Object-Oriented

Software Engineering

Roomie

Analysis Report - Revised

Group 3K

Deniz Alkışlar

Ekinsu Bozdağ

Eliz Tekcan

Serhat Aras

Selen Haysal

Supervisor:

- *Bora Güngören*

1. Introduction

Roomie is a single player, story-based Android game that is designed to simulate the life of a university student. Story-based games are recently popular in the app industry. The game proceeds as the player makes choices while answering the prompted questions- each possible answer leads the player to another scenario. While navigating through these scenarios, the player might earn unexpected gifts or face with challenging situations.

At the very beginning of the game, the player is expected to create a character by choosing a username and a gender. Roomie offers its users an extraordinary story line and various unexpected endings. Each player is equipped with a ‘backpack’ where he can use/sell the items that he has collected throughout the game. A status bar is devised to keep the players engaged and competitive in the game, it displays four different labels of statuses: health, money, sociality and grades. The choices that the player makes or the circumstances that the player faces may decrease or increase the levels of these statuses. If the player manages to end the game successfully, the status levels will be used to display an end message for the player(e.g “Congratulations! You are an engineer at Apple now!”)

The basic idea of Roomie is influenced by the game which is named Episode, it can be found on app stores.

2. Overview

Roomie is an Android application which is a story-based game and it requires user interaction to proceed through the story-line. The choices that the player makes during the game changes the flow of the game and how the story shapes. The story will take place in different places such as school, flat, and cafe. Each of these places will provide more interesting aspects to the game’s storyline. For example, a player would spend money on coffee at the cafe or solve a quiz question at the school to earn academic points on his status bar. While moving along with the storyline, some decisions of the player may result in unexpected consequences such as an accident or a failing grade. During the game, the player will be subject to several tasks to complete in order to either gain prizes or increase status levels. For example, at the school, a quiz might be given to the player which would change the academic status. The status bar is to challenge the player about his future decisions, for example if the player has a failing health status, he might choose to spend his money on healthy food instead of being more social which will cause dilemmas for the player. At the end of the game, if the player handles to survive all accidents or other undesirable events, a simple foresight about his future will be displayed.

2.1 Gameplay

Roomie is an Android application which is designed for Android devices. The player is expected to use the touch-screen of his phone. The interactions that are done by the means of the touch-screen will be used in Roomie, meaning that the user interface will be displayed on the touch screen. The users can use their phones' built-in buttons to exit from the game, or change the volume of the sound. Besides, while playing the game, the sensors that are built in their phones will be used to understand if the users are engaging in completing the missions that are assigned. For instance, the players are required to keep silent for at least ten second while they are in the "Library" mission. Roomie will use Android's sensors to implement this kind of missions. Also the phone's' speakers will be used to play the game's sounds.

2.2 Game Flow

In Roomie, the storyline proceeds in a sequential manner. To complete the game successfully, the player should always be careful in many aspects such as money, status levels, and tasks.

The player has some money when he starts the journey, his decisions affect his amount of money. If the money amount is low, the game is more critical to play. Since expenses like rent and tuition are obligatory.

Also throughout the game, the player is supplied with some prizes which he can use later to boost his status levels. Also by selling these prizes, he can earn some money too. Earning these prizes is also a critical point in the game.

Lastly, there are various tasks that should be completed by the player. Failing to do so might decrease the status levels or cause an unexpected end.

2.3 Events

The events are the main component of Roomie. The player navigates through the storyline by answering the questions that the events prompt. Those answers lead the player to different scenarios. The events will be implemented in a graph data structure to have a stable storyline. There will be many ends to the game, the pathway of the player will determine how the game will end.

2.4 The Scenes

In Roomie, there are various scenes that the game takes place such as school, disco, cafe, and house. These places are to make the storyline more interesting and detailed. In each different scene, the story shapes according to the scene. For example, at the disco, the player meets new people and dances- faces with questions that are related to the scene. Also, at each different scene, the player is assigned a related task. For example, at the library, the player is expected to keep quiet for some time to pass the task.

2.5 Backpack

The player is equipped with a backpack which he will store the items that he earns during the game. Later on, he might use or sell these items for different purposes such as increasing status levels or earning money.

2.6 Status Levels

There are 4 different status bars in Roomie: grades, health, sociality, and money. Answering questions and completing tasks might change the levels of statuses. These levels are critical since they determine the end of the game and likelihood of unexpected events appearance. For example, if the health level is so low, some illness might kill the player or if the player finishes the game with a low level of grades, then he might be a drop-out at the end.

3. Requirement Specification

3.1 Functional Requirements

3.1.1 Play Game

Roomie starts with a screen that prompts the player to create a character, the player chooses a username and a gender. After creating the character, the game starts at the house and prompts the player the first question. The player answers the question and according to his answer, the storyline shapes and the player is asked another question. The game flows like this and while answering those question, the player may face different situations where he needs to complete a task or gains a prize as a result of his answer to a specific question. Those prizes are kept in a backpack where the player can use or sell these items. Besides, a status bar is devised to keep track of the player's decisions and the player's answers affect the levels of these status bars. The game may end in various ways, if the status levels of the player is not sufficient, it might end the game because the player does not have enough health level. Also, an unexpected event may occur and it might kill the player- like an earthquake. These sudden and accidental details are to keep the player excited and awake. If the player manages to finish the game without dying, then he is presented an end message which is based on the status levels that the player finished the game with. After finishing the game, the player can restart.

3.1.2 Volume Change / Mute

While playing, the player can change the volume of the sound of the game. This event can be done by clicking the speaker icon that is on the screen.

3.1.3 Pause the Game

The player might pause the game from the main menu, when the game is paused the player cannot interact with the game. If the player likes to resume, the game continues from the same point that he is left. The player might exit as well using the main menu if he does not like to resume.

3.1.4 Using Built-in Sensors

In the roomie, the user must interact with the phone sensors since the game ensures the usage of the sensors with the events which is designed specifically to serve this purpose.

