
Bilkent University

Department of Computer Engineering

# CS 319
# Object Oriented Software Engineering

## *RUSH HOUR CC*

## Analysis Report

Fatbardh Feta, Masna Ahmed, Naisila Puka, Kunduz Efronova, Talha Zeeshan
Supervisor: Eray Tüzün

Analysis Report
October 21, 2018

# Contents

*1 Introduction* _____ *1*

*2 Proposed System* _____ *1*

**1.1 Overview** _____ **1**

**1.2 Functional Requirements** _____ **2**

**1.3 Non-functional Requirements** _____ **5**

**1.4 Pseudo Requirements** _____ **5**

**1.5 System Models** _____ **6**

1.1.1 Use-Case Scenarios and Model _____ 6
1.1.2 Object and Class Model _____ 12
1.1.3 Dynamic Models _____ 14
1.1.4 User Interface _____ 18

*3 Conclusion* _____ *24*

*4 References* _____ *25*

# Analysis Report

## *RUSH HOUR CC*

## 1 Introduction

RUSH HOUR CC (Rush Hour by Chain Coders) is a software implementation of the famous sliding block puzzle game Rush Hour [1], by the group of students called “Chain Coders”. We chose this famous board game among all the others because we believe Rush Hour is a good example to start with the basics of object-oriented software engineering.

In this classic remake of Rush Hour, player still has to get one main car out by maneuvering all the other cars out of its way, but we also added some additional features to make it more interesting. Player will have to think hard and plan! The score for each level and challenge will be determined by the number of moves and the time player takes to solve the puzzle and free the car. Game experience will be pleasing and the user will be motivated with additional bonuses like stars, crowns and coins.

This report contains general outline of our project including overview, functional and non-functional requirements, use case, sequence, class, activity and state diagrams. Following that there are mock-ups of user interface giving the taste of a game.

## 2 Proposed System

### 1.1 Overview

We will implement a Desktop based Java application to simulate Rush Hour in a software version. Briefly, the main properties of this version are:

- This is a personalized game: the user has his own identifier name and avatar, along with his achievements in this game.
- The user has a total set of 40 Rush Hour puzzles to solve, 10 for each of the following levels: beginner, intermediate, hard and advanced.
- Each Rush Hour puzzle grid consists of:
    1. A square board grid with 36 small squares, which is supposed to represent a traffic jam. The theme of the grid is chosen by the user in the settings, as will be explained in the other sections.
    2. A horizontal group of 6 consecutive squares, with the user’s car standing in the 2 leftmost squares, and the exit gate of the parking lot in the right of the rightmost square. The user should free this horizontal line of squares in order to arrive to the exit gate.

3. A configuration of a number of other cars in the grid which occupy squares, among which also the squares in (2.). Each car may occupy a horizontal/vertical group of 2/3 consecutive squares.
4. Each car may move either vertically forward/backward or horizontally forward/backward, depending on the car's direction in the grid. The user's car is always horizontally directed as in (2.). The user should move the cars in the grid accordingly in order to free the way for his/her car to go to exit gate. This is how the puzzle is solved.

- Once the puzzle is solved, the user wins a fixed amount of coins. Each puzzle has its minimum number of moves of cars in which it can be solved. The user may receive rewards in the form of stars if the number of moves in which he/she solves the puzzle is close to this minimum number. The stars are recognized as one way to represent player's achievements. They serve as multipliers for the coins.
- Along with the minimum number of moves, each puzzle has a measured time in which a good player can finish that puzzle. This time is decided by the Chain Coders. The user may receive rewards in the form of crowns if the time in which he/she solves the puzzle is close to the measured time. The crowns are recognized as another way to represent player's achievements. They serve as bonus for coins.
- The user will have a personalized garage with unlocked and locked cars. The user can unlock new cars based on the amount of coins that he has.
- The user will be able to change some game settings and look at game's instructions at any time. These will be explained in detail in the following sections.
- This is a single-player game: it is the user versus the puzzle he/she has to solve.
- There is no "lose" condition. In theory, the user may play infinite amount of time until the puzzle is solved. There is no move/time limit that may lead to losing the game.

## 1.2 Functional Requirements

We will provide the following functionalities to the game:

### Registration

As soon as the user downloads the game and opens it for the first time, a registration window will appear. Here the user will personalize the game by choosing a name and an avatar. From this point on, the game will save this

player's progress and information and the registration window won't appear again (unless the user decides to use another of our game functionalities described below: reset the game in the settings).

### Play

After registration, the user will be directed to game's main menu where he will see the option of playing the game. This will provide two consecutive functionalities:

- **Choose Level**

  The user can choose between four levels of difficulty, based on whether they are unlocked: Beginner, Intermediate, Hard and Expert level. The levels are unlocked based on the number of stars that the user has won.

- **Choose Puzzle and Start Game**

  Once the level is chosen, the user can choose between 10 different puzzles of that difficulty level. Once the user chooses a puzzle, the game directly displays the appropriate square board grid with a specific car configuration: the game has started! While playing, we provide the following functionalities to the user:

  - ❖ **Move a Car**

    The user can move a horizontal car left or right, and a vertical car up or down. This is done by drag-and-dropping it. The cars can be moved only in unoccupied squares (the user cannot put a car on top of another).

  - ❖ **Reset Puzzle**

    The user can reset the puzzle at any time while he is playing. This will put the grid in the original car configuration, and will reset the number of moves and the time.

  - ❖ **Leave Puzzle**

    The user may quit the game he is playing by pressing Back button in the screen. This will erase user progress in that specific puzzle: the next time he chooses to play this puzzle he will have to start over.

### Customize

While in main menu of the game, the user may go to Customize option. One of the things the user can do here is check his garage:

- **Go to My Garage**
  - ❖ **See Achievements**

    The user will be able to see the amount of coins, stars and crowns that he has. In general, coins represent finished puzzles, stars represent outstanding performance in terms of the number of

moves, and crowns represent outstanding performance in terms of the time it took for the user to finish a certain puzzle.

❖ **Change Car**

Among the unlocked cars, the user can choose the main car (the one he will free out of the traffic jam). By default, the user has a red Ferrari (I know right!).

❖ **Unlock Car**

The user can unlock new cars if he has enough coins! Each car costs a specific amount of coins to unlock.

At Customize we also provide the functionality of being able to change the following settings:

- **Change Mouse Sensitivity**

The user will change this by updating the integer for mouse sensitivity. Mouse sensitivity is the speed at which the mouse pointer moves on the screen. Increasing it makes the mouse pointer move faster and vice versa.

