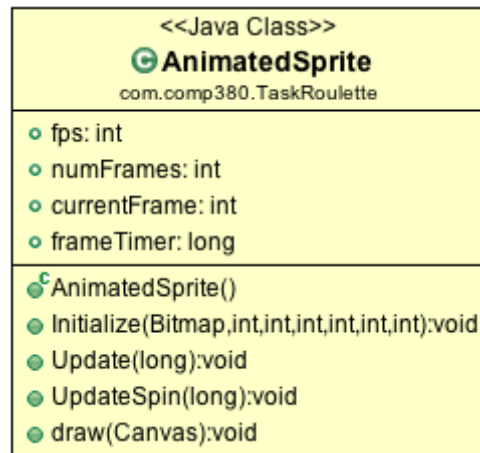


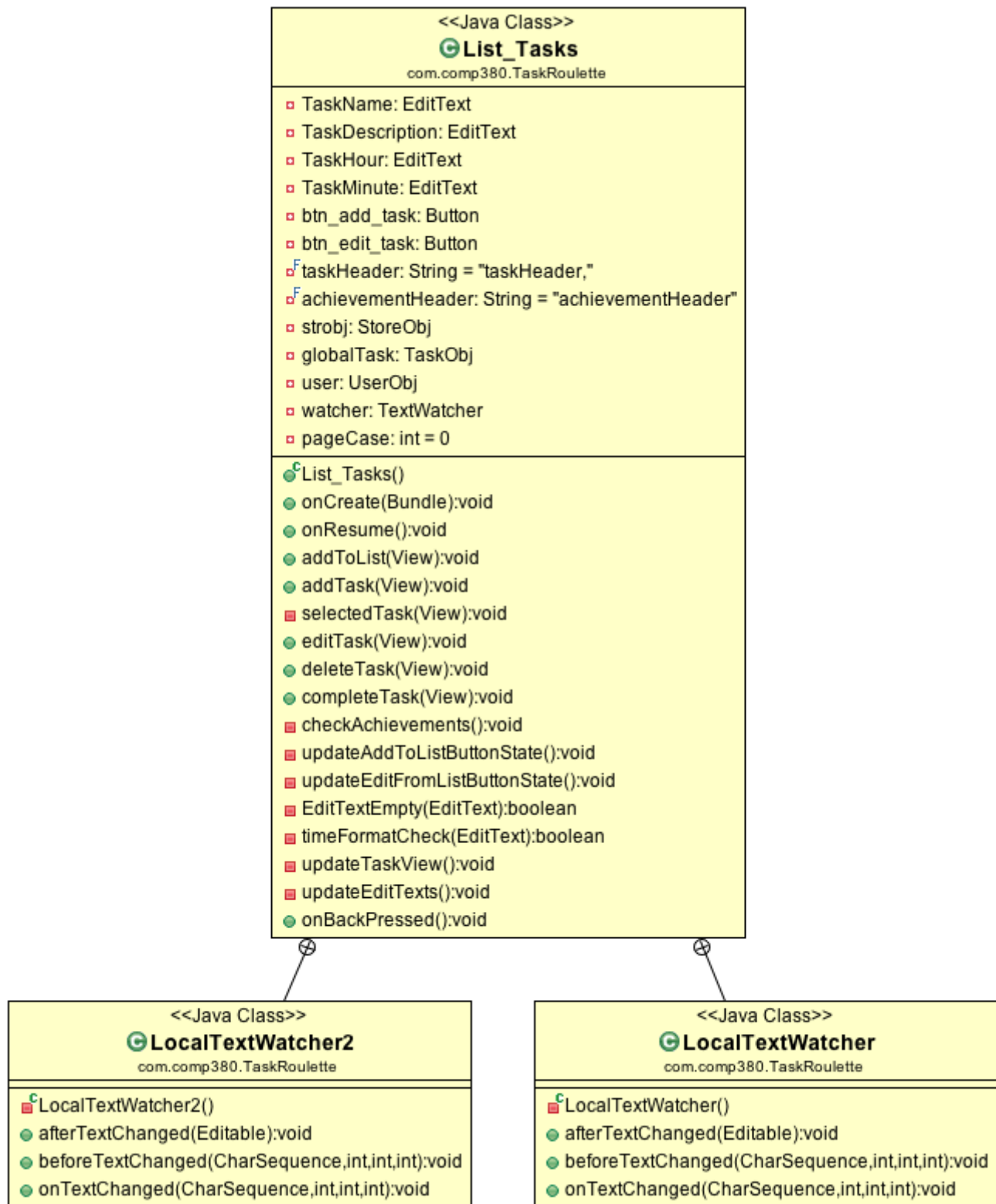
View Profile Sequence Diagram




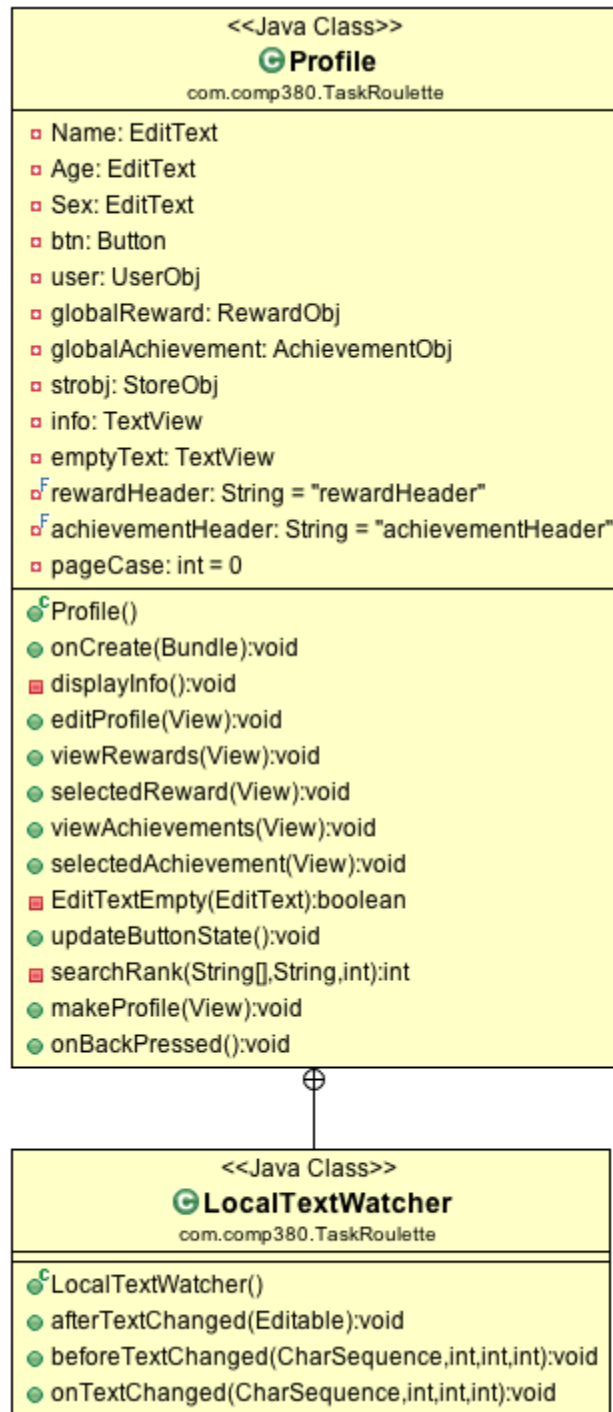
7. UML Class Diagrams























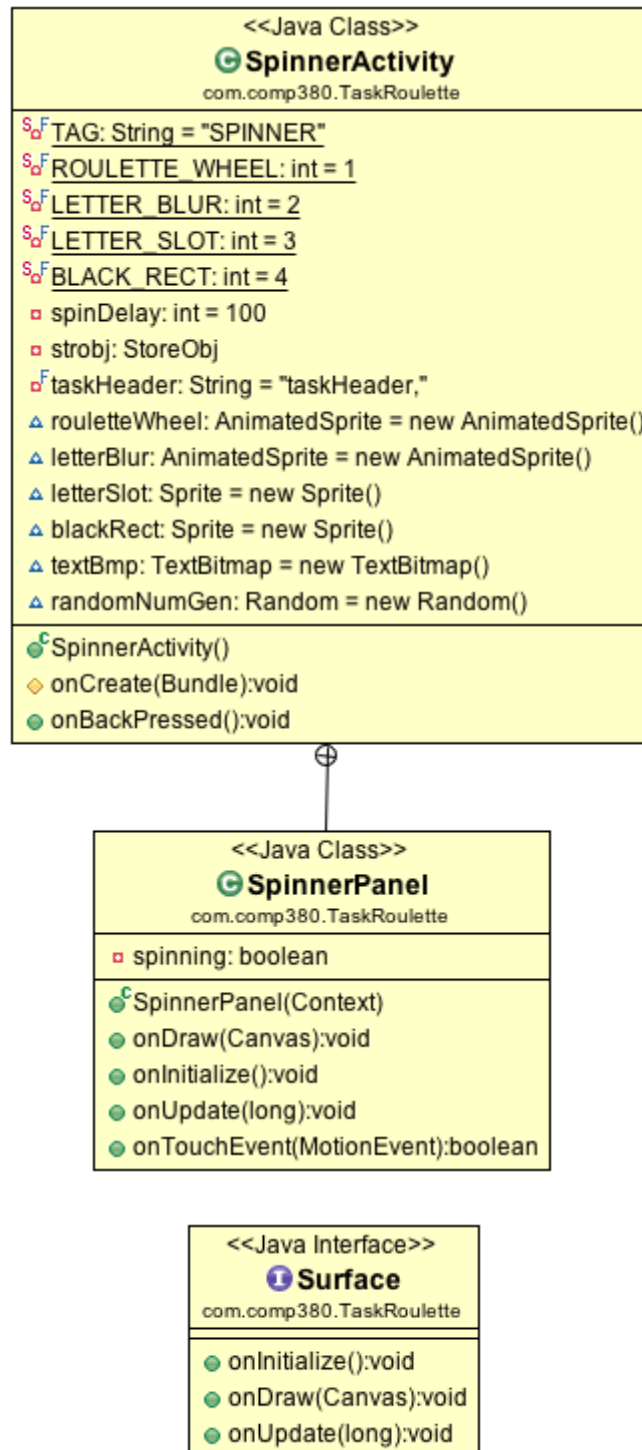


<<Java Class>>  Menu com.comp380.TaskRoulette
<ul style="list-style-type: none"> ▢ strobj: StoreObj ▢ ^FachievementFileName: String = "achievementList.txt" ▢ ^FrewardFileName: String = "rewardList.txt" ▢ ^FfileName: String = "StorageFile" ▢ doubleBackToMenuPressedOnce: boolean = false