3.1.5 Changing the Activities (View)

The activity flow of the Roomie will be based on the Activity concept of the Android API. Each of the main Activity (home environments, outdoor environments, and login) screens will be integrated and and maintained by the fragments of the specialized view. Therefore, the user will easily travel between views of the Roomie.

3.2 Non-functional Requirements

3.2.1 Reliability

The user should have an Android device since Roomie will run on that and the reliability of the system is dependent to the reliability of Android.

3.2.2 Availability

Since Android API 23 (Marshmallow) is going to be used in the project development, Roomie will be available to those with Android version 6.0 and above.

3.2.3 Maintainability

To keep our software's maintainability at a high level, we plan to keep our code clean and documented. We are planning to limit the number of external libraries used in the project. In addition to these, we are planning to avoid writing complicated code, and keep the code as simple as possible while being organized. Another approach we are going to take is testing the code in various devices to assure we do not encounter any problems.

3.2.4 Portability

The system is portable since it will run on any Android device with Android version 6.0 and above.

3.2.5 Extendibility

We plan to make Roomie suitable to be reused and extended for further versions and other works.

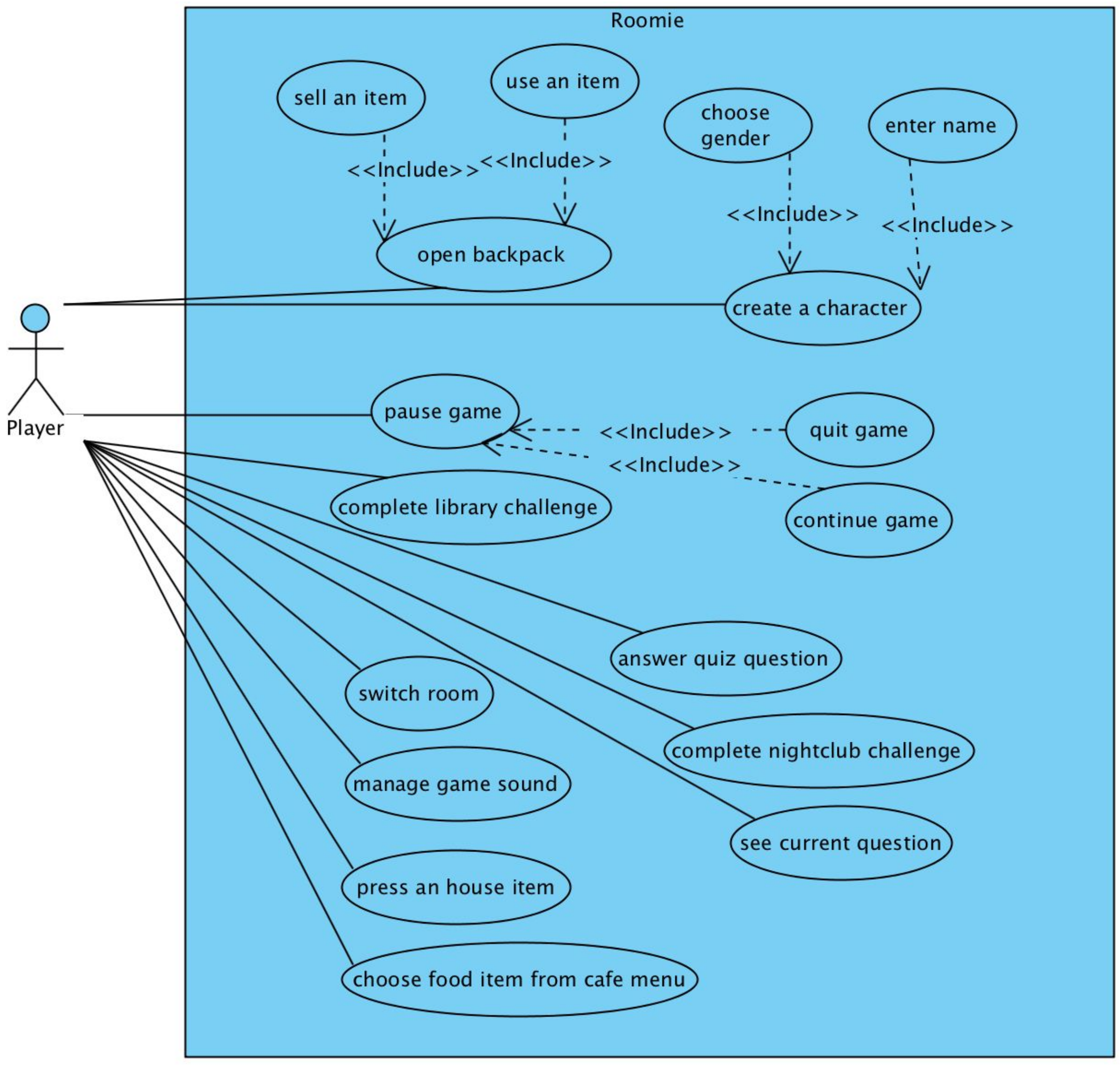
3.2.6 User-friendliness

We plan to build our application as user friendly as possible. Our first aim is to make the user interface easy to use. We plan to apply a minimal color range to our design. We will use icons to guide the users.

4. System Model

4.1 Use Case Model

In this section, the use cases of Roomie are demonstrated along with detailed information about each use case below.



4.1.1 Create a Character

Use Case Name: Create a Character

Primary Actor: Player

Stakeholders and Interests:

- Player wants to start a new game, for this a new character should be created first.
- System asks the player questions about the character that he wants to create.

Pre-conditions: The game is started.

Post-conditions: -

Entry Condition: Player selects “New Game” button from the main menu.

Exit Condition: Player selects “Back” button to return to the main menu.

Success Scenario Event Flow:

1. System displays character create layout, and the player is able to create his character. After creating, he is directed to the game.

