Development and Testing

Table of Contents

[Chapter Three: Development and Testing 2](file:///C:\Users\Aissa\Desktop\Project%20Template%20New%202020\NEW%20Developmenet%20and%20testing%20Template.docx#_Toc36797623)

[3.1 Introduction 3](#_Toc36797624)

[3.2.1 Stage 1: Building the Database 3](#_Toc36797625)

[3.2.1.1 SQLITE + Python Code 3](#_Toc36797626)

[3.2.1.2 Testing Databases 3](#_Toc36797627)

[3.1.3 Feedback from Stakeholder (if appropriate) 4](#_Toc36797628)

[3.1.4 Review and Corrections 4](#_Toc36797629)

[3.2.2 Stage 2: Building Log in 4](#_Toc36797630)

[3.2.2.1 The Code for Log in section (including validation) 4](#_Toc36797631)

[3.2.2.2 Testing Log in section (including validation) 4](#_Toc36797632)

[3.2.2.3 Feedback from Stakeholder (if appropriate) 4](#_Toc36797633)

[3.2.2.4 Improvements and Corrections 4](#_Toc36797634)

[3.2.3 Stage 3: Building …….. PART THREE …. 5](#_Toc36797635)

[3.2.3.1 Code for …Part THREE … including validation 5](#_Toc36797636)

[3.2.3.2 Testing …. Part THREE ….including validation 5](#_Toc36797637)

[3.2.3.3 Feedback from Stakeholder (if appropriate) 5](#_Toc36797638)

[3.2.3.4 Improvements and Corrections 5](#_Toc36797639)

[3.2.3 Stage 4: Building …….. PART FOUR …. 6](#_Toc36797640)

[3.2.3.1 Code For … part FOUR… (including validation) 6](#_Toc36797641)

[3.2.3.2 Testing …. Part FOUR ….including validation 6](#_Toc36797642)

[3.2.3.3 Feedback from Stakeholder (if appropriate) 6](#_Toc36797643)

[3.2.3.4 Improvements and Corrections 6](#_Toc36797644)

[3.2.4 Stage 2: Building …….. PART FIVE …. 7](#_Toc36797645)

[3.2.4.1 Code for … Part FIVE… including validation 7](#_Toc36797646)

[3.2.4.2 Testing …. Part FIVE ….including validation 7](#_Toc36797647)

[3.2.4.3 Feedback from Stakeholder (if appropriate) 7](#_Toc36797648)

[3.2.4.4 Improvements and Corrections 7](#_Toc36797649)

[3.3 Final Improvements and Corrective Actions 7](#_Toc36797650)

# Chapter Three: Development and Testing

Development and Testing

## 3.1 Introduction

When programming my project, I will use a modular approach. This means I will use the fact that I have decomposed the project into many smaller sections and I will first test each of these sections or modules separately. In the code, each module can import other modules, when necessary, in order to be able to run a function from another module. This approach allows for reusability of code and also improves the readability and ease of editing the code.

I will use sqlite3 in order to build by database. This will mean the database is stored locally on one machine rather than on an online server. This makes it more suitable for a low number of users or even a single user. Because my project is single-player and is not an online game, it is not important that the database can only be accessed from one device.

I have decided to use Python 3.1, because it is the programming language that I feel the most comfortable with. There is also a wide range of libraries available, some of which will be used frequently in my game.

I will use Tkinter for the login part of the project as well as displaying popup messages. This allows me to create a user-friendly, visually appealing interface that the user will use to log in as well as displaying error messages.

For creating my game, I have decided to use Pygame. This will allow me to display the GUI for the space game as well as displaying the game’s menus such as the highscore menu. Pygame is suited for my project because it allows for creation of sprites that are flexible and can be displayed on screen when needed. It also allows the player to input and control their character in the game using the keyboard.

I will use the bcrypt library which will allow me to hash and salt the passwords of users that are stored in the database. This will improve the security of the system.

When testing the project, I will use unittest in order to do automatic testing rather than manual testing. This will improve the efficiency when testing and allow modules to be tested quicker.

##### Modules

|  |  |  |  |
| --- | --- | --- | --- |
| Module name | Purpose | Relevant Design section | Development section |
| Admincontrol.py | Provides the UI for an admin to create or delete users |  |  |
| colour\_changer.py | Changing the colour hue of an image |  |  |
| dates.py | Handles getting the dates needed for storing a score |  |  |
| game.py | Main game module, runs the game GUI |  |  |
| HighscoresData.py | Connects to the database in order to read or edit scores |  |  |
| login.py | Provides the UI for a user to log in |  |  |
| LoginData.py | Connects to the database in order to read or edit user details |  |  |
| main\_app.py (Top-level module) | Starts the initial login window |  |  |
| messages.py | Displays Tkinter popup messages for error handling |  |  |
| settings.py | Manages editing games settings that are saved to a text file |  |  |
| sprites.py | Contains all sprite classes that are used in game.py |  |  |
| testing.py | Runs unit tests on each module |  |  |
| validation.py | Used anytime data must be validated |  |  |

## 3.2.3 Stage 1: Building the Error Handling

### 3.2.3.1 Code for Error handling section

##### Imported modules:

1. **import** tkinter **as** tk
2. **from** tkinter **import** messagebox

I have used Tkinter Messagebox to display the windows in this section

##### Prototype 1:

Show\_message procedure:

1. **def** show\_message(title, message):
2. root = tk.Tk()
3. **messagebox.showinfo(title=title, message=message)**
4. root.destroy()

I first built this show\_message procedure, which displays a message window using Tkinter and creating a window called root. The Tkinter.messagebox module is then used to display an information box with the title as the value of the parameter ‘title’ and a message as the parameter ‘message’. When OK is clicked, the message box will be closed and the parent window will be destroyed with .destroy(). I will first test this prototype before continuing to develop this module.

