IT PAT

Phase 2: Design Documentation

Name: Milaan Kassie

# Table of Contents

[Table of Contents 1](#_Toc209028804)

[2.1 User Interface Design 3](#_Toc209028805)

[Splash Screen 3](#_Toc209028806)

[Splash screen components 3](#_Toc209028807)

[Login Frame 4](#_Toc209028808)

[Login Frame components 4](#_Toc209028809)

[Welcome Frame 6](#_Toc209028810)

[Welcome Frame Components 6](#_Toc209028811)

[Main Menu 7](#_Toc209028812)

[Main Menu Components 7](#_Toc209028813)

[Patrons Table 8](#_Toc209028814)

[Patrons Table Components 9](#_Toc209028815)

[Events Table 12](#_Toc209028816)

[Events Table Components 13](#_Toc209028817)

[Visits Table 16](#_Toc209028818)

[Visits Table Components 17](#_Toc209028819)

[Reports Frame 20](#_Toc209028820)

[Reports Frame Components 21](#_Toc209028821)

[Help Frame 23](#_Toc209028822)

[Help Frame Components 23](#_Toc209028823)

[Exit Frame 24](#_Toc209028824)

[Exit Frame Components 24](#_Toc209028825)

[2.2 Program Flow Diagram 25](#_Toc209028826)

[2.3 Class design AND OOP Principles(Class Diagrams) 26](#_Toc209028827)

[Data classes 26](#_Toc209028828)

[Object classes 30](#_Toc209028829)

[Validation Class 36](#_Toc209028830)

[2.4 Secondary Storage Design 37](#_Toc209028831)

[TblUsers 37](#_Toc209028832)

[Design View 37](#_Toc209028833)

[Datasheet view 37](#_Toc209028834)

[Description 37](#_Toc209028835)

[TblVisits 38](#_Toc209028836)

[Design View 38](#_Toc209028837)

[Datasheet View 38](#_Toc209028838)

[Description 38](#_Toc209028839)

[TblEvents 39](#_Toc209028840)

[Design View 39](#_Toc209028841)

[Datasheet View 39](#_Toc209028842)

[Description 39](#_Toc209028843)

[TblPatrons 40](#_Toc209028844)

[Design View 40](#_Toc209028845)

[Datasheet view 40](#_Toc209028846)

[Description 40](#_Toc209028847)

[Relationships Diagram 41](#_Toc209028848)

[Help Textfiles 42](#_Toc209028849)

[2.5 Explanation of Secondary Storage 43](#_Toc209028850)

[Why Choose a Database Over JSON or Text Files 43](#_Toc209028851)

[Implications of Using a Database 44](#_Toc209028852)

[2.6 Explanation of how Primary Data Structures relate to Secondary Storage 45](#_Toc209028853)

# 2.1 User Interface Design

## A screen shot of a computer Description automatically generatedSplash Screen

### Splash screen components

* Layered Pane – Background
* Label - Loading GIF
* Label - Percent loaded
* Label - Company name
* Progress bar - Loading progress

| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| --- | --- | --- | --- |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  | Plays the loading GIF |
| 3 | Progress bar |  | Loads until completion |
| 4 | Label |  | Increases percentage from 0 to 100 |
| 5 | Label |  |  |

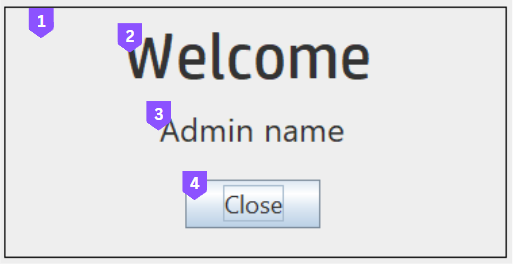
## Login Frame

### Login Frame components

* **Level 1: Layered Pane – Background**
  + Label - Title of screen (Admin Login)
  + Button - Help button in top right
  + Button - Exit button for closing application (leads to LogoutFrame)
  + Button - Login button (leads WelcomeFrame and then to MainMenuFrame)
    - **Level 2: Layered Pane - pane for the username and password** 
      * Label - Username label
      * Text Field - Entry for username
      * Label - Password label
      * Text Field - Entry for password
      * Checkbox - show password checkbox

| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| --- | --- | --- | --- |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Label |  |  |
| 5 | Layered pane |  |  |
| 6 | Label |  |  |
| 7 | Label |  |  |
| 8 | Text field | Keyboard |  |
| 9 | Text field | Keyboard |  |
| 10 | Check box | Mouse | Shows/Hides password |
| 11 | Button | Mouse | Leads up the Exit Frame to confirm closing the application |
| 12 | Button | Mouse | Logs user in if credentials are valid - Leads to Welcome Frame |

## Welcome Frame



### Welcome Frame Components

* Layered Pane – Background
* Label - Screen heading (welcome message)
* Label - Username (changes depending on user)
* Button - Close screen that closes this frame and leads to the MainMenuFrame

