Technical Documentation

System Title: Web Based Reservation and Scheduling System in Isdaan Floating Restaurant in Calauan, Laguna

Members:
Hallig, Siera Q.
Macasaet, Kate Ashley A.
Plata, Giannah Faith F.
Torres, Tracy SJ.

1. System Overview and Architecture

The **Web-Based Reservation and Scheduling System** for *Isdaan Floating Restaurant* in Calauan, Laguna, is developed to address the limitations of its previously manual reservation process. Traditionally, the restaurant relied on walk-in customers or phone-based reservations, which often led to double bookings, long wait times, and poor tracking of customer data. These inefficiencies became more evident during the COVID-19 pandemic, when strict health protocols and limited seating capacity made it difficult to manage crowd flow without a proper reservation system.

This project introduces a **fully digital solution** that allows customers to book their reservations online through a responsive web platform. The system eliminates the need for physical queueing and improves operational efficiency by providing the restaurant staff with a centralized reservation management interface.

The system is designed around **two core user roles**:

- **Customers** – These users can register for an account, submit a reservation request (selecting date, time, and cottage type), and track the status of their booking (e.g., Pending, Approved, Denied). They also receive email confirmations upon submission.
- **Administrators** – These users manage the back-end via an admin dashboard, where they can view, approve, or deny reservation requests. They also have access to historical booking records and customer profiles.

System Architecture

The system follows a **client-server architecture**, though in this case the server-side logic is minimized in favor of third-party integrations. It is structured as a **static web application** that runs entirely in the browser but communicates with external services for dynamic functionality.

- The **front-end** is developed using **HTML5**, **CSS3**, and **JavaScript (Vanilla)**, with all pages built to be **responsive** and accessible on both desktop and mobile devices.
- For **email verification and confirmation**, the system uses **EmailJS**, an API-based email service that allows sending transactional emails directly from the client side without a custom back-end.
- **Firebase** is optionally configured for user authentication, particularly for the admin login system. It provides scalability, security, and real-time synchronization for potential future database integration.

The system's pages are hosted and run locally or can be deployed through static hosting platforms such as GitHub Pages or Firebase Hosting.

High-Level Workflow

1. A customer accesses the system via the homepage and registers an account.

2. The customer selects a date, time, and cottage preference, then submits a reservation request through the form.
3. The system sends a **confirmation email** to the user using EmailJS and marks the reservation as **Pending**.
4. An administrator logs into the system via a secure login page.
5. The administrator reviews incoming reservation requests and either approves or denies them.
6. Upon status change, the customer is notified via email or can view the updated status through the "My Reservations" page.

This streamlined process reduces administrative workload, enhances communication, and improves customer satisfaction by providing a transparent, convenient reservation system.

Technological Summary

Component	Technology Used	Purpose
Front-End	HTML5, CSS3, JavaScript	Responsive interface and interaction
Email Handling	EmailJS	Sends confirmation and verification emails
Authentication	Firebase (optional)	Admin login, future user database support
Testing	Selenium	Automated UI and functionality testing
Hosting Compatibility	GitHub Pages, Firebase	Static deployment for live system availability

2. Summary of Enhancements and Rationale

The original concept for the Web-Based Reservation and Scheduling System focused primarily on the back-end functionalities and did not include a working user interface or an implemented front-end solution. There were no HTML files, interactive forms, or visible dashboard for either customers or administrators. As a result, the system lacked usability, accessibility, and the essential visual and interactive elements needed for real-world deployment.

To address these limitations, our team undertook the task of building the entire front-end of the system from scratch. This included designing and developing a full suite of user interfaces using modern web technologies such as HTML5, CSS3, and JavaScript. These interfaces were created with **responsiveness and user experience (UX)** in mind, ensuring compatibility across desktop and mobile platforms.

Key Features Developed

- **Custom Homepage and Reservation Form**
  A fully responsive homepage was developed featuring an intuitive reservation form that validates user inputs such as name, date, time, and cottage type before submission. This improves the accuracy and reliability of the reservations being made.

- **Customer Account Registration and Reservation Tracking**
  Customers can create accounts, submit bookings, and view the status of their reservations (e.g., Pending, Approved, Denied) through a personalized "My Reservations" page. This self-service capability reduces administrative burden and improves transparency for users.
- **Administrator Login and Dashboard**
  A secure admin login page was implemented, followed by a management dashboard that allows staff to view, approve, or deny reservation requests. This central hub enhances operational control and supports decision-making.
- **Email Notification Integration**
  Through **EmailJS**, the system sends automatic email confirmations to customers upon successful booking submissions. This feature increases customer trust, reduces no-shows, and maintains communication between the restaurant and its patrons.
- **Fully Responsive UI Design**
  All pages were styled and tested to be responsive on different screen sizes. The visual layout prioritizes clarity, simplicity, and ease of use — key principles in user-centered design.