- **Change Background Music**

The user will be able to change the music on the background. The default one is Eminem – Till I Collapse.

- **Change Music Volume**

The user will be able to change music volume by updating a slider.

- **Change Theme**

The user will be able to change the theme of the grids between different colors and backgrounds.

- **Change Name and Avatar**

The user can change the avatar and the name he has put in registration anytime he wants.

- **Reset Game**

The user may choose to reset everything and start over from the registration window! This button will erase all information about user's progress and the game will start assuming it was launched for the first time.

## Instructions

While in the main menu or solving a puzzle, the user may choose to open instructions and the game will provide him with a set of rules, as well as a video tutorial on how to play the game.

## Quit

The user may quit the game by pressing the Quit button on the main menu or by pressing the usual window quit.

## 1.3 Non-functional Requirements

### Usability Requirements

The software must provide an interface that is user-friendly. The following characteristics of the software will achieve this:

- Distinct Buttons
- UI layout this both well-arranged and consistent
- Aesthetically pleasing

### Reliability Requirements

The software must not crash in the event of an incorrect input. Possible incorrect inputs that will be checked include:

- Unlocking a car that the user cannot afford with the current number of coins he/she possesses
- Moving a car outside the game grid or in occupied squares
- Trying to move a vertically oriented car horizontally or a horizontally oriented car vertically
- Trying to access puzzles that are yet to be unlocked

### Performance Requirements

The software must meet certain performance goals to provide the user with a smooth experience of the game:

- The game must have a refresh rate of at least 60 times per second to allow an aesthetically smooth experience
- The visual elements of the game must be responsive not exceeding a response time of 2 seconds
- Text files must be used to allow the game to take as little storage space as possible

### Supportability Requirements

- The software must be available to the user as a single executable file.
- The software must be compatible with all major operating systems and must be runnable on their recent previous editions.
- The basis of the executable file must be a Java Archive (JAR) format.

## 1.4 Pseudo Requirements

The Software will be developed using the Java Programming Language.
Stock images will be used for the in-game car models.
The software shall be distributed as a single executable file in a Java Archive format (JAR).

## 1.5 System Models

### 1.1.1 Use-Case Scenarios and Model Scenarios

**1) Playing the game**

*Use Case Name:* Play

*Participating actors:* The Player

*Flow of Events:*

- The user launches RUSH HOUR then selects the Play button from the main menu.
    - RUSH HOUR responds by presenting the user with four levels of difficulties labeled Beginner, Intermediate, Hard and Expert and prompts the user to select the difficulty level of the game.
- The user selects the level of difficulty according to their comfort level.
    - RUSH HOUR responds by presenting the user a grid of puzzles of the selected difficulty and prompts the user to select any puzzle they wish to play.
- Player selects the puzzle they wish to play.
    - RUSH HOUR loads the selected puzzle and sets the timer within which player is encouraged to finish selected level.
- Player completes the level.
    - RUSH HOUR rewards players with a fixed amount of coins.
    - RUSH HOUR checks if player completed with minimum number of moves. If yes, RUSH HOUR rewards player 3 stars else rewards two stars if player takes 5 extra moves or just one star if neither condition applies. The reward of coins earned is multiplied according to stars earned.
    - RUSH HOUR checks if player completed level within time limit. If yes, RUSH HOUR rewards the user one, two or three crowns depending on the time limit remaining. If player takes longer than allotted time limit to complete the level, RUSH HOUR does not reward player any crowns. For each crown earned, player is rewarded a bonus of 1000 coins.
    - RUSH HOUR saves the amount of stars, coins and crowns earned by player so that player knows which levels they have completed and/or need to repeat to get a better score.

- RUSH HOUR prompts user if they wish to repeat level, go back to level grid or go back to difficulty level selection.

*Entry Condition:*

- The user launches RUSH HOUR desktop application and selects the "Play" option from the main menu.

*Exit Condition:*

- The user closes the application by invoking "Exit Game" use case.

**2) Resetting Game Level**

*Use Case Name:* Reset Level

*Participating Actors:* The Player

*Flow of Events:* ……

*Entry Condition:*

- The user has selected the level and the game has started.

*Requirement:*

- At any point when the user is playing a level, the user may wish to reset the level so that they may start over again. The "Reset Level" use case **extends** the "Play" use case and is initiated when player presses the appropriate button on the game grid. When this use case is invoked, the level is reset to its original setting and the timer is reset.

**3) Customize Game Play**

*Use Case Name:* Customize

*Participating Actors:* The Player

*Flow of Events:*

- Player invokes "Customize" use case after selecting the button from the main menu.
  - RUSH HOUR responds by providing player five options; Adjust Music Level, About Us, Report Bugs, Change avatar and Reset Game. These 5 options are shown as "Change Settings" use case in the diagram which extend the "Customize" use case.
- Player selects one of the five options.

*Optional Alternative Flows:*

- *Adjust Volume*
  - Player chooses to Adjust Volume of the game by selecting proper button.

        - RUSH HOUR prompts player to adjust volume of music according to their liking, ranging from 0 to 100.
    - Player adjusts volume of background music according to their liking.
        - RUSH HOUR saves any changes made.
- *About Us*
    - Player chooses to know more about the developer by selecting “About Us”.
        - RUSH HOUR provides information about the developers of the game, their names, general information about them and why they made the game.
- *Report Bugs*
    - Player chooses to Report Bugs by selecting proper button.
        - RUSH HOUR prompts user to enter their email address and a description of the bug they encountered during the game.
    - Player sends report of bug.
- *Change Avatar*
    - Player chooses to “Change Avatar” by selecting proper button
        - RUSH HOUR provides selection of different avatars and prompts player to choose avatar of their liking.
    - Player chooses avatar of their choice.
        - RUSH HOUR updates player avatar.
- *Reset Entire Game*
    - Player chooses to “Reset Game” by selecting proper button.
        - RUSH HOUR warns user that all progress made will be deleted and ask player to confirm again whether they want reset the game.
    - Player chooses to reset game.
        - RUSH HOUR deletes player progress and sets the state of the game as original.

*Entry Condition:*

- Player has launched application and selects “Customize” button from the menu.

*Exit Condition:*

- Player goes back from the customize menu to main menu.

**4) Unlocked Cars and Trophies**

*Use Case Name:* My Garage

*Participating Actors:* The Player

*Flow of Events:*

- Player selects “My Garage” button in Customize menu. “My Garage” use case extends “Customize” use case.
    - RUSH HOUR provides player with two choices. Player can either choose to view achievements he/she has earned during game or he/she may choose to unlock a new car model for customizing their gameplay.