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Label |  |  |
| 4 | Button | Mouse | Closes the Welcome Frame and leads to the Main Menu |

## A menu screen with text and purple labels Description automatically generated with medium confidenceMain Menu

### Main Menu Components

* Layered Pane – Background
  + Label - Screen heading/title (Main Menu Frame)
  + Button - Help button on top right
  + Button - VisitsTable
  + Button - EventsTable
  + Button - PatronsTable
  + Button - ReportsFrame
  + Button - Logout button

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Button | Mouse | Leads to the Visits Table Frame |
| 5 | Button | Mouse | Leads to the Events Table Frame |
| 6 | Button | Mouse | Leads to the Patrons Table Frame |
| 7 | Button | Mouse | Leads to the Reports Frame |
| 8 | Button | Mouse | Leads to the Admin Login Frame |

## Patrons Table

A screenshot of a computer

Description automatically generated

### Patrons Table Components

* **Level 1: Layered Pane – Background**
  + Label - Heading/Title label (Visits Table)
  + Button - Help button in top right
  + Button - Back button (leads back to LoginFrame)
* **Level 2: Layered Pane - Navigation (contains JTable)**
  + JTable - PatronsTable
  + Button - First record
  + Button - Previous record
  + Button - Next record
  + Button - Last record
* **Level 2: Layered Pane - Details**
  + Label - PatronID
  + Text Field - PatronID
  + Label - PatronName
  + Text Field - PatronName
  + Label - Surname
  + Text Field - Surname
  + Label - Gender
  + Combo Box - Gender
  + Label - DateOfBirth
  + JDateChooser- DateOfBirth
  + Label - HomeAddress
  + Text Field - HomeAdress
  + Label - Status
  + Combo Box - Status
  + Label - Registration Deadline
  + DateChooser - Registration Deadline
* **Level 2: Layered Pane – Data Handling**
  + **Level 3: Layered Pane - Options**
    - Button - Add (allows data to be entered into the details pane for the add function)
    - Button - Edit (allows data to be edited in the details pane for the edit function)
    - Button - Delete (deletes the currently selected record)
  + **Level 3: Layered Pane - Save Options**
    - Button - Save new (saves data entered in the details pane by calling the SQL method)
    - Button - Save edit (saves the data entered in the details pane by calling the SQL method)
    - Button - Cancel (makes the details pane inaccessible and removes all data entered/edited)
* **Level 2: Layered Pane - Search**
  + Radio Button - All (shows all records)
  + Radio Button - PatronID (deactivates all other radio buttons and enables the text field for entry of a PatronID)
  + Radio Button - FirstNam (deactivates all other radio buttons and enables the text field for entry of a FirstName)
  + Radio Button - CardLevel (deactivates all other radio buttons and enables the text field for entry of a CardLevel)
  + Radio Button – Gender (deactivates all other radio buttons and enables the text field for entry of a CardLevel)

| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| --- | --- | --- | --- |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Label |  |  |
| 5 | Layered pane |  |  |
| 6 | Table |  |  |
| 7 | Button | Mouse | Navigates to and displays the first record in the table |
| 8 | Button | Mouse | Navigates to and displays the next record in the table |
| 9 | Button | Mouse | Navigates to and displays the previous record in the table |
| 10 | Button | Mouse | Navigates to and displays the last record in the table |
| 11 | Label |  |  |
| 12 | Layered pane |  |  |
| 13 | Label |  |  |
| 14 | Label |  |  |
| 15 | Label |  |  |
| 16 | Label |  |  |
| 17 | Label |  |  |
| 18 | Text field | Keyboard |  |
| 19 | Check box | Mouse | Enables / Disables primary key field for manual editing |
| 20 | Text field | Keyboard |  |
| 21 | Text field | Keyboard |  |
| 22 | Combo box | Mouse |  |
| 23 | Date chooser | Mouse / Keyboard |  |
| 24 | Label |  |  |
| 25 | Label |  |  |
| 26 | Label |  |  |
| 27 | Label |  |  |
| 28 | Text field | Keyboard |  |
| 29 | Text field | Keyboard |  |
| 30 | Combo box | Mouse |  |
| 31 | Date chooser | Mouse / Keyboard |  |
| 32 | Label |  |  |
| 33 | Layered pane |  |  |
| 34 | Radio button | Mouse | Disables search text field and button & displays all records in table |
| 35 | Radio button | Mouse | Enables search text field and button |
| 36 | Radio button | Mouse | Enables search text field and button |
| 37 | Radio button | Mouse | Enables search text field and button |
| 38 | Radio button | Mouse | Enables search text field and button |
| 39 | Text field | Keyboard |  |
| 40 | Button | Mouse | Searches/sorts the table records according to the radio button selected and data entered |
| 41 | Label |  |  |
| 42 | Layered pane |  |  |
| 43 | Label |  |  |
| 44 | Label |  |  |
| 45 | Layered pane |  |  |
| 46 | Layered pane |  |  |
| 47 | Button | Mouse | Activates and clears relevant data entry fields |
| 48 | Button | Mouse | Activates and inputs current data into relevant data entry fields |
| 49 | Button | Mouse | Clears data and deactivates data entry fields - displays current data |
| 50 | Button | Mouse | Executes SQL to add a new record to the table using entered data |
| 51 | Button | Mouse | Executes SQL to edit a record in the table using the edited data |
| 52 | Button | Mouse | Executes SQL to delete a record in the table |
| 53 | Button | Mouse | Leads to the Main Menu Frame |