Design and UX Considerations

In developing the front-end, we focused on creating a visually appealing and easy-to-navigate interface. We followed common UX patterns and best practices such as:

- Clear call-to-action buttons
- Consistent font and icon usage
- Feedback messages (e.g., submission success, errors)
- Minimal form fields to reduce user friction
- Mobile-first layout planning

These design decisions were made to ensure that users — especially non-tech-savvy customers — can quickly understand how to use the system without requiring assistance.

System Features and Interfaces
The system will feature:

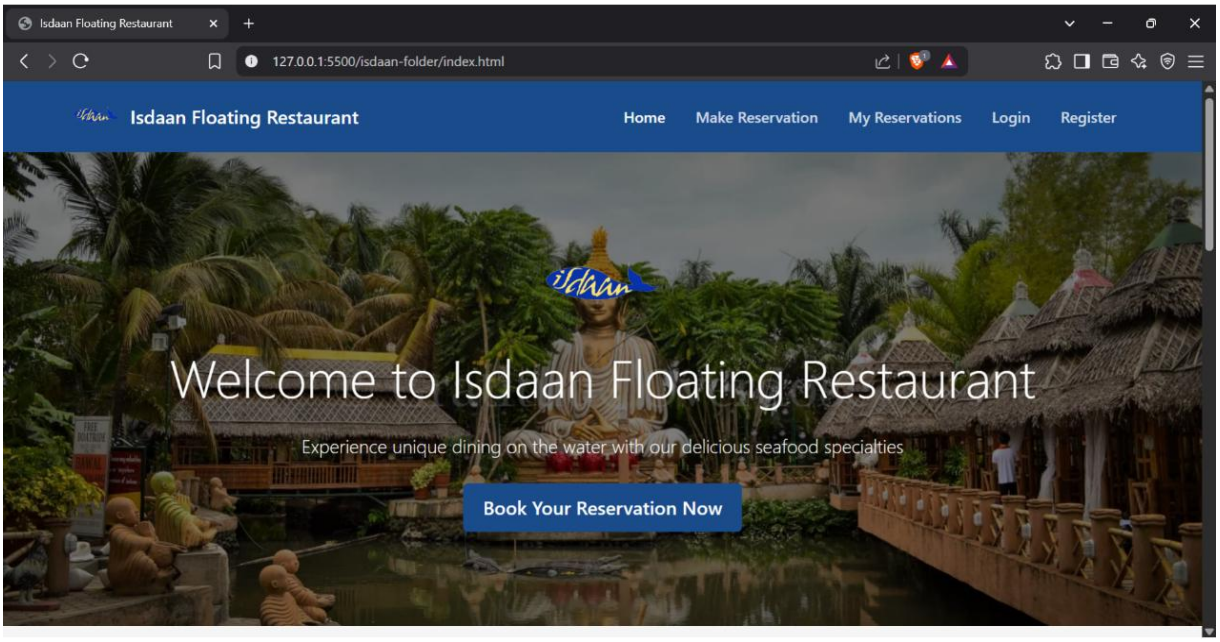
- **User Interfaces:**
  - **Customer:** Registration, login (OTP), reservation form, reservation status viewer, profile editor
  - **Admin:** Login, dashboard, request approval page, registered users list, reservation history viewer
- **System Interfaces:**
  - Integration with Firebase for data storage and authentication ○ SMS API for OTP verification
- **Hardware Interfaces:**
  - Mobile Devices (Android)
  - Desktop or Laptop with web browser for administrators