*Optional Alternative Flows:*

- View Achievements
    - The user invokes “See Achievements” use case if they choose to view the achievements they have earned during the game. If unlocked, they may select said achievement and learn the condition for which the achievement was unlocked.
- Unlock new Car
    - The user invokes the “Choose new Car” use case if they choose to unlock a new car.
        - RUSH HOUR presents a selection of model from which the user can choose from. Only models of whom the unlocking conditions are met are selectable. Each model can be bought for a defined number of coins.
- Player selects car model of their choosing.
    - RUSH HOUR asks for reconfirmation about player choice. If player confirms their choice, RUSH HOUR decreases the number of coins required for unlocking model from player’s collection and gives permission to player to select that model for next game session.

*Entry Condition:*

- Player has invoked “Customize” use case in the application.

*Exit Condition:*

- Player goes back to main menu.

**5) Game Instructions**

*Use Case Name:* Instructions

*Participating Actors:* The Player

*Flow of Events:*

- Player invokes “Instructions” use case by selecting instructions button from main menu.
    - RUSH HOUR provides information about the how the game is played. The objective of the game is outlined, rules are shown and information about earning crowns is provided.

*Entry Condition:*

- Player selects “Instructions” button from the menu.

*Exit Condition:*

- Player goes back from the instructions tab to main menu.

**6) Exit Game**

*Use Case Name:* Exit

*Participating Actors:* The Player

*Flow of Events:*

- Player invokes “Exit” use case by selecting instructions button from main menu.
    - RUSH HOUR prompts player to confirm if they want to exit the game.
- Player exits the game.

*Entry Condition:*

- The game is launched

*Exit Condition:*

- The user does not choose to exit game.

## Use-Case Diagram

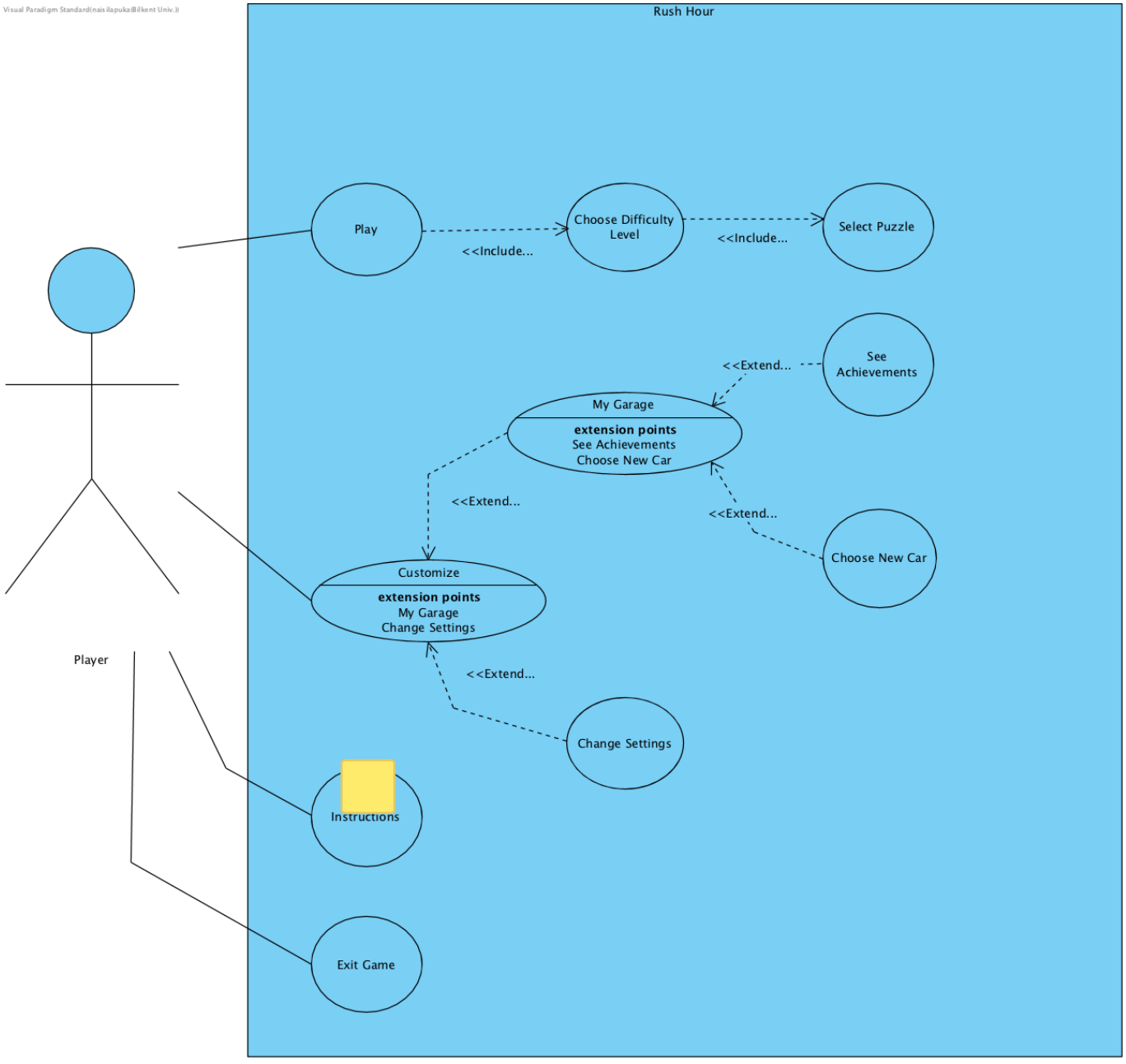


*Figure 1 Use Case Diagram for Rush Hour*

### 1.1.2 Object and Class Model

Here we provide the class diagram followed by class descriptions.

