# ADDICTION ASSISTANT

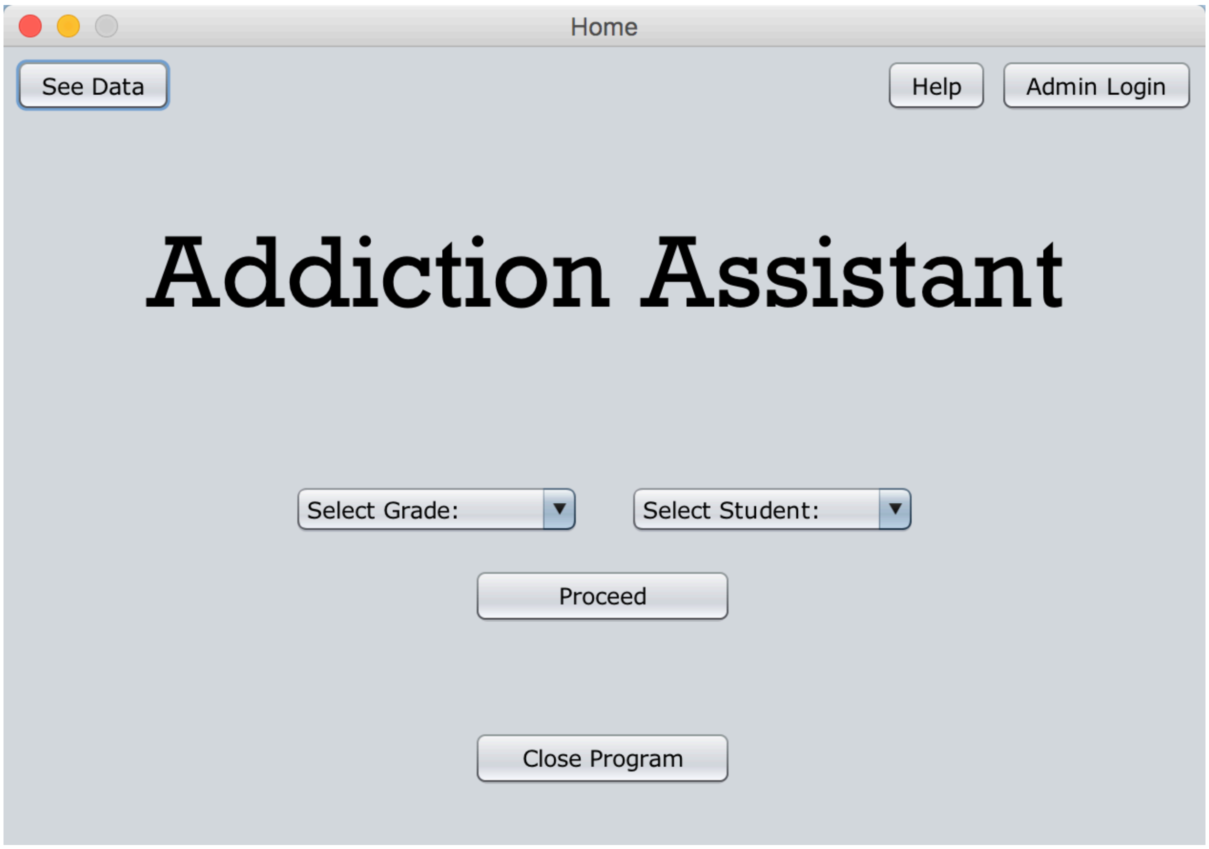
SYSTEM DESIGN DOCUMENT
PHASE 2

Luke Van Rooyen - 191094020935

## Table of Contents

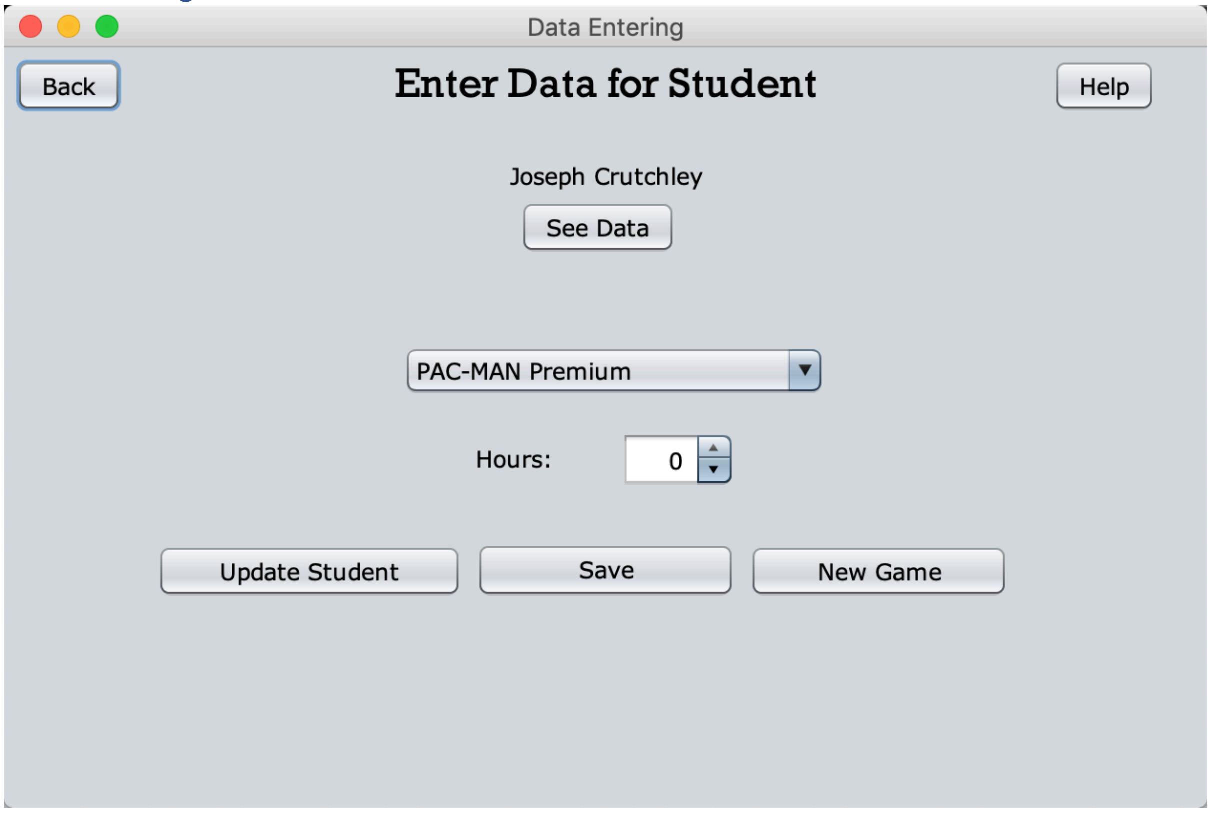
## User Interface Design

### Home Window



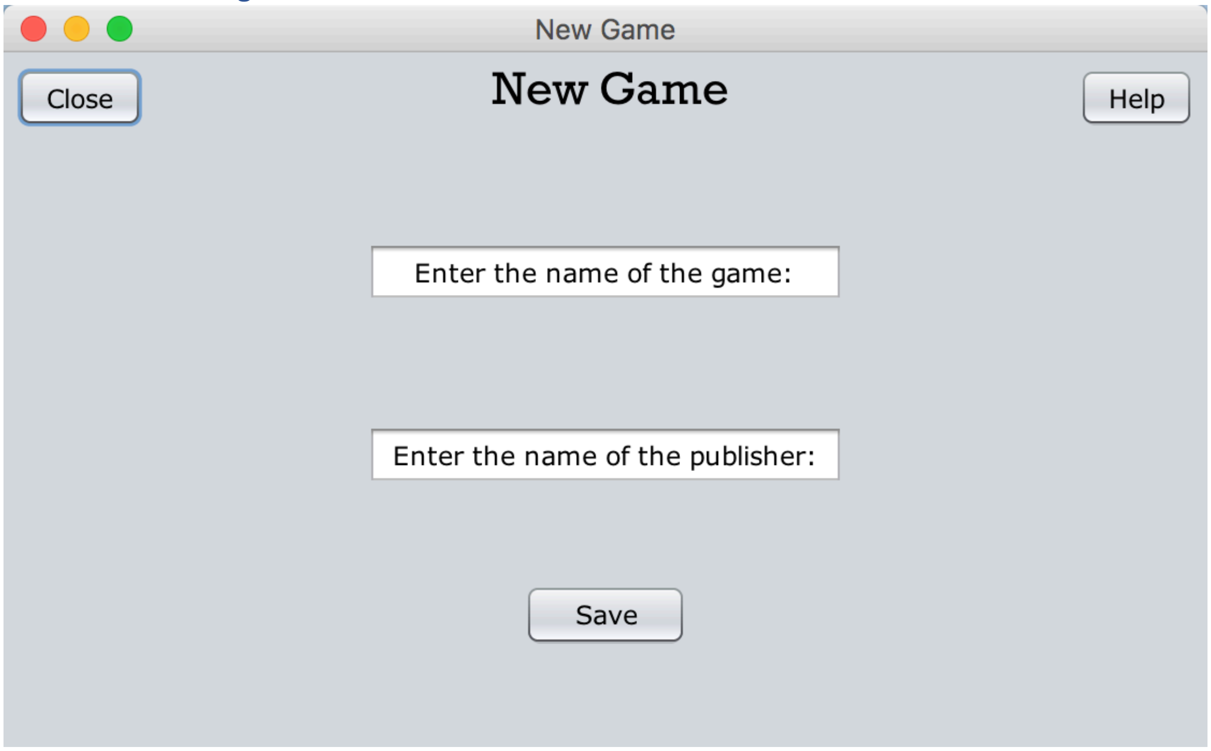
| GUI Component | Function |
|---|---|
| “See Data” Button | Takes the user to the Data Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Admin Login” Button | Takes the user to the Admin Login Window. |
| “Select Grade:” Combo Box | Allows the user to select the grade of the desired student. |
| “Select Student:” Combo Box | Allows the user to select the desired student from the grade selected. |
| “Proceed” Button | Takes the user to the Data Entering Window. |
| “Close Program” Button | Closes the program. |
| “Error” Label | Appears when an error occurs. |

## Data Entering Window



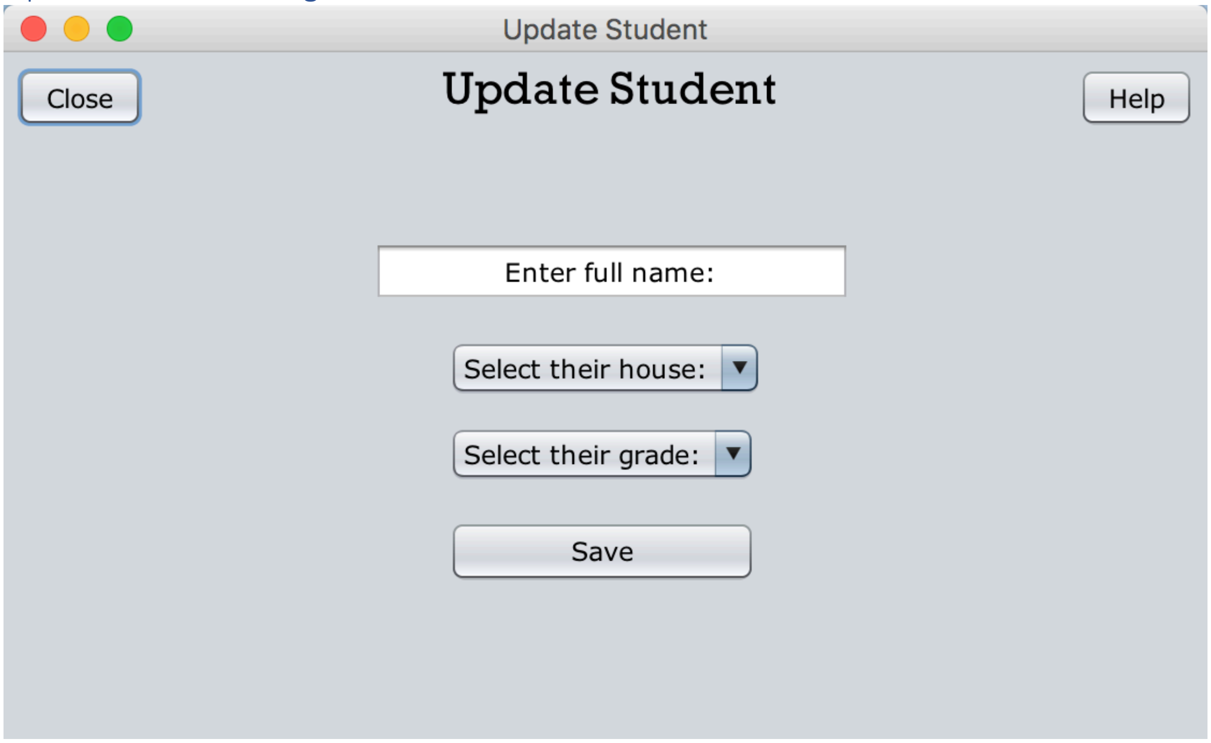
| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Home Window or the Edit Student Data Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “See Data” Button | Takes the user to the Student’s Data Dialog Box. |
| “Select Game:” Combo Box | Allows the user to select the desired game. |
| “Hours:” Spinner | Allows the user to enter the desired number of hours. |
| “Save” Button | Adds the game and hours to the selected student’s total hours in the data table. |
| “Update Student” Button | Takes the user to the Update Student Dialog Box. |
| “New Game” Button | Takes the user to the New Game Dialog Box. |
| “Error” Label | Appears when an error occurs. |

## New Game Dialog Box



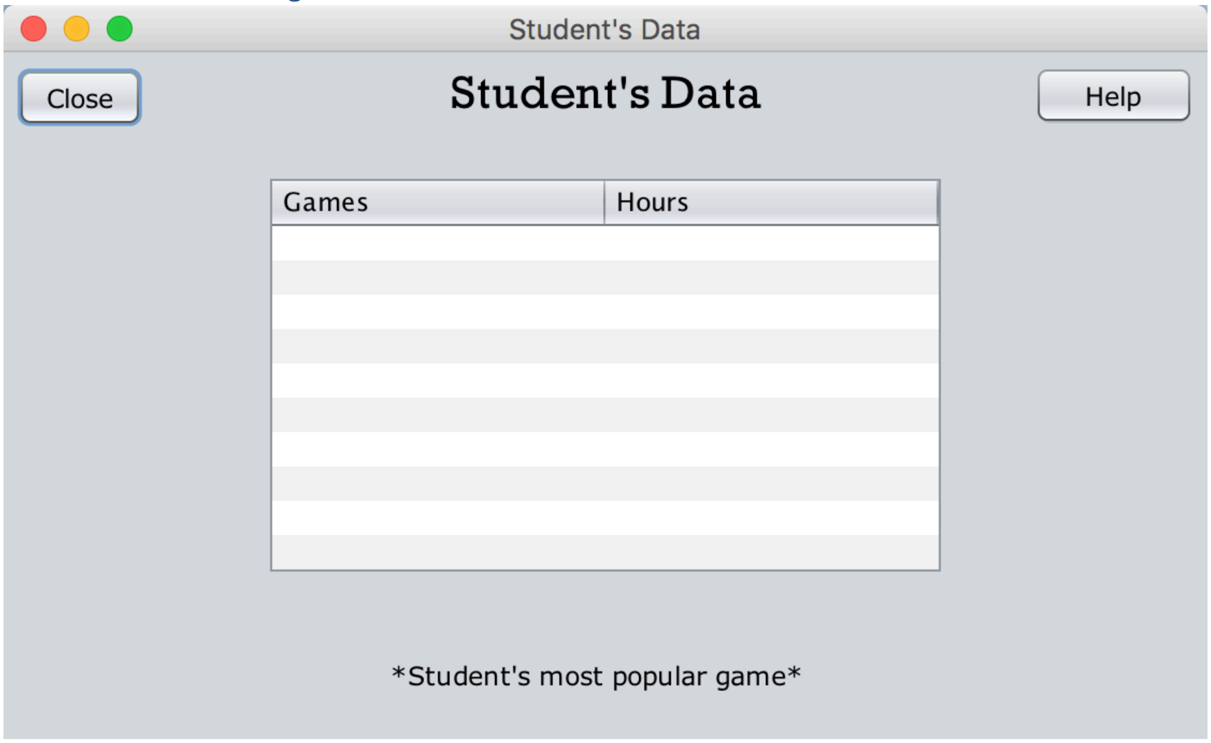
| GUI Component | Function |
|---|---|
| "Close" Button | Closes the current dialog box. |
| "Help" Button | Opens the Help Dialog Box at the relevant place. |
| "Game" Text Field | Allows the user to enter the name of a new game. |
| "Publisher" Text Field | Allows the user to enter the name of a new publisher. |
| "Save" Button | Saves the new game and publisher to the games table. |
| "Error" Label | Appears when an error occurs. |