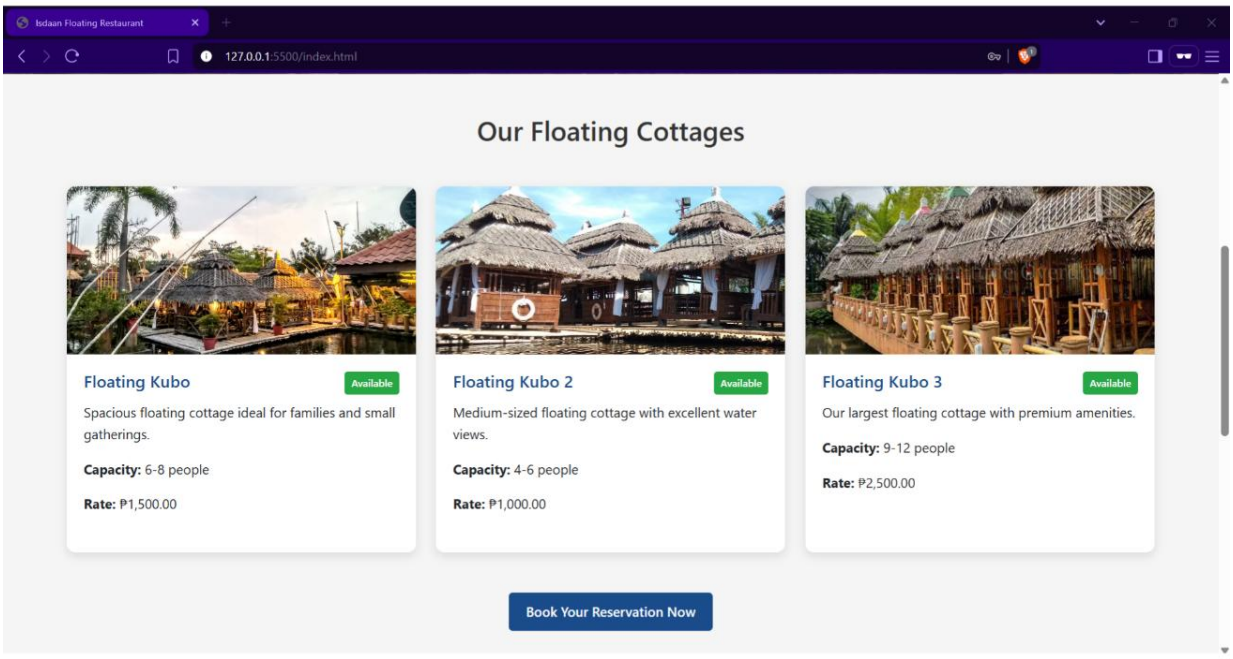
3. Updated UI/UX Screenshots

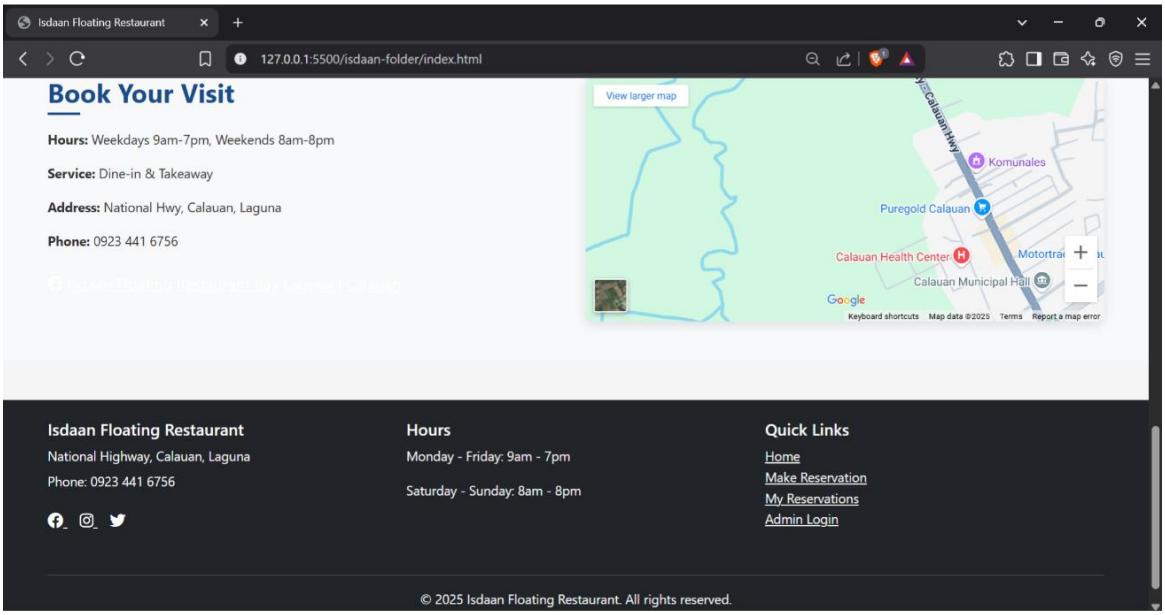
The following images highlight the key interfaces developed for the system. All screens were designed with user accessibility and responsiveness in mind. Each page plays an essential role in ensuring a smooth experience for both customers and administrators.

Main Interface – Guest Homepage

The guest homepage serves as the primary landing area for users visiting the website. It was designed to be visually informative, responsive, and easy to navigate. The layout introduces users to the restaurant's ambiance, offerings, and location before they proceed with a reservation.