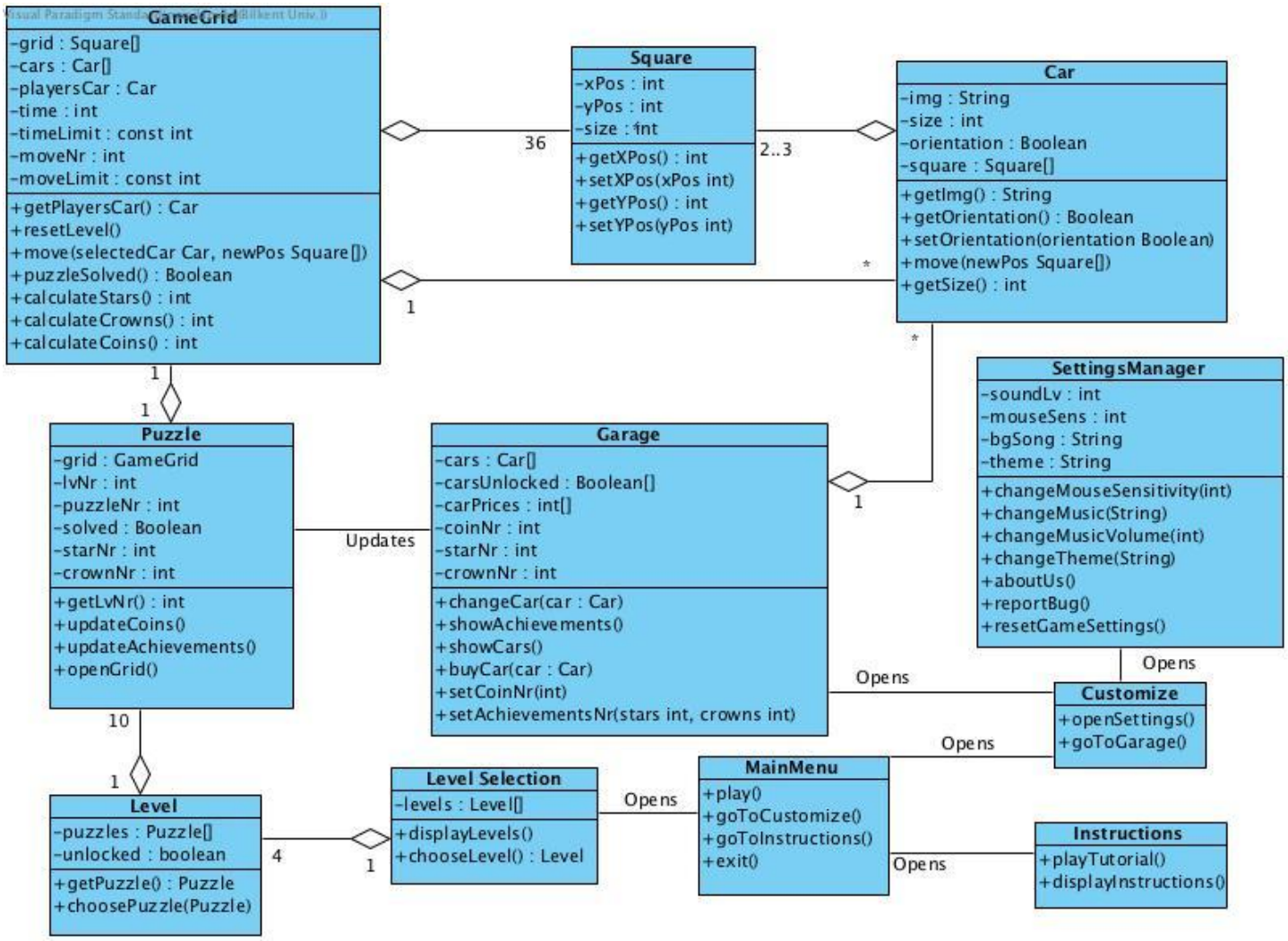


*Figure 2 Class diagram for Rush Hour*

#### Main Menu

Main Menu is designed to be the main interface of the program from where the user can chose to start playing, customize game settings, read instructions or exit game. When this object is created it reads a text file and sets the settings of the game.

#### Customize

Customize Menu will allow the user to open SettingsManager or go to Garage.

#### Settings

Settings Menu will allow the user to change the sound level, mouse sensibility, theme, report a bug, change his game nickname and avatar, read

more about the developers of the game and reset the game settings to their default. Whenever the settings of the game are edited a new SettingsManager object is created which updates the existing one. The new SettingsManager object is saved in a text file.

### Instructions

Instructions show a short text that will explain the rules of the game. It also includes a tutorial video on how to use the game controls as well as an interactive tutorial that will explain the game dynamics more clearly.

### Square

Square will be the main building block of the games architecture. Square object will be a JPanel object with a specific size and it will have methods to get and set its position in the grid.

### Car

Car object is composed of either two or three Square objects and the image of a car model. Car objects will have either horizontal or vertical inclination.

According to their inclination, a car can move only in one direction, e.g.: horizontal cars will move only left or right and vertical cars will move only up or down.

### Garage

Garage will have a list of cars with different models which will initially be locked. As the user progresses in the game and collects coins he will be able to unlock new car models and then use them in the game.

According to the performance, the user will be awarded a series of prices that he will be able to see at the Achievement Hall in Garage window.

Garage object will read a text file that is saved before. When the game is initialized for the first time the text file will be in its default form but when the user unlocks new cars or wins prices the saved text file will be updated.

### Game Grid

A Game Grid is composed by a series of squares that are organized in a square matrix structure. Each Game Grid has a series of cars arranged in vertical and horizontal positions. The user's car is found at the left most part of the grid while in the most right part of the grid there is found the exit of the grid where the user car must pass in order to win the game. Game Grid gets the user car model from the garage object.

Game Grid object is responsible for the movement of the cars in the field as well as keeping track of the movements and the time the user spends during this game. When the user wins a game, Game Grid should calculate the stars, crowns and the coins that the user should get from that puzzle.

### Level Selection

Level Selection menu will allow the user to select between four different game levels. At the beginning of the game only the first level will be unlocked. Each level is composed of ten Puzzle objects.

Whenever the user earns enough stars they can unlock a new level. Each game difficulty is composed of ten different puzzles. Whenever a player completes a certain amount of game grids they will be able to unlock a new difficulty level.

### Puzzle

Each Puzzle object has one distinct Game Grid object, which it should monitor. When the user solves the puzzle shown in the Game Grid object the Puzzle object should update the star, crown and coin number of the user. It also should check if the user has unlocked any achievements and update them in the Garage object.

## 1.1.3 Dynamic Models

### Sequence Diagrams

- **Level Selection and Game grid**

  The user chooses to play the game thereby opening the level selection menu. There he can choose one of four levels each of which contains multiple puzzles which upon selection open a grid containing cars arranged in a specific manner. The user can select a particular car and move it to any valid position to how the red car to exit the grid. Once the level finishes, a Boolean value is returned indicating the level's completion, signaling the puzzle to update the user's achievements (Figure 3).