## Update Student Dialog Box



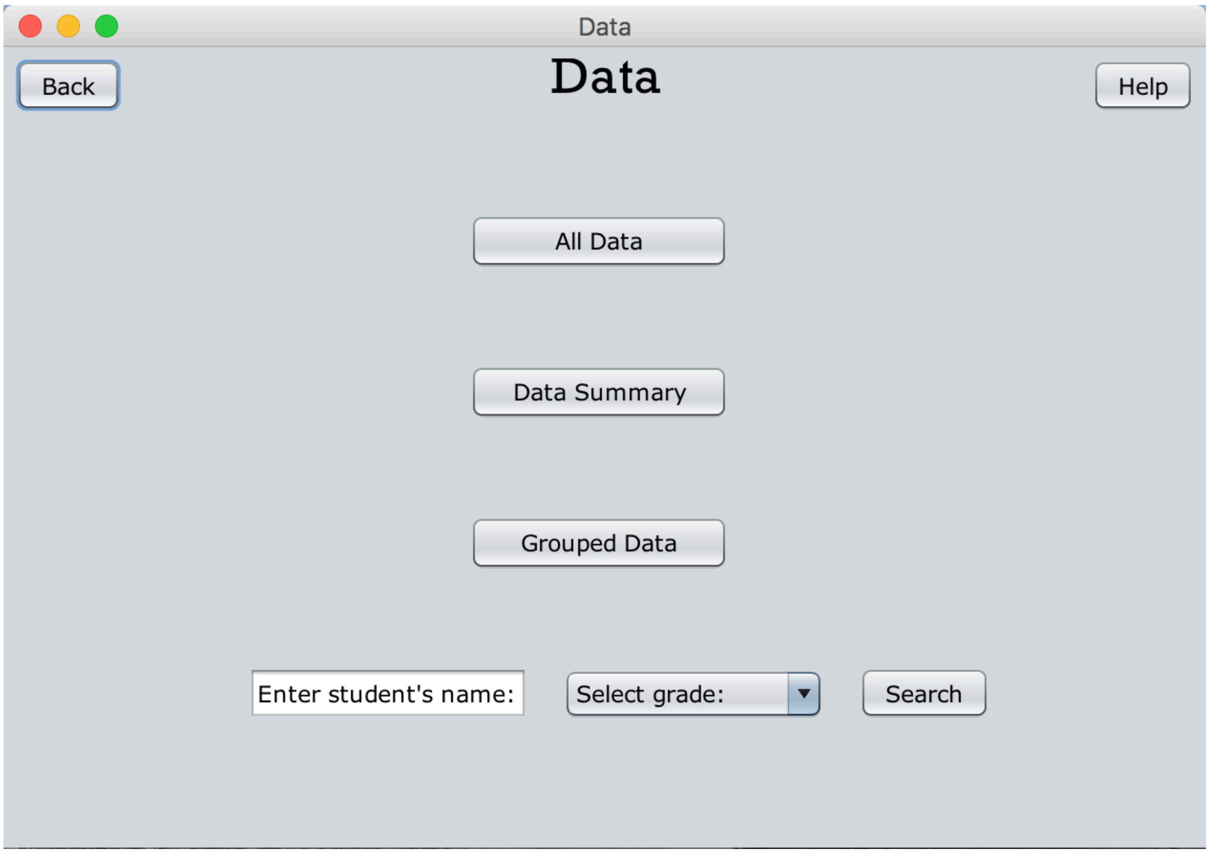
| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Full Name” Text Field | Allows the user to edit the student’s full name. |
| “Select House:” Combo Box | Allows the user to edit the student’s house. |
| “Select Grade:” Combo Box | Allows the user to edit the student’s grade. |
| “Save” Button | Saves any changes to the student table. |
| “Error” Label | Appears when an error occurs. |

## Student’s Data Dialog Box



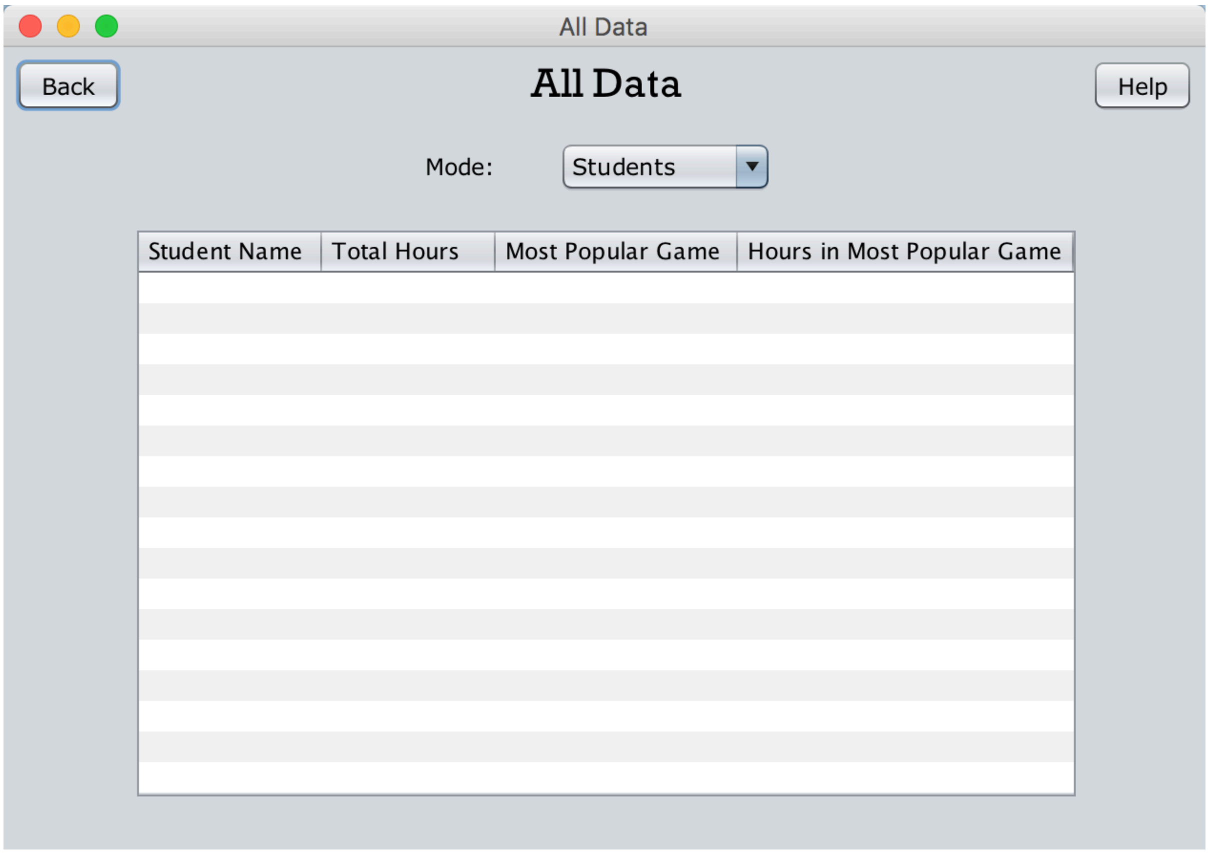
| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| Data Table | Displays the games and hours for the current student to the user. |
| “Most Popular Game” Label | Displays the current student’s most popular game. |

## Data Window



| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Home Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “All Data” Button | Takes the user to the All Data Window. |
| “Data Summary” Button | Takes the user to the Data Summary Window. |
| “Grouped Data” Button | Takes the user to the Grouped Data Window. |
| “Student Name” Text Field | Allows the user to enter the desired student’s name. |
| “Select Grade:” Combo Box | Allows the user to select the grade for the desired student. |
| “Search” Button | Takes the user to the Student’s Data Dialog Box for the desired student. |
| “Error” Label | Appears when an error occurs. |

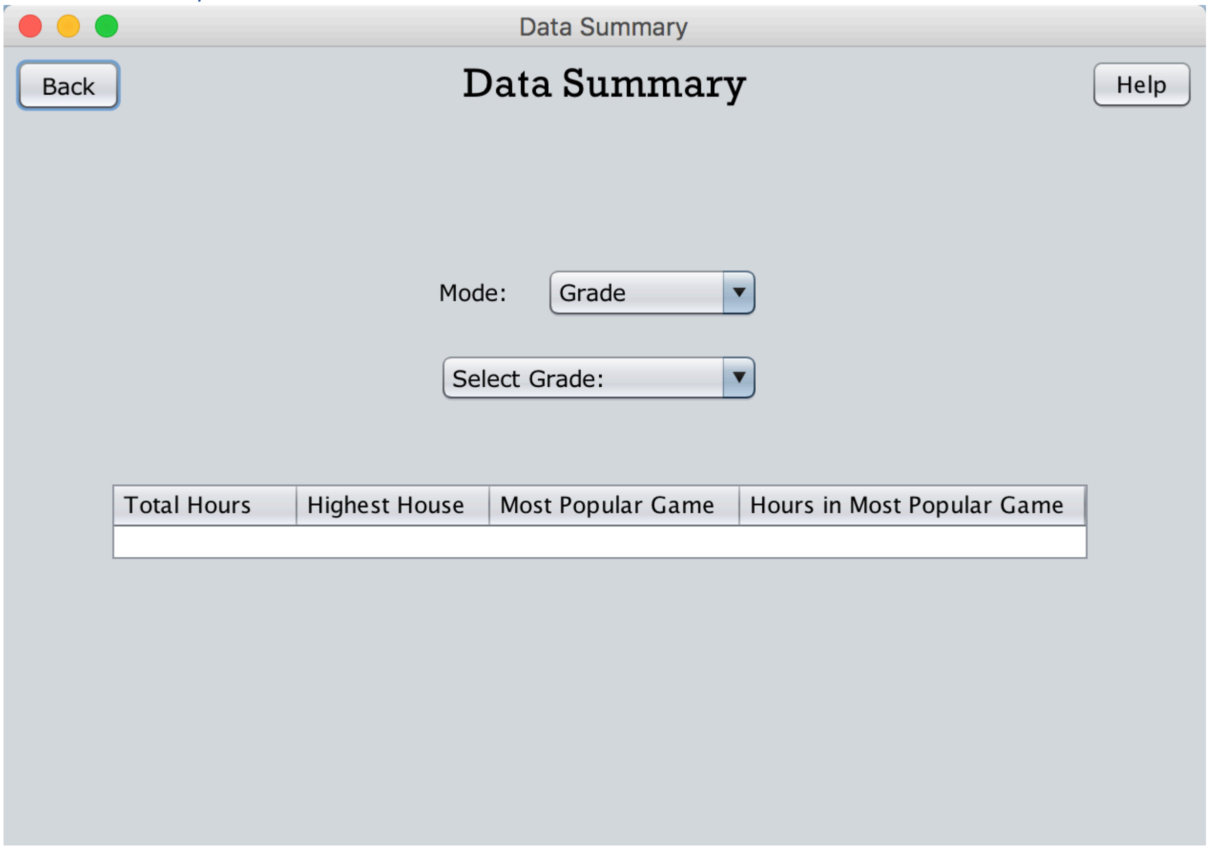
## All Data Window



| Student Name | Total Hours | Most Popular Game | Hours in Most Popular Game |
|---|---|---|---|

| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Data Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Mode” Combo Box | Allows the user to switch between “Students”, “Grades” or “Houses” mode. |
| Data Table | Displays the name, total hours, most popular game, and hours in most popular game for each student. |

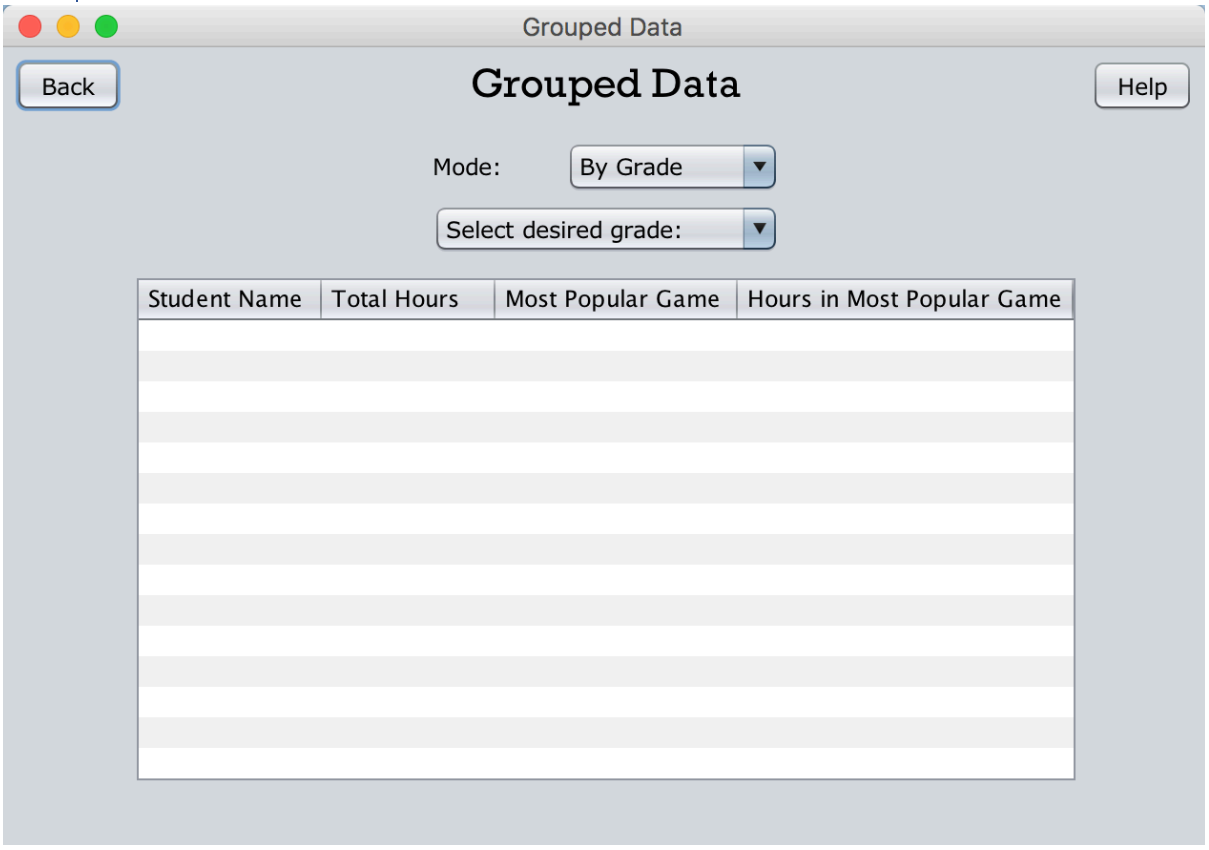
## Data Summary Window



| Total Hours | Highest House | Most Popular Game | Hours in Most Popular Game |
|---|---|---|---|
| | | | |

| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Data Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Mode” Combo Box | Allows the user to change between “Grade”, “House” and “School” mode. |
| “Select Grade:” Combo Box | Allows the user to select the desired grade. Will allow the user to select a house if in House mode. |
| Data Table | Displays the total hours, highest house or grade, most popular game, and hours in most popular game for each grade or house. |