## Events Table

### Events Table Components

* **Level 1: Layered Pane - Cover whole screen to provide a border**
  + Label - Heading/Title label (Visits Table)
  + Button - Help button in top right
  + Button - Back button (leads back to LoginFrame)
* **Level 2: Layered Pane - Navigation (contains JTable)**
  + JTable - EventsTable
  + Button - First record
  + Button - Previous record
  + Button - Next record
  + Button - Last record
* **Level 2: Layered Pane - Details**
  + Label - EventID
  + Text Field - EventID
  + Label - EventName
  + Text Field - EventName
  + Label - StartDate
  + Date Chooser - StartDate
  + Label - EndDate
  + Date Chooser - EndDate
  + Label - Capacity
  + Text Field - Capacity
  + Label - Location
  + Combo Box - Location
  + Label - Status
  + Combo Box - Status
  + Label - Registration Deadline
  + DateChooser - Registration Deadline
* **Level 2: Layered Pane – Data Handling**
  + **Level 3: Layered Pane - Options**
    - Button - Add (allows data to be entered into the details pane for the add function)
    - Button - Edit (allows data to be edited in the details pane for the edit function)
    - Button - Delete (deletes the currently selected record)
  + **Level 3: Layered Pane - Save Options**
    - Button - Save new (saves data entered in the details pane by calling the SQL method)
    - Button - Save edit (saves the data entered in the details pane by calling the SQL method)
    - Button - Cancel (makes the details pane inaccessible and removes all data entered/edited)
* **Level 2: Layered Pane - Search**
  + Radio Button - All (shows all records)
  + Radio Button - EventID (deactivates all other radio buttons and enables the text field for entry of a EventNo)
  + Radio Button - EventName (deactivates all other radio buttons and enables the text field for entry of a EventName)
  + Radio Button - Location (deactivates all other radio buttons and enables the text field for entry of a Location)
  + Radio Button – Status (deactivates all other radio buttons and enables the text field for entry of a Status)

| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| --- | --- | --- | --- |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Label |  |  |
| 5 | Layered pane |  |  |
| 6 | Table |  |  |
| 7 | Button | Mouse | Navigates to and displays the first record in the table |
| 8 | Button | Mouse | Navigates to and displays the next record in the table |
| 9 | Button | Mouse | Navigates to and displays the previous record in the table |
| 10 | Button | Mouse | Navigates to and displays the last record in the table |
| 11 | Label |  |  |
| 12 | Layered pane |  |  |
| 13 | Label |  |  |
| 14 | Label |  |  |
| 15 | Label |  |  |
| 16 | Text field | Keyboard |  |
| 17 | Check box | Mouse | Enables / Disables primary key field for manual editing |
| 18 | Text field | Keyboard |  |
| 19 | Date chooser | Mouse / Keyboard |  |
| 20 | Date chooser | Mouse / Keyboard |  |
| 21 | Label |  |  |
| 22 | Label |  |  |
| 23 | Label |  |  |
| 24 | Label |  |  |
| 25 | Text field | Keyboard |  |
| 26 | Combo box | Mouse |  |
| 27 | Combo box | Mouse |  |
| 28 | Date chooser | Mouse / Keyboard |  |
| 29 | Label |  |  |
| 30 | Layered pane |  |  |
| 31 | Radio button | Mouse | Disables search text field and button & displays all records in table |
| 32 | Radio button | Mouse | Enables search text field and button |
| 33 | Radio button | Mouse | Enables search text field and button |
| 34 | Radio button | Mouse | Enables search text field and button |
| 35 | Radio button | Mouse | Enables search text field and button |
| 36 | Text field | Keyboard |  |
| 37 | Button | Mouse | Searches/sorts the table records according to the radio button selected and data entered |
| 38 | Label |  |  |
| 39 | Layered pane |  |  |
| 40 | Label |  |  |
| 41 | Label |  |  |
| 42 | Layered pane |  |  |
| 43 | Layered pane |  |  |
| 44 | Button | Mouse | Activates and clears relevant data entry fields |
| 45 | Button | Mouse | Activates and inputs current data into relevant data entry fields |
| 46 | Button | Mouse | Clears data and deactivates data entry fields - displays current data |
| 47 | Button | Mouse | Executes SQL to add a new record to the table using entered data |
| 48 | Button | Mouse | Executes SQL to edit a record in the table using the edited data |
| 49 | Button | Mouse | Executes SQL to delete a record in the table |
| 50 | Button | Mouse | Leads to the Main Menu Frame |