###### Prototype 1 testing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Description** | **Test data** | **Expected outcome** | **Comment** |
| 1.1 | Attempt to display a message | Run show\_message() procedure with title = “Test title” and message = “Test message” | Message box with title “Test title” and message “Test message” displayed | Failed |

###### Prototype 1 results:

Test 1.1 failed because, while the message was displayed correctly, the Tkinter parent window was still visible alongside the message box. This should not be the case.

##### Prototype 2

Show\_message procedure (fixed):

1. **def** show\_message(title, message, opt):
2. root = tk.Tk()
3. root.withdraw() *# Hides tk window immediately*
4. messagebox.showinfo(title=title, message=message)
5. **root.destroy()**

In order to fix the bug found in test 1.1, I have added code that withdraws the blank Tkinter window before the message box is displayed. This should prevent the window from appearing and now only the message box should be displayed.

###### Prototype 2 testing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Description** | **Test data** | **Expected outcome** | **Comment** |
| 1.1 | Attempt to display a message | Run show\_message() procedure with title = “Test title” and message = “Test message” | Message box with title “Test title” and message “Test message” displayed | Success |
| 1.2 | Ok button clicked in the message box | Clicking on ‘OK’ | Message box closes | Success |

###### Prototype 2 results:

Test 1.1 was successful as shown in this evidence:

Test 1.2 was successful as when the ‘OK’ button was clicked the message box window closed and the parent Tkinter window was never visible.

##### Prototype 3:

Show\_message procedure (improved)

1. *# Options: 1-show info, 2-show error, 3-show warning, 4-ask y/n*
2. **def** show\_message(title, message, opt):
3. root = tk.Tk()
4. root.withdraw() *# Hides tk window immediately*
5. **if opt == 1:**
6. messagebox.showinfo(title=title, message=message)
7. **elif** opt == 2:
8. messagebox.showerror(title=title, message=message)
9. **elif** opt == 3:
10. **messagebox.showwarning(title=title, message=message)**
11. **else**:
12. **return** messagebox.askyesno(title=title, message=message)
13. root.destroy()

With the base show\_message() procedure now working as expected, I decided to add the feature to display different kinds of messages depending on option selected using the value of the ‘opt’ parameter.

##### Prototype 3 testing:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Description** | **Test data** | **Expected outcome** | **Comment** |
| 1.3 | Attempt to display a show info box. | Run show\_message() with title = “Info title” and message = “Info message”. opt = 1 | Info box displayed with title “Info title and message “Info message” | Success |
| 1.4 | Attempt to display an error box | Run show\_message() with title = “Error title” and message = “Error message”. opt = 2 | Error message displayed with title “Error title” and message “Error message” | Success |
| 1.5 | Attempt to display warning message box | Run show\_message() with title = “Warning title” and message = “Warning message”. opt = 3 | Warning message displayed with title “Warning title” and message “Warning message” | Success |
| 1.6 | Attempt to display question box | Run show\_message() with title = “Question title” and message = “Question message”. opt = 4 | Question box displayed with title “Question title” and message “Question message” | Success |
| 1.7 | Attempt to click “Yes” on question box | Clicking on the “Yes” button | Message box closes and returns True | Success |
| 1.8 | Attempt to click “No” on question box | Clicking on the “No” button | Message box closes and returns False | Success |

### 3.2.3.2 Testing …. Part THREE ….including validation

### 3.2.3.3 Feedback from Stakeholder (if appropriate)

### 3.2.3.4 Review

## 3.2.1 Stage 1: Building the Database

### 3.2.1.1 SQLITE + Python Code

### 3.2.1.2 Testing Databases

### 3.1.3 Feedback from Stakeholder (if appropriate)

### 3.1.4 Review

## 3.2.2 Stage 2: Building Log in

### 3.2.2.1 The Code for Log in section (including validation)

### 3.2.2.2 Testing Log in section (including validation)

### 3.2.2.3 Feedback from Stakeholder (if appropriate)

### 3.2.2.4 Review

## 3.2.3 Stage 3: Building …….. PART THREE ….

### 3.2.3.1 Code for …Part THREE … including validation

### 3.2.3.2 Testing …. Part THREE ….including validation

### 3.2.3.3 Feedback from Stakeholder (if appropriate)

### 3.2.3.4 Review

## 3.2.3 Stage 4: Building …….. PART FOUR ….

### 3.2.3.1 Code For … part FOUR… (including validation)

### 3.2.3.2 Testing …. Part FOUR ….including validation

### 3.2.3.3 Feedback from Stakeholder (if appropriate)

### 3.2.3.4 Review

## 3.2.4 Stage 2: Building …….. PART FIVE ….

### 3.2.4.1 Code for … Part FIVE… including validation

### 3.2.4.2 Testing …. Part FIVE ….including validation

### 3.2.4.3 Feedback from Stakeholder (if appropriate)

### 3.2.4.4 Review

## 3.3 Final Review, Improvements and Corrective Actions