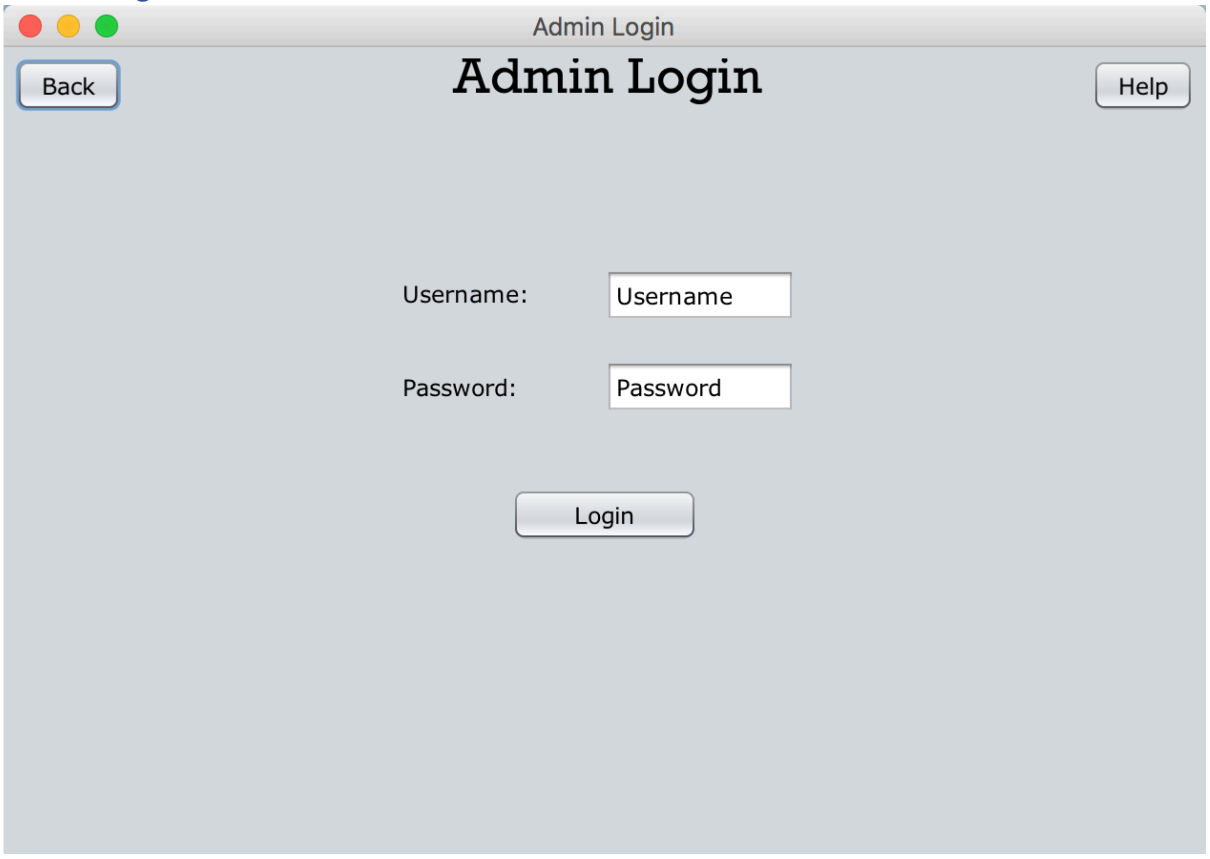
## Grouped Data Window



| Student Name | Total Hours | Most Popular Game | Hours in Most Popular Game |
|---|---|---|---|

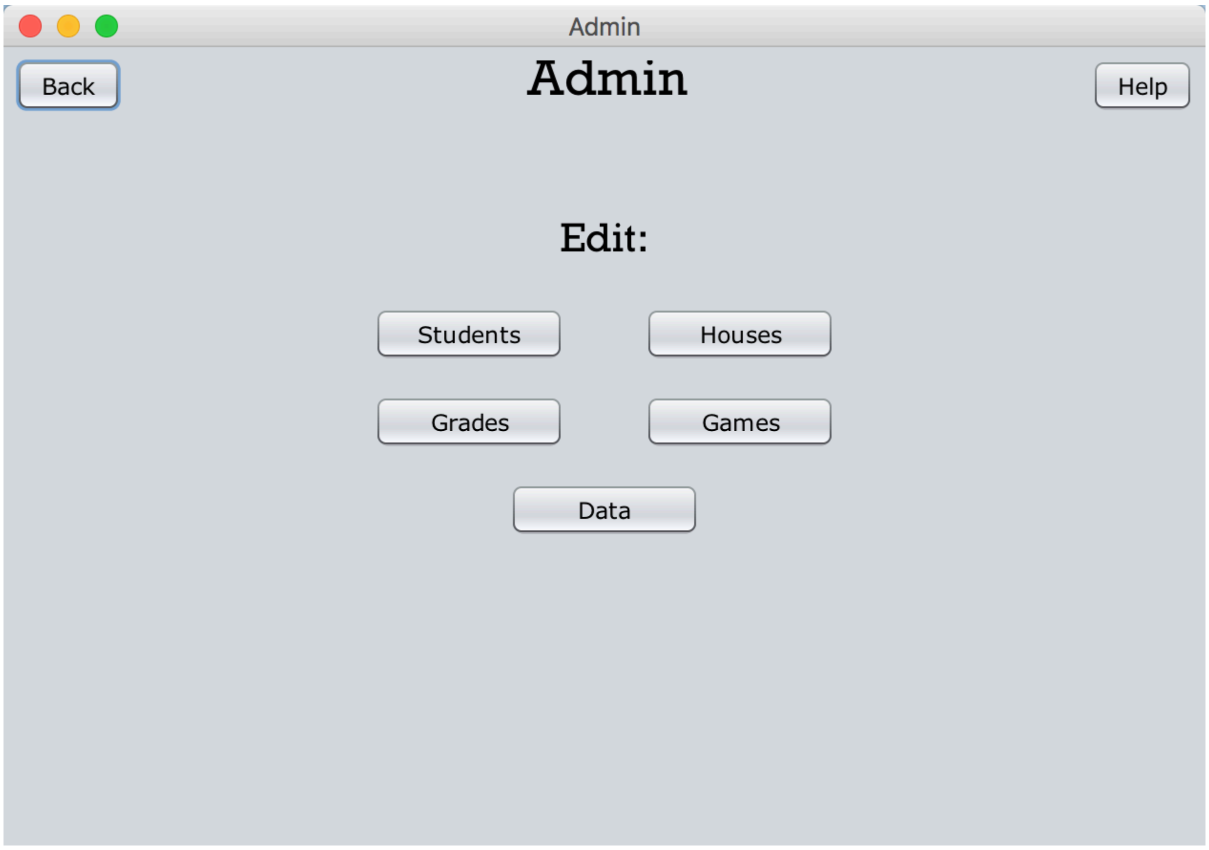
| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Data Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Mode” Combo Box | Allows the user to change between “By Grade” or “By House”. |
| “Select Desired Grade” Combo Box | Allows the user to select the grade, or house, if in By House mode, they want. |
| Data Table | Displays the name, total hours, most popular game, and hours in most popular game for each student. |

## Admin Login Window



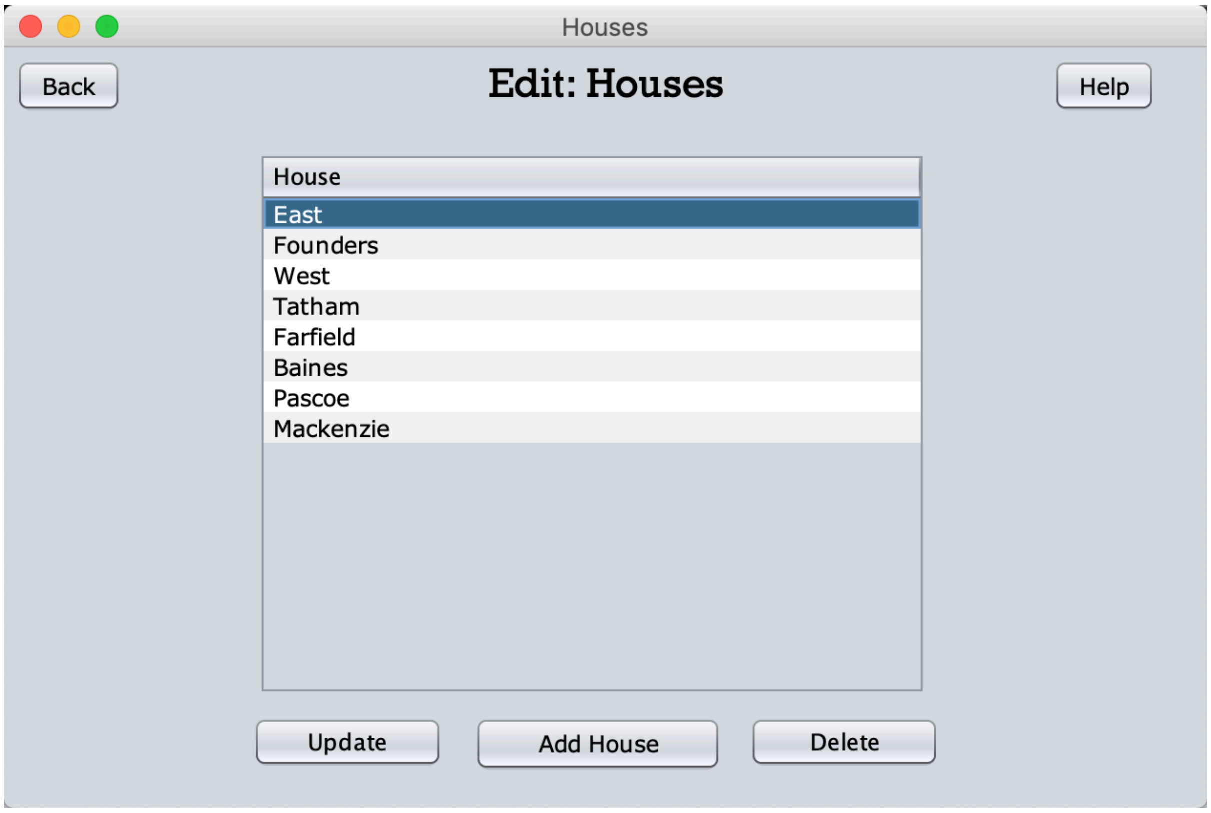
| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Home Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Username” Text Field | Allows the user to enter their username. |
| “Password” Text Field | Allows the user to enter their password. |
| “Login” Button | Takes the user to the Admin Window if the username and password are correct. |
| “Error” Label | Appears when an error occurs. |

## Admin Window



| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Home Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Students” Button | Takes the user to the Edit Students Window. |
| “Houses” Button | Takes the user to the Edit Houses Window. |
| “Grades” Button | Takes the user to the Edit Grades Window. |
| “Games” Button | Takes the user to the Edit Games Window. |
| “Data” Button | Takes the user to the Edit Data Window. |

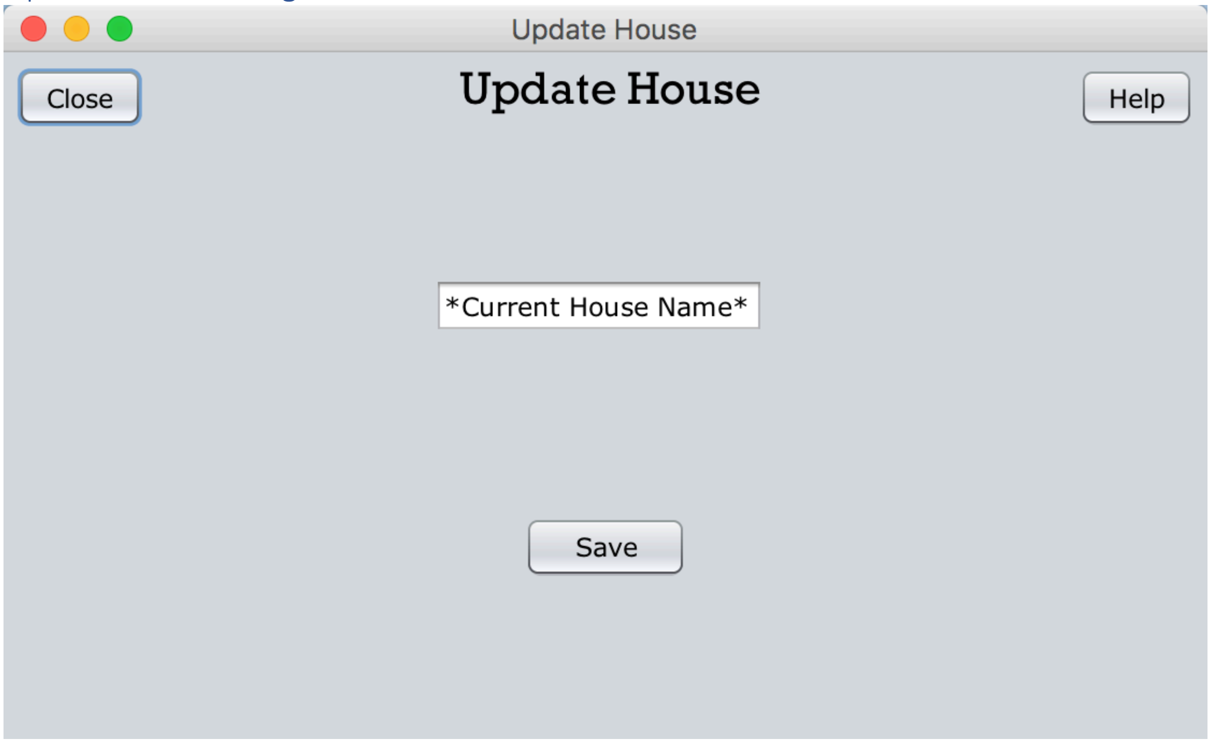
## Edit Houses Window



| House |
|---|
| East |
| Founders |
| West |
| Tatham |
| Farfield |
| Baines |
| Pascoe |
| Mackenzie |

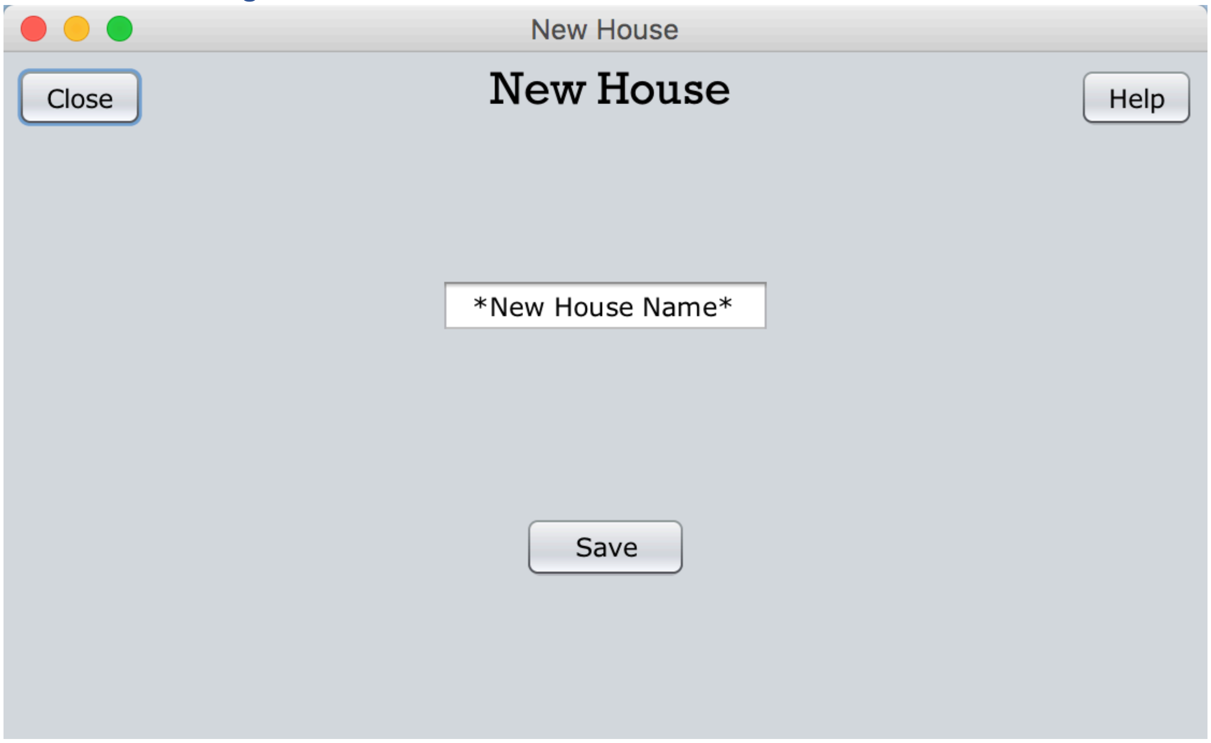
| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Admin Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| Data Table | Displays the house name. |
| “Add House” Button | Takes the user to the New House Dialog Box. |
| “Delete” Button | Appears when a House is selected and allows the user to delete the selected House. |
| “Update” Button | Appears when a House is selected and takes the user to the Update House Dialog Box. |
| “Error” Label | Appears when an error occurs. |