## Visits Table

### Visits Table Components

* **Level 1: Layered Pane - Cover whole screen to provide a border**
  + Label - Heading/Title label (Visits Table)
  + Button - Help button in top right
  + Button - Back button (leads back to LoginFrame)
* **Level 2: Layered Pane - Navigation (contains JTable)**
  + JTable - VisitsTable
  + Button - First record
  + Button - Previous record
  + Button - Next record
  + Button - Last record
* **Level 2: Layered Pane - Details**
  + Label - VisitNo
  + Text Field - VisitNo
  + Label - EventID
  + Text Field - EventID
  + Label - PatronID
  + Text Field - PatronID
  + Label - AmountSpent
  + Text Field - AmountSpent
  + Label - Username
  + Text Field - Username
  + Label - DateOfVisit
  + Date Chooser - DateOfVisit
* **Level 2: Layered Pane – Data Handling**
  + **Level 3: Layered Pane - Options**
    - Button - Add (allows data to be entered into the details pane for the add function)
    - Button - Edit (allows data to be edited in the details pane for the edit function)
    - Button - Delete (deletes the currently selected record)
  + **Level 3: Layered Pane - Save Options**
    - Button - Save new (saves data entered in the details pane by calling the SQL method)
    - Button - Save edit (saves the data entered in the details pane by calling the SQL method)
    - Button - Cancel (makes the details pane inaccessible and removes all data entered/edited)
* **Level 2: Layered Pane - Search**
  + Radio Button - All (shows all records)
  + Radio Button - VisitNo (deactivates all other radio buttons and enables the text field for entry of a VisitNo)
  + Radio Button - PatronID (deactivates all other radio buttons and enables the text field for entry of a PatronID)
  + Radio Button - EventID (deactivates all other radio buttons and enables the text field for entry of a EventID)

| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| --- | --- | --- | --- |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Label |  |  |
| 5 | Layered pane |  |  |
| 6 | Table |  |  |
| 7 | Button | Mouse | Navigates to and displays the first record in the table |
| 8 | Button | Mouse | Navigates to and displays the next record in the table |
| 9 | Button | Mouse | Navigates to and displays the previous record in the table |
| 10 | Button | Mouse | Navigates to and displays the last record in the table |
| 11 | Label |  |  |
| 12 | Layered pane |  |  |
| 13 | Label |  |  |
| 14 | Label |  |  |
| 15 | Label |  |  |
| 16 | Label |  |  |
| 17 | Label |  |  |
| 18 | Label |  |  |
| 19 | Text field | Keyboard |  |
| 20 | Check box | Mouse | Enables / Disables primary key field for manual editing |
| 21 | Combo box | Mouse |  |
| 22 | Combo box | Mouse |  |
| 23 | Text field | Keyboard |  |
| 24 | Combo box | Mouse |  |
| 25 | Date chooser | Mouse / Keyboard |  |
| 26 | Label |  |  |
| 27 | Layered pane |  |  |
| 28 | Radio button | Mouse | Disables search text field and button & displays all records in table |
| 29 | Radio button | Mouse | Enables search text field and button |
| 30 | Radio button | Mouse | Enables search text field and button |
| 31 | Radio button | Mouse | Enables search text field and button |
| 32 | Radio button | Mouse | Enables search text field and button |
| 33 | Text field | Keyboard |  |
| 34 | Button | Mouse | Searches/sorts the table records according to the radio button selected and data entered |
| 35 | Label |  |  |
| 36 | Layered pane |  |  |
| 37 | Label |  |  |
| 38 | Label |  |  |
| 39 | Layered pane |  |  |
| 40 | Layered pane |  |  |
| 41 | Button | Mouse | Activates and clears relevant data entry fields |
| 42 | Button | Mouse | Activates and inputs current data into relevant data entry fields |
| 43 | Button | Mouse | Clears data and deactivates data entry fields - displays current data |
| 44 | Button | Mouse | Executes SQL to add a new record to the table using entered data |
| 45 | Button | Mouse | Executes SQL to edit a record in the table using the edited data |
| 46 | Button | Mouse | Executes SQL to delete a record in the table |
| 47 | Button | Mouse | Leads to the Main Menu Frame |

