Hangman Revisited

Dragos Lucian

Linnaeus University

20 March 2019

Contents

1. Revision History

Date	Version	Description	Author
08.02.2019	1.0	Planning + Skeleton Code	Dragos Lucian
22.02.2019	1.1	Basic Mode + UML Diagrams	Dragos Lucian
08.03.2019	2.0	GUI + Testing	Dragos Lucian
20.03.2019	3.0	Final Iteration (feature-complete)	Dragos Lucian

2. General Information

Project Summary	
Project Name	Project ID
Hangman Revisited	1DV600_LD222JQ
Project Manager	Main Client
Dragos Lucian	Trivia enthusiasts
Key Stakeholders	
• Developer • Resource Manager • End-Users	
Executive Summary	
Hangman is a game concept in which users have a limited amount of chances to guess a randomly assigned word/phrase. “Hangman Revisited” gives a new and innovative perspective over the basic concept of the classic game by adding specific categories of words, progress tracking and individual score which make for prolonged play time.	

3. Vision

The system should allow users to play the game of “Hangman” i.e. guess the letters in a randomly assigned word/phrase from a specific category. The end goal is to guess as many words from the categories as possible.

The game will contain multiple categories of words and a timed mode which makes the game more challenging but at the same time more rewarding via the scoring system, all these features allow for diverse gameplay.

The individual progress will be seen for every category and a leaderboard will give the game a higher purpose than just guessing one word.

Every category should contain words that have something in common. The base categories will be Animals, Colors, European Countries, Movies and Songs. More categories can be added as updates for the game.

In the first mode (Classic), if a word is guessed, it’s taken out of the list, if not, it returns in the list, and will come back in the future. This mode allows for tracking the progress, the goal is to guess all the words from each set.

The second mode (Timed) increases the difficulty of the game but also gives more rewards. Guessing a “timed” word gives 3 points (compared to just one in the classic mode) but a greater risk is involved since if you don’t guess a word the first time, it won’t go back into the list.

Reflection

The vision is probably the most important part of this project because one can figure out whether the project is worth all the resources or not just by analyzing it. It also puts every team member “on the same page”, action crucial creating and maintaining a united team with a common goal. The vision is used throughout every step of developing and is the root of the project. Although some parts of the completed system might eventually differ from the initial vision, the vision acts as a map during the entire development.

4. Project Plan

The game will be developed in an iterative manner, divided into four major iterations. Along with all the steps new features will be added and tested. A basic version of the game in which letters from a word are guessed should be implemented right after planning in order to set a starting point. The total available time to develop the project is 2 months, with an iteration being released every two weeks.

4.1 Introduction

"Hangman Revisited" is a game in which the player must guess all the letters from a randomly assigned word or phrase from a selected category. The round is won if the word is completely "guessed" and lost if the player makes too many wrong guesses. Every correct word gives the player points depending on the mode selected which gives the game two end-goals: guess all words from the categories and have the highest score while doing so.

4.2 Justification

"Hangman Revisited" follows a "campaign" game structure, allowing players to track individual progress and score in order to compete against other users with the aid of the scoring system or just aim to complete all the categories via the progress trackers. The game is designed to scale with the addition of new categories as "updates" that will keep it new and actual over a longer period. These features make the game stand out and directly target Trivia enthusiasts who love longer or competitive gameplay.

4.3 Stakeholders

The **developer** works directly on the source code and follows the specifications given. The developer should write scalable code that can be easily improved and modified over the course of iterations as well as after the release.

The **resource manager** should focus on distributing the resources evenly throughout the development period. The resource manager will influence the project by assigning the number of resources available for a specific feature, setting the pace and importance.

Lastly, the **end-users** (Trivia Enthusiasts) have a large impact in the project by setting the initial specifications and taking part in testing the product, giving valuable feedback that will bring the game closer to the client's preferences.

