|  |  |
| --- | --- |
| Final Project– MODULAR STUDY APPLICATION | ABSTRACT  A Project Design for the development of a Modular Study Application for students. The design outlines the interface and functionality, application structure, build instructions and UML Diagram for the application.  Kory Bennett / Brock Allton / Dylan Sawyer / Justin Casey / Dan Page / Cale Ward  UMUC 495 6380 2 March 2020 |

TABLE OF CONTENTS

|  |  |
| --- | --- |
| **Revision History** | **2** |
| 1. **Overview** |  |
| 1. **Project Plan** |  |
| 1. **Requirements Specification** |  |
| 1. **System Specification** |  |
| 1. **User Guide** |  |
| 1. **Test Plan and Results** |  |
| 1. **Design and Alternate Design** |  |
| 1. **Development History** |  |
| 1. **Conclusions** |  |

Revision History

|  |  |  |
| --- | --- | --- |
| Date | Description | Author |
| 02 Mar 2020 | Initial Draft | K. Bennett |
| 04 Mar 2020 | Update User guide | K. Bennett |
| 05 Mar 2020 | Review / Conclusion Update | C. Ward |
| 05 Mar 2020 | Review / Lessons Learned | D. Sawyer |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Final

1. **Overview**

The Modular Study System is a program that can be utilized by students at any level from grade school through college. The purpose is to allow the user, a student, to load practice test questions on each subject and then test themselves on those questions. The program will store the questions that are loaded, the questions that are asked, the answer given for those questions, and keep track of how many times the questions were answered correctly or incorrectly.

1. **Project Plan**

**Project Description & Objective**

The Modular Study System is a program that can be utilized by students at any level from grade school through college. The purpose is to allow the user, a student, to load practice test questions on each subject and then test themselves on those questions. The program will store the questions that are loaded, the questions that are asked, the answer given for those questions, and keep track of how many times the questions were answered correctly or incorrectly.

**Project Management**

**Methodology**

The Modular Study System team will be utilizing Trello and Gantt charts for progress tracking of this project. Due to the rather quick deadline for the project, and the different items that will be in constant motion, the Gantt chart will provide an excellent visual representation of all tasks displayed against a timeline of the life of the project. Gantt charts can adjust automatically and display up to date project schedule and timelines. Since meetings will not be taking place daily, the Gantt chart will make it easy for all team members to understand and follow the progress of the project. Before the Gantt chart can be created all requirements of the program will be thoroughly laid out and understood by all team members.

For the purpose of tracking each portion of the project, the team will utilize Trello. The Trello system provides an excellent source for not only keeping track of to-do-lists, files can be shared amongst members, cards can be easily inserted and removed as required as the project progresses and are edited with comments to easily keep team members informed of the tasks current state. Below is a list of cards that will be used to track the progress of the project:

* Project Plan: tracks all tasks for the completion of this plan
* Test plan: tracks all tasks for the completion of the test plan
* Coding: track the different modules being created for the project
* Testing: track the testing of the different modules as they are completed and being put together

Each of the boards listed will be used on three separate lists: Not Started, In Progress and Completed. The Project Manager will be responsible for upkeep of the cards once the task is assigned. The cards will reflect the specific item to be worked, who is to work on it, when the due date for that item is completed and the positioning of the card on the appropriate list. Although constant reminders will be sent out on what is going on for the week, due dates or items and so on, all team members will be responsible for constant monitoring of Trello and the Gantt chart.

**Communication Tools**

The primary method of communication for the Modular Study System team is via Hangout. This app will allow all members of the team to stay in near constant contact with one other through messages as the app can easily be installed on a phone allowing access anywhere and anytime. The application also allows for video conferencing in the case that face to face communication needs to happen at that moment.

The Modular Study System team will also conduct two video conferences per week, one on Monday and one on Saturday. These meetings will be conducted through WebEx. The meetings on Monday will be to discuss upcoming assignments for the week, who will be responsible for what aspect of the assignment, and when the due date will be for the deliverables. The meeting on Saturday will primarily be for reviews of the class deliverables due that Sunday and to address any concerns or feedback from individuals not brought up throughout the week.

**Analysis, Design & Development Tools**

* + - Plan Monitoring and Analysis Tools
      * Trello for plan progression and assignment monitoring
      * Excel for deliverables due dates and requirements
    - Design Tools
      * Draw.io for program diagrams
      * Lucidchart for class UML diagrams
    - Development Tools
      * GitHub for documentation sharing/submission
      * NetBeans for all class designs
      * MySQL for questions database

**Peer Reviews**

All team members will conduct peer reviews on all deliverables before final submission. The items for review will be available for peer review on the Friday before the due date, and all reviews will be completed by the Saturday before the due date. All comments and suggestions from the reviews will be incorporated into the final product for delivery. Peer input on material can also be done during the Saturday video conference if needed to allow for any last-minute inputs.