## Reports Frame

### Reports Frame Components

* **Level 1: Layered Pane – Background**
  + Label - Heading/Title label (Visits Table)
  + Button - Help button in top right
  + Button - Back button (leads back to LoginFrame)
  + Button - Clear Text Area
* **Level 2: Layered Pane - Output Pane (contains Text Area)**
  + Text Area - Display all queries when called
* **Level 2: Layered Pane - Queries (Contains buttons for each query)**
  + Button - Highest Spender
  + Button - Most Visits
  + Button - Patron Card Types
  + Button - Gender
  + Button - Event Location
  + Button - High-Capacity Events

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Label |  |  |
| 5 | Layered pane |  |  |
| 6 | Text area |  |  |
| 7 | Label |  |  |
| 8 | Layered pane |  |  |
| 9 | Button | Mouse | Displays the result set for the Highest Spender query - The patron that spent the most |
| 10 | Button | Mouse | Displays the result set for the Most Visits query - The patrons that made the most visits |
| 11 | Button | Mouse | Displays the result set for the Patron Card Type query - The number of each card type |
| 12 | Button | Mouse | Displays the result set for the Gender query - The number of male and female patrons |
| 13 | Button | Mouse | Displays the result set for the Event Locations query - Number of events in each location |
| 14 | Button | Mouse | Displays the result set for the High Capacity Events query - Events with >1000 patrons |
| 15 | Button | Mouse | Clears the text area of all text |
| 16 | Button | Mouse | Leads to the Main Menu Frame |

## Help Frame

### Help Frame Components

* Layered Pane – Background
  + Label – Heading
  + Button – Help
  + Button – Close
  + Button – Prints the contents of the text area
  + Text Area – Displays the help text from the help text files

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| 0 | Frame |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Button | Mouse | Brings up the Help Frame |
| 4 | Text area |  |  |
| 5 | Button | Mouse | Starts the printing process - brings up the printing configurations to begin printing |
| 6 | Button | Mouse | Closes the Help Frame |

## Exit Frame



### Exit Frame Components

* Layered Pane – Background
  + Label – Title(You are about to exit)
  + Label – Confirmation message(Are you sure you want to exit
  + Button – Exit(closes application)
  + Button – Return(goes back to LoginFrame)

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **COMPONENT** | **INPUT** | **EVENT** |
| 0 | Frame (not visible) |  |  |
| 1 | Layered pane |  |  |
| 2 | Label |  |  |
| 3 | Label |  |  |
| 4 | Button | Mouse | Leads to the Admin Login Frame |
| 5 | Button | Mouse | Closes the application |

# 2.2 Program Flow Diagram

Splash

Login

Main Menu

o

Visits Tables

* View Visit Logs
* Search Visits Items
* Add Visit
* Edit Visit
* Delete Visit

Patrons Table

* View Patrons
* Search Patrons
* Add Patron
* Edit Patron
* Delete Patron

Reports Tables

* View SQL reports
* View data graphs
* Print reports

Events Table

* View Events
* Search Events
* Add Events
* Edit Events
* Delete Events

Help

# 2.3 Class design AND OOP Principles(Class Diagrams)

## Data classes

|  |
| --- |
| **Users** |
| - username : string |
| - password : string |
| - firstName : string  - surname : string |
|
| + Constructor() |
| + Constructor(username : string, password : string, firstName : string, surname : string) |
| + getUsername() : string |
| + getPassword() : string |
| + getFirstName() : string |
| + getSurname() : string |
| + setUsername(username : string) |
| + setPassword(password : string) |
| + setFirstName(firstName : string) |
| + setSurname(surname : string) |
| + toString() : string |

|  |
| --- |
| **Visits** |
| - visitNo : integer |
| - eventID : string |
| - patronID : string  - amountSpent : real |
| - username : string |
| - dateOfVisit : Date |
| + Constructor() |
| + Constructor(visitNo : integer, eventID : string, patronID : string, amountSpent : real, username : string, dateOfVisits : Date) |
| + getVisitNo() : string |
| + getEventID() : string |
| + getPatronID() : string |
| + getAmountSpent() : real |
| + getUsername() : string |
| + getDateOfVisit() : Date |
| + setVisitNo(visitNo : string) |
| + setEventID(eventID : string) |
| + setPatronID(patronID : string) |
| + setAmountSpent(amountSpent : real) |
| + setUsername(username : string) |
| + setDateOfVisit(dateOfVisit : Date) |
| + toString() : string |