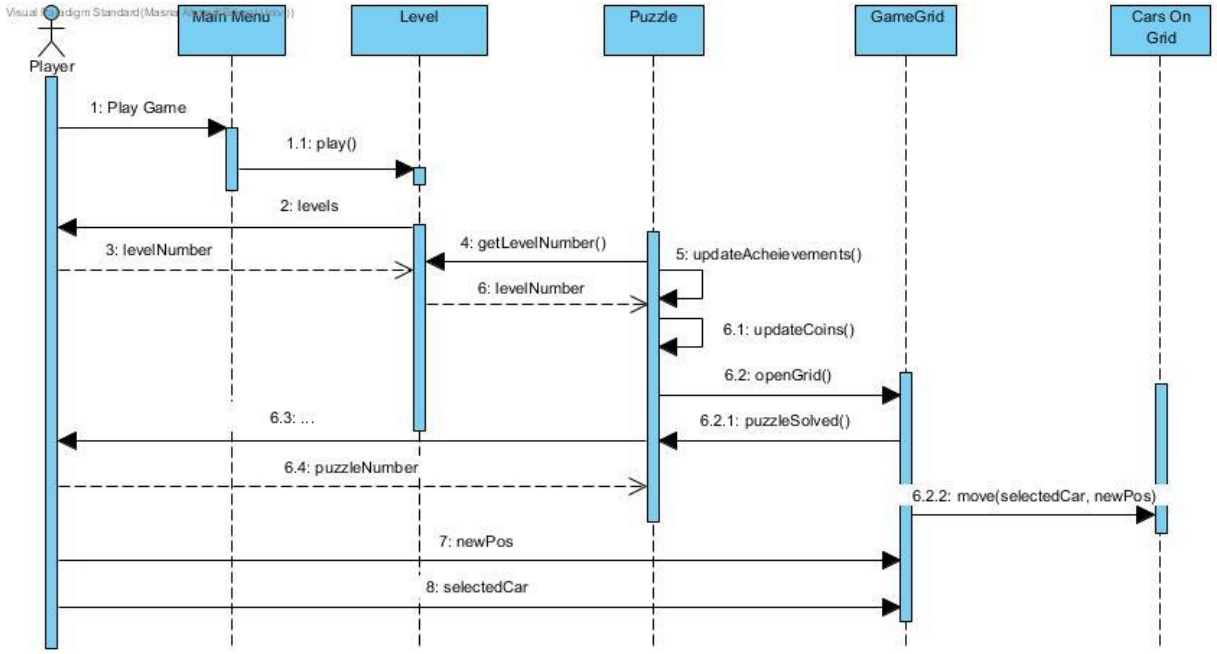


*Figure 3 Sequence Diagram for Level Selection and Game play*

- **Settings Menu and Garage**

The user opens settings wherein he can either alter the game settings such changing game volume or the user's avatar. He can also choose to go to the garage where he can view his achievements, view his unlocked cars and unlock new ones. He can also select a particular car as his main car.(Figure 4)

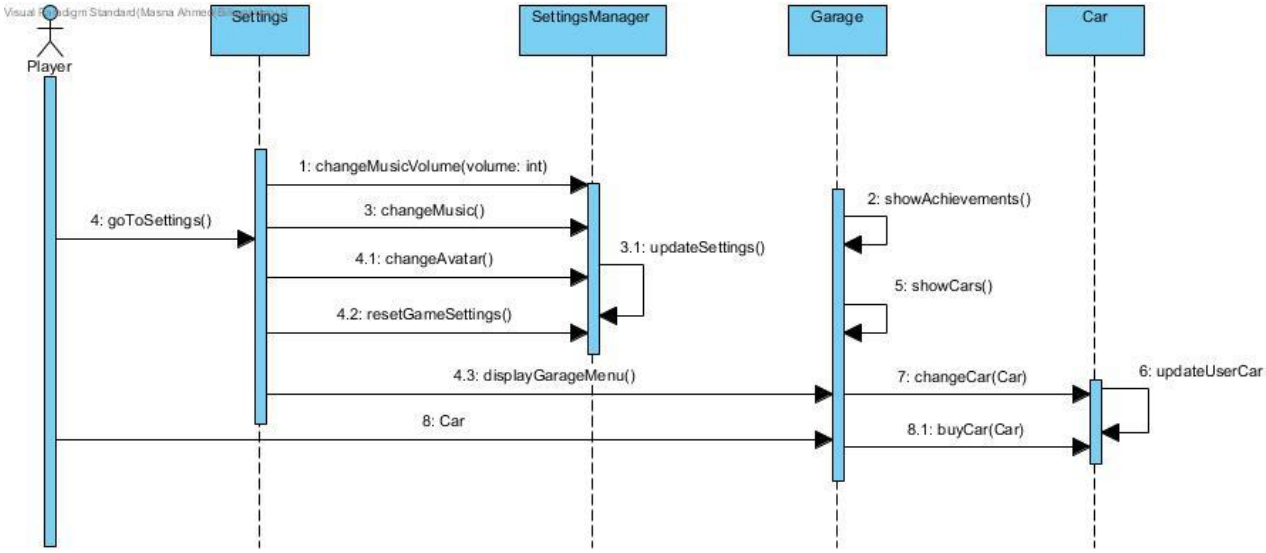


*Figure 4 Sequence Diagram for Settings Menu and Garage*

## Activity Diagram

Activity diagram represents flow of possible activities without event (trigger mechanism) throughput the game. (Figure 5) When game is started system gets user input for levels and initializes game. When the game is in the progress, system checks if the main user's car is at exit is empty. If car is not at exit, game gets updated. If the car is at exit, system retrieves time spent and number of moves used and according to them updates stars, crowns and coins for current game. After updating score menu, it restarts game flow.