**Resources**

This project will employ the use of various software products that will be used in the capacity communications, program review and coding. Below is a listing of the resources used:

|  |  |
| --- | --- |
| **Name** | **Use** |
| GitHub | Documentation Repository |
| Google Docs | Project Review |
| IDE Software | Coding |
| Excel | Gantt Chart / Milestones / Peer Reviews |
| Word | Written Report |
| Google Hangouts | Team Communications |

**Roles and Responsibilities:**

|  |  |
| --- | --- |
| Name: **Kory Bennett** |  |
| Role | Team Lead |
| Responsibility | Team organizer / Documentation |
|  |  |
| Name: **Brock Alton** |  |
| Role | Project Manager |
| Responsibility | Co Team Lead / Documentation |
|  |  |
| Name: **Dylan Sawyer** |  |
| Role | Team Member |
| Responsibility | Lead Developer |
|  |  |
| Name: **Dan Page** |  |
| Role | Team Member |
| Responsibility | Software Developer |
|  |  |
| Name: **Justin Casey** |  |
| Role | Team Member |
| Responsibility | Software Developer |
|  |  |
| Name: **Cale Ward** |  |
| Role | Team Member |
| Responsibility | Software Developer |

**Risks**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Impact** | **Action** |
| Team Availability | Team is split across two time zones and with different work schedules. Lack of communication could cause delays in project development/deliverables. | Rely on online communication via hangout and email correspondence. All team members are aware to check Hangouts and email on a regular basis, and to notify when unable to attend a scheduled meeting. |
| Compressed Schedule | 8-weeks is a very quick timeline for a group project, especially when team members are geographically separated. Possibilities of missing deadlines. | Ensure all team members are aware that this is a joint effort, and each contribution is significant. If one person cannot meet obligations, then informing Team Lead and Project Manager is paramount and another person or people can pick up the task. |
| Family/Military obligations | Family and military obligations will always play a primary concern in day to day activities, and can have a large impact on the availability for meetings and having sufficient time to completed assigned tasks. | Communication is key to inform team supervision of possible conflicts stemming from family or military. When a conflict arises, tasks will be adjusted as needed to meet deadlines. |

**Project Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Duration (Days) | Start Date | End Date |
| **1. Planning and Design** | **45** | 20 January | 08 March |
| 1.1 Project Plan | 7 | 20 January | 26 January |
| 1.1.1 Documentation Writing | 5 | 22 January | 26 January |
| 1.1.2 Document Review / Submit | 2 | 25 January | 26 January |
| **1.2 Test Plan / User guide** | **7** | 27 January | 02 February |
| 1.2.1 Test plan | 5 | 27 January | 31 January |
| 1.2.2 User Guide | 1 | 01 February | 01 February |
| 1.2.3 Document Review / Submit | 1 | 02 February | 02 February |
| **1.3 Project Design** | **7** | 03 February | 09 February |
| 1.3.1 Initial Design | 5 | 10 February | 14 February |
| 1.3.2 Design Review | 1 | 15 February | 15 February |
| 1.3.3 Document Review / Submit | 1 | 16 February | 16 February |
| **2. Testing and Implementation** | **14** | 17 February | 01 March |
| 2.1 Sprint | 7 | 17 February | 23 February |
| 2.2 Sprint | 7 | 24 February | 01 March |
| **3. Project Close and Deliverable** | **7** | 02 March | 08 March |
| 3.1 Code Review | 2 | 02 March | 03 March |
| 3.1.2 Test Data | 2 | 04 March | 05 March |
| 3.1.3 User Guide Review and Revision | 2 | 06 March | 07 march |
| 3.1.4 Project Submission | 1 | 08 March | 1. arch |

1. **Requirements and Specifications**

|  |  |
| --- | --- |
| Requirement # | Description |
| 1 | The application will open with a Java-based Graphic Interface |
| 2 | Several test subjects will be available for user selection |
| 3 | User will have the ability to customize the number of questions |
| 4 | Questions will be multiple choice with only one correct answer |
| 5 | Questions will display in a random sequence until finished |
| 6 | Test results will display at the end of the test and be accurate |
| 7 | User may return to the main menu and select another test |
| 8 | A previous score will not affect the next test score |
| 9 | Questions will only display for the chosen subject |
| 10 | Entering more questions than available will alert the user |

During the development process, all requirements were met except for requirements 7 and 8. This is due to the decision to remove the Restart button and simply exit the program instead, allowing the user to re-run the program if desired. This effectively accomplished the same task, so requirement 8 could be technically considered met. However- the reason the restart button was removed was due to resetting complications. The exact source of the bugs is still unknown but several attempts were made by several developers on the team to determine the cause. It was confirmed that tracking and resetting variables were working properly. The “best guess” for the cause was attributed to issues with Java Swing not reporting selected radio buttons properly upon reinstancing the GUI windows. Which would explain the issues encountered in Phase III (documented via screenshots).

1. **System Specifications**

**Main.java**

* Started by the user
* Initializes and handles GUI.java
* Queries Maria DB for the above information through Database.java
* Parses database results for optimal viewing in GUI
* Receives requests from GUI.java
* Sends results to GUI.java
* Serves as the primary interface to the database
* Responsible for randomizing questions
* Tracks questions completed out of total
* Tracks number of correct answers out of total

**Database.java**

* Responsible for maintaining the connection to SQL database
* Receives queries from Main.java
* Sends information to Main.java
  + Available Databases
  + Number of Questions within each database
  + Individual Questions

**Question.java**

* Helper class for storing question information
* Created per question, generated from Database.java
* Resulting questions are sent to Main.java