|  |
| --- |
| **Patrons** |
| - patronID: string |
| - firstName: string |
| - surname: string |
| - gender: string |
| - dateOfBirth: Date |
| - homeAddress: string |
| - emailAddress: string |
| - cardLevel: string |
| - joinDate: Date |
| + Constructor() |
| + Constructor(patronID: string, firstName: string, surname: string, gender: string, dateOfBirth: Date, homeAddress: string, emailAddress: string, cardLevel: string, joinDate: Date) |
| + getPatronID(): string |
| + getFirstName(): string |
| + getSurname(): string |
| + getGender(): string |
| + getDateOfBirth(): Date |
| + getHomeAddress(): string |
| + getEmailAddress(): string |
| + getCardLevel(): string |
| + getJoinDate(): Date |
| + setPatronID(patronID: string) |
| + setFirstName(firstName: string) |
| + setSurname(surname: string) |
| + setGender(gender: string) |
| + setDateOfBirth(dateOfBirth: Date) |
| + setHomeAddress(homeAddress: string) |
| + setEmailAddress(emailAddress: string) |
| + setCardLevel(cardLevel: string) |
| + setJoinDate(joinDate: Date) |
| + toString(): string |
| **Events** |
| - eventID: string |
| - eventName: string |
| - startDate: Date |
| - endDate: Date |
| - location: string |
| - capacity: integer |
| - status: string |
| - registrationDeadline: Date |
| + Constructor() |
| + Constructor(eventID: string, eventName: string, startDate: Date, endDate: Date, location: string, capacity: integer, status: string, registrationDeadline: Date) |
| + getEventID(): string |
| + getEventName(): string |
| + getStartDate(): Date |
| + getEndDate(): Date |
| + getLocation(): string |
| + getCapacity(): integer |
| + getStatus(): string |
| + getRegistrationDeadline(): Date |
| + setEventID(eventID: string) |
| + setEventName(eventName: string) |
| + setStartDate(startDate: Date) |
| + setEndDate(endDate: Date) |
| + setLocation(location: string) |
| + setCapacity(capacity: integer) |
| + setStatus(status: string) |
| + setRegistrationDeadline(registrationDeadline: Date) |
| + toString(): string |

## Object classes

|  |
| --- |
| **UsersData** |
| - db: DbManager |
| - userList: List<Users> |
|  |
| + Constructor()  + getAllUsers(): |
|
| + getAllUsers(username: string): |
| + getAllUsers(userLevel: integer): |
| + getUsersList(sql: string): |
| + getUsername(username: string): Users |
| - db: DbManager |
| - userList: List<Users> |
| + UsersData() |
| + getAllUsers(): |
| + getAllUsers(username: string): |
| + getAllUsers(userLevel: integer): |
| + getUsersList(sql: string): |

|  |
| --- |
| **VisitsData** |
| - db: DbManager |
| - ed: EventsData |
| - pd: PatronsData |
| - ud: UsersData |
| - visitsList: List<Visits> |
|  |
| + Constructor() throws SQLException |
| + getAllVisits(): throws SQLException |
| + getAllVisitsVisitNo(visitNo: integer): throws SQLException |
| + getAllVisitsPatronID(patronID: string): throws SQLException |
| + getAllVisitsEventID(eventID: string): throws SQLException |
| + getAllVisitsUsername(username: string): throws SQLException |
| + getVisitsList(sql: string): throws SQLException |
| + getVisit(patronID: string): Visits |
| + getVisitPosition(visitNo: integer): integer |
| + addVisit(visitNo: integer, eventID: string, patronID: string, amountSpent: real, username: string, dateOfVisit: string): throws SQLException |
| + editVisit(visitNo: integer, eventID: string, patronID: string, amountSpent: real, username: string, dateOfVisit: string): throws SQLException |
| + deleteVisit(visitNo: integer): throws SQLException |
| + populateEventIDJComboBox(eventIDCombo: javax.swing.JComboBox): |
| + populatePatronIDJComboBox(patronIDCombo: javax.swing.JComboBox): |
| + populateUsernameJComboBox(usernameCombo: javax.swing.JComboBox): |
| + populateJTable(tblEvents: javax.swing.JTable, rowSelect: integer): |

|  |
| --- |
| **PatronsData** |
| - db: DbManager |
| - evd: EventsData |
| - patronsList: List<Patrons> |
|  |
| + Constructor() throws SQLException |
| + getAllPatrons(): throws SQLException |
| + getAllPatronsPatronID(patronID: string): throws SQLException |
| + getAllPatronsFirstName(firstName: string): throws SQLException |
| + getAllPatronsCardLevel(cardLevel: string): throws SQLException |
| + getAllPatronsGender(gender: string): throws SQLException |
| + getPatronList(sql: string): throws SQLException |
| + getPatron(patronID: string): Patrons |
| + getPatronPosition(firstName: string): integer |
| + addPatron(patronID: string, firstName: string, surname: string, gender: string, dateOfBirth: string, homeAddress: string, emailAddress: string, cardLevel: string, joinDate: string): throws SQLException |
| + editPatron(patronID: string, firstName: string, surname: string, gender: string, dateOfBirth: string, homeAddress: string, emailAddress: string, cardLevel: string, joinDate: string): throws SQLException |
| + deletePatron(patronID: string): throws SQLException |
| + populateJComboBoxCardLevel(patronCombo: javax.swing.JComboBox): |
| + populateJComboBoxGender(patronCombo: javax.swing.JComboBox): |
| + populateJTable(userTable: javax.swing.JTable, rowSelect: integer): |