Alternative Flows:

A. If player desires to return :

A.1. Player selects “Back” button to return to the main menu.

A.2. System displays the main menu.

4.1.2 Choose Gender

Use Case Name: Choose Gender

Primary Actor: Player

Stakeholders and Interests:

- Player wants to start a new game, for this a new character should be created first, inorder to do that choosing its gender is necessary.

Pre-conditions: The game is started.

Post-conditions: -

Entry Condition: Player selects “New Game” button from the main menu.

Exit Condition: Player selects “Back” button to return to the main menu.

Success Scenario Event Flow:

1. System displays character create layout, and the player is able to choose its gender.

Alternative Flows:

A. If player desires to return :

A.1. Player selects “Back” button to return to the main menu.

A.2. System displays the main menu.

4.1.3. Enter Name

Use Case Name: Enter Name

Primary Actor: Player

Stakeholders and Interests:

- Player wants to start a new game, for this a new character should be created first, inorder to do that entering its name is necessary.

Pre-conditions: The game is started.

Post-conditions: -

Entry Condition: Player selects “New Game” button from the main menu.

Exit Condition: Player selects “Back” button to return to the main menu.

Success Scenario Event Flow:

System displays character create layout, and the player is able to enter its name.

Alternative Flows:

A. If player desires to return :

A.1. Player selects “Back” button to return to the main menu.

A.2. System displays the main menu.

4.1.4. Press an House Item

Use Case Name: Press an House Item

Primary Actor: Player

Stakeholders and Interests:

- The house item pressed by the user represents the answer to the prompted question.
- Player is prompted questions and expected to answer those questions to move along the game flow.
- System changes the game setting according to the player’s answers.
- System keeps the status bar of the player updated based on the answers to the questions.

Pre-conditions: -

Post-conditions: -

Entry Condition: A character is created and the player pressed “start” button afterwards.

Exit Condition: Player selects “quit the game” button from the pause menu..

Success Scenario Event Flow:

1. A question is prompted to the player by the system.
2. The player answers the question (makes a decision by pressing an house item).
3. The game setting is updated based on the given answer.
4. If necessary, the status bar is updated based on the given answer.
5. New task or a new question is assigned to the player.

6. Same steps are repeated until the player makes the last decision before dying or completing all of the questions.

Alternative Flows:

A. If the player desires to pause:

A.1. The player selects “pause” button to pause the game.

A.2. Exit menu is displayed.

A.3. The player either selects quit or returns back to the game.

B. If the player collects items from his/her answers:

B.1. The player presses an house item.

B.2 His/her decision brings him/her an item.

B.3 The item is stored in the backpack.

4.1.5. Switch Room

Use Case Name: Switch Room

Primary Actor: Player

Stakeholders and Interests:

- The player wants to change the room that he/she is currenty in.

Pre-conditions: The starting room is the living room of the house at the beginning.

Post-conditions: The room is changed.

Entry Condition: The player selects the name of the room that he wants to switch to.

Exit Condition: -

Success Scenario Event Flow:

1. The player touches the arrows that leads the room to change.
2. The game setting is changed, the interface is updated.
3. The steps that are above could be repeated to switch between other rooms.

4.1.6. Complete Night Club Challenge

Use Case Name: Complete Night Club Challenge

Primary Actor: Player

Stakeholders and Interests:

- The player is prompted a night club task previously which requires him/her to shake his/her phone.

- The player wants to complete the assignment.

Pre-conditions: A task is assigned to the player.

Post-conditions: -

Entry Condition: The player accepts to complete the task.

Exit Condition: The player cancels the task.

Success Scenario Event Flow:

1. The player shakes his/her phone to complete the task.
2. The task is completed.
3. The status bar is updated.
4. The backpack is updated with the new item that the player has gained.

4.1.7. Complete Library Challenge

Use Case Name: Complete Library Challenge

Primary Actor: Player

Stakeholders and Interests:

- The player is prompted a library task previously which requires him/her to use his/her phone’s microphone.
- The player wants to complete the assignment.

Pre-conditions: A task is assigned to the player.

Post-conditions: -

Entry Condition: The player accepts to complete the task.

Exit Condition: The player cancels the task.

Success Scenario Event Flow:

1. The player uses his/her phone’s microphone to complete the task.
2. The task is completed.
3. The status bar is updated.
4. The backpack is updated with the new item that the player has gained.

4.1.8. Open Backpack

Use Case Name: Open Backpack

Primary Actor: Player

Stakeholders and Interests:

- The player wants to see the items that he/she has been collecting during the game.
- The backpack collection is displayed to the player.

Pre-conditions: The backpack is empty at the beginning, and gets updated throughout the game.

Post-conditions: -

Entry Condition: The player selects the backpack button.

Exit Condition: The player touches on the backpack button again to close the backpack layout.

Success Scenario Event Flow:

1. The player selects the backpack button to display his/her backpack.
2. The backpack collection is displayed by the System.

Alternative Flows:

A. If the player wants to return to the game:

A.1. The player selects the backpack button again.

A.2. The backpack display closes.

4.1.9. Sell an Item

Use Case Name: Sell an Item

Primary Actor: Player

Stakeholders and Interests:

- The player wants to sell an item from his/her backpack.
- After the player sells an item, the item is removed from the list.
- The money bar of the player is updated with money that he/she earned.

Pre-conditions: There is at least one item to be sold.

Post-conditions: The selected item is removed.

Entry Condition: The player selects the backpack button first, and then selects the sell item button on the item that he/she wants to sell.

Exit Condition: After selling the item, the player can close the backpack display.

Success Scenario Event Flow:

1. The player selects the backpack button to display his/her backpack.
2. The backpack collection is displayed by the System.
3. The player selects the sell item button.

4. The item is removed from the collection.
5. The money bar of the player is increased by the value of the item.

Alternative Flows:

A. If the player wants to return to the game:

A.1. The player selects the backpack button again.

A.2. The backpack display closes.