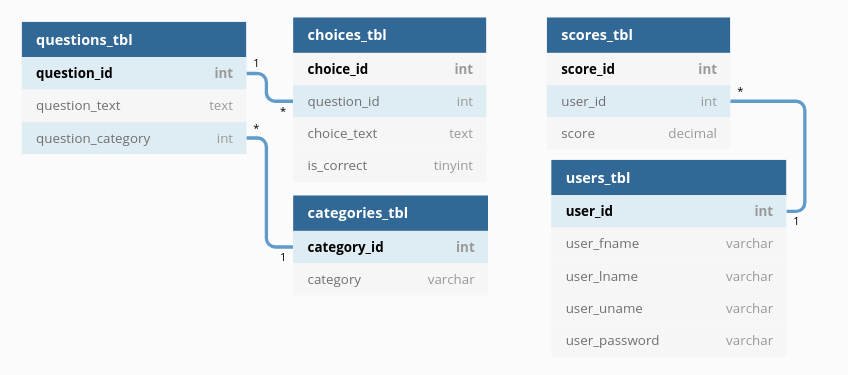
**Choice.java**

* Helper class for storing answers to questions
* Created per choice, generated from Database.java
* Resulting choices are attached to corresponding questions
* Choices are compared by Main.java to determine if correct

**SQL**

* Stores all subject and question information
* Only interfaced by Database.java
* Hosted remotely to mitigate database setup complications

Entity Relationship Diagram



**Build Instructions**

**cd into the ‘Code’ directory**

* 1. **Compile your Java code, generating class files**

*javac -cp “.:./mysql-connector-java-8.0.19.jar” <ALL\_JAVA\_FILES>*

* 1. **Create a JAR file called study**

*jar cmf manifest study <ALL\_CLASS\_FILES>*

* 1. **Execute the JAR file**

*java -jar study*

1. **User Guide**

The Modular Study System is a program that can be utilized by students at any level from grade school through college. The purpose is to allow the user, a student, to load practice test questions on each subject and then test themselves on those questions. The program will store the questions that are loaded, the questions that are asked, the answer given for those questions, and keep track of how many times the questions were answered correctly or incorrectly.

**Maria Db Backend:**

Maria Db, a SQL database variant will be responsible for housing the content of the Modular Study System. Containing a variety of subjects and questions within each subject. The content of the database will be provided to the user via a “Middle Man” java processor into the GUI Frontend.

**GUI Frontend:**

The Graphical User Interface will serve as the primary interaction with the user. The GUI is java based and will handle the input/output (IO) functions of the project. From here the user can select study subjects, answer questions and receive test scores.

**System Features:**

* Multiple study subject selection
* Multiple choice testing
* End of test score feedback
* Randomized questions
* Variable test lengths

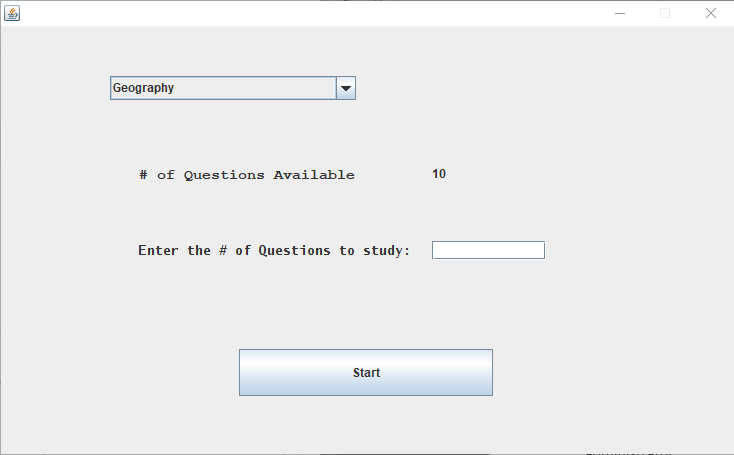
**Getting Started:**

**System Requirements:**

* Intel Core i5 or equivalent
* 2 GB memory (32-bit) or 4 GB memory (64-bit)
* 100 GB Storage Capacity
* Windows 7 or higher

**Compiling:**

The user will need to compile the code from the command line and will have two options to run the code. From the command line of the directory that all of the code is located the user can “shift” right click on the Command Prompt and “run the command prompt as administrator”

* + - 1. From there the user will navigate to the execution directory of the code.  
            
         Directory will vary depending where the user places the files.
      2. From here the user will need to compile the code:  
         
      3. After compiling, the user can type the run.bat command:  
         
      4. The java program will then execute:  
         

**Selecting a Test:**

Upon executing the program, the initial state will open a Jframe containing the controls to select a test subject. Navigate through the list of subjects, selecting the desired test.

Next select the number of questions desired for the test.

The number of questions available will appear next to the number entry.   
  
Select “Start”