|  |
| --- |
| **EventsData** |
| - db: DbManager |
| - eventsList: List<Events> |
| - ev: EventsData |
|  |
| + Constructor() throws SQLException |
| + getAllEvents(): throws SQLException |
| + getAllEventsEventID(eventID: string): throws SQLException |
| + getAllEventsEventName(eventName: string): throws SQLException |
| + getAllEventsLocation(location: string): throws SQLException |
| + getEventsList(sql: string): throws SQLException |
| + getEvent(eventID: string): Events |
| + getEventPosition(eventName: string): integer |
| + addEvent(eventID: string, eventName: string, startDate: string, endDate: string, location: string, capacity: integer, status: string, registrationDeadline: string): throws SQLException |
| + editEvent(eventID: string, eventName: string, startDate: string, endDate: string, location: string, capacity: integer, status: string, registrationDeadline: string): throws SQLException |
| + deleteEvent(eventID: string): throws SQLException |
| + populateLocationJComboBox(locationCombo: javax.swing.JComboBox): |
| + populateStatusJComboBox(statusCombo: javax.swing.JComboBox): |
| + populateJTable(tblEvents: javax.swing.JTable, rowSelect: integer): |

|  |
| --- |
| **DbManager** |
| conn: Connection |
| + Constructor() |
| + query(SQL: string): ResultSet throws SQLException |
| + update(SQL: string): integer throws SQLException |
| + updateReturnID(SQL: string): integer throws SQLException |

|  |
| --- |
| **ReportsData** |
| db: DbManager |
| + Constructor() throws SQLException |
| + pieChart(): throws SQLException |
| + barGraph(): throws SQLException |
| + getHighestSpender(): String throws SQLException |
| + getMostVisitsPatron(): String throws SQLException |
| + getEventLocation(): String throws SQLException |
| + getPatronCardType(): String throws SQLException |
| + getPatronGender(): String throws SQLException |
| + getHighCapacityEvents(): String throws SQLException |

## Validation Class

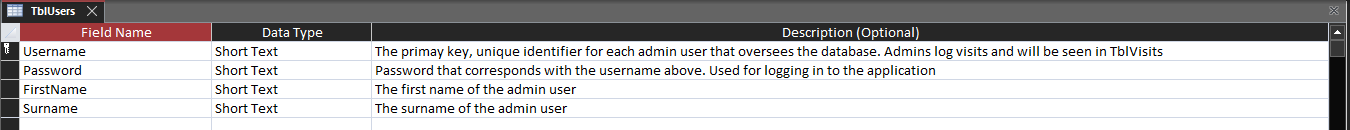
|  |
| --- |
| **Validation** |
| db: DbManager |
| + vVisitNoCheck(valueToCheck: string): real |
| + vPKCheck(valueToCheck: string, initial: char, primaryKeyField: string, table: string): real |
| + vAgeMin(dob: Date, minAge: integer): real |
| + vDateMin(endDate: Date, startDate: Date): real |
| + vDateMax(entry: Date, startDate: Date): real |
| + vBasicPresenceCheck(field: string): real |
| + v2IntialCharacters(numChars: integer, condition: string, entry: string): real |
| + vVirtualCombo(entry: string, option1: string, option2: string): real |
| + vVirtualCombo3(entry: string, option1: string, option2: string, option3: string): real |
| + vVirtualCombo4(entry: string, option1: string, option2: string, option3: string, option4: string): real |

# 2.4 Secondary Storage Design

Database will require the following tables containing appropriate data with the following structure. Sample data is also provided.

## TblUsers

### Design View



### Datasheet view

A screenshot of a computer

Description automatically generated

### Description

TblUsers stores information about the admin users which oversee the database. The username and password data will be used for logging in to the application upon startup. These admins also make visit logs found in TblVisits. As a result, their username will appear in TblVisits next to the data of the visit they have logged. This is to ensure accountability and easy tracing of logs.

## TblVisits

### Design View

A screenshot of a computer

Description automatically generated

### A screenshot of a computer Description automatically generatedDatasheet View

### Description

TblVisits is a linking table that makes use of data from all other tables. It acts as a log book that records any and all visits to the casino made by patrons as well as data related to the activity of the patron during the visit

## TblEvents

### Design View

A screenshot of a computer

Description automatically generated

### Datasheet View

### Description

TblEvents stores data about all the events of the casino, active, inactive, upcoming, cancelled and so on. All events are stored for recording purposes.

## TblPatrons

### Design View

A screenshot of a computer

Description automatically generated

### Datasheet view

### Description

TblPatrons is a contact list type of table that holds all the patrons and their related information.