## Update House Dialog Box



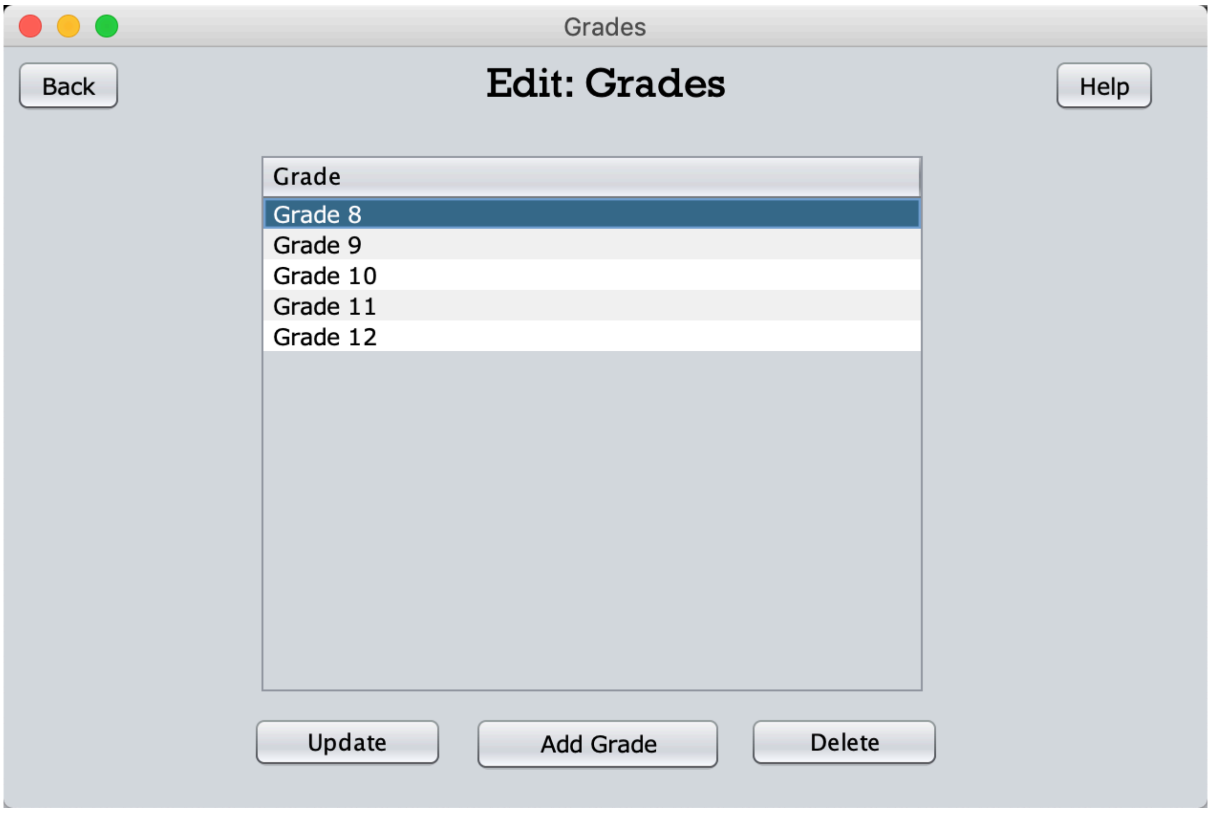
| GUI Component | Function |
|---|---|
| "Close" Button | Closes the current dialog box. |
| "Help" Button | Opens the Help Dialog Box at the relevant place. |
| "Current Name" Text Field | Allows the user to edit the current house name. |
| "Save" Button | Saves any changes to the house table. |
| "Error" Label | Appears when an error occurs. |

## New House Dialog Box



| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “New House” Text Field | Allows the user to enter the name of a new house. |
| “Save” Button | Saves any changes to the house table. |
| “Error” Label | Appears when an error occurs. |

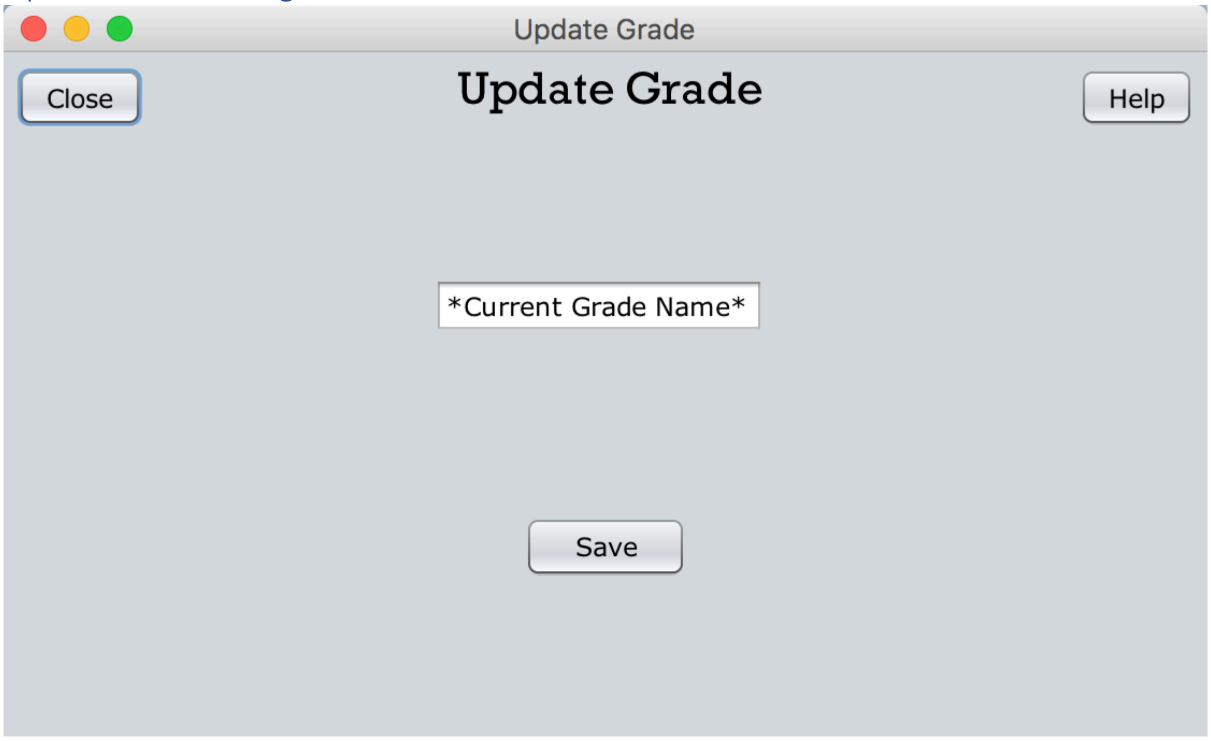
## Edit Grades Window



| Grade |
|---|
| Grade 8 |
| Grade 9 |
| Grade 10 |
| Grade 11 |
| Grade 12 |

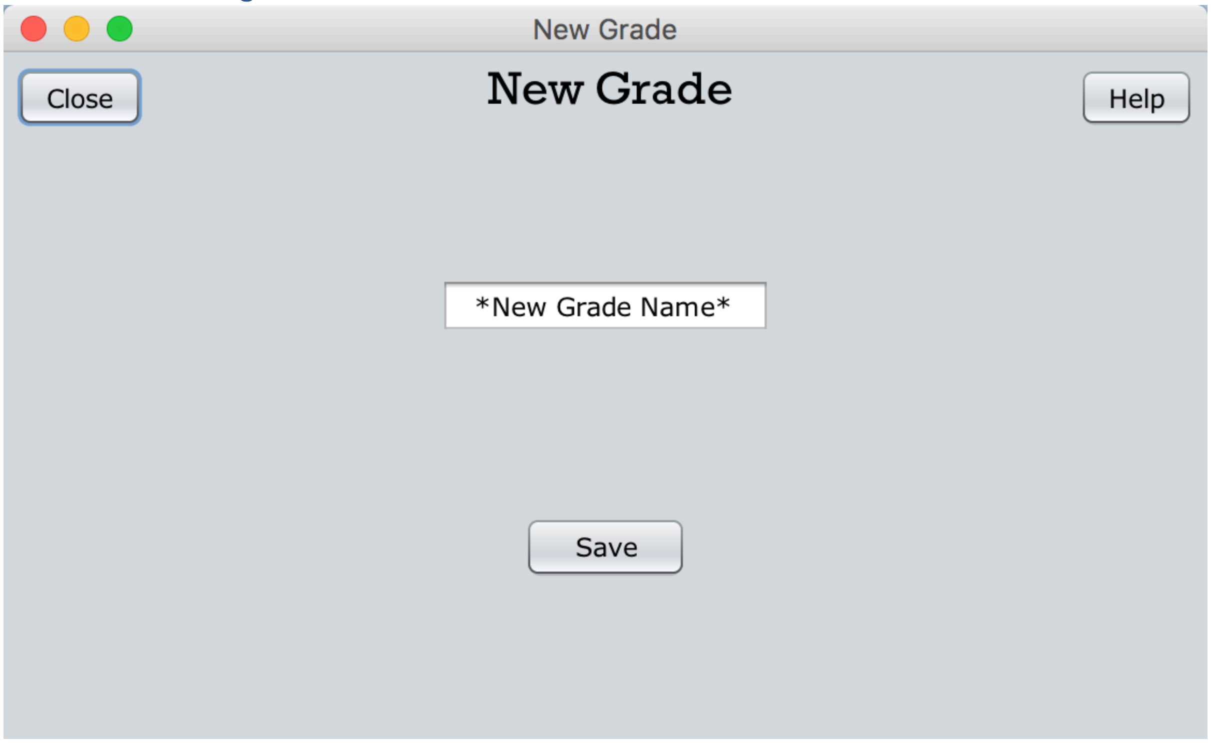
| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Admin Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| Data Table | Displays the grade name. |
| “Add Grade” Button | Takes the user to the New Grade Dialog Box. |
| “Delete” Button | Appears when a Grade is selected and allows the user to delete the selected Grade. |
| “Update” Button | Appears when a Grade is selected and takes the user to the Update Grade Dialog Box. |
| “Error” Label | Appears when an error occurs. |

## Update Grade Dialog Box



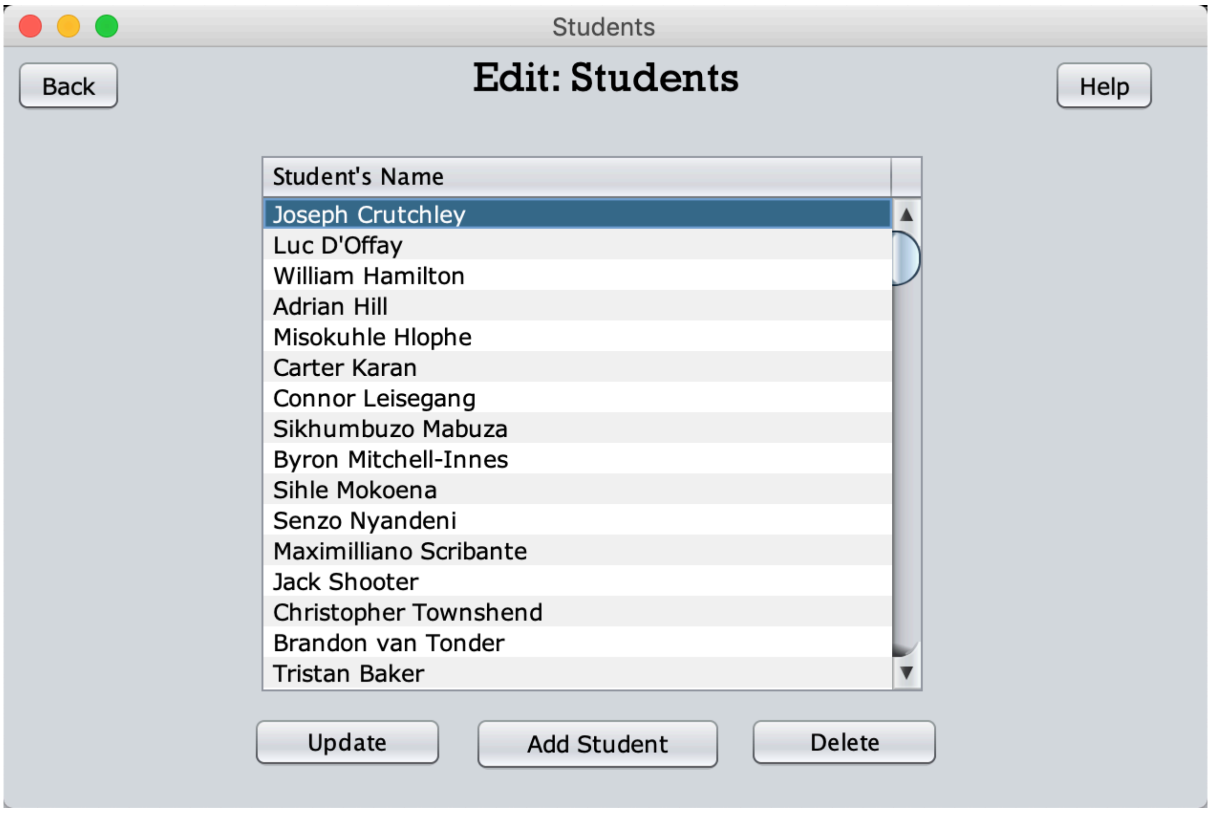
| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Current Name” Text Field | Allows the user to edit the current grade name. |
| “Save” Button | Saves any changes to the grade table. |
| “Error” Label | Appears when an error occurs. |

## New Grade Dialog Box



| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “New Grade” Text Field | Allows the user to enter the name of a new grade. |
| “Save” Button | Saves any changes to the grade table. |
| “Error” Label | Appears when an error occurs. |