4.4 Resources

- Time: 8 weeks (57 days, from 22 January to 21 March 2019)
- Literature: Software Engineering 10th ed. by Ian Sommerville
- Hardware: Asus G551JW (Intel® i7, 8GB RAM, GeForce® GTX 960M)
- Software: Windows 10 (64bit) running Eclipse Java 2018-09, JDK 11.0.2, JUnit 5 & JavaFX 11

4.5 Hardware and Software Requirements

Requirements to develop the game:	
Operating System	Windows® XP/Vista/7/8/8.1/10 (32/64 bit)
Processor	Intel® Core™ i3 or higher
Memory	4096MB RAM
Graphics Card	Minimum 2GB
Software	Eclipse IDE, JDK 11.0.2, JUnit 5, JavaFX 11
Minimum requirements to run the game:	
Operating System	Windows® XP/Vista/7/8/8.1/10 (32/64 bit)
Processor	Intel® Core™ i3 or higher
Memory	2048MB RAM
Graphics Card	Minimum 1GB
JRE	11

4.6 Overall Project Schedule

- Project Plan & Skeleton Code: Friday, February 8, 2019, 23:55
- Basic Game & UML Models: Friday, February 22, 2019, 23:55
- Testing of the game: Friday, March 8, 2019, 23:55
- Final Iteration with complete features: Friday, March 21, 2019, 23:55

4.7 Scope, Constraints and Assumptions

This project will contain a game in which the user has to guess randomly assigned words/phrases. The scope of the game is represented by having 5 different categories to choose from, a user database to store credentials and a simple graphical user interface. Having more categories, storing the user credentials in a more advanced and secure manner, adding a category creator as well as having a professional-looking design is out of scope because of the time limitations as well as having a low functional impact.

The game will run using a graphical user interface with a fixed resolution of 600x800px. The progress, user details and categories will be stored locally on the machine that runs the game.

To start the game, the GUI.java class located in the src/basicGame directory needs to be run. The intergace is built using JavaFX 11 which was manually added to the project and run configuration. All the game resources (code, textures, categories and database) are stored in the source folder of the game. When a new account is created, it is added to the database (db.txt) and a copy of the categories assigned to the user is created inside the directory userLibraries.

Reflections

The project plan was the most difficult part of the documentation to put together since the size of the project makes it hard to come up with Stakeholders, Requirements and Scope. But on the other hand, going through all the planning of a project helped me a lot to visualize and get a better grasp on the project as a whole, not just individual assignments. The different sections in this part make the project feel “professional” and gave a glimpse of what I expect to be working with during my future career.

5. Iterations

The four iterations of this product represent important steps in developing an application. The final project needs to be the result of all the development steps that are represented by the different iterations (planning, modelling, testing and implementing)

TASK	DESCRIPTION	ESTIMATED TIME
Brainstorm ideas for game features	Come up with ideas for developing the game as well as unique features	2 days
ITERATION 1 (DUE DATE 8th of February)		
Writing the vision	Describe why the game should be developed	1:00 H
Writing the Project plan	Describe the process of development in detail	2:00 H
Analyzing risks	Write down all the risks that might be encountered while developing the app	1:00 H
Writing the Skeleton Code	Write the basic structure of the system (doesn't have to be playable)	2h
ITERATION 2 (DUE DATE 22nd of February)		
Designing Use-Cases	Create some Use-Cases to show different features of the game	1:30 H
Expanding Use-Case-Model	Add a diagram to represent the Use Cases	0:30 H
Design State Machine for "Play Game"	Show the functionality of the code representing the "Play Game" use case	1:00 H
Design Complete State Machine	Show complete functionality of the game	1:30 H
Implementation of base game	Make the game playable (one round of hangman)	3:00 H
Design Class Diagram	Design a diagram to show the relationships between the classes	1:00 H
ITERATION 3 (DUE DATE 8th of March)		
Create Test Plan	Plan the manner of testing and show why it is relevant	1:30 H
Design Manual Test-Cases	Write Manual Test Cases for the game that follow multiple scenarios	1:30 H
Test Manual Test-Cases	Test the scenarios described by the test-cases	0:20 H
Implement Unit Tests	Use JUnit to implement tests for all the methods in the system	3:00 H
ITERATION 4 (DUE DATE 22nd of March)		
Implement Visual Interface	Replace the console input-output with a GUI	4:00 H
Implement User & "Database" Feature	Add a log in feature and find a way to store accounts	3:00 H
Implement Timed Game Mode	Different game mode that has a time factor making it harder to solve	1:00 H
Update Project Documentation	Update time log and polish all sections	3:00 H

