**SDLC**

Software Development Life Cycle Documentation for a Client-Server System

**Project Overview**

The purpose of this project is to show basic ideas that can be implemented in a simple Registration process. This could be useful for hotels, restaurants, Airbnb’s, Airline reservations, etc. We tried to make the Graphical User Interface as easy to use as possible, given the limitations of the 1997 graphics library, or Java Foundation Classes, better known as Swing.

This project involves developing a client-server system for managing client records, including adding, updating, and deleting clients. The system features a graphical user interface (GUI) offering a user-friendly environment with text labels, text fields, spinners, radio buttons, a text area, and functionality to upload a client's picture. The server component manages the MySQL database operations, while the client component interacts with the server to perform CRUD (Create, Read, Update, Delete) operations.

**1. Requirements Gathering and Analysis**

**Functional Requirements**:

***Client Management***: Add new client records with the following details:

1. Name (First, Middle, Last)
2. Phone Numbers (Primary, Secondary)
3. Address (Street, City, State, Zip Code)
4. Email
5. Birthdate (Day, Month, Year)
6. Gender
7. Marital Status
8. Password and Confirmation
9. Notification Preferences
10. Notes
11. Profile Picture (Image upload)
12. Update existing client records.
13. Delete client records.
14. Display client information in a Table for easy viewing and selection.

*GUI Components*:

* 1. Text Fields: For inputting text data such as name, address, and email.
  2. Spinners: For selecting birthdate (day, month, year).
  3. Radio Buttons: For selecting gender and marital status.
  4. Text Area: For entering additional notes.
  5. File Upload: For uploading a client's profile picture.
  6. Buttons: To submit, update, delete, and clear form data.
  7. Socket connections between client and server.

***Server-Side:***

* 1. Handle database connections and queries (MySQL).
  2. Provide APIs for client-side operations: add, update, delete, and fetch client data.
  3. Ensure secure data transmission between client and server.

**Non-Functional Requirements:**

* 1. ***Performance***: The system should handle multiple simultaneous client connections efficiently.
  2. ***Security***: Implement secure communication (e.g., using SSL) between the client and server. Use prepared statements to prevent SQL injection.
  3. ***Scalability***: Design the system to accommodate future expansion (e.g., additional client fields or operations).
  4. ***Usability***: The GUI should be intuitive and easy to use, with clear labels and appropriate input validation.

**2. System Design**

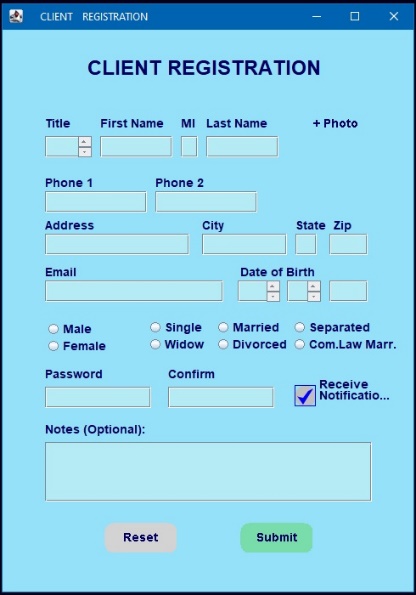
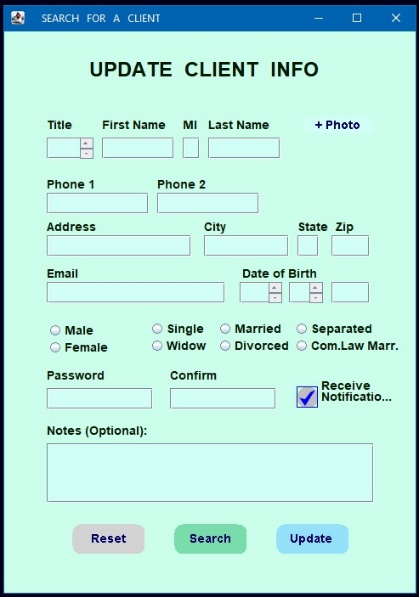
**High-Level Client-Server Architecture:**

- ***Client Application***: A Java Swing-based GUI that interacts with the user and sends requests to the server.

- ***Server Application***: A Java-based server that processes requests from clients, interacts with the database, and returns responses.

**Component Design:**

A screenshot of a web page

Description automatically generated  

***- Client Application:*** The following elements have been developed in customized methods.

* 1. Main JFrame: The main window containing all GUI components.
  2. JTextField: For entering text data.
  3. JSpinner: For selecting birthdates.
  4. JRadioButton: For gender and marital status selection.
  5. JTextArea: For entering notes.
  6. JFileChooser: For uploading a client's picture.
  7. JButton: For triggering add, update, delete operations.
  8. JTable: For displaying and selecting client records.
  9. ActionListener: For handling button clicks and other actions.