4.1.10 Use an Item

Use Case Name: Use an Item

Primary Actor: Player

Stakeholders and Interests:

- The player wants to use the items that he/she has been collecting during the game.
- The backpack collection is displayed to the player and the item is marked as used.

Pre-conditions: There is at least one item to be used.

Post-conditions: The item that is selected is marked as used.

Entry Condition: The player selects the backpack button first, and then selects the use item button on the item that he/she wants to use.

Exit Condition: After using the item, the player can close the backpack display.

Success Scenario Event Flow:

1. The player selects the backpack button to display his/her backpack.
2. The backpack collection is displayed by the System.
3. The player selects the use item button.
4. The item is marked as used.
5. The status bar of the player is updated based on the qualities of the used item.

Alternative Flows:

A. If the player wants to return to the game:

A.1. The player selects the backpack button again.

A.2. The backpack display closes.

4.1.11 Manage Sound

Use Case Name: Manage Sound

Primary Actor: Player

Stakeholders and Interests:

- The player wants to mute/play the sound.

Pre-conditions: The sound is already muted/played.

Post-conditions: The sound becomes muted/played.

Entry Condition: The player selects the audio button that is on the right-up corner.

Exit Condition: -

Success Scenario Event Flow:

1. The player touches on the audio button to change the sound status.

4.1.12 Pause Game

Use Case Name: Pause Game

Primary Actor: Player

Stakeholders and Interests:

- The player wants to pause the game.

Pre-conditions: -

Post-conditions: The game is paused.

Entry Condition: The player touches the pause button.

Exit Condition: -

4.1.13 Quit Game

Use Case Name: Quit Game

Primary Actor: Player

Stakeholders and Interests:

- The player wants to quit the game.

Pre-conditions: -

Post-conditions: The game is quited.

Entry Condition: The player exits the application or touches the exit button.

Exit Condition: -

Success Scenario Event Flow:

The player touches on the exit button or exits the application.

4.1.14 Continue Game

Use Case Name: Continue Game

Primary Actor: Player

Stakeholders and Interests:

- The player wants to continue the game.

Pre-conditions: -

Post-conditions: The game is continued.

Entry Condition: The player enters the application or touches the continue button after pausing.

Exit Condition: -

Success Scenario Event Flow:

The player enters the application or touches the continue button after pausing.

4.1.15 See Current Question

Use Case Name: See Current Question

Primary Actor: Player

Stakeholders and Interests:

- The player wants to see the current question.

Pre-conditions: -

Post-conditions: The user is prompted an event.

Entry Condition: The player touches the image on the left top corner.

Exit Condition: -

Success Scenario Event Flow:

The player touches the image on the left top corner.

4.2 Dynamic Models

4.2.1 Sequence Diagram

Scenario: The player starts new game by entering the characters name and gender, after the new game is initialized the player is prompted a question wheter he/she wants to go to the library to study for midterms or stay at home and join their friends who are playing video games. The player touches the house item to play video games. This causes his stats points to be affected, and the game refreshes the game environment to the library and the stats bar. After playing video games the player decides to use the band aid in his/her backpack. After using it, his/her stats are updated. Then by the user decides to take a break and pauses the game, and decides to continue in 5 minutes. He/she decides the mute the game.