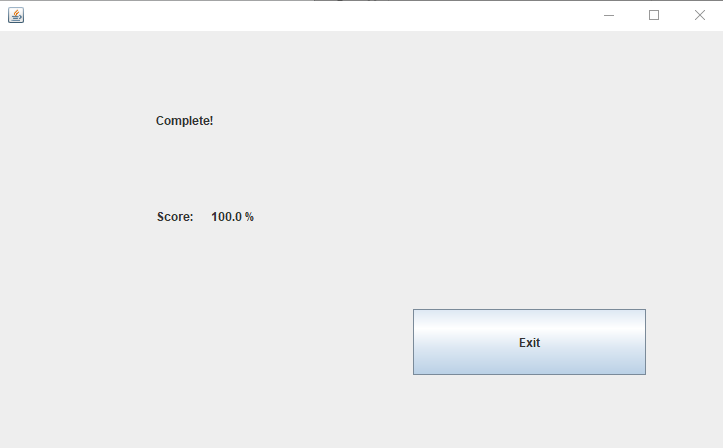
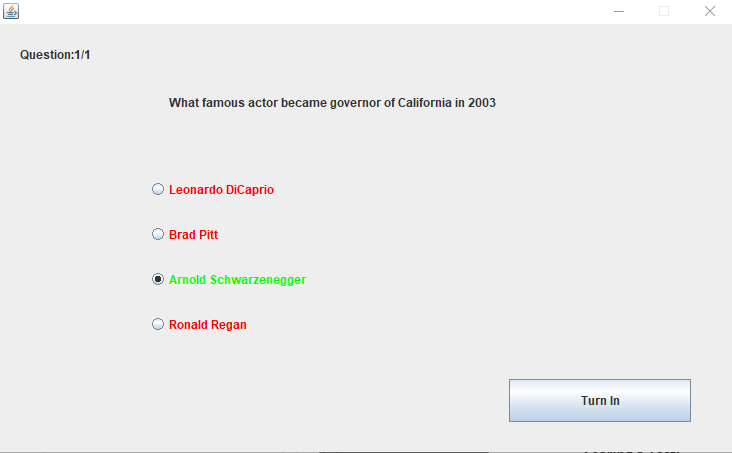
(Note: If more questions are requested than are available, the user will be notified.)

**Taking a Test:**

Once the start button is clicked after selecting a test, the questions will be displayed in random order to the user.

1. The question format is multiple choice with four answers and only one correct answer.
2. Click the radio button next to the correct answer to select it.
3. Click the Submit button to check the answer.   
   (If the answer was correct it will be highlight in green and the rest of the answers will be red. This is true if you select the wrong answer, it will be highlighted in red and the correct answer will highlight green.)
4. Now select next for the next question.

**Scoring a Test:**

Once the final question has been answered, a “Turn In” button will appear. Pressing this button will display the test results as a percentage of corrected answers divided by the total questions.  


Once finished the user may exit the program.

(Note: Scores of one test does not affect the score of any other test.)

1. **Test Cases and Results**

|  |  |
| --- | --- |
| **Test Scenario** | Use Case 1 |
| **Test Scenario Description** | History Study Session |
| **Test Environment** | Windows 10 Pro |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Action Performed | Input | Expected Result | Actual Result | Results |
| 1 | Click study.jar to run application | N/A | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Pass |
| 2 | Select History Subject | Dropdown Menu | The subject will now be set to History. | The subject is set to History. | Pass |
| 3 | Set quantity of questions to 3 | Textfield | The textfield should be populated with the number 3. | The textfield is populated with the number 3. | Pass |
| 4 | Press Start | Button | Questions begin to be displayed. | Questions begin to be displayed. | Pass |
| 5 | Select an Answer | Radio Button | Radio Button is selected. | Radio Button is selected. | Pass |
| 6 | Confirm Answer | Button | Answer is accepted and correct answer is displayed. | Answer is accepted and correct answer is displayed. | Pass |
| 7 | Next Question | Button | Next question is displayed. | Next question is displayed. | Pass |
| 8 | Repeat steps 6 through 8. | N/A | Repeat steps 6-8. | Repeat steps 6-8. | Pass |
| 9 | Complete Study Session | Button | A grade will be displayed depending on amount of correct answers. | A grade will be displayed depending on amount of correct answers. | Pass |
| 10 | End application | Button | Application closes without issue. | Application closed without issue. | Pass |

Use Case 1 Requirements Met: 1, 2, 3, 4, 6, 7

|  |  |
| --- | --- |
| **Test Scenario** | Use Case 2 |
| **Test Scenario Description** | Math Study Session |
| **Test Environment** | Raspbian 10 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Action Performed | Input | Expected Result | Actual Result | Results |
| 1 | Click study.jar to run application | N/A | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Pass |
| 2 | Select History Subject | Dropdown Menu | The subject will now be set to Math. | The subject is set to Math. | Pass |
| 3 | Set quantity of questions to 3 | Textfield | The textfield should be populated with the number 4. | The textfield is populated with the number 4. | Pass |
| 4 | Press Start | Button | Questions begin to be displayed. | Questions begin to be displayed. | Pass |
| 5 | Select an Answer | Radio Button | Radio Button is selected. | Radio Button is selected. | Pass |
| 6 | Confirm Answer | Button | Answer is accepted and correct answer is displayed. | Answer is accepted and correct answer is displayed. | Pass |
| 7 | Next Question | Button | Next question is displayed. | Next question is displayed. | Pass |
| 8 | Repeat steps 6 through 8. | N/A | Repeat steps 6-8. | Repeat steps 6-8. | Pass |
| 9 | Complete Study Session | Button | A grade will be displayed depending on amount of correct answers. | A grade will be displayed depending on amount of correct answers. | Pass |
| 10 | End application | Button | Application closes without issue. | Application closed without issue. | Pass |