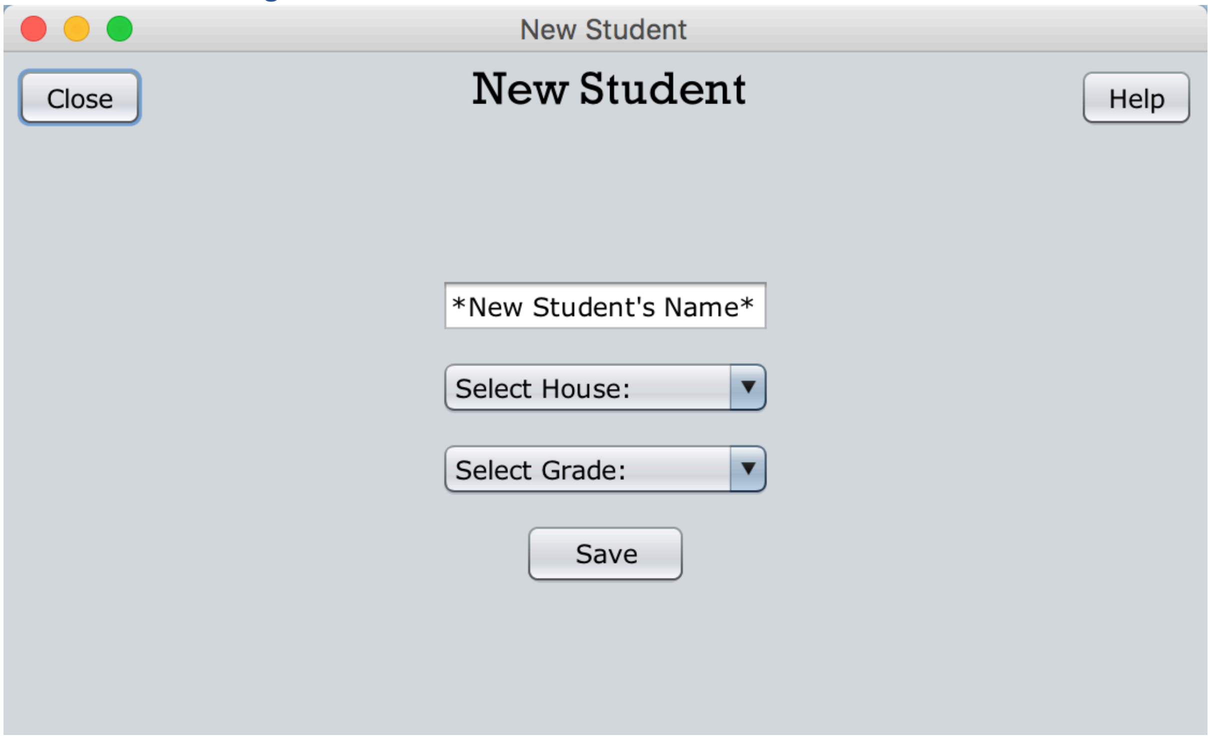
## Edit Students Window



| Student's Name |
|---|
| Joseph Crutchley |
| Luc D'Offay |
| William Hamilton |
| Adrian Hill |
| Misokuhle Hlophe |
| Carter Karan |
| Connor Leisegang |
| Sikhumbuzo Mabuza |
| Byron Mitchell-Innes |
| Sihle Mokoena |
| Senzo Nyandeni |
| Maximilliano Scribante |
| Jack Shooter |
| Christopher Townshend |
| Brandon van Tonder |
| Tristan Baker |

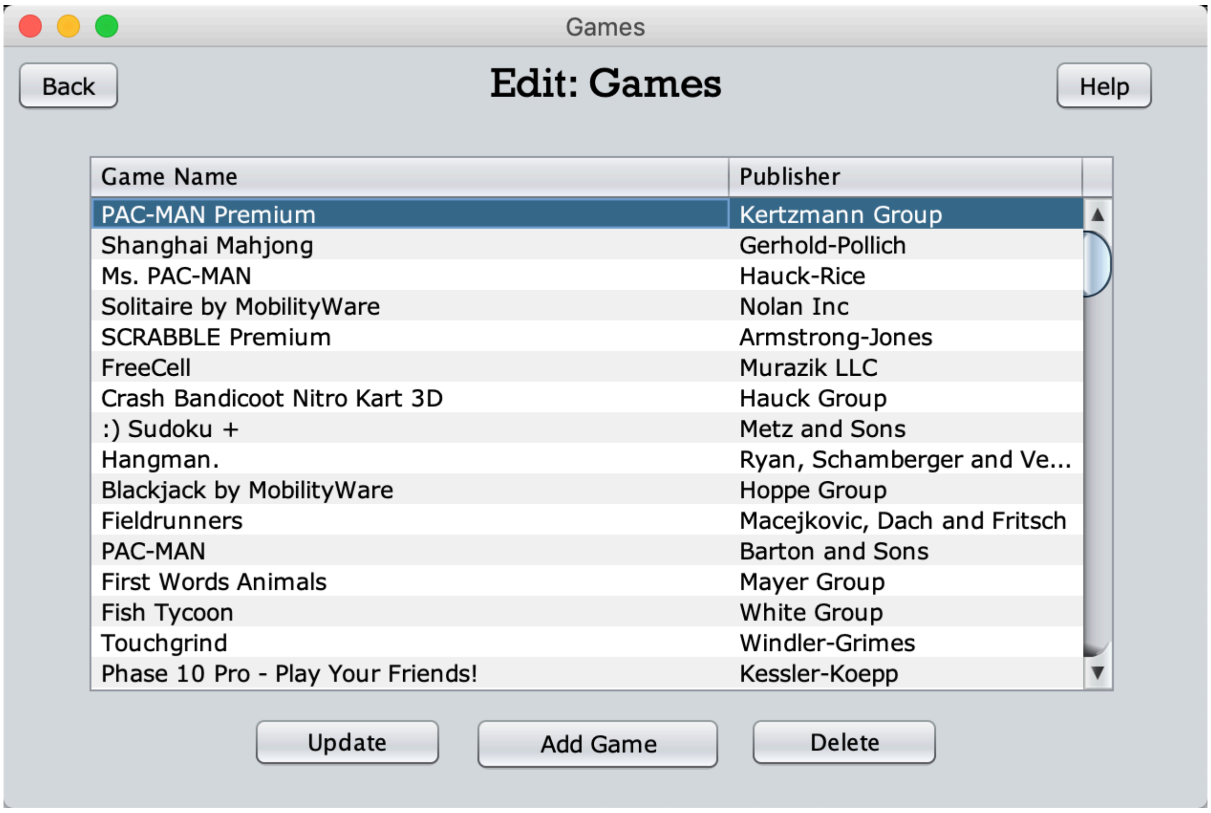
| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Admin Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| Data Table | Displays the student name. |
| “Add Student” Button | Takes the user to the New Student Dialog Box. |
| “Delete” Button | Appears when a Student is selected and allows the user to delete the selected Student. |
| “Update” Button | Appears when a Student is selected and takes the user to the Update Student Dialog Box. |
| “Error” Label | Appears when an error occurs. |

## New Student Dialog Box



| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “New Name” Text Field | Allows the user to enter the name of a new student. |
| “Select House:” Combo Box | Allows the user to select the desired house for the student. |
| “Select Grade:” Combo Box | Allows the user to select the desired grade for the student. |
| “Save” Button | Saves any changes to the student table. |
| “Error” Label | Appears when an error occurs. |

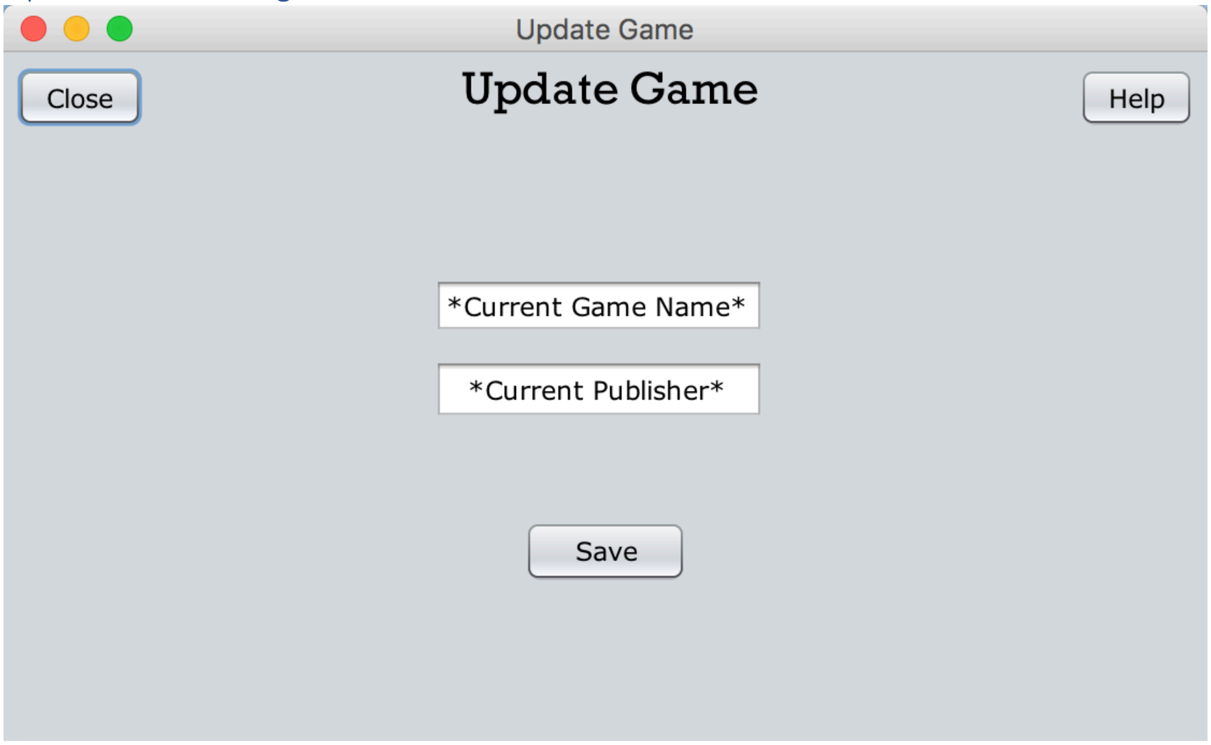
## Edit Games Window



| Game Name | Publisher |
|---|---|
| PAC-MAN Premium | Kertzmann Group |
| Shanghai Mahjong | Gerhold-Pollich |
| Ms. PAC-MAN | Hauck-Rice |
| Solitaire by MobilityWare | Nolan Inc |
| SCRABBLE Premium | Armstrong-Jones |
| FreeCell | Murazik LLC |
| Crash Bandicoot Nitro Kart 3D | Hauck Group |
| :) Sudoku + | Metz and Sons |
| Hangman. | Ryan, Schamberger and Ve... |
| Blackjack by MobilityWare | Hoppe Group |
| Fieldrunners | Macejkovic, Dach and Fritsch |
| PAC-MAN | Barton and Sons |
| First Words Animals | Mayer Group |
| Fish Tycoon | White Group |
| Touchgrind | Windler-Grimes |
| Phase 10 Pro - Play Your Friends! | Kessler-Koepp |

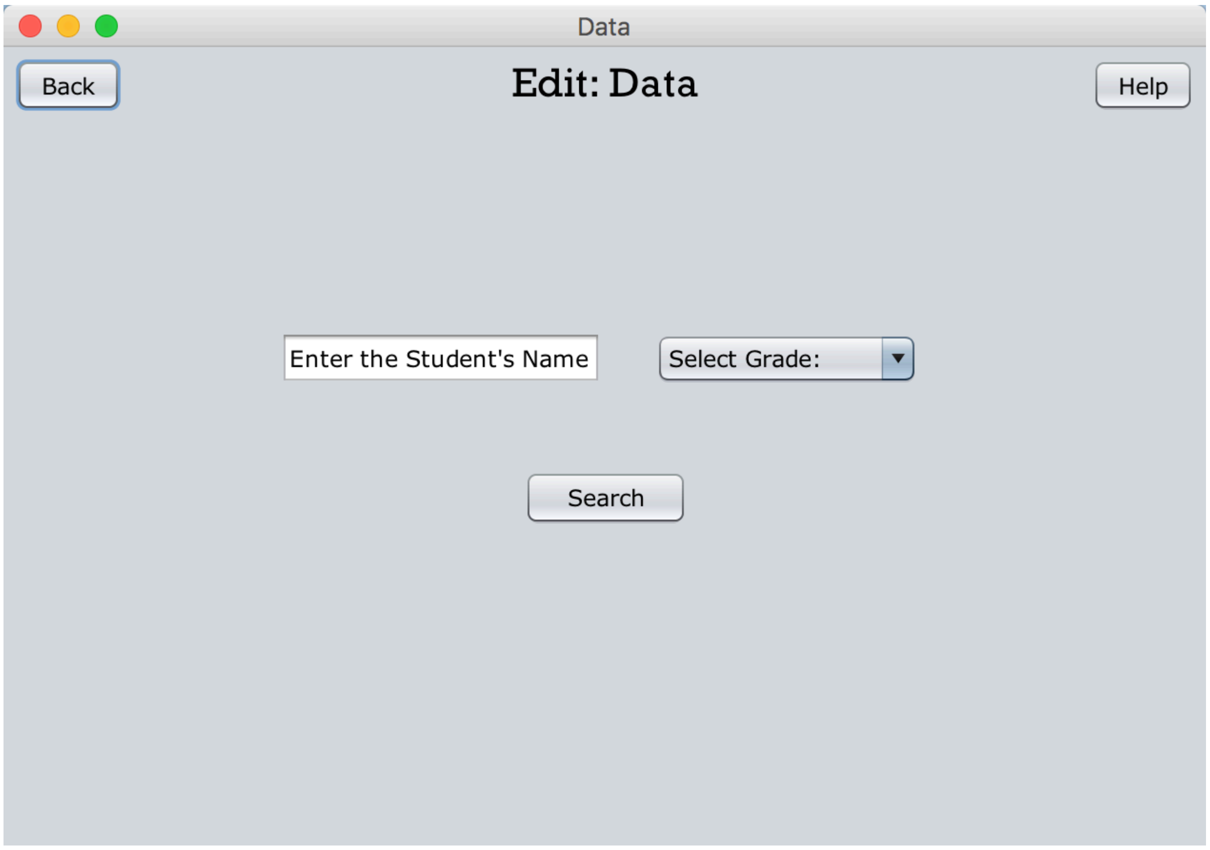
| GUI Component | Function |
|---|---|
| "Back" Button | Takes the user to the Admin Window. |
| "Help" Button | Opens the Help Dialog Box at the relevant place. |
| Data Table | Displays the game name and the game publisher. |
| "Add Game" Button | Takes the user to the New Game Dialog Box. |
| "Delete" Button | Appears when a Game is selected and allows the user to delete the selected Game. |
| "Update" Button | Appears when a Game is selected and takes the user to the Update Game Dialog Box. |
| "Error" Label | Appears when an error occurs. |