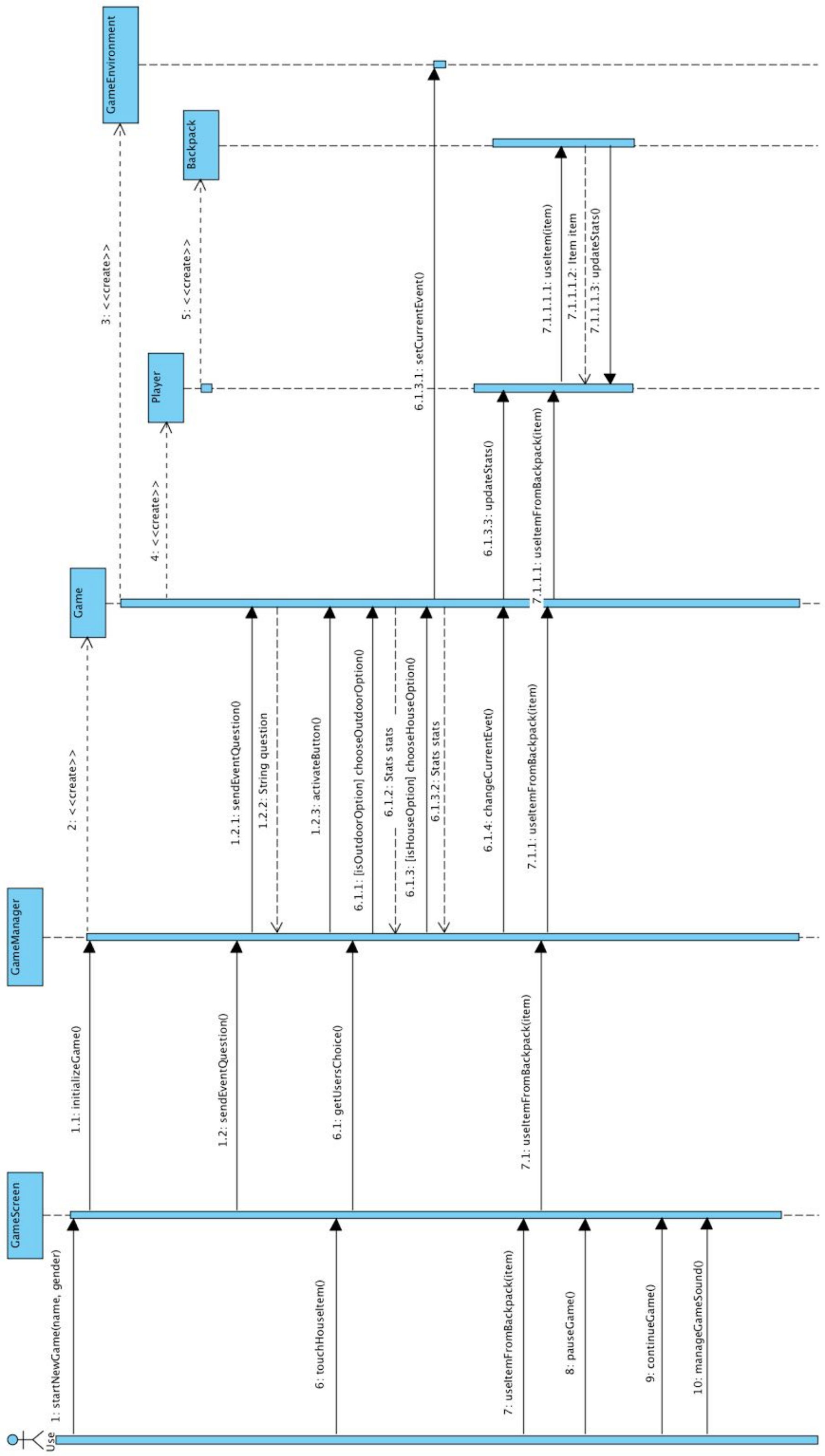


Figure: Illustrates the Sequence Diagram for Roomie

4.2.2 Activity Diagram

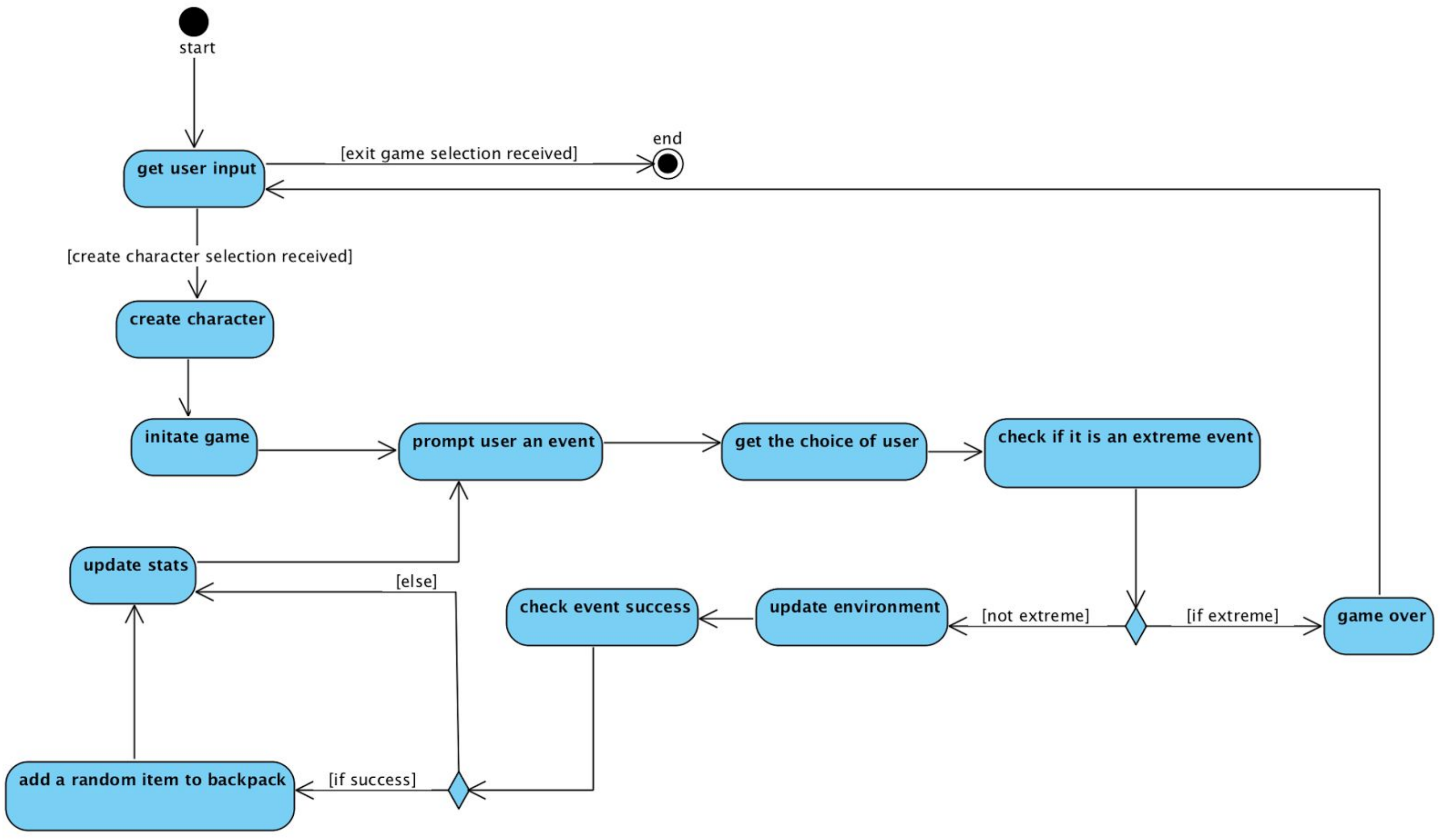


Figure: Illustrates the Activity Diagram for Roomie

Description of the Activity Diagram

When user enters his/her name and presses “Start Game” button from the login screen, the game starts. It shows user an event by using a dialog. Then the game gets input from the user with the image buttons. Pressed button represents the choosen option. The game first checks the choosen option to see if it is extreme. If it is an extreme option, the game is over. Unless it is an extreme option, the game checks the event’s success and update the stats of the player. If the event is successful a random item is added to the player’s backpack. The game creates a new event and shows it to the player.