Use Case 2 Requirements Met: 1, 2, 3, 4, 6, 7

|  |  |
| --- | --- |
| **Test Scenario** | Use Case 3 |
| **Test Scenario Description** | History Study Session |
| **Test Environment** | Ubuntu 16.04 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Action Performed | Input | Expected Result | Actual Result | Results |
| 1 | Click study.jar to run application | N/A | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Pass |
| 2 | Select History Subject | Dropdown Menu | The subject will now be set to Geography. | The subject is set to Geography. | Pass |
| 3 | Set quantity of questions to 3 | Textfield | The textfield should be populated with the number 5. | The textfield is populated with the number 5. | Pass |
| 4 | Press Start | Button | Questions begin to be displayed. | Questions begin to be displayed. | Pass |
| 5 | Select an Answer | Radio Button | Radio Button is selected. | Radio Button is selected. | Pass |
| 6 | Confirm Answer | Button | Answer is accepted and correct answer is displayed. | Answer is accepted and correct answer is displayed. | Pass |
| 7 | Next Question | Button | Next question is displayed. | Next question is displayed. | Pass |
| 8 | Repeat steps 6 through 8. | N/A | Repeat steps 6-8. | Repeat steps 6-8. | Pass |
| 9 | Complete Study Session | Button | A grade will be displayed depending on amount of correct answers. | A grade will be displayed depending on amount of correct answers. | Pass |
| 10 | End application | Button | Application closes without issue. | Application closed without issue. | Pass |

Use Case 3 Requirements Met: 1, 2, 3, 4, 6, 7

|  |  |
| --- | --- |
| **Test Scenario** | Error Handling 1 |
| **Test Scenario Description** | Demonstrate Application handles errors |
| **Test Environment** | Windows 10 Pro |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Action Performed | Input | Expected Result | Actual Result | Results |
| 1 | Click study.jar to run application | N/A | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Java GUI opens and presents user with subject choice drop down, quantity of questions available, a textfield to enter amount of questions to study and a Start button. | Pass |
| 2 | Select a Subject | Dropdown Menu | The subject will now be set to Geography. | The subject is set to Geography. | Pass |
| 3 | Press Start | Button | Without entering a number of questions press start and a popup notice alerting user appears. | Without entering a number of questions press start and a popup notice alerting user appears. | Pass |
| 4 | Enter number of questions | Textfield | Enter 11 in the textfield and 11 is displayed. | Entered 11 into the number of questions to study field. | Pass |
| 5 | Test max number of questions | Textfield | A popup alerting the user should be displayed. | A popup informing the user that 11 is an invalid number appears. | Pass |
| 6 | Set quantity of questions | Textfield | Enter 1 in the textfield. | Enter 1 in the textfield. | Pass |
| 7 | Press Start | Button | Question populates the GUI. | Question populates the GUI. | Pass |
| 8 | Press Next | Button | Without selecting an answer, press next. A popup should appear alerting the user. | A popup appears alerting the user of lack of answer selection. | Pass |
| 9 | Select an answer | Radio Button | Radio button for select answer is populated. | Radio button for select answer is populated. | Pass |
| 10 | Submit answer | Button | Answer is submitted and correct answer displayed. | Answer is submitted and correct answer displayed. | Pass |
| 11 | Turn in answer | Button | Answer is turned in and scored. | Answer is turned in and scored. | Pass |
| 12 | Exit application | Button | Application closes without issue. | Application closes without issue. | Pass |

Error Handling 1 Requirements Met: 8

Test Case Images:

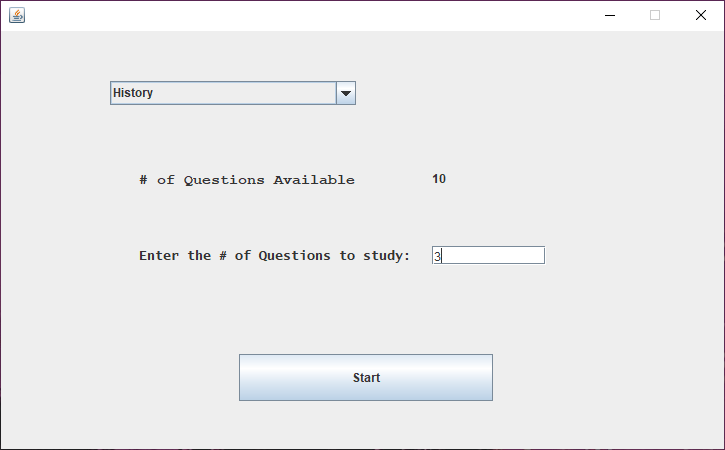


Figure 1 – Use Case 1 Setup

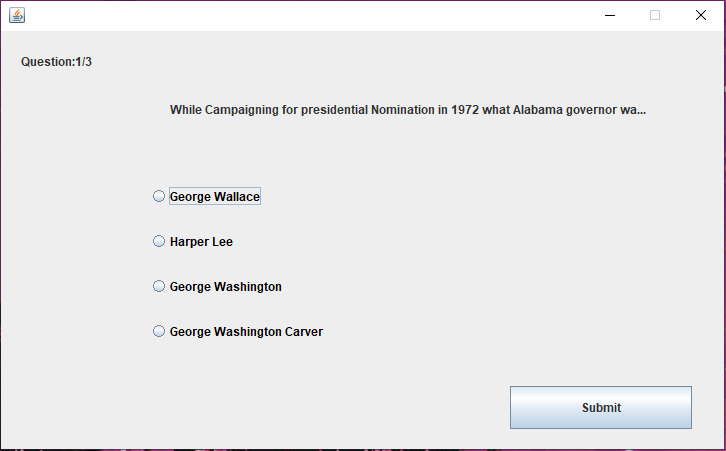


Figure 2 – Question 1/3 without an answer selected for Use Case 1

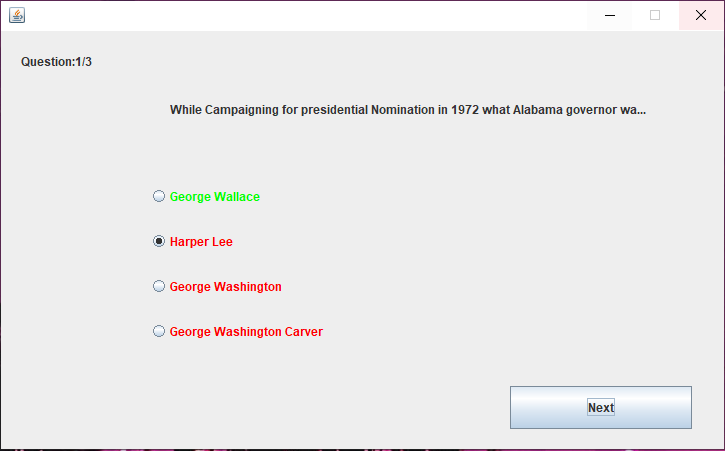


Figure 3 – Question 1/3 with an answer submitted

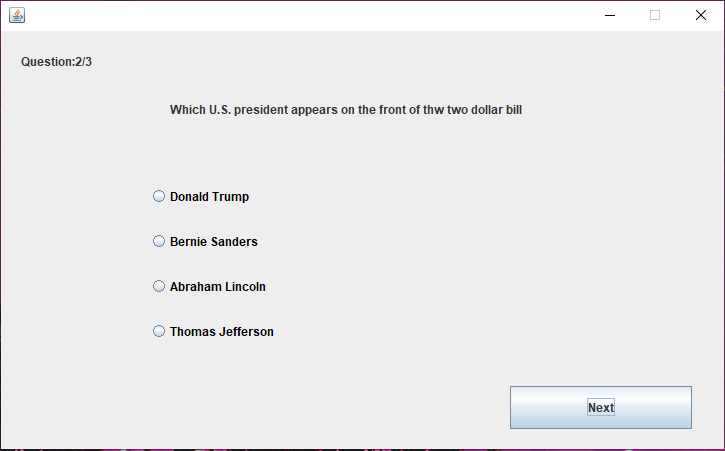


Figure 4 – Question 2/3 without an answer submitted for Use Case 1

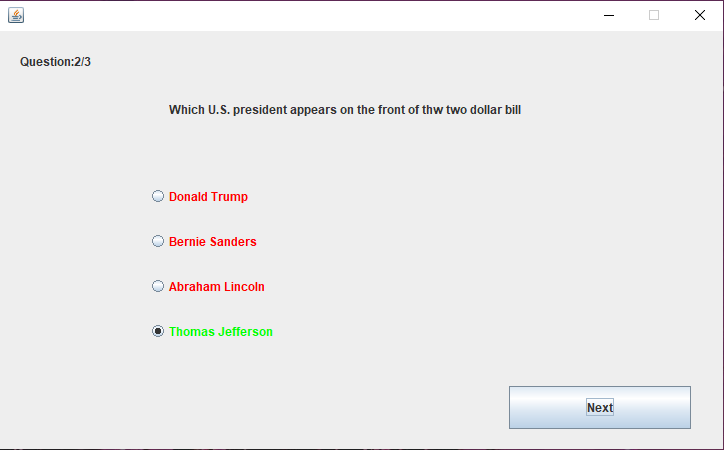


Figure 5 – Question 2/3 with an answer submitted for Use Case 1

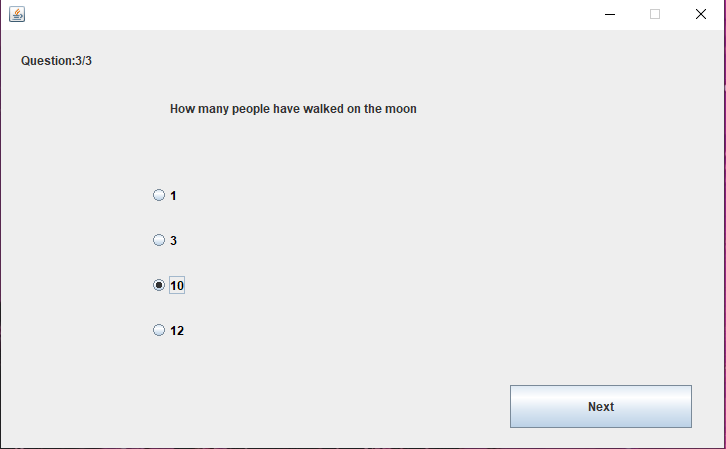


Figure 6 – Question 3/3 with an answer selected but not submitted for Use Case 1

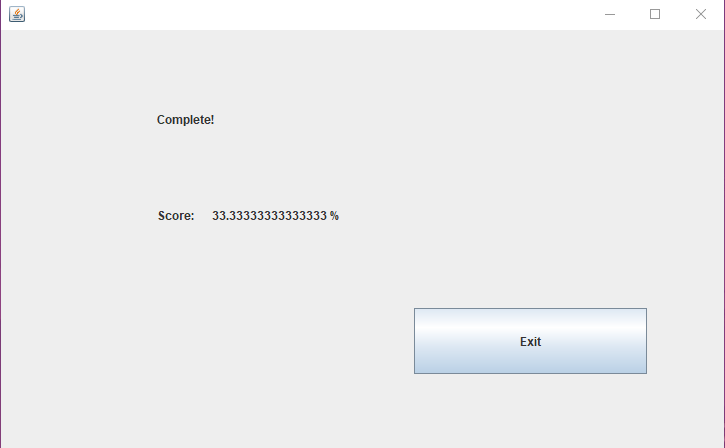


Figure 7 – Final score for Use Case 1 was 1 out of 3

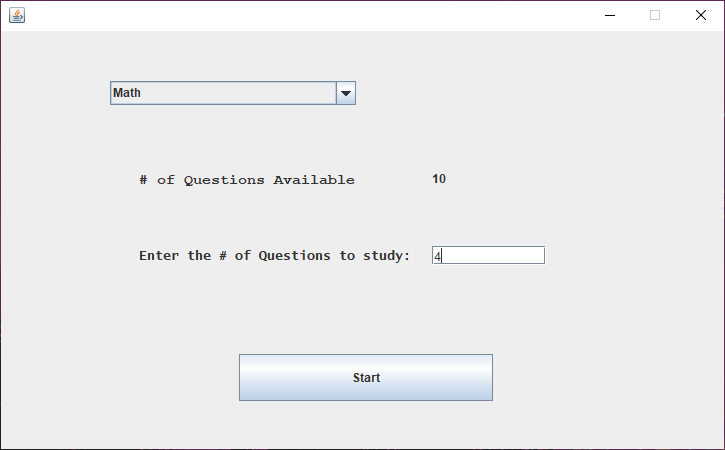


Figure 8 – Use Case 2 Setup

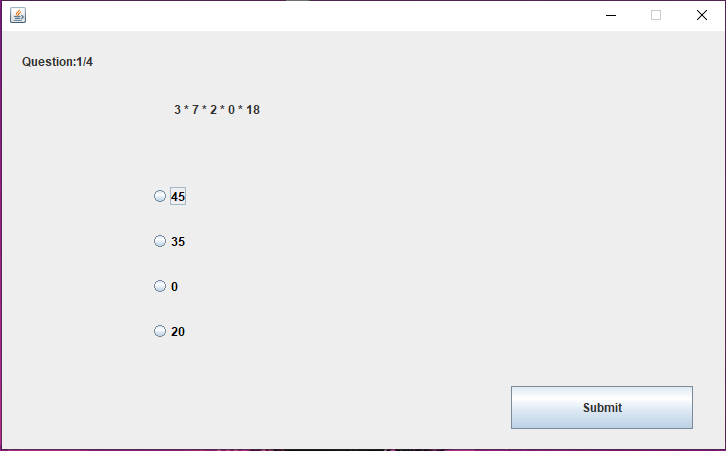


Figure 9 – Question 1/4 for Use Case 2 with no answer selected for Use Case 2

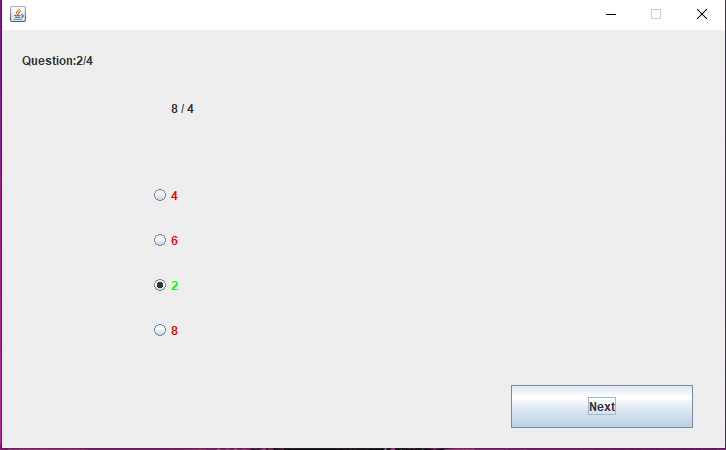


Figure 10 – Question 2/4 with the correct answer submitted for Use Case 2

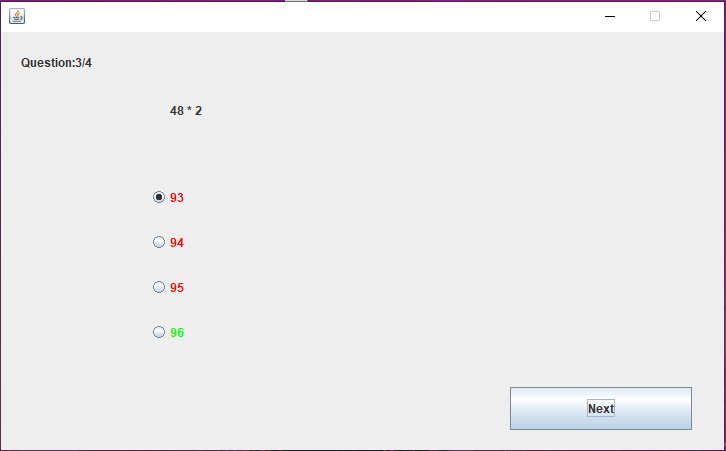


Figure 11 – Question 3/4 with the correct answer submitted for Use Case 2

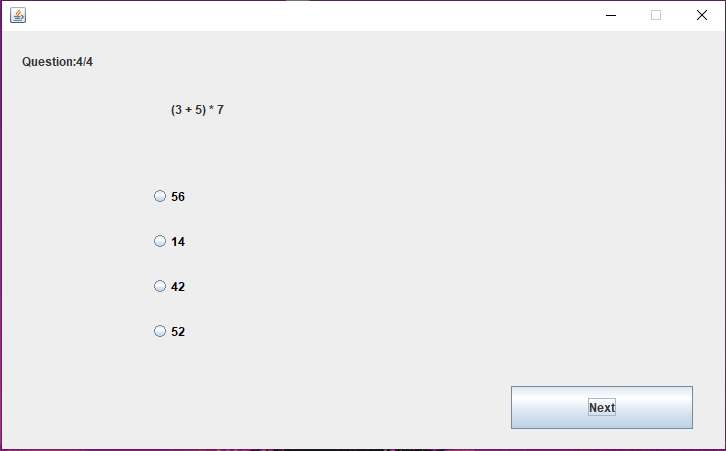


Figure 12 – Question 4/4 with no answer selected for Use Case 2

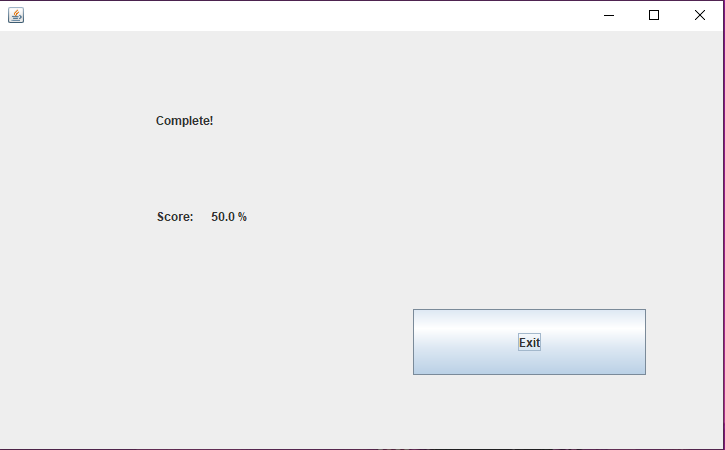
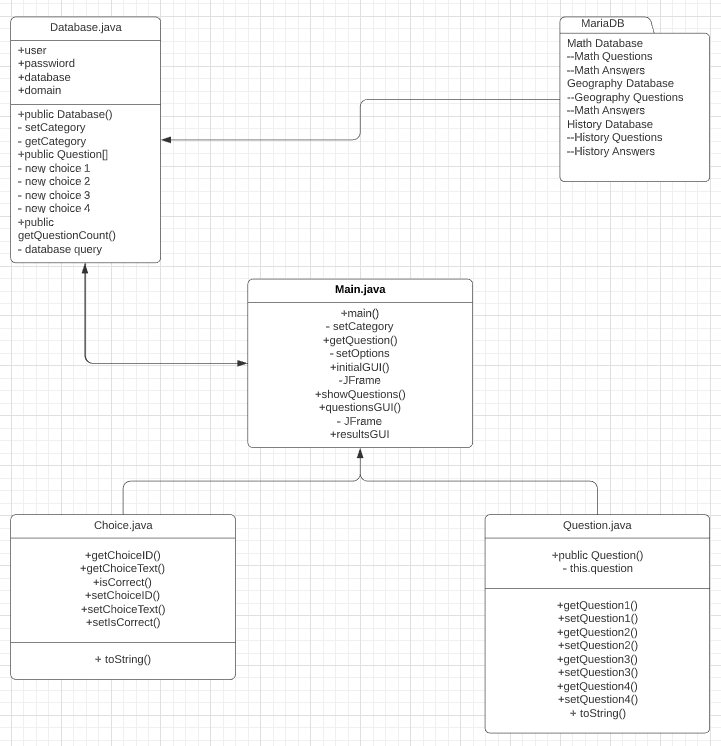


Figure 13 – Final score for Use Case 2 was 2/4

