Jamaine Drakes 400005037

This document outlines the steps taken and special features included, while creating the graphical user interface (GUI)

**Project B DBMS**

**Write Up**

Contents

[**Introduction** 2](#_Toc119587194)

[**Home** 3](#_Toc119587195)

[**Display Student Information** 5](#_Toc119587196)

[**Data Entry** 6](#_Toc119587197)

[**Data Search and Results** 8](#_Toc119587198)

[**Data Modification** 10](#_Toc119587199)

[**Supporting Data Access & Integrity:** 11](#_Toc119587200)

[**Conclusion** 12](#_Toc119587201)

[**Appendix A** 13](#_Toc119587202)

[**Appendix B** 14](#_Toc119587203)

# **Introduction**

The web application provided is a functioning graphical user interface (GUI) to help *The Higher Learning Academy* with their Textbook Loan Scheme. The GUI facilitates handling this system which will lead to improved productivity and overall, greater satisfaction amongst users. The element chosen for this project was the student entity wherein the GUI would make it much easier to manage the student relation in the database. This is the case since the information on the web application is presented in an intuitive and user-friendly manner which eliminates the need for users to have technical knowledge to achieve success with the application. This report was written to outline the possibilities of the application as well as to showcase the application in use and demonstrate its effectiveness. In addition, all special features will be discussed, along with the reason behind including them with respect to the users.

# **Home**

1. **Screenshot:**

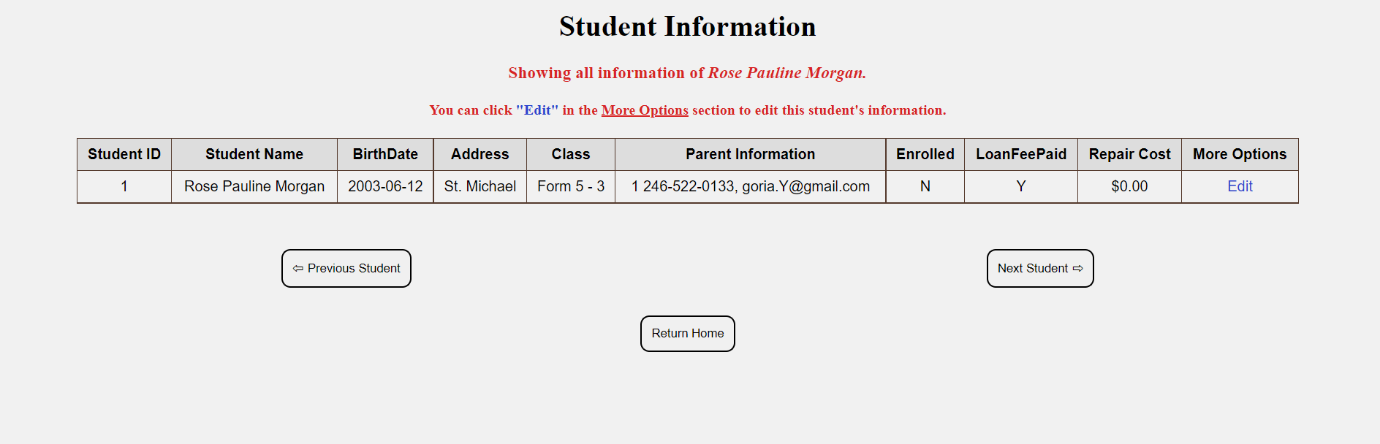
**Table

Description automatically generated with low confidence**Screenshot of index.php

1. **Description:** This is the first page the user sees and provides a means for the users to gain access to the different functions of the web application. The page displays all students currently in the database as well as links to all other pages that are accessible to the users.
2. **Importance:** The purpose of this page as mentioned before was to provide the users with a way to access the other features of the application where they can manipulate the student relation in the database. However, it also serves as a way to greet our users and give them a warm welcome to the application.
3. **Special Features:** Clear instructions are provided to aid in the user’s understanding of the application. Moreover, all clickable elements on the page have some distinguishing mark, be it a different colour or a change in its shape, to easily notify the users that it has a special function. An example of this would be if you hover over the add or search button it will change its colour and border slightly. Please Note: this special feature on the buttons is the same on all other pages and will not be mentioned any further to avoid repetition.