4.3 Class and Object Diagram

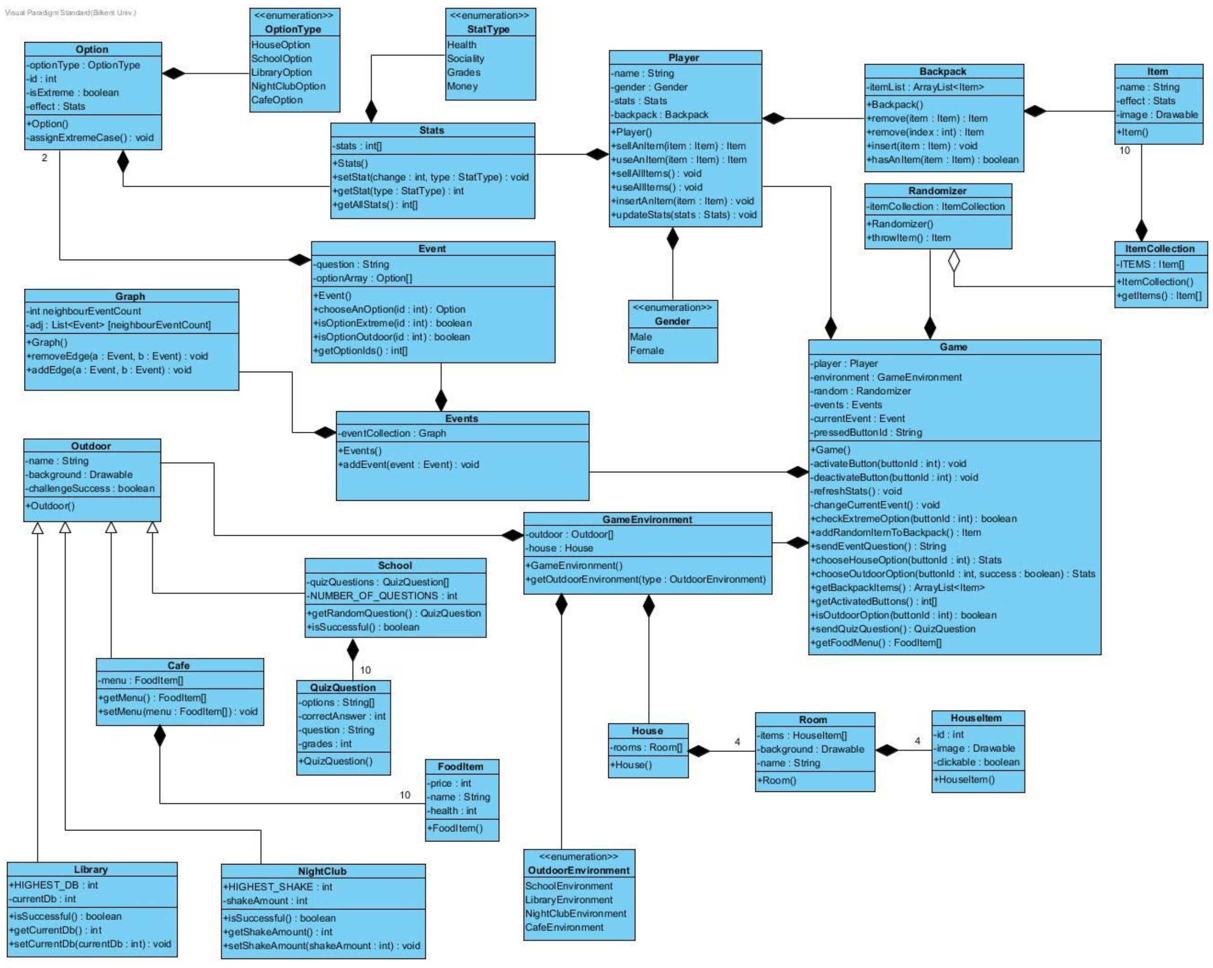


Figure: Illustrates the Class Diagram for Roomie

Description of the Class Diagram

Game Class is the main class of Roomie that interacts with other classes to maintain the whole game logic. **Event Class** is to implement the questions that the players will answer to iterate through the game. **Events Class** is to keep a collection of event objects in a graph structure. This graph structure represents the game flow, the player will choose a path inside this graph by answering questions.

Player Class is to implement the functionalities and the properties of the players. The player has a backpack and stats bar. **Backpack Class** contains an ArrayList of items. **Item Class**

implements the items that the player will store in his/her backpack, items have boostPoints and a price. The player may sell their items to gain money or boost their stats.

GameEnvironment Class is about setting the environment that the game takes place. It can be either set from **Outdoor Class** or **House Class. Outdoor Class** is extended by outdoor place classes such as **Library Class**, **NightClub Class** and **Cafe Class. House Class** implements the house environment which is composed of rooms that are implemented by **Room Class.**