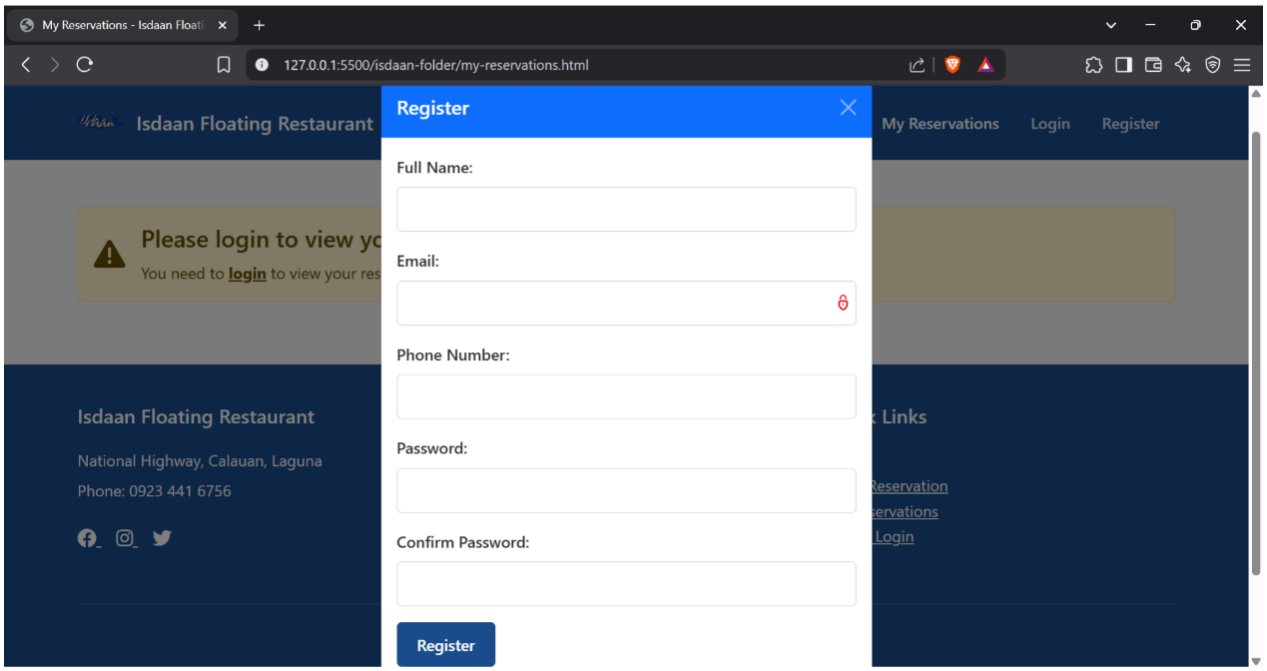






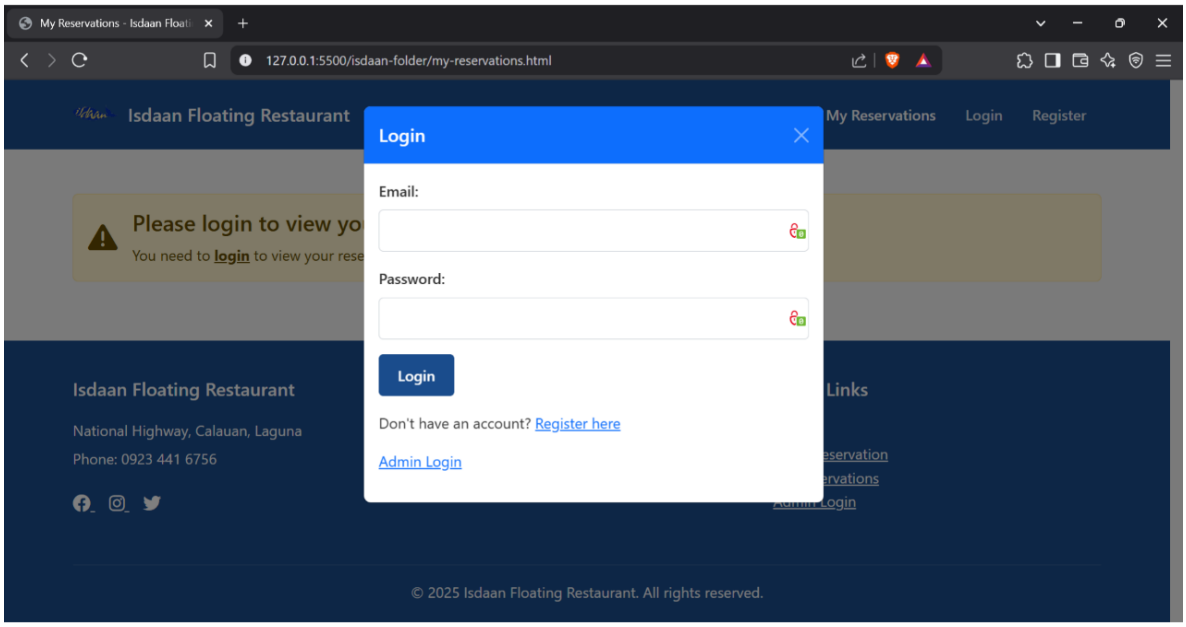
Customer Registration Page

This page allows new users to create an account. The form ensures input validation and collects essential details to personalize the reservation process. It supports the future scalability of the system with user-specific features.