## Relationships Diagram

A screenshot of a computer

Description automatically generated

## Help Textfiles

**A screenshot of a computer

Description automatically generatedAdmin Login help Screen**

A screenshot of a computer

Description automatically generated**Table Help Screen (.txt file content will differ for each table)**

**A screenshot of a computer

Description automatically generatedReports Frame Help Screen**

# 2.5 Explanation of Secondary Storage

### Why Choose a Database Over JSON or Text Files

#### Data Security, Integrity, and Consistency

**Databases**: Databases ensure data integrity and consistency through constraints (e.g., primary keys, foreign keys), validation techniques and so on. The ability to make sure data is valid is a huge concern. Data is also safer due to the password protection and other safety features present in database software.

**JSON/Text Files**: JSON or text files lack built-in mechanisms to enforce data integrity and consistency. Working with data without these mechanisms is error-prone and inefficient. JSON/Textfiles are largery unsafe due to their inablity to use password protection and other security features.

#### Complex Queries and Relationships

**Databases**: Databases excel at handling complex queries and relationships between entities. SQL allows for sophisticated querying and reporting, which is essential for generating insights from my casino data (e.g., patron visit history, event participation).

**JSON/Text Files**: Performing complex queries on JSON or text files is cumbersome and slow. You would need to load and parse entire files, which is inefficient for large datasets.

#### Scalability and Performance

**Database**: Databases are optimized for performance and can handle large volumes of data efficiently. Indexes, query optimization, and caching improve data retrieval times.

**JSON/Text Files**: As the data grows, the performance of JSON or text files degrades significantly. File I/O operations are slower, and parsing large files in memory can be resource-intensive.

#### Backup and Recovery

**Databases**: Databases provide robust backup and recovery options, including automated backups, point-in-time recovery, and replication.

**JSON/Text Files**: Implementing reliable backup and recovery processes for JSON or text files requires additional effort and custom solutions.

### Implications of Using a Database

#### Structured Data Storage

My design makes use of relational database tables (TblUsers, TblVisits, TblEvents, TblPatrons) to store date in a structured and efficient manner. Each table has clearly defined fields and data types, ensuring data is stored in a consistent format.

#### Data Relationships

The use of foreign keys (e.g., EventID in TblVisits referencing TblEvents, PatronID in TblVisits referencing TblPatrons) allows for establishing relationships between tables. This supports complex queries to retrieve related data across multiple tables, such as finding all visits made by a particular patron to specific events.

#### Data Integrity

Primary keys (e.g., Username in TblUsers, VisitNo in TblVisits, EventID in TblEvents, PatronID in TblPatrons) ensure each record is uniquely identifiable, preventing duplication and maintaining data integrity.

#### Efficient Data Retrieval

Indexes can be created on frequently queried fields (e.g., Username, EventID, PatronID) to speed up data retrieval operations, improving the overall performance of your casino management system.

#### Scalability

As your casino grows and the volume of data increases, the database can scale to accommodate the growth. Advanced database features like partitioning, indexing, and replication support scalability and high availability.

#### Security

Databases provide robust security features, including user authentication, role-based access control, and encryption. This is vital for protecting sensitive information about patrons, events, and visits.

# 2.6 Explanation of how Primary Data Structures relate to Secondary Storage

For every table that is present in secondary storage, there is a corresponding Object Class in the primary storage design. The object class corresponds with the table using different fields/class variables to represent each column in the table, each field being of the same data type as its corresponding column. A simple object in the object class will have data that relates to one record (row) in its corresponding table.

Each of these primary storage object classes has an associated data class. A data class refers to a class in which there is a list. This list stores many objects from one object class and thus stores multiple records from a table in the database. Auxiliary methods are used to process data in secondary storage and allow this data/each record to be stored in a list.

This relationship between secondary storage, primary storage and the associated data class is explained further through the following table and paragraph points:

|  |  |  |
| --- | --- | --- |
| Primary Storage | | Secondary Storage |
| Object class | Data Class | Database Table |
| Users | UsersData | TblUsers |
| Events | EventsData | TblEvents |
| Patrons | PatronsData | TblPatrons |
| Visits | VisitsData | TblVisits |

**Users**:

The primary function of users is to facilitate login. The Users class stores the attributes. The methods in the UsersData class are primarily for checking the credentials against the database to facilitate the login procedure. The method from the UsersData class is called by in the login button in the Admin Login screen

Data is pulled from the database when login credentials need to be verified.

**Events**:

The primary function of events is to keep a list of all events. The Events class stores the attributes. The methods in the EventsData class are primarily methods that are used in the GUI frame for displaying, searching/sorting and table functions.

**Patrons**:

The primary function of patrons is to keep a list of all patrons. The Patrons class stores the attributes. The methods in the PatronsData class are primarily methods that are used in the GUI frame for displaying, searching/sorting and table functions.

**Visits**:

The primary function of visits is to keep a list of all casino visits. The Visit class stores the attributes. The methods in the VisitsData class are primarily methods that are used in the GUI frame for displaying, searching/sorting and table functions.