## Update Game Dialog Box



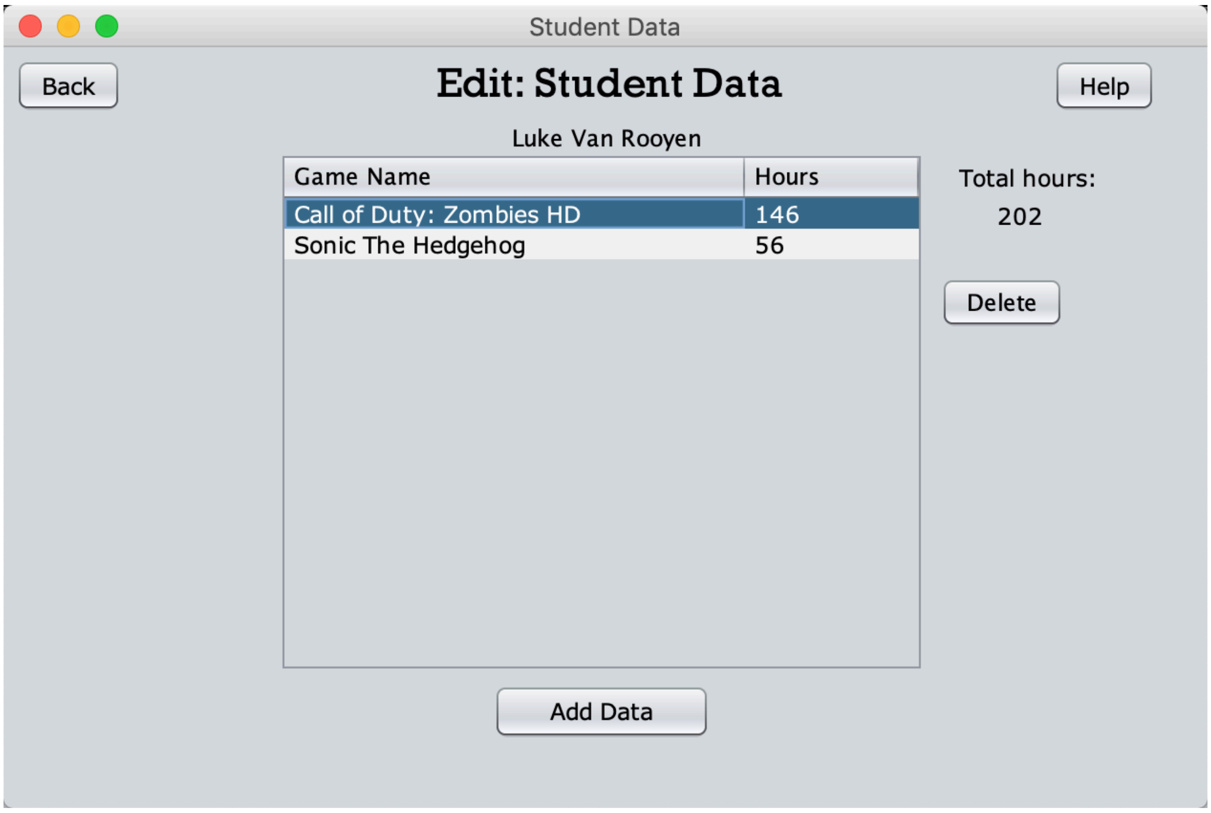
| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Name” Text Field | Allows the user to edit the current game name. |
| “Publisher” Text Field | Allows the user to edit the current publisher. |
| “Save” Button | Saves any changes to the game table. |
| “Error” Label | Appears when an error occurs. |

## Edit Data Window



| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Admin Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| “Student’s Name” Text Field | Allows the user to enter the desired student’s name. |
| “Select Grade:” Combo Box | Allows the user to select the desired student’s grade. |
| “Search” Button | Finds the student and takes the user to the Edit Student Data Window. |
| “Error” Label | Appears when an error occurs. |

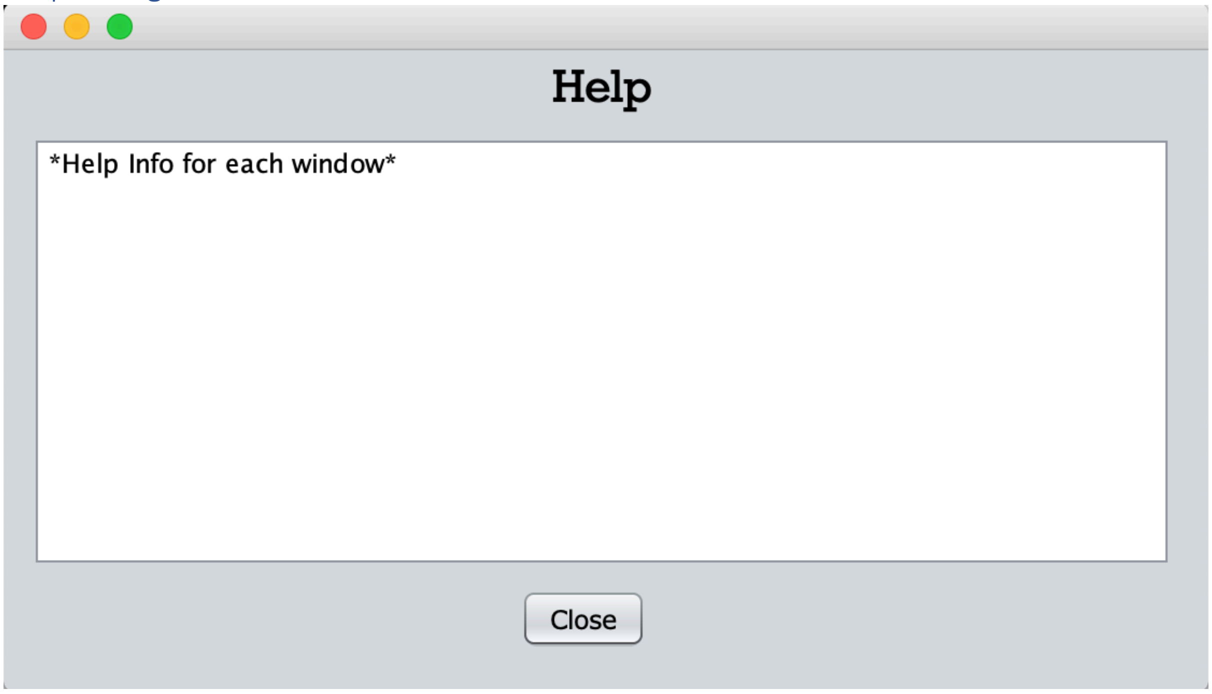
## Edit Student Data Window



| Game Name | Hours |
|---|---|
| Call of Duty: Zombies HD | 146 |
| Sonic The Hedgehog | 56 |

| GUI Component | Function |
|---|---|
| “Back” Button | Takes the user to the Edit Data Window. |
| “Help” Button | Opens the Help Dialog Box at the relevant place. |
| Data Table | Displays the game name and the number of hours. |
| “Add Data” Button | Takes the user to the Data Entering Window. |
| “Save” Button | Saves any changes to the data table. |
| “Delete” Button | Appears when a row is selected and allows the user to delete the selected row. |
| “Error” Label | Appears when an error occurs. |

## Help Dialog Box



*Help Info for each window*

| GUI Component | Function |
|---|---|
| “Close” Button | Closes the current dialog box. |

## Sequencing Interface

### Home Window

Home Window opens.

> "See Data" Button is pressed.
>> Data Window is opened.
>> Home Window is closed.

> "Help" Button is pressed.
>> Help Dialog Box is opened.

> "Admin Login" Button is pressed.
>> Admin Login Window is opened.
>> Home Window is closed.

> "Proceed" Button is pressed.
>> Data Entering Window is opened.
>> Home Window is closed.

> "Close Program" Button is pressed.
>> Home Window is closed.

### Data Entering Window

Data Entering Window opens.

> "Back" Button is pressed.
>> Home Window is opened.
>> Data Entering Window is closed.

> "Help" Button is pressed.
>> Help Dialog Box is opened.

> "See Data" Button is pressed.
>> Student's Data Dialog Box is opened.

> "Save" Button is pressed.
>> The name of the game and the number of hours from the spinner is saved to the database.
>>
>> Error: if the spinner contains a negative number an error message will be displayed.

“Update Student” Button is pressed.
Update Student Dialog Box is opened.

“New Game” Button is pressed.
New Game Dialog Box is opened.

## New Game Dialog Box

New Game Dialog Box opens.

“Close” Button is pressed.
New Game Dialog Box is closed.

“Help” Button is pressed.
Help Dialog Box is opened.

“Save” Button is pressed.
The new game name and publisher are saved to the database.

Error: if the text boxes are blank, contain an apostrophe or start with a space an error message will be displayed.

## Update Student Dialog Box

Update Student Dialog Box opens.

“Close” Button is pressed.
Update Game Dialog Box is closed.

“Help” Button is pressed.
Help Dialog Box is opened.

“Save” Button is pressed.
The name, grade and house for the relevant student are updated in the database.

Error: if the full name text box is blank, contains an apostrophe or starts with a space an error message will be displayed.

## Student’s Data Dialog Box

Student’s Data Dialog Box opens.

> Data Table is populated from the database.
>
> “Close” Button is pressed.
>> New Game Dialog Box is closed.
>
> “Help” Button is pressed.
>> Help Dialog Box is opened.

## Data Window

Data Window opens.

> “Back” Button is pressed.
>> Home Window is opened.
>> Data Window is closed.
>
> “Help” Button is pressed.
>> Help Dialog Box is opened.
>
> “All Data” Button is pressed.
>> All Data Window is opened.
>> Data Window is closed.
>
> “Data Summary” Button is pressed.
>> Data Summary Window is opened.
>> Data Window is closed.
>
> “Grouped Data” Button is pressed.
>> Grouped Data Window is opened.
>> Data Window is closed.
>
> “Search” Button is pressed.
>> Student’s Data Dialog Box is opened.
>>
>> Error: if the student name text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

## All Data Window

```
All Data Window opens.

    Data Table is populated from the database.

    “Back” Button is pressed.
        Data Window is opened.
        All Data Window is closed.

    “Help” Button is pressed.
        Help Dialog Box is opened.

    “Mode” Combo Box is used.

        If “Students” mode is selected
            Then the “Student Name” column is displayed in the table and each
            row relates to one student.

        If “Grades” mode is selected
            Then the “Grade” column is displayed in the table and each row
            relates to one grade.

        If “Houses” mode is selected
            Then the “House” column is displayed in the table and each row
            relates to one house.
```

## Data Summary Window

```
Data Summary Window opens.

    Data Table is populated from the database.

    “Back” Button is pressed.
        Data Window is opened.
        Data Summary Window is closed.

    “Help” Button
        Help Dialog Box is opened.
```

“Mode” Combo Box is used.

If “Grade” mode is selected
Then the “Highest House” column is displayed in the table and the “Select Grade:” combo box is shown.

If “House” mode is selected
Then the “Highest Grade” column is displayed in the table and the “Select House:” combo box is shown.

“Select House:” / “Select Grade:” Combo Box is used.
Data Table is refreshed to show new data.

## Grouped Data Window

Grouped Data Window opens.

Data table is populated from the database.

“Back” Button is pressed.
Data Window is opened.
Grouped Data Window is closed.

“Help” Button is pressed.
Help Dialog Box is opened.

“Mode” Combo Box is used.

If “By Grade” mode is selected
Then the “Select Desired Grade:” combo box is shown and the data table is filtered by the selected grade.

If “By House” mode is selected
Then the “Select Desired House:” combo box is shown and the data table is filtered by the selected house.

“Filter” Combo Box is used.
Data Table is refreshed using the selected house or grade as a filter.

## Admin Login Window

Admin Login Window opens.

“Back” Button is pressed.
Home Window is opened.
Admin Login Window is closed.

```
“Help” Button is pressed.
    Help Dialog Box is opened.

“Login” Button is pressed.
    The username and password will be validated.
    If they are correct, then
            Admin Window is opened.
            Admin Login Window is closed.

        Error: if the username and password are incorrect or blank, contain an
        apostrophe or start with a space an error message will be displayed.
```

### Admin Window

```
Admin Window opens.

    “Back” Button is pressed.
        Home Window is opened.
        Admin Window is closed.

    “Help” Button is pressed.
        Help Dialog Box is opened.

    “Students” Button is pressed.
        Edit Students Window is opened.
        Admin Window is closed.

    “Houses” Button is pressed.
        Edit Houses Window is opened.
        Admin Window is closed.

    “Grades” Button is pressed.
        Edit Grades Window is opened.
        Admin Window is closed.

    “Games” Button is pressed.
        Edit Games Window is opened.
        Admin Window is closed.

    “Data” Button is pressed.
        Edit Data Window is opened.
        Admin Window is closed.
```

## Edit Houses Window

Edit Houses Window opens.

Data table is populated from the database.

"Back" Button is pressed.
Admin Window is opened.
Edit Houses Window is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Update" Button is pressed.
Update House Dialog Box is opened.

Error: if no house is selected an error message will be displayed.

"Add House" Button is pressed.
New House Dialog Box is opened.

"Delete" Button is pressed.
The current row is deleted.

## Update House Dialog Box

Update House Dialog Box opens.

"Close" Button is pressed.
Update House Dialog Box is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Save" Button is pressed.
Any changes or additions are saved to the database.

Error: if the text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

## New House Dialog Box

New House Dialog Box opens.

"Close" Button is pressed.
New House Dialog Box is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Save" Button is pressed.
Any changes or additions are saved to the database.

Error: if the text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

### Edit Grades Window

Edit Grades Window opens.

Data table is populated from the database.

"Back" Button is pressed.
Admin Window is opened.
Edit Grades Window is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Update" Button is pressed.
Update Grade Dialog Box is opened.

Error: if no grade is selected an error message will be displayed.

"Add Grade" Button is pressed.
New Grade Dialog Box is opened.

"Delete" Button is pressed.
The current row is deleted.

### Update Grade Dialog Box

Update Grade Dialog Box opens.

"Close" Button is pressed.
Update Grade Dialog Box is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Save" Button is pressed.
Any changes or additions are saved to the database.

Error: if the text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

## New Grade Dialog Box

New Grade Dialog Box opens.

“Close” Button is pressed.
New Grade Dialog Box is closed.

“Help” Button is pressed.
Help Dialog Box is opened.

“Save” Button is pressed.
Any changes or additions are saved to the database.

Error: if the text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

## Edit Students Window

Edit Students Window opens.

Data table is populated from the database.

“Back” Button is pressed.
Admin Window is opened.
Edit Students Window is closed.

“Help” Button is pressed.
Help Dialog Box is opened.

“Update” Button is pressed.
Update Student Dialog Box is opened.

Error: if no student is selected an error message will be displayed.

“Add Student” Button is pressed.
New Student Dialog Box is opened.

“Delete” Button is pressed.
The current row is deleted.

## New Student Dialog Box

New Student Dialog Box opens.

"Close" Button is pressed.
New Student Dialog Box is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Save" Button is pressed.
Any changes or additions are saved to the database.

Error: if the text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

## Edit Games Window

Edit Games Window opens.

Data table is populated from the database.

"Back" Button is pressed.
Admin Window is opened.
Edit Games Window is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

"Update" Button is pressed.
Update Game Dialog Box is opened.

Error: if no game is selected an error message will be displayed.

"Add Game" Button is pressed.
New Game Dialog Box is opened.

"Delete" Button is pressed.
The current row is deleted.

## Update Game Dialog Box

Update Game Dialog Box opens.

"Close" Button is pressed.
Update Game Dialog Box is closed.

"Help" Button is pressed.
Help Dialog Box is opened.

“Save” Button is pressed.

Any changes or additions are saved to the database.

Error: if the text field is blank, contains an apostrophe or starts with a space an error message will be displayed.

### Edit Data Window

Edit Data Window opens.

“Back” Button is pressed.

Admin Window is opened.
Edit Data Window is closed.

“Help” Button is pressed.

Help Dialog Box is opened.

“Search” Button is pressed.

Edit Student Data Window is opened.
Edit Data Window is closed.

Error: if the text field is blank, contains an apostrophe or starts with a space or the student cannot be found an error message will be displayed.

### Edit Student Data Window

Edit Student Data Window opens.

Data table is populated from the database.

“Back” Button is pressed.

Edit Data Window is opened.
Edit Student Data Window is closed.

“Help” Button is pressed.

Help Dialog Box is opened.

“Add Data” Button is pressed.

Data Entering Window is opened.
Edit Student Data Window is closed.

“Delete” Button is pressed.

The current row is deleted.

Error: if no record is selected an error message will be displayed.

### Help Dialog Box

Help Dialog Box opens.

    “Close” Button is pressed.
        Help Dialog Box is closed.