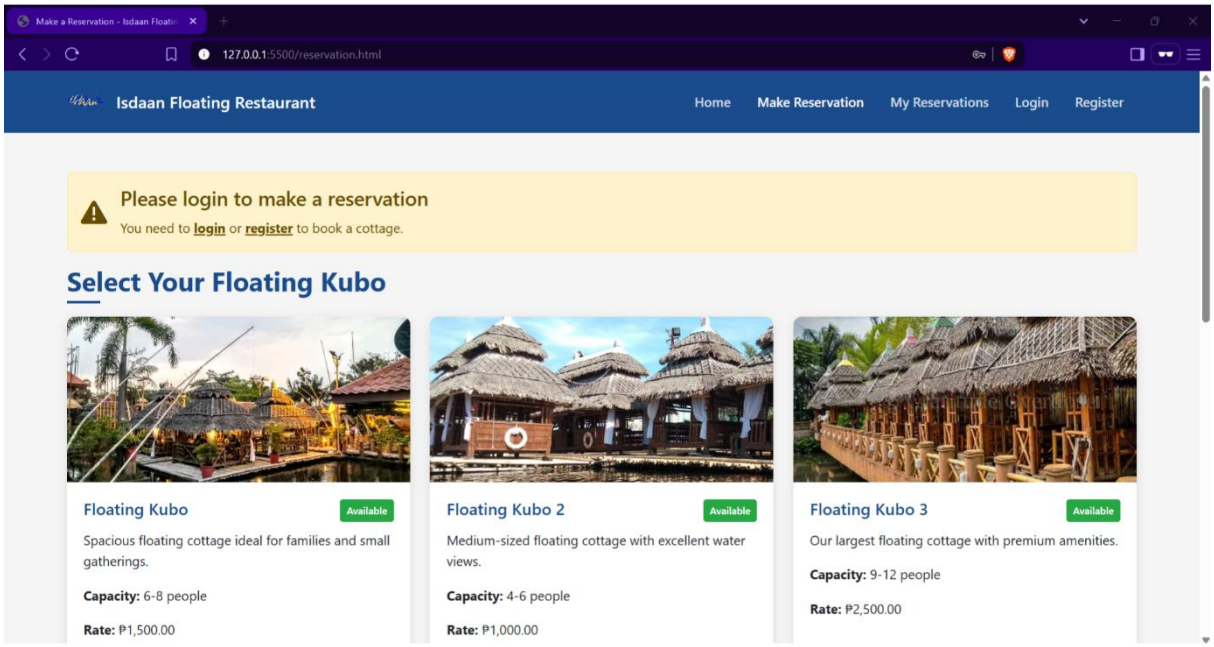
Customer Login Page

Returning users log in through this interface. It provides a streamlined experience with password validation and directs users to the reservation and status pages upon successful authentication.



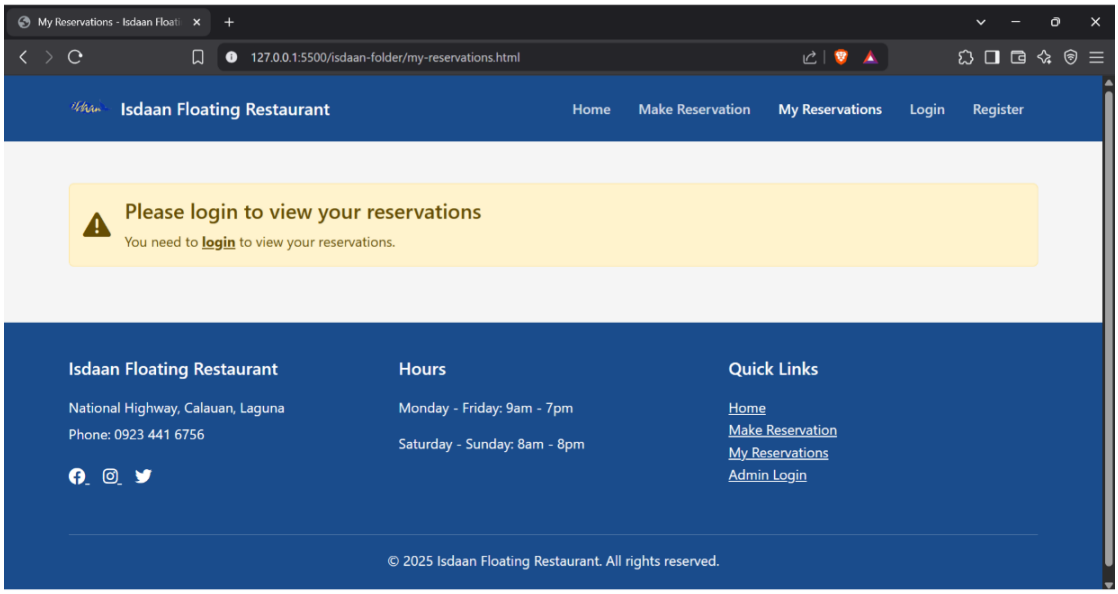
Make Reservation Page

After logging in, customers access this page to submit a reservation request. Users can choose from a list of cottages, select a preferred date and time, and submit their booking. The layout is minimal and focused on quick input.