# **Display Student Information**

1. **Screenshot:**

****Screenshot of studentInformation.php

1. **Description:** This page exists to show all the information about a selected student. The information is presented in an extremely efficient and uncomplicated format for the users’ sake.
2. **Importance:** This page was needed to avoid a clustered view for the users. It provides a separate and isolated place where the users can carefully examine a student’s information and make decisions without having too much information on the page.
3. **Special Features:** The edit link in the More Options column of the table provides a quick way for the users to go to the edit page to update a student’s information once they have looked at the information. Furthermore, the ‘Next Student’ and ‘Previous Student’ buttons allow the user to traverse the students in the database, one student at a time, if they so please. Lastly, there is a ‘Return Home’ button present on the page which allows the users to easily return to the Home Page (Home page outlined above) Please Note: the ‘Return Home’ button is present on every page other than the home page and will not be mentioned again to avoid repetition.

# **Data Entry**

1. **Screenshot:**

**Graphical user interface

Description automatically generated**Screenshot of addStudent.php

1. **Description:** If the user wants to add a student into the database, they will be given this screen. This page will allow users to easily enter the necessary information for the student they wish to add. Once the user clicks the ‘Add Student’ button and there are no errors on the form then a message will be returned, indicating whether the add has been successful or not.
2. **Importance:** Implementing a GUI for this page to allow users to add a student into the database in an effective manner since it grants the user a much more desired way to interact with the database. No technical knowledge is required to add a student when using the GUI as it is displayed using a form for data entry which the users would be accustomed to filling out.
3. **Special Features:** In the input fields there are regular expressions present to help prevent the user from entering erroneous data. In conjunction with this if the user hovers their cursor over the question marks next to the input fields, they will be able to see which characters are allowed for the given input field. This acts as a guide for the users which makes the interface even more user friendly. Furthermore, the user will not have to worry about entering a student ID number since it will be automatically generated when the user adds a student. A smaller mention can be made to the fact that once the user hovers or focuses on an input field or dropdown box it changes shape and colour to grab the user’s attention.

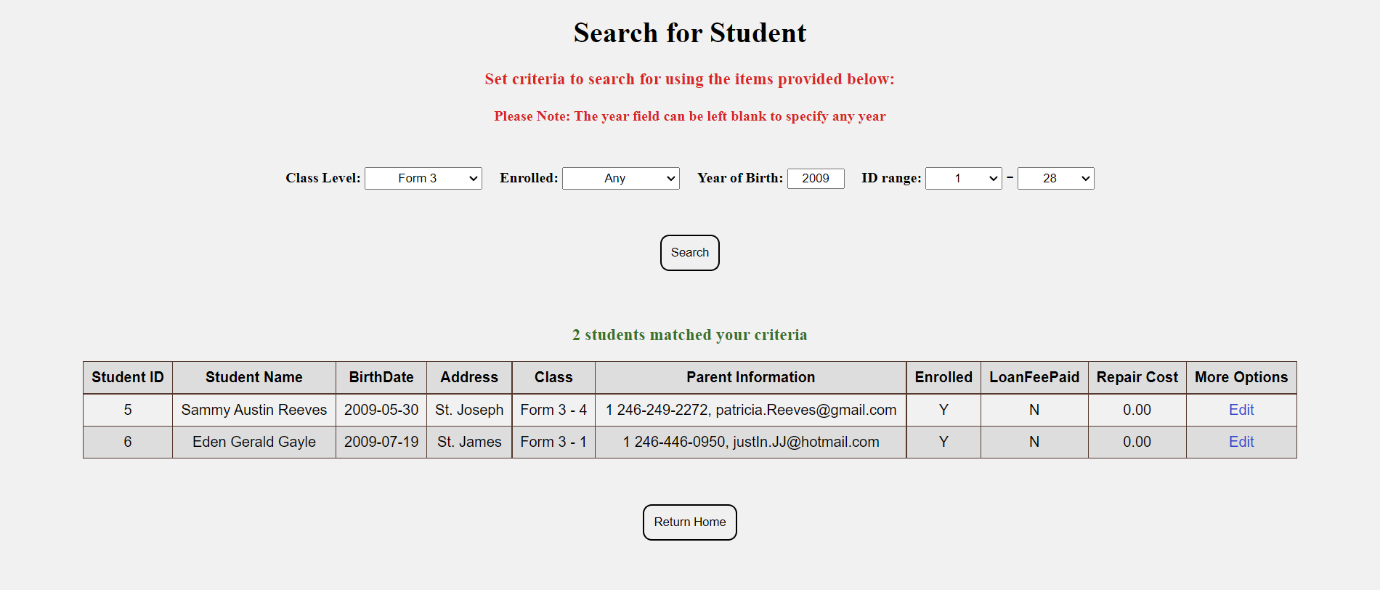
# **Data Search and Results**

1. **Screenshot:**

Graphical user interface, text, application

Description automatically generated

Screen shot of searchStudent.php (dafault)