## Class Design

| Admin Class | |
|---|---|
| **Fields** | |
| **- user_id: int** | Stores the user_id. |
| **- username: String** | Stores the username. |
| **- password: String** | Stores the password. |
| **Methods** | |
| **+ Admin(user_id: int, username: String, password: String): void** | Instantiates an object using the given user_id, username, and password values. |
| **+ getUser_id(): int** | Returns the user_id as an integer. |
| **+ setUser_id(user_id: int): void** | Sets the user_id to a certain value, given an integer. |
| **+ getUsername(): String** | Returns the username as an String. |
| **+ setUsername(username: String): void** | Sets the username to a certain value, given an String. |
| **+ getPassword(): String** | Returns the password as an String. |
| **+ setPassword(password: String): void** | Sets the password to a certain value, given an String. |
| **+ toString(): String** | Returns a summary of all the fields in the form of a string. |

| Game Class | |
|---|---|
| **Fields** | |
| **- game_id: int** | Stores the game_id. |
| **- game: String** | Stores the game name. |
| **- publisher: String** | Stores the game publisher. |
| **Methods** | |
| **+ Game(game_id: int, game: String, publisher: String): void** | Instantiates an object using the given game_id, game, and publisher values. |
| **+ getGame_id(): int** | Returns the game_id as an integer. |
| **+ setGame_id(game_id: int): void** | Sets the game_id to a certain value, given an integer. |
| **+ getGame(): String** | Returns the game as an String. |

| | |
|---|---|
| **+ setGame(game: String): void** | Sets the game to a certain value, given an String. |
| **+ getPublisher(): String** | Returns the publisher as an String. |
| **+ setPublisher(publisher: String): void** | Sets the publisher to a certain value, given an String. |

| Grade Class | |
|---|---|
| **Fields** | |
| **- grade_id: int** | Stores the grade_id. |
| **- grade: String** | Stores the grade name. |
| **Methods** | |
| **+ Grade(grade_id: int, grade: String): void** | Instantiates an object using the given grade_id and grade. |
| **+ getGrade_id(): int** | Returns the grade_id as an integer. |
| **+ setGrade_id(grade_id: int): void** | Sets the grade_id to a certain value, given an integer. |
| **+ getGrade(): String** | Returns the grade as an String. |
| **+ setGrade(grade: String): void** | Sets the grade to a certain value, given an String. |
| **+ toString(): String** | Returns the grade. |

| Grouped Data Class | |
|---|---|
| **Fields** | |
| **- full_name: String** | Stores the full_name. |
| **- total_hours: int** | Stores the total_hours. |
| **- game: String** | Stores the game. |
| **- hours: int** | Stores the hours. |
| **Methods** | |
| **+ GroupedData(full_name: String, total_hours: int, game: String, hours: int): void** | Instantiates an object using the given full_name, total_hours, game and hours. |
| **+ getFull_name(): String** | Returns the full_name as an String. |
| **+ setFull_name(full_name: String): void** | Sets the full_name to a certain value, given an String. |
| **+ getTotal_hours(): int** | Returns the total_hours as an integer. |
| **+ setTotal_hours(total_hours: int): void** | Sets the total_hours to a certain value, given an integer. |
| **+ getGame(): String** | Returns the game as an String. |
| **+ setGame(game: String): void** | Sets the game to a certain value, given an String. |
| **+ getHours(): int** | Returns the hours as an integer. |
| **+ setHours(hours: int): void** | Sets the hours to a certain value, given an integer. |

| | |
|---|---|
| **+ toString(): String** | Returns the full_name. |

| Hours Class | |
|---|---|
| **Fields** | |
| **- hours_id: int** | Stores the hours_id. |
| **- student_id: int** | Stores the student_id. |
| **- game_id: int** | Stores the game_id. |
| **- hours: int** | Stores the hours. |
| **Methods** | |
| **+ Hours(hours_id: int, student_id: int, game_id: int, hours: int): void** | Instantiates an object using the given hours_id, student_id, game_id and hours. |
| **+ getHours_id(): int** | Returns the hours_id as an integer. |
| **+ setHours_id(hours: int): void** | Sets the hours_id to a certain value, given an integer. |
| **+ getStudent_id(): int** | Returns the student_id as an integer. |
| **+ setStudent_id(hours: int): void** | Sets the student_id to a certain value, given an integer. |
| **+ getGame_id(): int** | Returns the game_id as an integer. |
| **+ setGame_id(hours: int): void** | Sets the game_id to a certain value, given an integer. |
| **+ getHours(): int** | Returns the hours as an integer. |
| **+ setHours(hours: int): void** | Sets the hours to a certain value, given an integer. |

| House Class | |
|---|---|
| **Fields** | |
| **- house_id: int** | Stores the house_id. |
| **- house: String** | Stores the house name. |
| **Methods** | |
| **+ House(house_id: int, house: String): void** | Instantiates an object using the given house_id and house. |
| **+ getHouse_id(): int** | Returns the house_id as an integer. |
| **+ setHouse _id(grade_id: int): void** | Sets the house_id to a certain value, given an integer. |
| **+ getHouse(): String** | Returns the house as an String. |
| **+ setHouse (grade: String): void** | Sets the house to a certain value, given an String. |

| Student Class | |
|---|---|
| **Fields** | |
| **- student_id: int** | Stores the student_id. |
| **- full_name: String** | Stores the full_name. |

| | |
|---|---|
| **- house_id: int** | Stores the house_id. |
| **- grade_id: int** | Stores the grade_id. |
| **Methods** | |
| **+ Student(student_id: int, full_name: String, house_id: int, grade_id: int): void** | Instantiates an object using the given student_id, full_name, house_id and grade_id. |
| **+ getStudent_id(): int** | Returns the student_id as an integer. |
| **+ setStudent_id(hours: int): void** | Sets the student_id to a certain value, given an integer. |
| **+ getFull_name(): String** | Returns the full_name as an String. |
| **+ setFull_name(full_name: String): void** | Sets the full_name to a certain value, given an String. |
| **+ getHouse_id(): int** | Returns the house_id as an integer. |
| **+ setHouse _id(grade_id: int): void** | Sets the house_id to a certain value, given an integer. |
| **+ getGrade_id(): int** | Returns the grade_id as an integer. |
| **+ setGrade_id(grade_id: int): void** | Sets the grade_id to a certain value, given an integer. |

| StudentData Class | |
|---|---|
| **Fields** | |
| **- game: String** | Stores the game. |
| **- hours: int** | Stores the hours. |
| **- hours_id: int** | Stores the hours_id. |
| **Methods** | |
| **+ StudentData(game: String, hours: int, hours_id: int): void** | Instantiates an object using the given game, hours and hours_id. |
| **+ getGame(): String** | Returns the game as an String. |
| **+ setGame(game: String): void** | Sets the game to a certain value, given an String. |
| **+ getHours(): int** | Returns the hours as an integer. |
| **+ setHours(hours: int): void** | Sets the hours to a certain value, given an integer. |
| **+ getHours_id(): int** | Returns the hours_id as an integer. |
| **+ setHours_id(hours_id: int): void** | Sets the hours_id to a certain value, given an integer. |

| SummaryData Class | |
|---|---|
| **Fields** | |
| **- totalhours: int** | Stores the totalhours. |
| **- highest: String** | Stores the highest house or grade. |
| **- popgame: String** | Stores the most popular game. |
| **- hourspopgame: int** | Stores the hours in the most popular game. |

| Methods | |
|---|---|
| **+ SummaryData(totalhours: int, highest: String, popgame: String, hourspopgame: int): void** | Instantiates an object using the given totalhours, highest, popgame and hourspopgame. |
| **+ getTotalhours(): int** | Returns the totalhours as an integer. |
| **+ setTotalhours (totalhours: int): void** | Sets the totalhours to a certain value, given an integer. |
| **+ getHighest(): String** | Returns the highest as an String. |
| **+ setHighest(highest: String): void** | Sets the highest to a certain value, given an String. |
| **+ getPopgame(): String** | Returns the popgame as an String. |
| **+ setPopgame(popgame: String): void** | Sets the popgame to a certain value, given an String. |
| **+ getHourspopgame(): int** | Returns the hourspopgame as an integer. |
| **+ setHourspopgame(hourspopgame: int): void** | Sets the hourspopgame to a certain value, given an integer. |

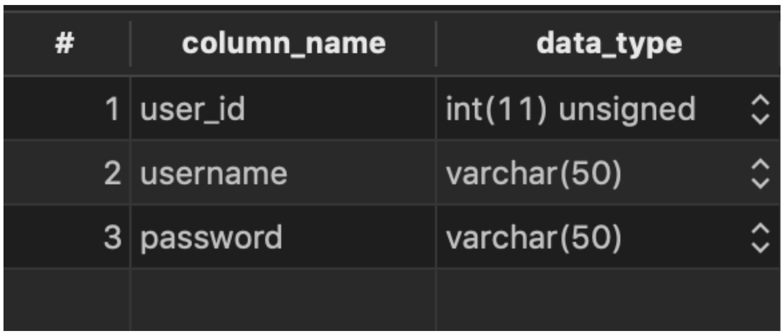
| DBManager Class | |
|---|---|
| **Fields** | |
| **- conn: Connection** | Stores the details for the connection between the database software and Java. |
| **Methods** | |
| **+ DBManager(): void** | Instantiates an object with the conn variable. |
| **+ addGame(game: String, publisher: String): Boolean** | Adds a game to the database, given the name and publisher, and returns a Boolean. |
| **+ addGrade(grade: String): Boolean** | Adds a grade to the database, given the name, and returns a Boolean. |
| **+ addHours(stud_id: int, game_id: int, hours: int): Boolean** | Adds an hour record to the database, given the student_id, game_id and number of hours, and returns a Boolean. |
| **+ addHouse(house: String): Boolean** | Adds a house to the database, given the name, and returns a Boolean. |
| **+ addStudent(full_name: String, house_id: int, grade_id: int): Boolean** | Adds a student to the database, given their full_name, house_id and grade_id, and returns a Boolean. |
| **+ delGameRow(selgame: Game): Boolean** | Deletes a game from the database, given the game object, and returns a Boolean. |
| **+ delGradeRow(selgrade: Grade): Boolean** | Deletes a grade from the database, given the grade object, and returns a Boolean. |
| **+ delHoursRow(studdata: StudentData): Boolean** | Deletes an hours record from the database, given the studentdata object, and returns a Boolean. |
| **+ delHouseRow(selhouse: House): Boolean** | Deletes a house from the database, given the house object, and returns a Boolean. |

| | |
|---|---|
| **+ delStudentRow(selstudent: Student): Boolean** | Deletes a student from the database, given the student object, and returns a Boolean. |
| **+ getAllAdmins(): ArrayList<Admin>** | Returns an ArrayList of all the admins from the database. |
| **+ getAllData(mode: int): ArrayList<GroupedData>** | Returns an ArrayList of groupeddata objects using an SQL query and given the mode as an integer. |
| **+ getAllGames(): ArrayList<Game>** | Returns an ArrayList of all the games from the database. |
| **+ getAllGrades(): ArrayList<Grade>** | Returns an ArrayList of all the grades from the database. |
| **+ getAllHours(): ArrayList<Hours>** | Returns an ArrayList of all the hour records from the database. |
| **+ getAllHouses(): ArrayList<Houses>** | Returns an ArrayList of all the houses from the database. |
| **+ getAllStudents(): ArrayList<Student>** | Returns an ArrayList of all the students from the database. |
| **+ getGradeStudents(grade_id: int): ArrayList<Student>** | Returns an ArrayList of all the students in a certain grade from the database, given the grade_id. |
| **+ getGroupedData(mode: int, filter: int): ArrayList<GroupedData>** | Returns an ArrayList of groupeddata objects using an SQL query, given the mode and filter. |
| **+ getStudentData(selstudent: Student): ArrayList<StudentData>** | Returns an ArrayList of studentdata objects using an SQL query and given the selected student. |
| **+ getStudentHours(selstudent: Student): int** | Returns an integer for the number of hours for a certain student from the database, using an SQL query and given the student. |
| **+ getStudent(full_name: String, grade_id: int): Student** | Returns a student object from the database, given their full_name and grade_id. |
| **+ getSummData(mode: int, filter: int): ArrayList<SummaryData>** | Returns an ArrayList of summarydata objects from the database, using an SQL query and given the mode and filter. |
| **+ testLogin(username: String, password: String): Boolean** | Requires the username and password, as strings, and returns a Boolean based on whether they match the database. |
| **+ updateGame(selgame: Game): Boolean** | Updates a game in the database and returns a Boolena. |
| **+ updateGrade(selgrade: Grade): Boolean** | Updates a grade in the database and returns a Boolena. |
| **+ updateHouse(selhouse: House): Boolean** | Updates a house in the database and returns a Boolena. |
| **+ updateStudent(selstudent: Student): Boolean** | Updates a student in the database and returns a Boolena. |

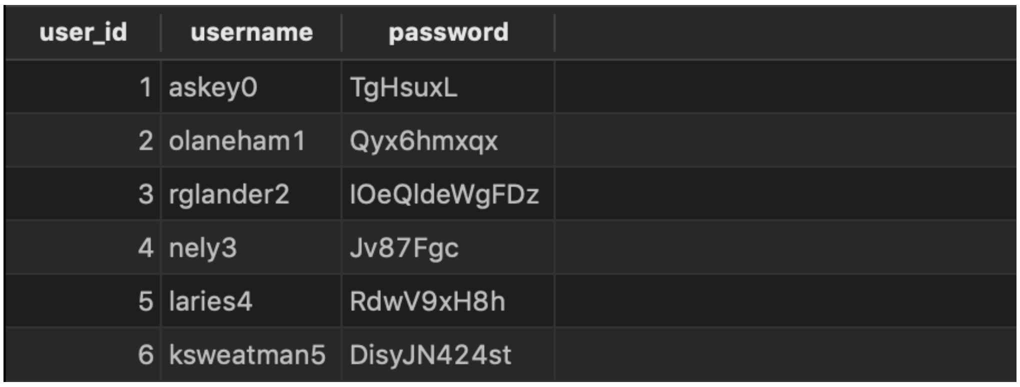
## Database Design

All the data for the program is stored in a single database consisting of 6 tables, as shown below. The database will be centralised, meaning the users will have to connect to the database using their devices in order to view or update their data.

### tblAdmins



| # | column_name | data_type |
|---|---|---|
| 1 | user_id | int(11) unsigned |
| 2 | username | varchar(50) |
| 3 | password | varchar(50) |
| | | |



| user_id | username | password |
|---|---|---|
| 1 | askey0 | TgHsuxL |
| 2 | olaneham1 | Qyx6hmxqx |
| 3 | rglander2 | IOeQIdeWgFDz |
| 4 | nely3 | Jv87Fgc |
| 5 | laries4 | RdwV9xH8h |
| 6 | ksweatman5 | DisyJN424st |

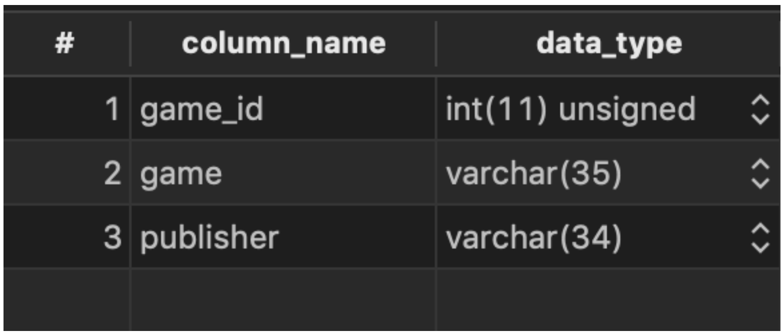
A list of strings for the admin usernames, passwords and a user_id for each will be saved in this table.

**Primary Key:** user_id.

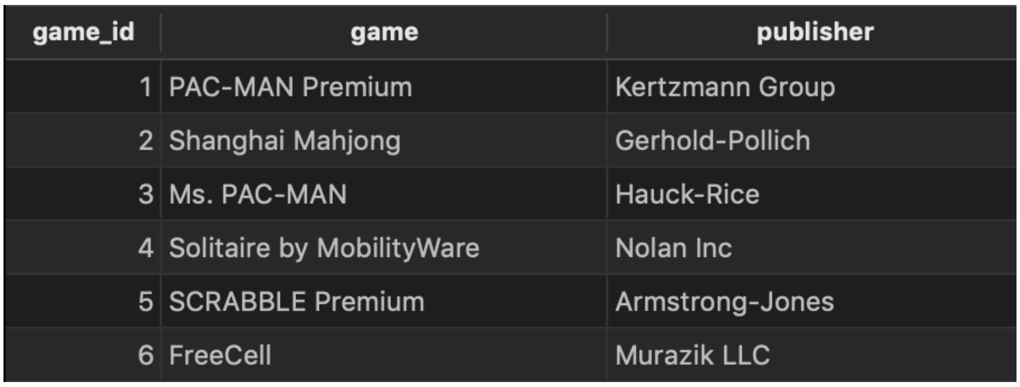
**Foreign Keys:** none.