5. User Interface

5.1 Navigational Path

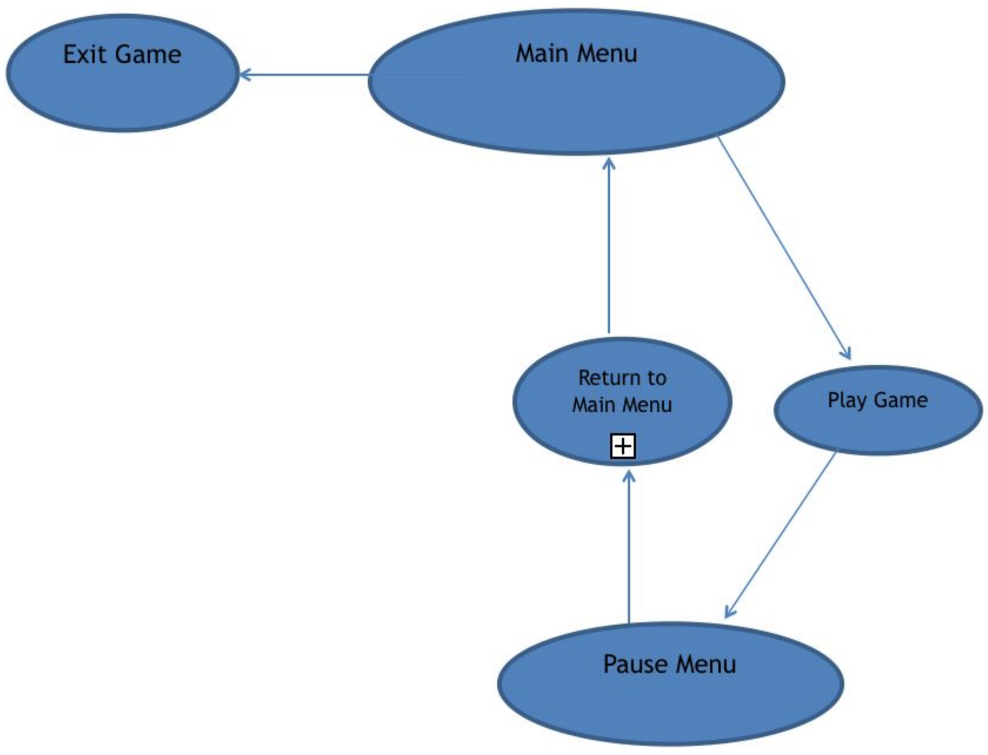


Figure: Illustrates the Navigational Path of Roomie

5.2 Screen Mock-ups

Roomie consists of two main activities. One of the activities is the welcome activity. Welcome activity is shown on the first launch of Roomie. Welcome activity has a text input which asks name of the user. It also has radio buttons to take gender of the user.

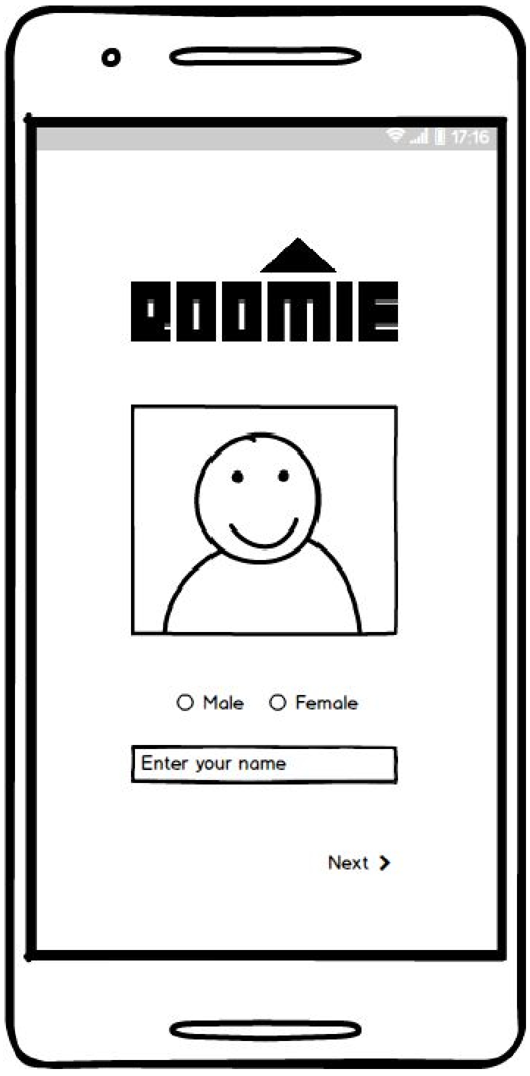


Figure: A Mock-up of the scene where the user creates his character (Welcome Activity)

The other activity is main activity. Main activity is where Roomie proceeds. The screen has three parts which are top, middle and bottom. On the left top there is a picture of the player. It is also a button which shows the current question as a dialog when it is clicked. On the right of the player's picture there is a button for showing items in the player's backpack. On the right top, there is a button to mute sound of Roomie. Middle part of the screen shows rooms of the player's house. It shows one room at a time and the user could switch between rooms by using buttons at below. Finally, on the bottom part there are four indicators which show player's current status.