<ul style="list-style-type: none"> ● ^CMenu() ● onCreate(Bundle):void ● onResume():void ● onStop():void ● displayUser(View):void ● displayLists(View):void ● displayRewards(View):void ● displayAchievements(View):void ● displaySpin(View):void ▢ readAchievements():void ▢ readRewards():void ● onBackPressed():void

























<<Java Class>>  RewardObj com.comp380.TaskRoulette
<ul style="list-style-type: none"> ▣ Name: String = "" ▣ Earned: boolean = false ▣ Cost: String = "" ▣ imageID: String = ""
<ul style="list-style-type: none">  RewardObj()  RewardObj(String,String,String,boolean)  getName():String  getEarned():boolean  setEarned(boolean):void  getCost():String  getImageID():String


<<Java Class>>  Rewards com.comp380.TaskRoulette
<ul style="list-style-type: none"> ▣ stobj: StoreObj ▣ globalReward: RewardObj  rewardHeader: String = "rewardHeader" ▣ Name: TextView ▣ Cost: TextView ▣ image: ImageView ▣ chips: TextView ▣ btn_buy: Button ▣ emptyText: TextView ▣ user: UserObj ▣ pageCase: int = 0
<ul style="list-style-type: none">  Rewards()  onCreate(Bundle):void  rewardView():void  searchRank(String[],String,int):int  selectedReward(View):void  buyReward(View):void  updateBuyButtonState():void  onBackPressed():void