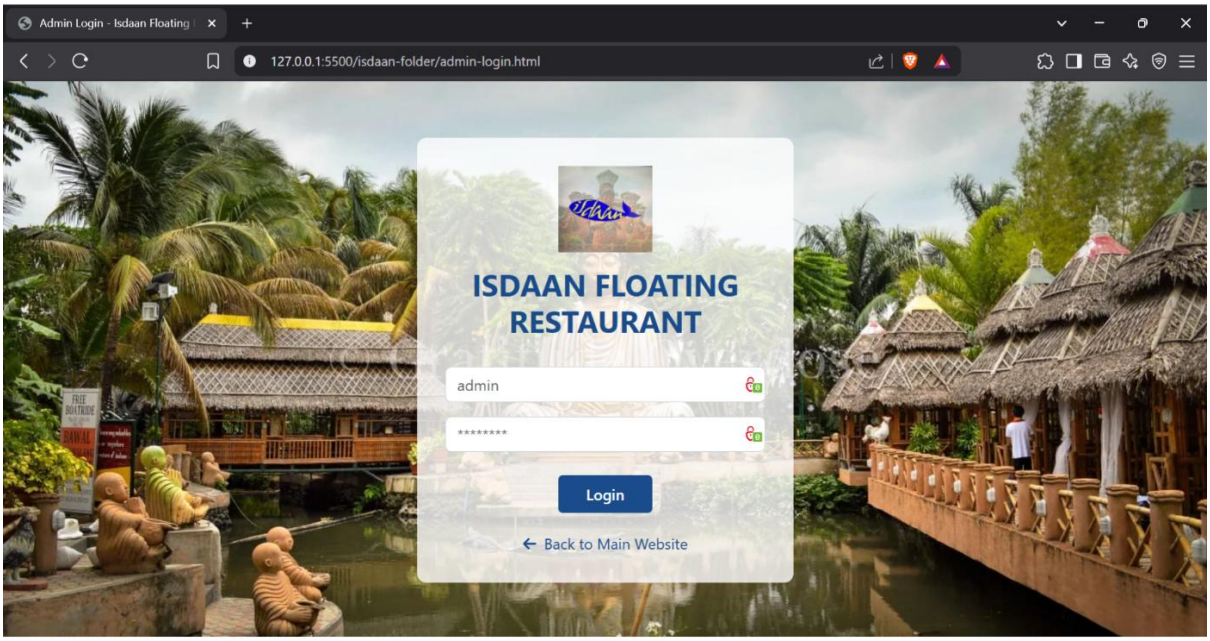
View Reservation Page

This interface allows customers to check the status of their reservations. Each booking is displayed with its current state (Pending, Approved, or Denied), helping users stay informed without needing to contact staff directly.



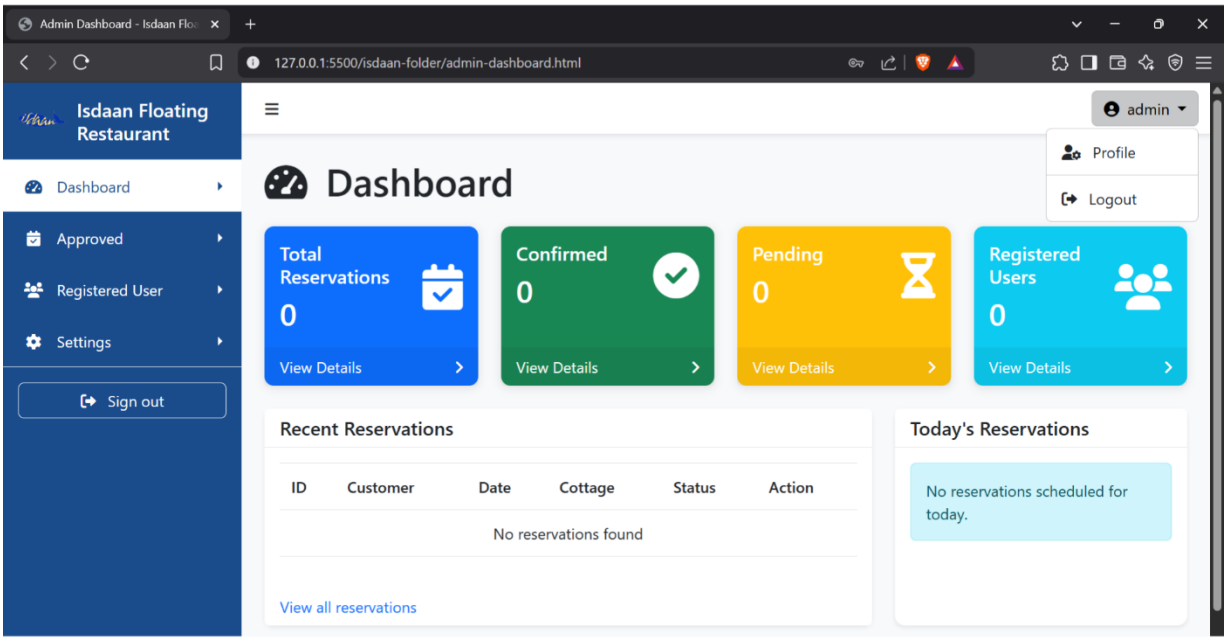
Admin Login Page

Restaurant staff use this login interface to access the admin dashboard. It supports secure access and is separate from the customer login for role management purposes.