5.1 Iteration 1 (Planning)

This iteration started once the concept of the game was set. The concept should be represented by the vision and the Project Plan should showcase the steps to be taken in order to develop the game. Risks should also be taken into consideration and strategies to prevent or easily recover from any of the risks need to be developed.

After the vision and Plan are complete, the basic structure of the game should be implemented in the so-called “Skeleton Code”, this will help plan the following iterations and divide the work-load better over the course of the following iteration.

5.2 Iteration 2 (Modelling)

The second iteration should focus on modelling and representing different parts/features of the game using UML diagrams. The first step is to write Use Cases that will represent different features and scenarios of the game. To complete the Use-Case Model, a diagram representing the use cases as well as the actors should be designed.

To help in developing the application, a state machine that describes the functionality of the game needs to be created. Expanding this application to cover all the features could help see all the connections and interactions that will be encountered during the implementing phase.

Before the due date, the game should be in a “playable” state. This could be a good moment to decide on what kind of interface should be used. This state should be represented using a class diagram to have an overview of all the relationships between the classes that are or will be developed.

5.3 Iteration 3 (Testing)

The third iteration will be focused on testing the game. In order to cover both Manual and Automated tests, a Test Plan that divides and organizes the work is required. Manual tests will be written such that they will follow multiple scenarios from the Use Cases. After going through the manual tests, the Unit Tests will be written with the purpose of covering as many features of the game as possible. Having the tests for the basic version of the game should help preserve the code integrity when new features will be added.

5.4 Iteration 4 (Final Implementation)

The last iteration should be focused on adding the remaining features to the game. This is where the key aspects mentioned in the Vision need to be implemented on an already stable build of the game. The interface of the game should be developed as well as the overall aspect and user experience of it.

Polishing all parts of the system is important and the application should be thoroughly tested in order to find any errors that can be fixed before delivering the final application.

6 Risk Analysis

As this project only has one developer, most of the risks have to do with his/her Physical and Mental wellness. Planning and managing the time required to develop this project is crucial. The biggest risk that will be encountered has to do with the lack of motivation and external distractions. Having a well-organized schedule and work style will reduce the probability of any risk occurrence. Planning with slack and dividing the workload all throughout the time allocated is the ideal solution to prevent most of the risks encountered.

6.1 List of risks

Risk	Description	Probability	Impact
Computer breaking down	Loss of progress after the last release	Unlikely	Severe
Bad interpretation of requirements	Having more work than expected	Likely	Severe
Procrastinating	Rushed and unfinished project	Likely	Significant
Getting sick	Pause in development	Unlikely	Significant
Gold Plating	Not focusing on important aspects	Likely	Significant
Missing deadlines	Staying behind the required pace	Possible	Moderate
Not attending lectures	Missing key aspect of the project	Unlikely	Moderate

6.2 Strategies

Risk	Strategy
Computer Breaking down	Back-up project often and take care of computer
Bad interpretation of requirements	Read the assignments carefully and have slack
Procrastinating	Ignore distractions and focus on important things
Getting sick	Eat fruits, workout and in case of sickness use medicine
Gold Plating	Write tasks in order of importance
Missing deadlines	Check deadlines often and plan with slack
Not attending lectures	Watch live streams, read and set reminders

Reflection

Listing risks for an individual project like this one felt odd in the beginning but after I read into different risk categories and started writing them down, I realized that they could really impact the development process. I think writing the risks made me plan with more slack and made me feel a bit more confident towards the whole project.

