Software Design Specification (SDS)

Project Name: [GlowVibe beauty clinic Web Application]
Prepared By: [Rofida, Menna, Abeer, Sara, Manar]
Date: [9/11/2024]

---

1. Introduction

1.1 Purpose

The purpose of this document is to describe the design, architecture, and technical specifications of the **GlowVibe** platform. It outlines the functionality, system components, and design decisions to be followed during development. The goal is to create a user-friendly and efficient platform that simplifies appointment scheduling and consultation management for beauty services, enhancing the experience for clients and professionals alike.

1.2 Scope

This SDS covers the design and implementation details of the **GlowVibe** platform. The software will perform the following major tasks:

- **User Registration and Profile Management**: Allow clients and professionals to register, log in, and manage their profiles.
- **Appointment Booking and Scheduling**: Enable clients to view available times and book, modify, or cancel appointments with beauty professionals.
- **Service Catalog and Reviews**: Display a categorized list of services, allow clients to filter services by category and price, and enable clients to leave reviews and ratings for professionals.

---

2. System Overview

The system consists of the following components:

- **Frontend:** HTML, CSS, and JavaScript for a responsive and interactive user interface.
- **Backend:** Python (object-oriented programming) for handling server-side logic.
- **Database:** MySQL to manage customer data, bookings, and service information.

---

3. System Architecture

3.1 Architectural Design

This project follows the [Insert architecture type: e.g., client-server, microservices, monolithic] architecture, where:

- **Frontend** communicates with the backend using [API/Protocol].

- Restful API and HTTPS protocol to ensure secure and reliable data transmission.
- **Backend** interacts with the database to manage and retrieve data.

- the backend interacts with my SQL database to store and manage data related to the users

3.2 Data Flow

1. **User Interaction**: The user interacts with the UI to perform an action (e.g., create a new entry, update an existing record).

   the users interact with the frontend interface by selecting the service, booking an appointment, or viewing a profile

2. **Request Processing**: The frontend sends an API request to the backend server.
   the frontend users send API requests to the backend with any necessary data

3. **Data Handling**: upon receiving the request the backend processes the data validations and then interacts with the database (my SQL), for example: if the user wants to booking will check the availability first and then store the details

4. **Response**: The backend sends the response back to the frontend, updating the UI.

   the backend sends the response to the frontend, providing the result of the action (confirmation or error message)the frontend updates the user UI.

---

4. Database Design

4.1 Database Schema

The system will store data in relational MySQL with the following entities and relationships:

Table 1: **USERS** stores information about all users of the system, including clients, beauty professionals, doctors, and admins

- **UserID**: Unique identifier for each user
- **UserType**: Specifies if the user is a client, beauty professional, doctor, or admin
- **Email**: User's email address
- **PasswordHash**: Encrypted password for secure authentication

Table 2: **APPOINTMENTS** tracks all appointment bookings made by clients with beauty professionals or doctors.

- **AppointmentID**: Unique identifier for each appointment
- **UserID**: Foreign key linking to the Users table
- **ServiceID**: Foreign key linking to the Services table
- **AppointmentTime**: Date and time of the appointment
- **Status**: Current status of the appointment (e.g., scheduled, canceled)

Table 3: **Services** contains details about the various services offered by beauty professionals and doctors.

- **ServiceID**: Unique identifier for each service
- **ProfessionalID**: Foreign key linking to the Users table (for beauty professionals)
- **ServiceName**: Name of the service
- **Description**: Detailed description of the service
- **Price**: Cost of the service

Table 4: Reviews stores clients' feedback on their experiences with beauty professionals and services.

- **ReviewID**: Unique identifier for each review
- **UserID**: Foreign key linking to the Users table
- **ProfessionalID**: Foreign key linking to the Users table
- **Rating**: Numeric rating given by the client
- **Comment**: Textual feedback provided by the client

Relationships:

Users and **Appointments**: One-to-Many

Users and **Services**: One-to-Many

Users and **Reviews**: One-to-Many

Appointments and **Services**: Many-to-One

---

5. Technology Stack

- **Frontend**: React.
- **Backend**: Node.js.
- **Database**: MySQL.
- **Hosting**: Azure, heroku

---

6. Testing Plan

6.1 Unit Testing

Each module and function will undergo unit testing to ensure that individual components are working as expected.
Tools: Jest for frontend (React), Pytest for any Python-based backend components, or Mocha for Node.js.
test individual UI components, backend logic, for example: (booking functionality, authentication) and database interactions independently to confirm that every tiny component operates as planned.
Testing will continue during development and after every significant code update.

6.2 Integration Testing

Integration tests will validate that different modules (frontend and backend, or backend and database) work together as expected.
We will Focus on scenarios where multiple components must work together, such as when a user books an appointment (frontend requests the backend, backend retrieves data from the database
Tools: Postman for API endpoint testing and Supertest for HTTP assertions in Node.js.
We will test after each feature or module integration and before any deployment.

6.3 User Acceptance Testing (UAT)
End users will be involved in testing the system to verify that it meets their requirements and expectations.
Test important user processes like registering for an account, scheduling appointments, looking up professional profiles, and giving feedback.
Tools: Test scripts and questionnaires for feedback and Google form for collecting feedback

6.4 Performance Testing
Stress and load testing will be conducted to ensure the system can handle the required number of users and operations without degradation in performance.
Tools: Apache JMeter or Gatling for load testing we will test scenarios with high user concurrency and examine server load, stability, and performance.

---

7. Conclusion
GlowVibe is designed to satisfy the functional and non-functional requirements detailed in this document. Its system architecture ensures that it will be dependable, user-friendly, and capable of adapting to future needs. By simplifying the appointment booking and service management processes, **GlowVibe** aims to offer a valuable and intuitive platform for beauty and wellness clients and professionals.