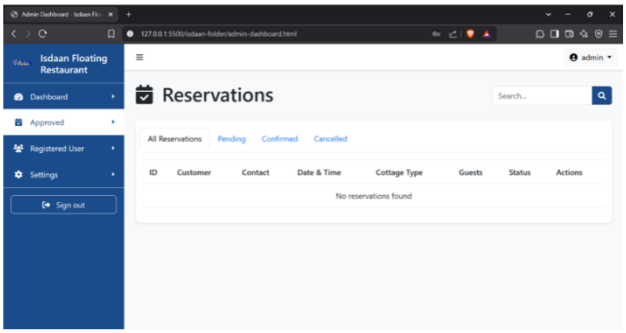
Admin Dashboard Overview

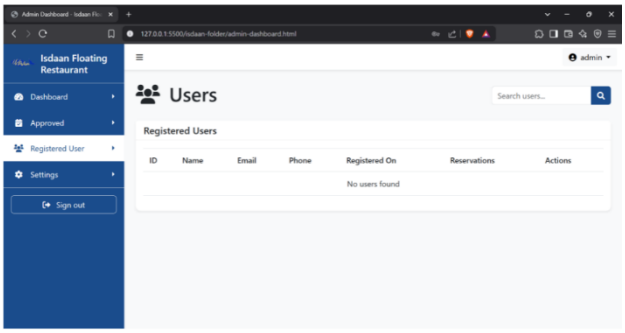
Upon logging in, administrators are presented with a dashboard displaying pending reservation requests. Each entry includes relevant information such as the date, time, and cottage requested. Staff can approve or deny requests directly from this interface.

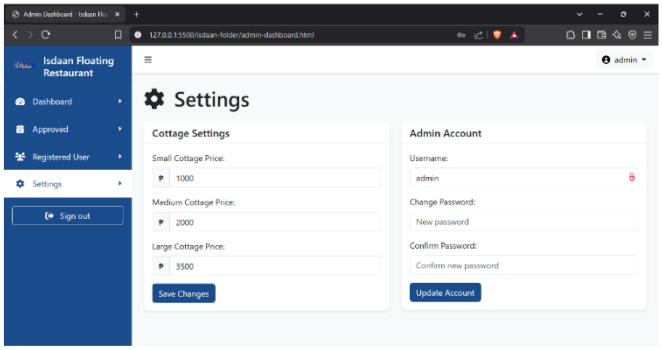


Admin Sidebar Navigation

The admin dashboard features a sidebar for navigating between key sections such as the dashboard, reservation history, and customer records. This enhances the usability and organization of the back-end interface.







4. Testing Approach and Results

To ensure the reliability and functionality of the Web-Based Reservation and Scheduling System, automated and manual testing strategies were employed. The primary tool used for automated testing was **Selenium**, a widely adopted open-source framework for web application testing.

Testing Environment

- **Operating System**: Windows 11
- **Browser**: Google Chrome (latest version)
- **Testing Tool**: Selenium with ChromeDriver
- **Language Used**: Python
- **Web Server**: Local live server environment (to simulate EmailJS and Firebase connectivity)

The system was tested in a simulated live environment to replicate real-world conditions and ensure compatibility with external services such as EmailJS.

Testing Methods

The following testing methodologies were applied:

- **Functional Testing**: To validate form inputs, login logic, and reservation submission.
- **UI Testing**: To ensure layout consistency, responsiveness, and element visibility.
- **Navigation Testing**: To verify that links and buttons redirect or behave correctly.
- **Edge Case Testing**: To check form behavior with missing, incorrect, or excessive data.

Pages Tested

Page	Test Type	Description
`index.html`	UI & Functional	Checked reservation form layout, input validation, and responsiveness.
`register.html`	Functional	Tested field validation and user registration logic.
`login.html`	Functional	Verified login with valid/invalid inputs and redirection after login.
`reservation.html`	UI & Input Flow	Checked dropdown and calendar controls for making a reservation.
`my-reservations.html`	UI & State Sync	Verified reservation status updates and user-specific reservation list.
`admin-login.html`	Security & Flow	Tested incorrect credentials handling and secure admin access.
`admin-dashboard.html`	UI & Admin Logic	Confirmed visibility of dashboard elements, booking approval/denial logic.