<<Java Class>> TaskObj com.comp380.TaskRoulette
<ul style="list-style-type: none"> Name: String = "" Description: String = "" Hour: int = 0 Min: int = 0 TimeStamp: long
<ul style="list-style-type: none"> TaskObj() TaskObj(String,String,int,int) setName(String):void getName():String setDescription(String):void getDescription():String setTime(int,int):void getHour():int getMin():int getTimeStamp():long

<<Java Class>> Sprite com.comp380.TaskRoulette
<ul style="list-style-type: none"> ROULETTE_WHEEL: int = 1 LETTER_BLUR: int = 2 LETTER_SLOT: int = 3 BLACK_RECT: int = 4 image: Bitmap xPos: int yPos: int sRectangle: Rect spriteHeight: int spriteWidth: int scale: int imgType: int
<ul style="list-style-type: none"> Sprite() Initialize(Bitmap,int,int,int,int):void getXPos():int getYPos():int setXPos(int):void setYPos(int):void draw(Canvas):void

<<Java Class>>  StoreObj com.comp380.TaskRoulette
□ ^F fileName: String = "StorageFile" □ taskList: ArrayList<TaskObj> = new ArrayList<TaskObj>() □ USER: UserObj = null □ rewardList: ArrayList<RewardObj> = new ArrayList<RewardObj>() □ achievementList: ArrayList<AchievementObj> = new ArrayList<AchievementObj>() □ ^F userHeader: String = "userHeader," □ ^F taskHeader: String = "taskHeader," □ ^F rewardHeader: String = "rewardHeader" □ ^F achievementHeader: String = "achievementHeader" □ ^F endOfLine: String = "\u00B6"
 StoreObj()  storeObject(TaskObj):void  storeObject(UserObj):void  storeObject(RewardObj):void  storeObject(AchievementObj):void  removeObject(int,String):void  removeObject(TaskObj):boolean  removeObject(UserObj):boolean  removeObject(RewardObj):boolean  removeObject(AchievementObj):boolean  getUser():UserObj  getTask(int):TaskObj  getReward(int):RewardObj  getAchievement(int):AchievementObj  makeIDList(ArrayList):String  makeIDList():String  storeList():void  getList():void  getListName(String):String[]  getListName(String,boolean):String[]  spinnerMath(int,boolean,boolean):int

<<Java Class>>	
 TextBitmap com.comp380.TaskRoulette	
S	F TAG: String = "CORY" F charXDim: int = 11 F charWidth: int = 6 F charHeight: int = 10 F textFieldWidth: int = 120 F textFieldHeight: int = 12 F fontMap: Bitmap F text: String F symbol: char[] F width: int[] F xPos: int F yPos: int F srcRect: Rect F dstRect: Rect F spinSrcRect: Rect F spinDstRect: Rect F scale: int F totalWidth: int F spinWidth: int F iSpunDown: int F spinningSymbolIndex: int F doneSpinning: boolean F spinDelay: int F fpsDelay: int F numFrames: int F currentFrame: int F frameTimer: long
C	TextBitmap() Initialize(Bitmap,String,int):void setSpinning():void isDoneSpinning():boolean getXPos():int setText(String):void getYPos():int setXPos(int):void setYPos(int):void Update(long):void draw(Canvas):void

<<Java Class>>  UserObj com.comp380.TaskRoulette
<ul style="list-style-type: none"> ▣ Name: String = "" ▣ Age: String = "" ▣ Sex: String = "" ▣ TimeStamp: long ▣ chips: int = 0 ▣ hours: int = 0 ▣ minutes: int = 0 ▣ taskCount: int = 0 ▣ completedTaskCount: int = 0
<ul style="list-style-type: none"> ●^c UserObj() ●^c UserObj(String,String,String) ● setName(String):void ● getName():String ● setAge(String):void ● getAge():String ● setSex(String):void ● getSex():String ● getInfo():String ● getTimeStamp():long ● getHours():int ● addMinutes(int):void ● getChips():int ● addChips(int):void ● deleteChips(int):void ● incrementTaskCount():void ● getCompletedTaskCount():int ● incrementCompletedTaskCount():void