9. Use Case Model

In order to better understand the goals of the users, the required behavior of the system in satisfying these goals and the interactions between the users and the system a use case model was created.

9.1 Use Cases

The system is divided into five different use cases that cover the main scenario as well as the alternative options.

UC 1 Login

Precondition: none.
Postcondition: the category menu is shown.

Main scenario

1. Starts when the user wants to begin a session of the hangman game.
2. The system shows the options “Login”, “Leaderboard” and “Quit”
3. The user selects the Login option
4. The system asks for the user details and gives the option to login, register a new account, go back to menu or quit.
5. The user inputs the details and selects the login button.
6. The system shows the Category menu (Execute use case 3)

Alternative scenarios

2.1 The user selects the Leaderboard option
- The system shows the and the option to return to menu

5.1 User inputs wrong details
- The system shows an error message and returns to step 4.

5.2 User selects the register option
- Execute Use Case 2

5.3 User selects the back to menu option
- Go back to step 2

2.2/5.4 The user selects the quit option
- Execute Use Case 5

UC 2 Register

Precondition: the register menu is accessed.
Postcondition: a new user is registered, and the category menu is shown.

Main scenario

1. Starts when the user wants to register a new account.
2. The system asks for the user details and gives the option to confirm, go back to menu or quit.
3. The user inputs the details and selects the confirm button.
4. The system shows the Category menu (Execute use case 3)

Alternative scenarios

2.1 User selects the back to menu option
- Go to Use Case 1

2.2 The user selects the quit option
- Execute Use Case 5

3.1 User inputs invalid credentials
- The system shows an error message and returns to step 2.

UC 3 Select a category

Precondition: The user is logged in/registered a new account.
Postcondition: A round begins.

Main scenario

1. Starts when the user successfully logs in or registers a new account
2. The system presents the available categories with the user progress, the timed-mode checkbox, the option to go back to menu and quit.
3. The user selects a category.
4. The system presents a random word from the category.

Alternative scenarios

2.1 All of the categories are completed

- The system shows a complete game screen and an option to quit.

2.2 User selects the back to menu option

- Go to Use Case 1

2.3 The user selects the quit option

- Execute Use Case 4

UC 4 Play Game

Precondition: A category was selected.
Postcondition: The game has ended.

Main scenario

1. Starts when the user chooses a category.
2. The system presents a random word from the category, the image of the hangman and the options to go back to categories, back to menu and quit.
3. The user guesses a letter from the word
1. Repeat step 3 until all the letters are guessed.
4. The system shows that the word was guessed, increases the user score by one and asks if the user wants to keep playing, select a different category go back to menu or quit.
5. The user selects to quit the game.
6. The system quits the game (Execute Use Case 5).

Alternative scenarios

*. At any point of time, user wants to go back to menu

1. System asks for confirmation

A. 2. User confirms

3. System Goes back to menu

B. 2. User doesn't confirm

3. System returns to previous state

*. At any point of time, user wants to exit

- Execute Use Case 5

3.1 The user selected the Timed Man difficulty (UC 3)

- Same behavior, but in addition, every 10 seconds of inactivity, one "life" point is lost

3.2 The user guesses a letter that's not in the word

- The user loses one "life".

3.3 The user runs out of "lives"

- The system shows a "Game over" message and goes to step 4.

4.1 The Timed-Mode difficulty is selected

- The system increases the score by 3 points per word instead of 1

4.2 The word guessed was the last one in the category

- The user cannot select the "Keep Playing" option

5.1 The user selects "Keep Playing"

- Go to step 2

5.1 The user selects "Back to categories"

- Execute Use Case 3

5.2 The user selects "Back to Menu"

- Execute Use Case 1

5.3 The user selects "Quit"

- Execute Use Case 5

UC 5 Quit Game
Precondition: The game is running.
Postcondition: The game is terminated.

Main scenario

1. Starts when the user wants to quit the game.
2. The system prompts for confirmation.
3. The user confirms.
4. The system shows a message and terminates.