Screenshot of searchStudent.php (with results)

1. **Description:** This page exists to allow to users to search for a student by specifying some criteria. Upon success all students that match the user’s criteria along with their information will be displayed. If no students match the criteria then an appropriate message will be shown to the users.
2. **Importance:** This page provides the user’s with a convenient way to view all the informaiton of students based on the user’s specifications. Similar to adding a student there is no need for technical knowledge to accomplish the search, thus, decreasing the need for users to actually enter the database. In addition, the search process is extremely intuitive. Therefore, implementing the data search and results using a GUI will undoubtedly increase the user experience and user satisfaction.
3. **Speaicl Features:** Most of the informaiton that can be used as criteria, was provided using a dropdown boxes which already has informaiton populated within it. This greatly reduces the data entry required from the user, and consequently reduces the chances of having errors preventing the search from succeding on the side of the user. Moreover the number of students that match the search criteria will be clearly shown to the user, and in the results there will be a link to the edit page avaliable for each student. Lastly, if no criteria was specified the search will still function, albeit with all students currently in the database being shown to the user.

# **Data Modification**

1. **Screenshot:**

A picture containing graphical user interface

Description automatically generatedScreenshot of editStudent.php

1. **Description:** If updates need to be made for a particular student then the user will be able to select the required student from a previous page and then directed to this screen. Here the user will be able to change the attributes of the student as they desire, however it must be noted that the student’s ID number, name or birthdate cannot be changed from within the GUI. Once the user clicks the ‘Edit’ button then a message will be returned indicating whether the update has been successful or not.
2. **Importance:** The GUI makes it much easier for the user to see what they are doing as opposed to trying to accomplish the same task in the mySQL console. This is extremely crucial when it comes to updating a database, thus, a GUI is a much better way to approach this, especially for someone who lacks the understanding of the mySQL console.
3. **Special Features:** Firstly, all of the student’s current informaiton will be displayed in the appropriate fields upon the user arriving at the page. This has two man advantages, the first one being that if only a minor change needs to be made to the students information then the user will simply be able to insert that required informaiton into what is already there, thus decreasing the work for the user. The second advantage is that it will act as sample data, allowing the user to see what informaiton and format is required for the particular input field on the page. Another special feature is that if the user does happen to enter eroneous data then they will be made aware of this situation theough a gentle message. This prevents data from entering the database that will cause issues.

# **Supporting Data Access & Integrity:**

1. **Reason For GUI:** Using a GUI for data access makes the system more convenient and more importantly, much less intimidating to use than using the mySQL console. This is important as we want our users to feel comfortable and encouraged while using our services, making the GUI a great choice. Using a GUI allows the user to accomplish many tasks inside of the database without them needing to know that they are interacting with a database, which makes this approach wonderful. The data can be presented in a format that the users are used to seeing, for example a table or form, while the backend would perform the more complicated parts of the process. Moreover, the GUI makes it easier to guide users on what to do by leaving several cues and hints throughout the application to keep the user on track, which greatly helps with user satisfaction.
2. **Data Integrity Maintenance:** The main way this was accomplished during the design of this GUI was the by actively decreasing the input necessary from the user. This was the main goal since it reduced the chances of problematic data being sent into the database which can obstruct the database in catastrophic ways. If this were to happen then it would not only be hard for the programmers to deal with but the users would no longer be able to use the GUI effectively, which is a situation that must be avoided. Another way integrity was maintained was by not allowing the user to alter the primary key or the student entity in any way throughout the use of the GUI. In addition, no deletions of records can be made into the database through the GUI. These restrictions on what could be done on the GUI were necessary to ensure no damage or even irreversible change was made to the database.

# **Conclusion**

To conclude, the use of the GUI is beneficial to the users since it makes operating the Textbook Loan Scheme much more efficient and easy to handle. This is done by drastically decreasing user input when accomplishing necessary tasks while leaving most of the work to the application’s backend. This then leads to an increase in user satisfaction which is the ultimate goal.

# **Appendix A**

1. Changed the starting point of the student IDs from 1000 to 1. In addition, whenever a new student is added a query is run to get the max (largest) ID from the list of students and then add one to it creating a unique ID for the new student. If students are only added from the GUI this should not be an issue. However, if someone enters the database and manually enters an ID that is much higher than the previous highest ID then the list of sequential IDs will start from that point leaving wasted space.
2. Only the middle name can be left black when entering the information for a new student.
3. All parents’ phone numbers will start with 1 246-.
4. Changed the length of the parent email attribute in the database from VARCHAR (20) to VARCHAR (50) to accommodate larger email addresses.