***- Server Application:***

* 1. Database Connection Class: Handles the connection to the MySQL database.
  2. CRUD Operations Class: Handles adding, updating, deleting, and retrieving client data from the database.
  3. ServerSocket Class: Listens for client requests and manages client connections.
  4. Request Handler Class: Processes client requests and interacts with the database to perform CRUD operations.

**Data Model:**

***Users Table (MySQL) schema:***

* 1. `ID` (Primary Key, Auto Increment)
  2. `FirstName` (VARCHAR)
  3. `MiddleName` (VARCHAR)
  4. `LastName` (VARCHAR)
  5. `Phone1` (VARCHAR)
  6. `Phone2` (VARCHAR)
  7. `Address` (VARCHAR)
  8. `City` (VARCHAR)
  9. `State` (VARCHAR)
  10. `ZipCode` (VARCHAR)
  11. `Email` (VARCHAR)
  12. `BirthDate` (DATE)
  13. `Gender` (ENUM)
  14. `MaritalStatus` (ENUM)
  15. `Password` (VARCHAR)
  16. `ConfirmPassword` (VARCHAR)
  17. `Notifications` (BOOLEAN)
  18. `Notes` (TEXT)
  19. `ProfilePicture` (BLOB)

**3. Implementation**

**Step 1: Setting Up the Server**

- ***Database Setup***: Create the `client` MySQL database and the `users` table.

- ***Server Setup***:

* 1. Develop the `DatabaseConnection` class for managing database connections.
  2. Implement the `CRUDOperations` class with methods to add, update, delete, and retrieve client records.
  3. Create the `ServerSocket` and `RequestHandler` classes to listen for client connections and process requests.

**Step 2: Developing the Client Application**

***- GUI Design:***

* 1. Create the main `JFrame` and organize the layout using `JPanel` components.
  2. Add `JTextField`, `JSpinner`, `JRadioButton`, `JTextArea`, and `JFileChooser` components to the form.
  3. Implement `ActionListener` classes to handle form submissions, updates, and deletions.
  4. Use `JTable` to display the client list and allow for selection and editing.
  5. Implement input validation to ensure that the user enters valid data.
  6. Client-Server Communication:
  7. Develop methods to send requests to the server and handle responses (e.g., using `Socket` and `PrintWriter`).
  8. Ensure secure transmission of sensitive data (e.g., using SSL).
  9. Implement methods to update the GUI based on server responses (e.g., updating the `JTable` with the latest client data).

**4. Testing**

**Unit Testing:**

- Test individual components (e.g., database connection, CRUD operations, GUI components) for correctness.

- Use JUnit to automate testing of server-side logic.

**Integration Testing:**

- Test the interaction between the client and server, ensuring that requests and responses are handled correctly.

- Simulate multiple client connections to test server concurrency and performance.

**User Acceptance Testing:**

- Involve end-users to test the GUI for usability and functionality.

- Gather feedback and make necessary adjustments to the design or functionality.

**5. Deployment**

**Server Deployment:**

* 1. Deploy the server application on a dedicated server or cloud platform.
  2. Ensure that the database is properly configured and accessible by the server application.
  3. Configure the server to start automatically and handle multiple client connections.

**Client Deployment:**

* 1. Package the client application into an executable JAR file.
  2. Distribute the client application to users with instructions for installation and use.

**6. Maintenance**

**Post-Deployment Support:**

* 1. Monitor the system for any issues or bugs reported by users.
  2. Provide regular updates to improve functionality or address any security vulnerabilities.
  3. Maintain the database, ensuring regular backups and performance tuning.

**Conclusion**

This SDLC documentation outlines the process of developing a client-server system with a Swing-based GUI for managing client records. By following these steps, the project can be executed smoothly, ensuring that the final product meets the functional and non-functional requirements. Regular testing and maintenance will ensure the system remains reliable and efficient for end-users.

Addendum

**User Manual**

This is a guide for end-users on how to use the client’s interface. The Welcome screen displays a small logo of the company and a set of buttons under the heading Client Registration.

Sections:

1. **Adding a client**: On the welcome screen click on the Add New button for the Client Registration form to appear. Fill as many fields as possible.
2. **Uploading a photo**: On the top right corner of the Client Registration form, there is a “ + Photo” label that is actually a button. Once clicked a browser window will pop up to help you find and “upload” the client’s photo.
3. **Updating a client**: To change any part of a client's information, or photo, there are to ways to upload new data. First, you can click on the “Update User” button on the Welcome screen. Or second, you can click on the “Update” button on the client’s table, but first you have to click on any row for the button to pop up. This will open up the client’s information screen, in which you can change any thing and the click the “Update” button.
4. **Deleting** **a client**: On the same table and after clicking on a row, the “delete” button will also pop up. The previously selected record, highlighted in a yellow row, will be deleted from the database.

After a small description of our company and our goals, the Welcome screen displays also some of our engineers who are also included in the database. By clicking on any photo you will open their client screen, with their information.

**Project’s Gantt Chart**

A screenshot of a computer

Description automatically generated