1. **Design and Alternate Designs**

**UML Diagram:**



1. **Development History**

In week 2 after the group was finalized, each member submitted ideas for the Modular Study Application. The pool of features was rather large and fairly unrealistic to finish in the span of the class. We decided a Middleman class would handle the lifting of the database and the GUI and the database class and GUI class would be fairly light. This proved to be hard to pull off.

We shifted the design to more code in the GUI yet handling logic in the GUI code made for some bulky and unnecessary coding. It was at this point we chose to create Main.java to handle the initial GUI and logic as well as the database query. Database.java was created to maintain the connection with the database enabling Main to make the queries for the questions and answers.

1. **Conclusions, Lessons Learned and Future Improvements**

In summary, the project succeeded with the goal to provide a simple and easy-to-use self-quiz program for a variety of subjects. Extra features such as the ability to restart within the GUI and store user accounts on the database that were password protected were not met due to time constraints and programming complexity.

**Conclusions:**

The program was a success, it fulfilled the majority of initial requirements and provides the core concept of what the project was initially set out to achieve. Overall the initial proposal of the project with its included features was slightly more ambitious than what was realistic for the given time frame. However, this provided a clear and thorough exploration of what features were needed and what features were not necessary.

**Lessons Learned:**

Communication was a key factor in the success of the project, as well as team members being individually invested and putting forth quality effort. The only things not achieved were the initial proposal of having user accounts, which was scrapped early on due to programming complexity, and then later the removal of a restart button due to limitations of the Java GUI components working with the overall program structure.

**Future Improvements:**

There were many issues our team ran into throughout the eight weeks. There were some ambitious ideas in the beginning that can still be applied to this project for the future. A login screen that will save a User’s session and track their scores is an area for improvement. The ability for the program to reset and allow the User to test again without exiting the application is another area for improvement. Another improvement would be to add another option on the initial window for the User to add questions to the database. This would be a great customization feature for the User.