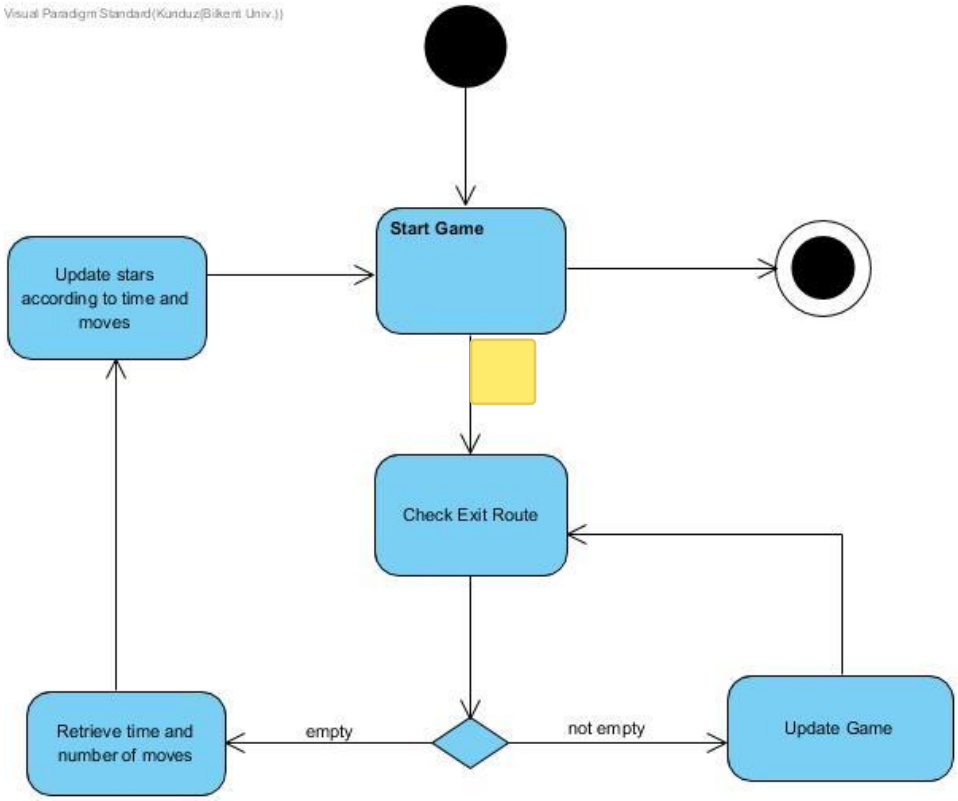


*Figure 5 Activity Diagram for Game Play*

## State Diagrams

- **State Diagram of Player playing the game**

At first player is at state of choosing level and puzzles. (Figure 6) After making choices, player goes to state where he or she starts the game. During the game there will be 2 states of player, a default state in which player can move cars up, down, left and right. Player will be in default state until puzzle is solved. In our game player cannot possibly lose because even if level is completed with extra time and too many moves, level will still be considered as a complete, that is why the second state is when player completes level and goes to win state. Alternative state is when player can choose "reset level" option and goes back to "start game" state. Anytime during the game player can end the game and go to final state.

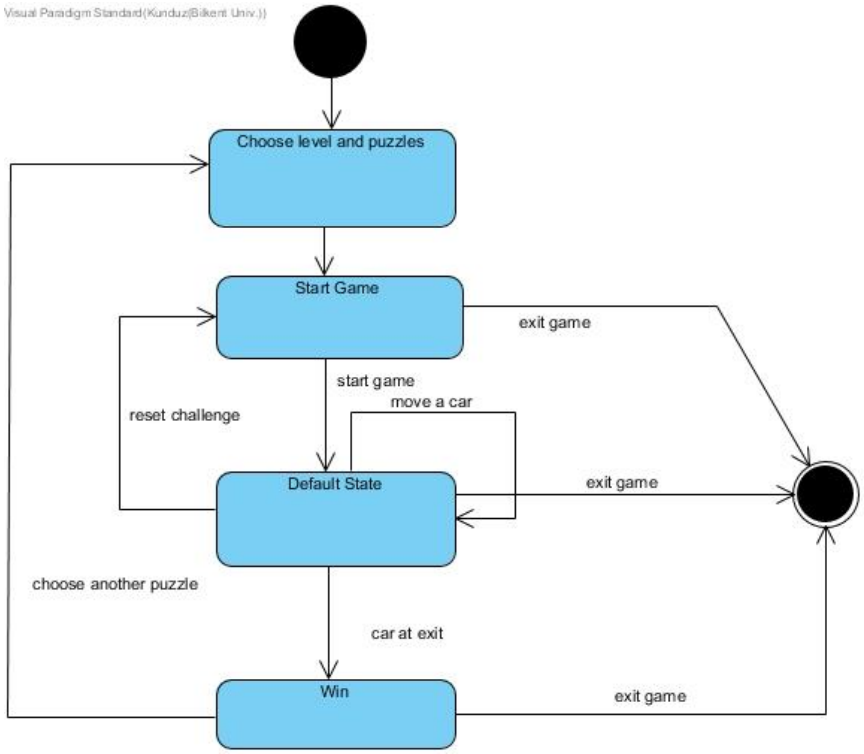


*Figure 6 State Diagram of Player playing the game*

- **State Diagram of Unlocked Levels**

Initially player is in the start state. (Figure 7) Right after that player continues to “Register to the game” state, in this state player chooses level. In the beginning, regardless of the progress, “Beginner level” will be unlocked in the next state. In order to proceed to next level player needs to collect 20 stars in total and then “Intermediate level” will be unlocked. Similarly, in order to go to next state, player needs to collect enough number of stars, but this time 38 stars in total including previous stars from previous levels. Next state requires 53 stars in total. As it showed in diagram in every state after “Register to the game” state, player can choose to reset whole game and go back to initial state “Register to the game”.

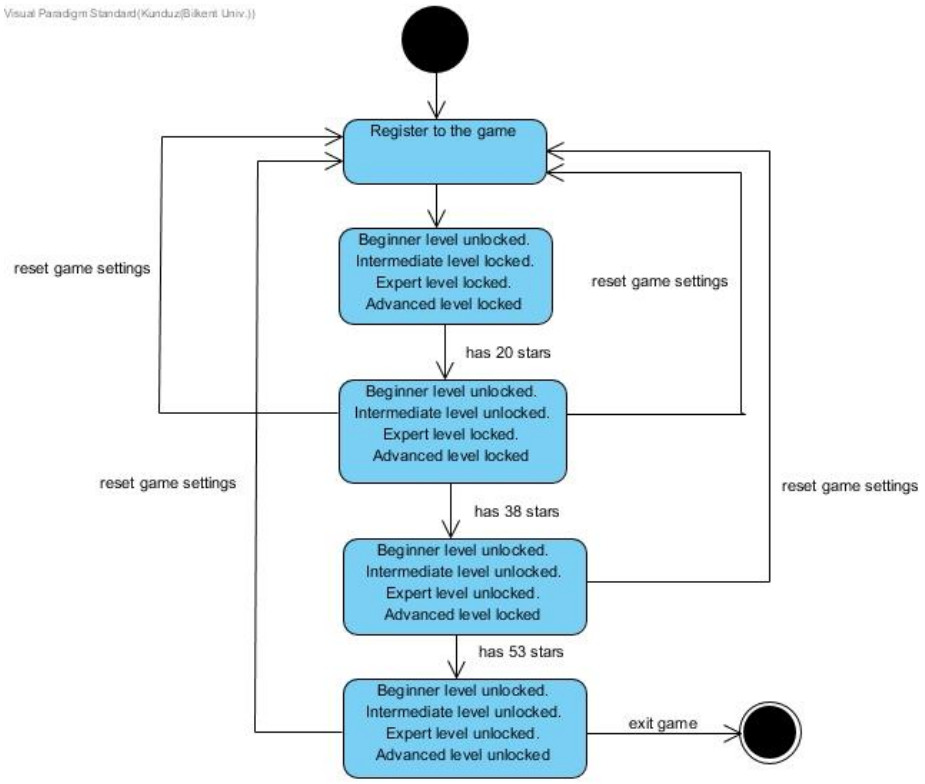


*Figure 7 State Diagram of Unlocked Levels*

### 1.1.4 User Interface

We have used Pidoco (Online Wireframe and UX Prototyping Tool)[2] to create the mock-ups and Flaming Text (Online Logo Creator)[3] for text art.