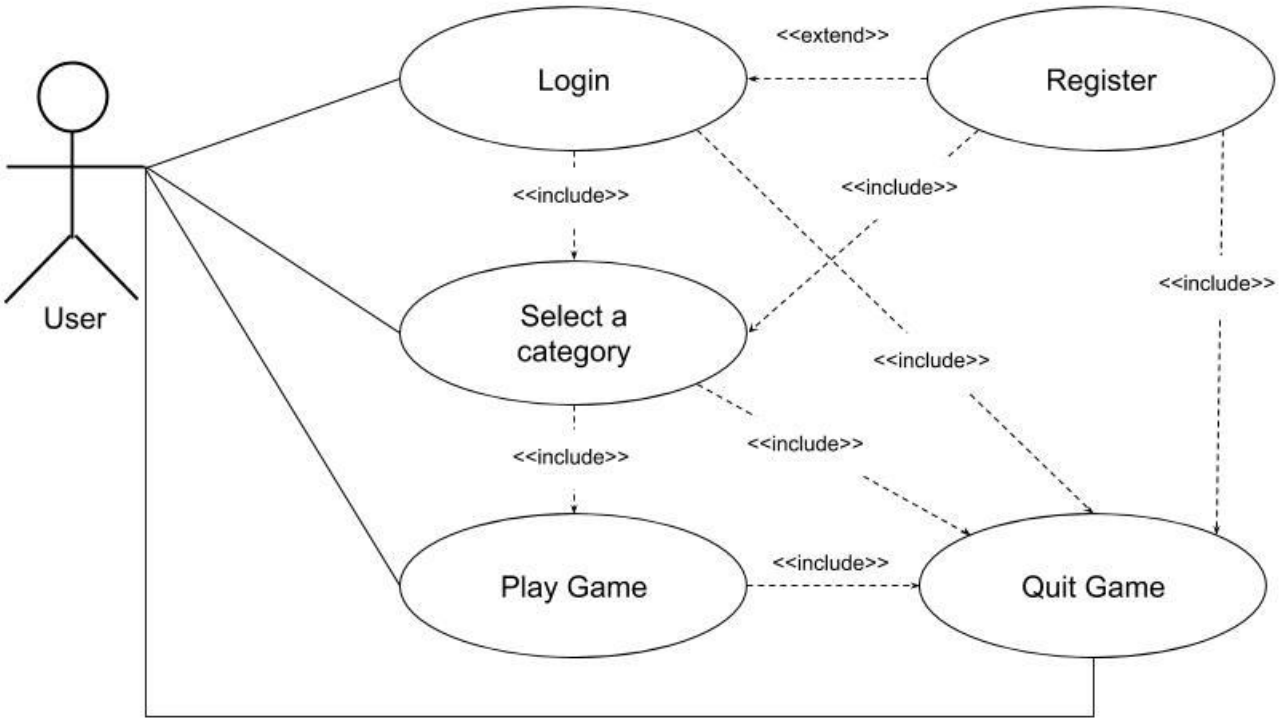
Alternative scenarios

3.1. The user does not confirm

- The system returns to its previous state

9.2 Use Case Diagram

To showcase the actors and the relationships between the use cases, a diagram was created:

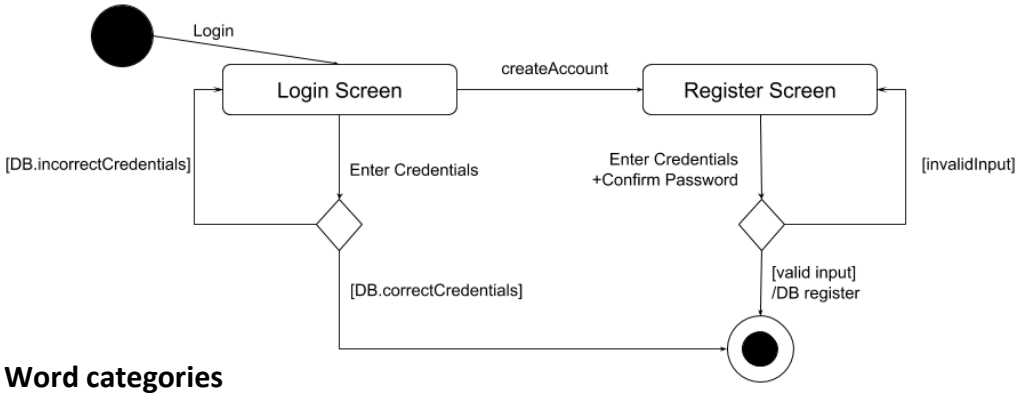


8. Additional Features

In the last iteration, three additional features were added:

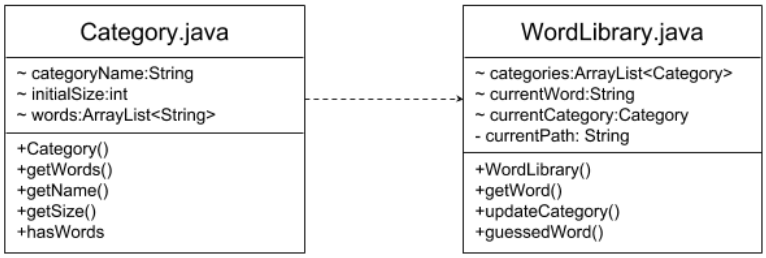
- **Login page**

In order to play the game, the user needs to register with a username and password or log in using an account previously created. The credentials are stored in a text file together with the personal score of the user.



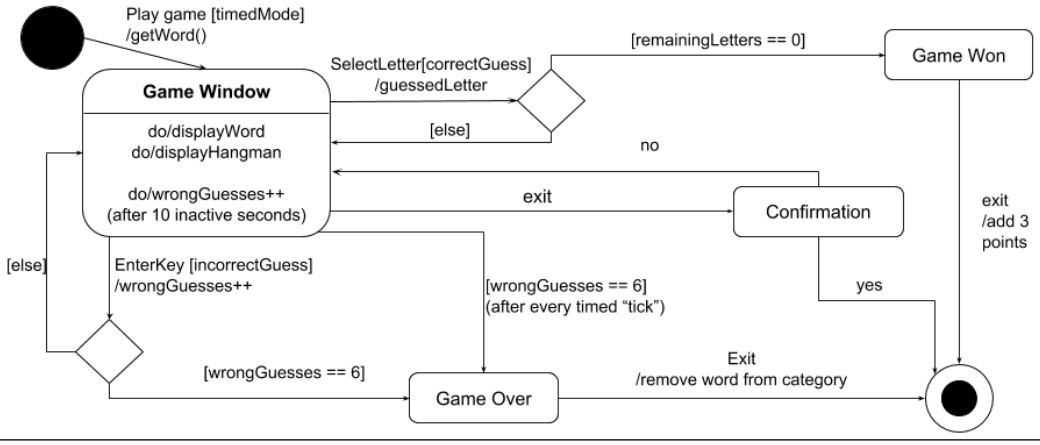
- **Word categories**

To make the game more diverse, categories containing words/phrases with similar origin are now part of the game. Their format is a txt file containing the category name, number of words and the actual words. This feature is scalable because in order to add a new category, all you need to do is to create a file in the game's "categoryFolder". All the categories will be stored in a Library which is unique for every user.



- **Timed-mode**

This feature is crucial as it affects a lot of other components and gives the game a more challenging and competitive feel. Timed-mode can be started by selecting the check box in the category menu. Playing this mode makes the user lose a "life" every 10 seconds of inactivity, rewards every word with 3 points instead of 1 and removes a word from the game even if it wasn't guessed.



9. Testing

Unit testing was first implemented during the 3rd iteration for all components of the system besides the GUI which was manually tested.

9.1 Manual testing

Multiple scenarios from the use cases of the 2nd iteration were tested with an 100% success rate. From that point forward every new feature or change of old one was accompanied by a series of manual tests in order to verify the functionality of the program.