# **Appendix B**

|  |  |
| --- | --- |
| **html/index.php** | |
|  |
| SELECT studid, CONCAT(FirstName, ' ', LastName) AS 'Student Name', CONCAT(ClassLevel, ' - ', ClassSuffix) AS 'Class' FROM student | Query to get the student ID, first and last name (concatenated) as well as class level and class number (concatenated) of all students in the students table in the database. |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| **html/studentInformation.php** | |  |
|  |
| SELECT studid as 'list' FROM student | This query was used to get all the ID's currently in the database. Afterwards you can see if the ID that was to this page is one that is within the database |  |
|  |
|  |
|  |
|  |
| SELECT studid, CONCAT(FirstName, ' ',MiddleName, ' ', LastName) AS 'Student Name', BirthDate, Address, CONCAT(Classlevel, ' - ', ClassSuffix) AS Class, CONCAT(ParentPhoneNumber,', ', ParentEmail) AS 'Parent Information', Enrolled, LoanFeePaid, RepairCost  FROM student WHERE studid =" . $studentID | Query to return information of the student that was requested ($studentID has the ID of the student that was requested) |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| SELECT MAX(studid) AS 'maxID', MIN(studid) AS 'minID' FROM STUDENT | Query to get the largest and smallest student ID in the database. |  |
|  |
|  |
| **html/addForm.php** | |  |
|  |
| SELECT ClassLevel FROM class | Query to get all class levels from the class relation |  |
|  |
| **resources/addHandler.php** | |  |
|  |
| SELECT (MAX(studid)+1) AS 'newID' FROM STUDENT | Query to get an ID that is one greater than the largest ID in the database |  |
|  |
|  |
| INSERT INTO student VALUES ('$newID', '$firstName', '$middleName', '$lastName', '$address', '$birthdate', '$classLevel', '$classSuffix', '$parentPhoneNumber', '$parentEmail', '$enrolled', '$loanfee', '$repairCost') | Add a record to the database using the new id generated along with the values entered by the user |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| **html/editForm.php** | |  |
|  |
| SELECT studid, CONCAT(firstname, ' ', middlename, ' ', lastname) AS Name, birthdate, ClassLevel, ClassSuffix, Enrolled, LoanFeePaid, Address, (SELECT RIGHT(ParentPhoneNumber,8) ) AS ParentNumber, ParentEmail, RepairCost FROM student WHERE studid =" . $\_REQUEST['studid'] | Query to get all the current information of the student with the ID that was requested so it can be displayed in the edit form |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| SELECT ClassLevel FROM class | Query to get all class levels from the class relation |  |
|  |
| **resources/editHandler.php** | |  |
|  |
| Update student Set … | Query to update the student with the information the user has entered. This query is dynamically created and will change depending on what the user specified. |  |
|  |
|  |
|  |
| SELECT studid, CONCAT(FirstName, ' ', LastName) AS 'name' FROM student WHERE studid = ". $\_POST['studID'] | Query to get some information of the student that was updated so it can be displayed to the user. |  |
|  |
|  |
|  |
|  |
| **html/searchStudent.php** | |  |
|  |
| SELECT ClassLevel FROM class | Query to get all class levels from the class relation |  |
|  |
| SELECT MIN(studid) AS 'minID' FROM STUDENT | query to get the smallest ID currently in the database |  |
|  |
| SELECT MAX(studid) AS 'maxID' FROM STUDENT | query to get the largest ID currently in the database |  |
|  |
| SELECT studid FROM student | Get all the students IDs in the student relation |  |
|  |
| SELECT studid, CONCAT(FirstName, ' ',MiddleName, ' ', LastName) AS 'Student Name', BirthDate, Address, CONCAT(Classlevel, ' - ', ClassSuffix) AS Class, CONCAT(ParentPhoneNumber,', ', ParentEmail) AS 'Parent Information', Enrolled, LoanFeePaid, RepairCost FROM student WHERE studid BETWEEN " . $MinID. " AND " . $MaxID … | Get all the information of the students that match the search criteria. This query is dynamically generated and will change depending on what the user entered, however it will always start like this. |  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| SELECT COUNT(studid) AS 'amount' FROM student | Get the number of students currently in the database, to see if the search results return all the students |  |
|  |
|  |