## Registration

This is the welcoming window. The user enters his name and chooses an avatar to register to the game (Figure 8).


*Figure 8 Rush Hour Registration Mock-up*

## Main Menu

In the main menu of Rush Hour game, the user can see his name, avatar and coins on the top right corner, and he may choose to navigate to the game, instructions, customize or quit (Figure 9).


*Figure 9 Rush Hour Main Menu Mock-up*

## Choose Level

Right after pressing play from the main menu, the user is directed to Choose Level window, where he can choose between the unlocked levels of the game. Here he can also see the current number of stars and the necessary stars needed to unlock the next level (Figure 10).


*Figure 10 Rush Hour Level Choice Mock-up*

## Choose Puzzle

Right after choosing the level, the user is directed to Choose Puzzle window, where he can choose between 10 different puzzles to solve. The puzzles that he has already solved before are indicated together with the obtained achievements (Figure 11).

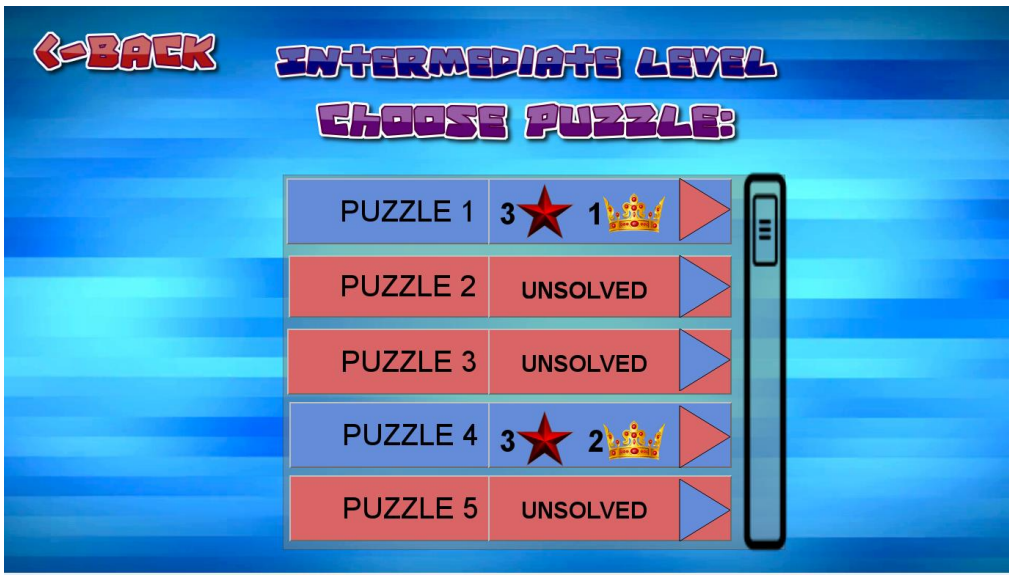


*Figure 11 Rush Hour Puzzle Choice Mock-up*

### Rush Hour Game

Right after choosing the puzzle, the user is directed to a specific game configuration. He will see a specific arrangement of different cars with his car on the leftmost middle and the exit on the rightmost middle (Figure 12). The user is able to move a vertically aligned car up or down and a horizontally aligned one left or right. He can also see the current number of moves accompanied by the minimum number of moves necessary to complete the game, as well as the current time taken. The number of moves and time is infinite in theory, but if the user manages to finish with the minimum number of moves and average time, he will receive stars and crowns, which will reflect in this puzzle specifications as well as player's total achievements.


*Figure 12 Rush Hour Game Play Mock-up*

### Instructions

If the user presses Instructions from the main menu, he will be directed to Instructions window which will have the game rules and a video tutorial.

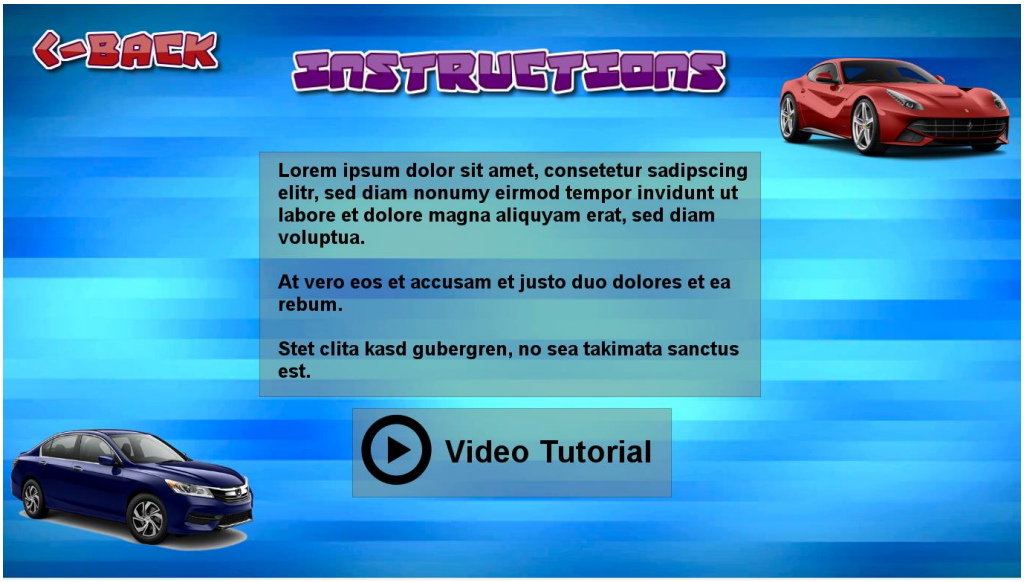


*Figure 13 Rush Hour Instructions Mock-up*

### Customize

If the user presses Customize from the main menu, he will be directed to that window which will provide the game settings and a button to user's garage. The user can see/modify different game settings such as volume, theme, background music etc (Figure 14). The user may choose to reset the game: this will erase all of the user's information and will redirect to Registration window. Also he may report a bug to the game owners and see info about the game owners.