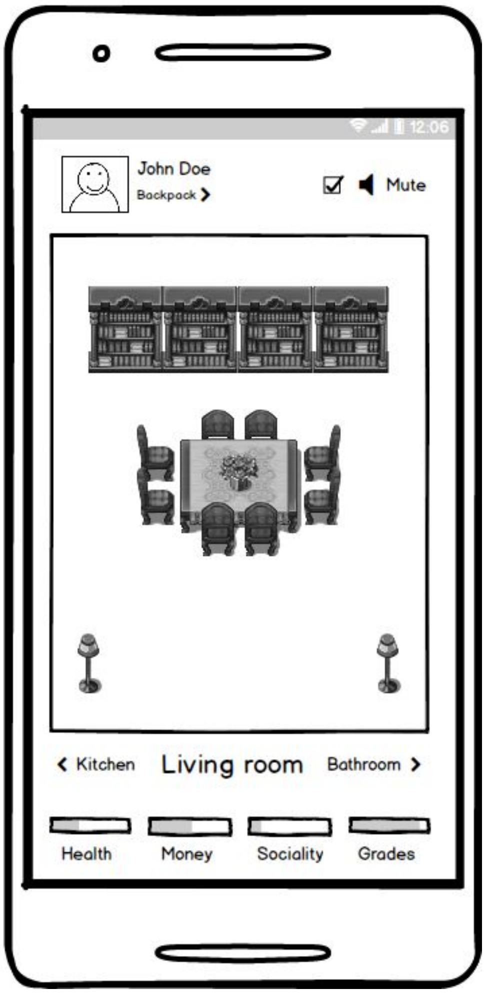


Figure: A Mock-up screening the main game scene, the Student Flat

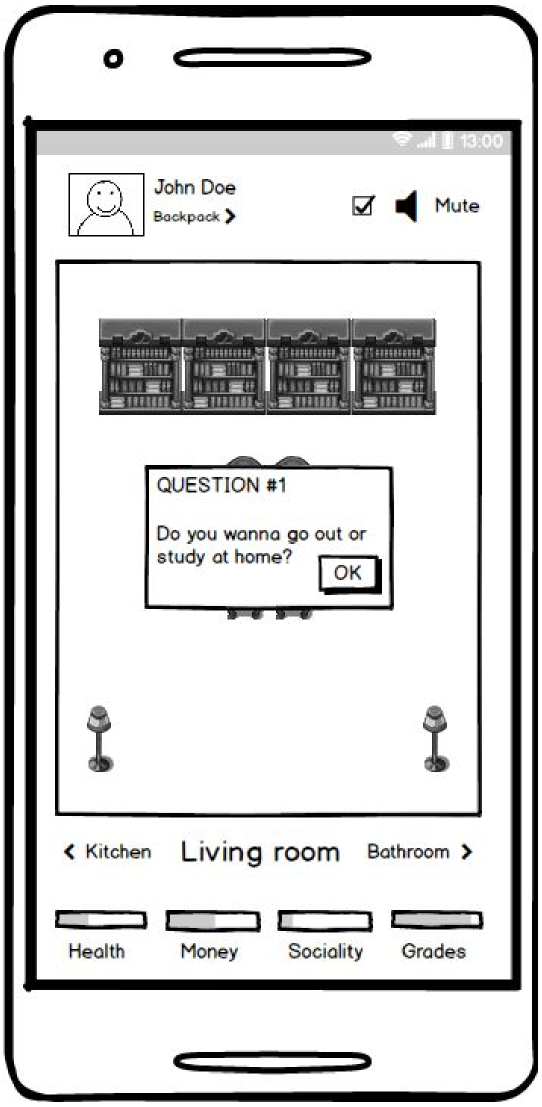


Figure: A Mock-up screening a question that is prompted

The questions are shown as dialog box. The user is going to click items in the rooms to answer the questions. For instance, if one question is “do you want to go out or study at home?” the user is going to click door for going out and click desk for studying.

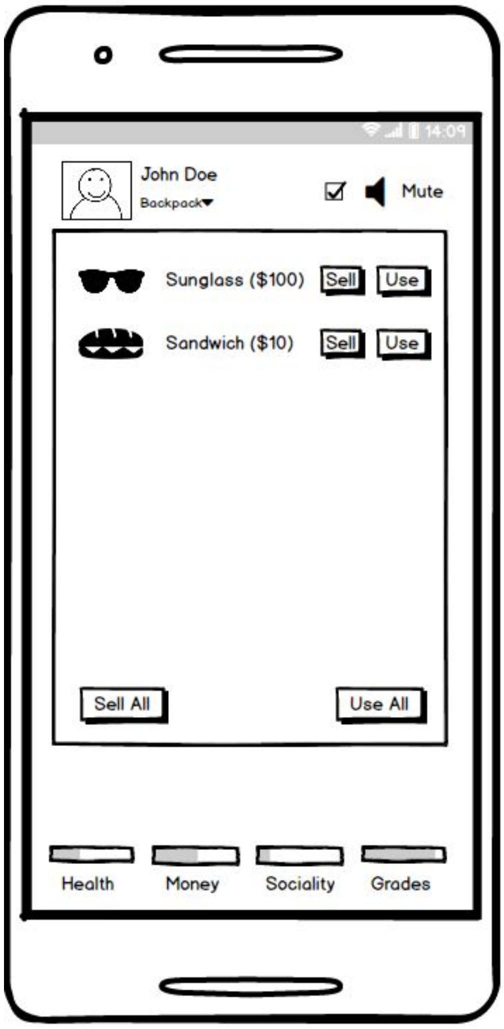


Figure: A Mock-up screening the ‘backpack’ of the player

Backpack of the player is shown when the user clicks the button on the top. Items in the backpack will be shown as a list. The user could use or sell a specific item from the list. If the player uses an item, indicators will be changed. If the player sells an item, money indicator will be increased. Using or selling all the items at once is also possible with the buttons at below.

Roomie is going to start on an Android screen with its initial boot page . The prior input to Roomie will be the touch motion of the user which is listened by the screen to initiate actions. Additional inputs will be used for required cases, for instance the gravitational motion which readed by motion sensors of the phone will be used in cases like shaking the phone to dance or collection sound data to complete missions that contains sound inputs. The sensors will be activated only when it is required. Therefore, all sensors will not be active fully during the play time.

6.Conclusion

In this report, the aim was to analyze through our story-based game, Roomie. This analysis report is composed of two major parts which are the requirement specification and the system design.

In the first part of this report, Roomie and its core components were explained in detail. The logic of Roomie, and the elements such as the questions and the backpack were described.

The functional requirements and the non-functional requirements of Roomie were revealed later in the report. These were the requirements that we think that players will need and utilize during their use of Roomie. In our design and implementation, we plan to adapt to these requirements as much as possible to satisfy our players and their possible needs.

In the system design, there were four main focuses including use case model, dynamic models, class model and user-interface. We tried to be as specific as possible in our system design since we know that these models will lead us in the implementation part. We designed mock-ups to specify Roomie’s user-interface.

7.References

[1]-Motion Sensors. (2017, August 08). Retrieved October 07, 2017, from https://developer.android.com/guide/topics/sensors/sensors_motion.html

[2]-AudioRecord. (2017, July 24). Retrieved October 07, 2017, from https://developer.android.com/reference/android/media/AudioRecord.html

[3]-Android 6.0 APIs. (2016, October 07). Retrieved October 07, 2017, from https://developer.android.com/about/versions/marshmallow/android-6.0.html

[4]-The Activity Lifecycle (2017, September 26). Retrieved October 07, 2017, from https://developer.android.com/guide/components/activities/activity-lifecycle.html

[5]- Activity. (2017, July 24). Retrieved October 07, 2017, from https://developer.android.com/reference/android/app/Activity.html