After implementing all the features, the program was tested by both the developer and a selected group of end users in order to discover errors. This final testing activity revealed a total of 5 errors which were not previously noticed:

- The timed-mode option remained selected after another user logged in.
- Accessing the leaderboards two times in the same session created duplicates.
- The first timer in a timed-mode round began from 9
- The 10 second timer resets every time a letter is selected, but the visual output doesn't
- Pressing TAB to navigate in the register menu jumps from "password" to "quit" instead of "confirm password"

All these errors were analyzed, and the fault that caused them was fixed. The aid of the end-users was crucial in order to identify these errors which could have had a devastating outcome after the final release.

9.2 Unit testing

Every time a method was added/changed, the existing tests were re-run and additional tests were created in order to maintain coverage. The total coverage is not 100% because not all errors generated from Files and Directories were handled since all the files are stored in the application's directory and should not be damaged by any means.

Reflection

The testing part of the course felt useful directly towards my system and the following projects/work environments. I came across multiple issues during testing since my source code was hard to test and had multiple interconnections between the methods/classes. But after adjusting them, it went smooth and I hope that from now on testing will always in the back of my head while coding.

The only real problem I came across was the fast transition from the last part of the course(modelling) to the testing one, in which I had no additional time to develop the game.

10. Time log

Task	Date	Time Estimated	Actual Time
Project Plan	05.02.2019	2:00 H	3:30 H
Skeleton Code	06.02.2019	1:00 H	1:30 H
Plan 2nd Assignment	19.02.2019	1:00 H	0:30 H
Use Case Model	20.02.2019	1:30 H	2:30 H
State Machine(basic)	20.02.2019	1:00 H	2:00 H
Implement Basic Game	20.02.2019	3:00 H	2:00 H
Class Diagram	20.02.2019	1:00 H	0:30 H
State Machine(extra)	21.02.2019	1:30 H	2:00 H
Plan 3rd Assignment	04.03.2019	0:30 H	1:00 H
Code Inspection	04.03.2019	1:00 H	1: 30 H
Manual TC	06.03.2019	1:30 H	1: 00 H
Running Manual Tests	06.03.2019	0:20 H	0:10 H
Unit Tests	08.03.2019	3:00 H	4:00 H
Test report	08.03.2019	1: 00 H	0:45 H
Plan features to add	13.03.2019	2:00 H	2:30 H
Implement GUI	14-15.03.2019	4:00 H	5:30 H
Implement Categories	15.03.2019	3:00 H	2:00 H
Implement Login page	15.03.2019	3:00 H	4:00 H
Implement Timed Mode	16.03.2019	1:00 H	2:30 H
Design and add textures	16.03.2019	1:00 H	1:00 H
Populate with categories	16.03.2019	0:15 H	0:15 H
User-Test & find bugs	17.03.2019	2:00 H	2:30 H
Fix said bugs	17.03.2019	3:00 H	2:00 H
Tidy code up	18.03.2019	1:00 H	1:30 H
Finish project documentation	19.03.2019	3:00 H	4:00 H

Some of the tasks took more time than estimated but there were also a few that took less time to implement. The tasks that had a noticeable difference between the estimated and actual time are:

- Project Plan – putting together the plan didn't seem as difficult in the beginning but going through all aspects that need to be described turned out to be very time-consuming.
- UML diagrams – a bad strategy was used while designing the diagrams (i.e. trying to learn about all their particularities during their development) which made the actual time be almost two times as large as the expected one. This was taken as a lesson for further projects, never learn how to use an important tool "on-the-go".
- Unit tests – implementing the tests was a bit more difficult than expected because the methods from the system were developed without having testing in mind. Although by dividing the methods into smaller logical structures made the testing process easier and did not cause an impediment in the project.
- GUI – the transition from the console input-output to the GUI took longer as some methods had some instructions specific for the console and required adjustments in order to work with the GUI

Besides the mentioned aspects, I think the estimations were quite accurate. The most important aspect which was omitted was adding slack to every task in order to be sure there is some time to adjust unpredicted outcomes.