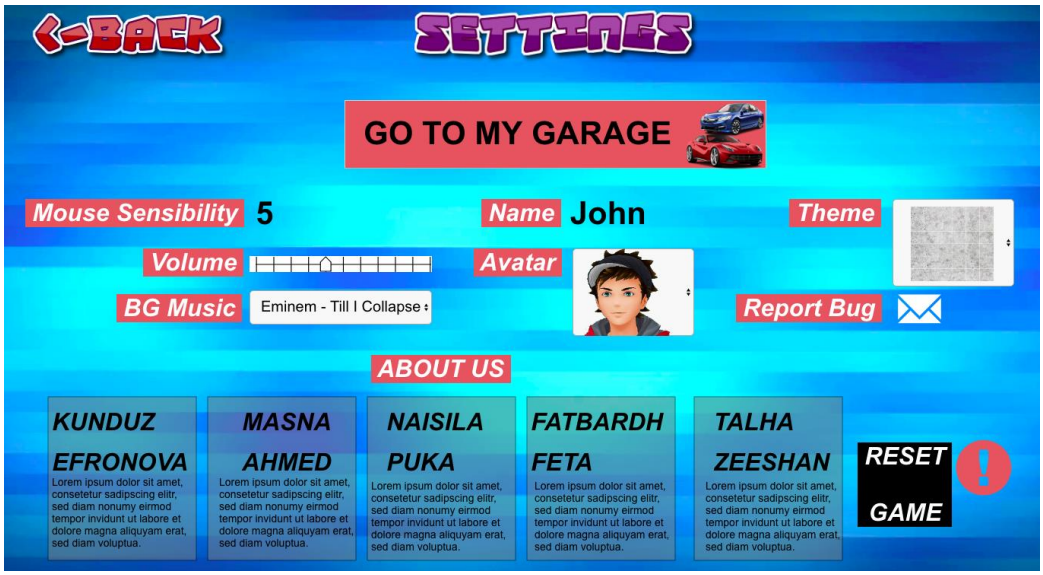


*Figure 14 Rush Hour Customize Mock-up*

## My Garage

From the Customize window, the user may press “Go to My Garage” button where he will be able to see his unlocked cars and his money (Figure 15). He may change his current car to another unlocked one. He may also unlock another car if he has enough money.


*Figure 15 Rush Hour My Cars Mock-up*

## My Achievements

From My Garage window, the user may press “My Achievements” and see his avatar and name, accompanied by the number of stars, crowns and coins he has gained.

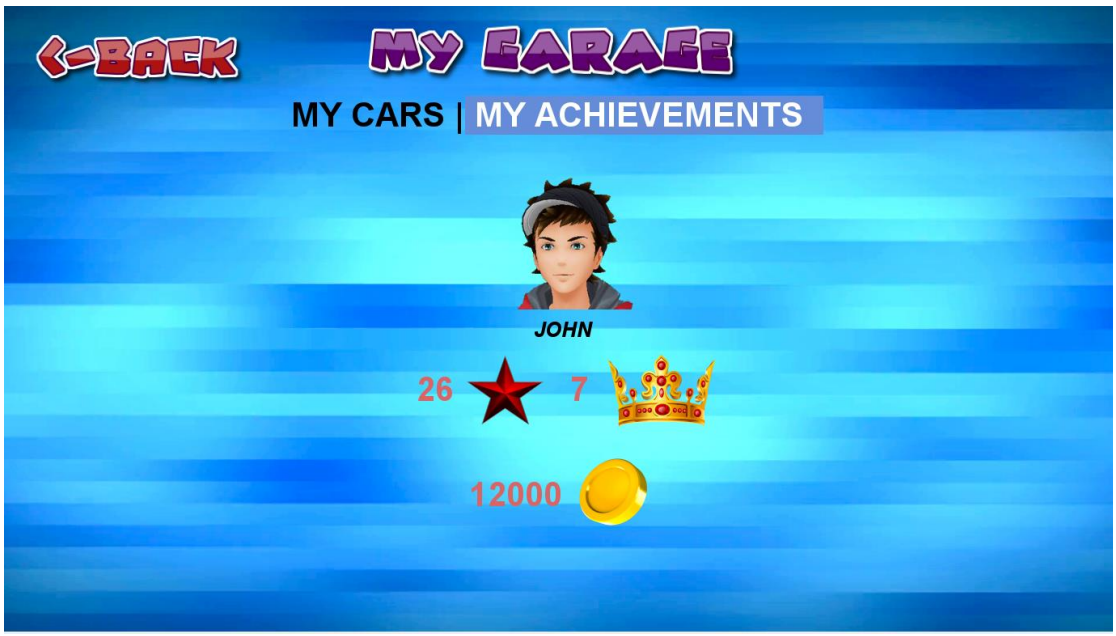


*Figure 16 Rush Hour My Achievements Mock-up*

# 3 Conclusion

In this report, we have explained our analysis of the design and implementation of the classic game “Rush Hour”. The report consists of two main sections; the requirement analysis and the system models.

The requirement analysis section of the report consists of an overview of the project as well as the functional and non- functional requirements that the project entails. The functional requirement describes the behavior and functionality of the rush hour game system we have designed and all the functionalities the user can interact with. The non-functional requirement section elaborates on the performance characteristics of our designed system such as usability and reliability.

The second section of this report focuses on the system models. The diagrams used to design and describe the system models are: Use Case Diagram, Sequence Diagram, Class Diagram, Activity Diagram and State Diagram. Each diagram mentioned describes the individual system models that are essential parts of the project. Inspiration for the design of the system models was taken from a similar game called “Unlock Me Free” [4] that is available on both the apple store and play store. This analysis report serves as a guide line for a smooth, clear and hassle free implementation of the project.

# 4 References


[1] “Rush Hour” https://www.thinkfun.com/products/rush-hour/ [Accessed 21 October 2018].

[2] “Pidoco - Online Wireframe and UX Prototyping Tool” https://pidoco.com/en [Accessed 21 October 2018].

[3] “Flaming Text (Online Logo Creator” https://flamingtext.com/ [Accessed 21 October 2018].

[4] “Unlock Me Free” https://play.google.com/store/apps/details?id=com.kiragames.unblockmefree&hl=en_US [Accessed 21 October 2018].