**Other Fields:** username, password.

### tblGames



| # | column_name | data_type |
|---|---|---|
| 1 | game_id | int(11) unsigned |
| 2 | game | varchar(35) |
| 3 | publisher | varchar(34) |
| | | |



| game_id | game | publisher |
|---|---|---|
| 1 | PAC-MAN Premium | Kertzmann Group |
| 2 | Shanghai Mahjong | Gerhold-Pollich |
| 3 | Ms. PAC-MAN | Hauck-Rice |
| 4 | Solitaire by MobilityWare | Nolan Inc |
| 5 | SCRABBLE Premium | Armstrong-Jones |
| 6 | FreeCell | Murazik LLC |

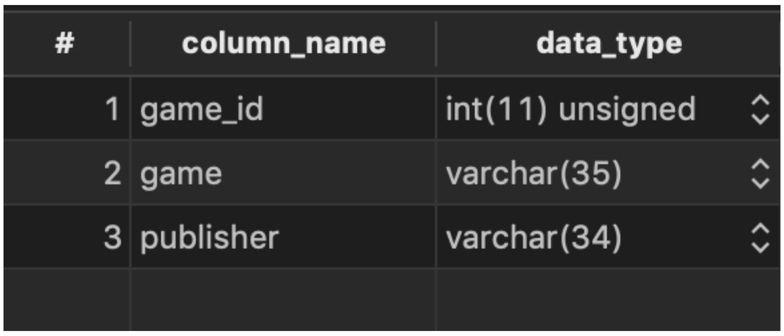
Games will be stored in this table with a string for their name and a string for their publisher as well as an integer for each game_id.

**Primary Key:** game_id.

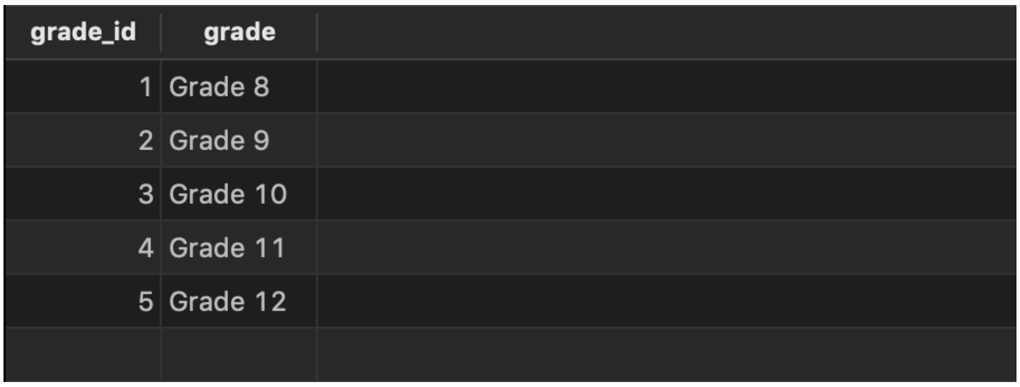
**Foreign Keys:** none.

**Other Fields:** game, publisher.

### tblGrades



| # | column_name | data_type |
|---|---|---|
| 1 | game_id | int(11) unsigned |
| 2 | game | varchar(35) |
| 3 | publisher | varchar(34) |



| grade_id | grade |
|---|---|
| 1 | Grade 8 |
| 2 | Grade 9 |
| 3 | Grade 10 |
| 4 | Grade 11 |
| 5 | Grade 12 |

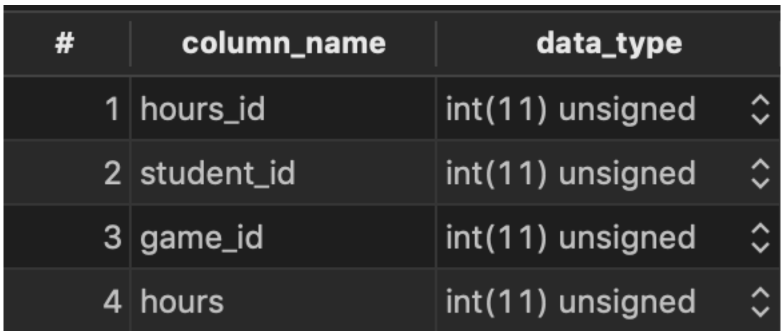
The grades will be stored in this table with an string for each grade and an integer for each grade_id.

**Primary Key:** grade_id.

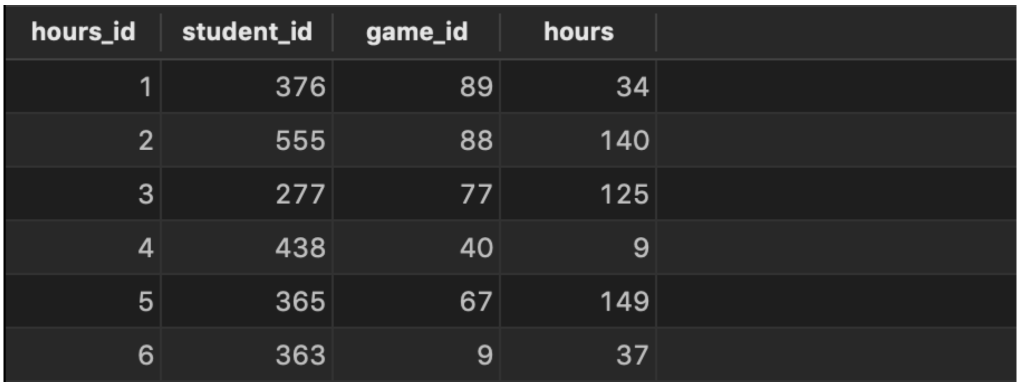
**Foreign Keys:** none.

**Other Fields:** grade.

### tblHours



| # | column_name | data_type |
|---|---|---|
| 1 | hours_id | int(11) unsigned |
| 2 | student_id | int(11) unsigned |
| 3 | game_id | int(11) unsigned |
| 4 | hours | int(11) unsigned |



| hours_id | student_id | game_id | hours |
|---|---|---|---|
| 1 | 376 | 89 | 34 |
| 2 | 555 | 88 | 140 |
| 3 | 277 | 77 | 125 |
| 4 | 438 | 40 | 9 |
| 5 | 365 | 67 | 149 |
| 6 | 363 | 9 | 37 |

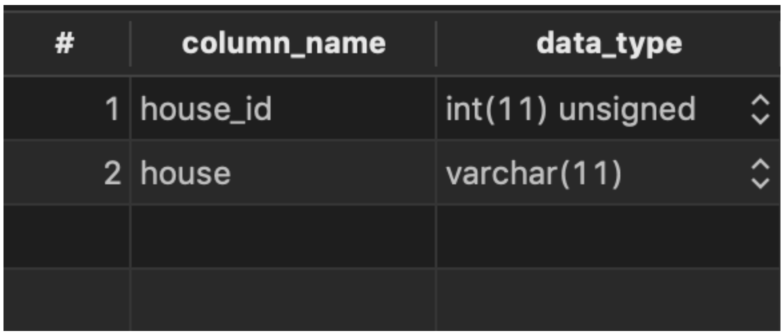
This table will save the number of hours in each game for each student. It will have an integer, hours_id, as well as the student_id, game_id and hours for each student.

**Primary Key:** hours_id.

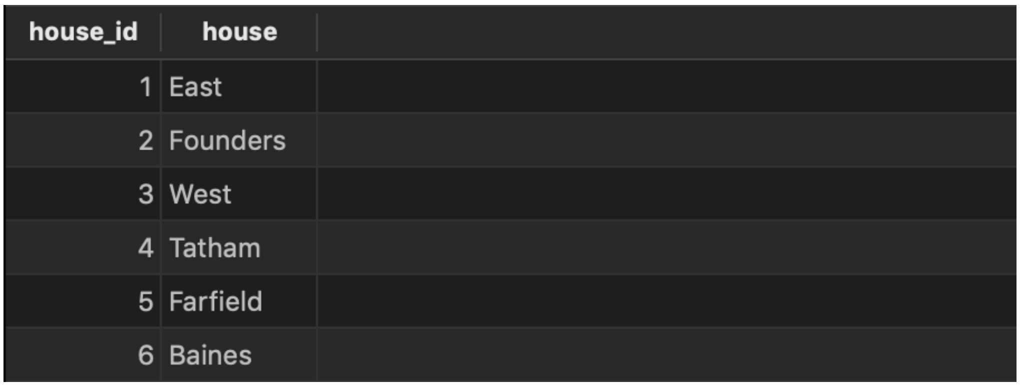
**Foreign Keys:** student_id, game_id.

**Other Fields:** hours.

### tblHouses



| # | column_name | data_type |
|---|---|---|
| 1 | house_id | int(11) unsigned |
| 2 | house | varchar(11) |
| | | |
| | | |



| house_id | house |
|---|---|
| 1 | East |
| 2 | Founders |
| 3 | West |
| 4 | Tatham |
| 5 | Farfield |
| 6 | Baines |

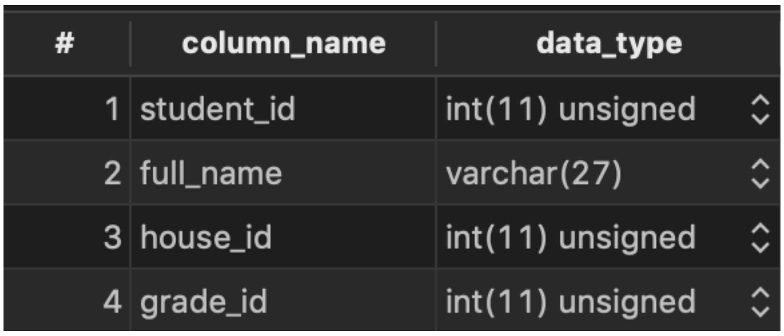
Houses will be stored in a table with a string for each house name and a house_id, an integer.

**Primary Key:** house_id.

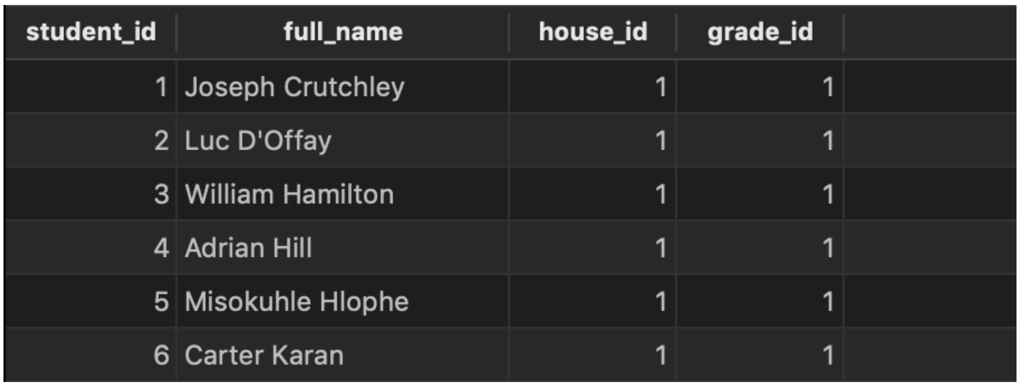
**Foreign Keys:** none.

**Other Fields:** house.

### tblStudents



| # | column_name | data_type |
|---|---|---|
| 1 | student_id | int(11) unsigned |
| 2 | full_name | varchar(27) |
| 3 | house_id | int(11) unsigned |
| 4 | grade_id | int(11) unsigned |



| student_id | full_name | house_id | grade_id |
|---|---|---|---|
| 1 | Joseph Crutchley | 1 | 1 |
| 2 | Luc D'Offay | 1 | 1 |
| 3 | William Hamilton | 1 | 1 |
| 4 | Adrian Hill | 1 | 1 |
| 5 | Misokuhle Hlophe | 1 | 1 |
| 6 | Carter Karan | 1 | 1 |

The students will be stored in this table with a string for their name, an integer for their grade_id and an integer for their house_id as well as another integer, student_id.

**Primary Key:** student_id.

**Foreign Keys:** house_id, grade_id.

**Other Fields:** full_name.

## Explanation for Database Design

I've chosen to store all the related data for my program in a database, rather than text files or spreadsheets. The main reason for this is that data will need to be continuously updated, deleted and removed, thus a database in conjunction with SQL makes the most sense, as it will allow me to manipulate my data easily. This could be achieved using text files or spreadsheets but implementing it would be considerably more complicated and time-consuming. A database also allows me to create relations between fields and tables as well as reduce redundancy and the likelihood of errors or anomalies. The final reason is that there is a very large amount of data that needs to be stored and database management programs, such as Sequel Pro or TablePlus, allow the user to easily import records and tables from CSV or SQL files.

From all of these reasons it is clear that a database is the obvious choice for storing this data.

The database will consist of many tables that will be edited by my program using SQL queries and the JDBC driver. I have chosen to normalize my database in the way described above because it allows me to add houses, grades, games, students, hours and admins all without interfering or interacting with the other tables, thus preventing insertion anomalies. It also allows me to delete individual houses, grades, games, students, hours and admins without causing any unnecessary or unforeseen data loss, thus preventing deletion anomalies. Finally, normalizing my database as described allows me to easily edit the details of any houses, grades, games, students, hours and admins

### Tables:

- tblAdmins
  - This table will store the user_id, username, and password for each admin user. The user_id will be stored as an integer, username as a string, and password also as a string.
  - Storing the data in this way will allow the program to easily re-access, edit, and use the data every time.
  - This data is stored permanently as it is a crucial part of the Admin Login window, and thus the editing windows.
  - I chose user_id as the primary key as it uniquely identifies each record in the table.
- tblGames
  - This table will store the game_id, game, and publisher for each record. The game_id will be stored as an integer, game as a string, and publisher also as a string.
  - Storing the data in this way will allow the program to easily re-access, edit, and use the data every time.
  - This data is stored permanently as it is a crucial part of all the data and game windows and needs to be reused whenever the program runs.

    - I chose game_id as the primary key as it uniquely identifies each record in the table.
- tblGrades
    - This table will store the grade_id and grade for each record. The grade_id will be stored as an integer and grade as a string.
    - Storing the data in this way will allow the program to easily re-access, edit, and use the data every time.
    - This data is stored permanently as it is a crucial part of the all the grade and data windows, and thus shall be needed every time the program runs.
    - I chose grade_id as the primary key as it uniquely identifies each record in the table.
- tblHours
    - This table will store the hours_id, student_id, game_id, and hours for each record. The hours_id will be stored as an integer as well as the student_id, game_id and hours.
    - Storing the data in this way will allow the program to easily re-access and use the data every time.
    - This data is stored permanently as it is a crucial part of the various data windows, as well as the editing windows.
    - I chose hours_id as the primary key as it uniquely identifies each record in the table.
- tblHouses
    - This table will store the house_id and house for each record. The house_id will be stored as an integer and house as a string.
    - Storing the data in this way will allow the program to easily re-access and use the data every time.
    - This data is stored permanently as it is a crucial part of all the house windows and the data windows.
    - I chose house_id as the primary key as it uniquely identifies each record in the table.
- tblStudents
    - This table will store the student_id, full_name, house_id and grade_id for each record. The student_id, house_id and grade_id will be stored as integers and full_name as a string.
    - Storing the data in this way will allow the program to easily re-access and use the data every time.
    - This data is stored permanently as it is a crucial part of the student windows, and as well as the data windows.
    - I chose student_id as the primary key as it uniquely identifies each record in the table.