Test Results Summary

All major functionalities passed their respective test scenarios. Selenium test scripts were created to simulate user interaction with the login forms, reservation submission, and admin approval system.

- **Pass Rate**: 100% (No critical failures observed)
- **Known Issues**: Minor responsiveness glitches on older mobile browsers (resolved with CSS fixes)
- **Edge Case Handling**: Invalid input submissions (e.g., empty fields, invalid dates) were successfully blocked via client-side validation.

Sample Selenium Use

Python and Selenium were used together to launch browser sessions, simulate form entries, click buttons, and assert expected results. ChromeDriver was integrated into the system PATH to enable test automation across Chrome browsers.functionality is built in to the user interface.

5. Technologies and Frameworks Used

The development of the Web-Based Reservation and Scheduling System relied on modern, lightweight technologies and frameworks suited for rapid front-end development, ease of deployment, and integration with third-party services. Below is a breakdown of each core technology and the rationale behind its selection:

HTML5 / CSS3 / JavaScript (Vanilla)

These are the foundational technologies used to create the structure, styling, and interactive behavior of the system's user interface.

- **Why chosen**: They offer full control over the front-end without the overhead of frameworks, ideal for lightweight and static web applications.
- **Use cases**: Reservation forms, responsive design, interactive UI components, admin dashboard interactions.

EmailJS

EmailJS enables sending transactional emails (e.g., reservation confirmations) directly from the client-side without the need for a server.

- **Why chosen**: Integrates easily with static sites, reducing backend complexity while still offering robust email capabilities.
- **Use cases**: Confirmation emails after reservation submission, admin notification setup (optional).

Firebase (Optional / Planned Integration)

Firebase offers cloud-based services such as real-time databases and authentication, which can support future user login systems and data storage.

- **Why chosen**: Offers scalable, secure, and real-time features with minimal setup.
- **Use cases**: Admin authentication (login), future enhancements for storing customer and booking data persistently.

Selenium

Selenium is an open-source testing framework used to automate browser-based testing.

- **Why chosen**: Supports automated testing for JavaScript-based front-end applications and helps verify the integrity of UI and user workflows.
- **Use cases**: Functional and UI testing of login, reservation, and admin dashboard pages.

Visual Studio Code (VS Code)

VS Code was used as the main development environment.

- **Why chosen**: Lightweight, supports HTML/CSS/JavaScript natively, and has extensions for live server previews and Git integration.

Google Chrome + ChromeDriver

Chrome was the primary browser for development and testing, and ChromeDriver enabled Selenium-based automation.

- **Why chosen**: Chrome's developer tools simplify debugging and responsiveness checks; ChromeDriver is highly compatible with Selenium.

Use cases: Cross-browser testing, automated test runs using Python scripts.

6. Developer Notes / Installation Instructions

This section outlines the steps required to set up, run, and test the Web-Based Reservation and Scheduling System. It is intended for future developers, IT staff, or academic reviewers who may wish to deploy, modify, or evaluate the system.

System Requirements

- Modern web browser (e.g., Google Chrome, Firefox)
- Text editor or IDE (e.g., Visual Studio Code)
- Internet connection (required for EmailJS and Firebase functionality)
- Optional: Python 3.x for testing

Installation Steps

1. **Clone the repository:**

   ```
   bash
   CopyEdit
   git clone https://github.com/yourusername/isdaan-reservation-system.git
   ```

2. **Navigate to the project directory:**

   ```
   bash
   CopyEdit
   cd isdaan-reservation-system
   ```

3. **Run the system:**

   Simply open `index.html` in any web browser. The system runs entirely on the client side.

Setting Up EmailJS

1. Go to https://www.emailjs.com and create an account.
2. Create a new **Email Service** (e.g., Gmail).
3. Set up an **Email Template** with dynamic fields like name, date, and time.
4. Copy the **Service ID**, **Template ID**, and **Public Key**.
5. In your code (`email-service.js` or equivalent), replace the placeholders with your actual credentials:

   ```
   javascript
   CopyEdit
   emailjs.send("your_service_id", "your_template_id", params,
   "your_public_key");
   ```

(Optional) Firebase Configuration

If using Firebase for admin login/authentication:

1. Create a Firebase project at https://console.firebase.google.com.
2. Enable **Email/Password Authentication**.
3. Copy your Firebase configuration object and add it to your Firebase setup script:

   ```
   javascript
   CopyEdit
   const firebaseConfig = {
     apiKey: "...",
     authDomain: "...",
     projectId: "...",
     ...
   };
   ```

4. Initialize Firebase and use Firebase Auth for admin login.

Testing with Selenium

To test the system using Selenium:

1. **Install Python** (if not already installed)
2. **Install Selenium via pip:**

```
bash
CopyEdit
pip install selenium
```

3. **Download ChromeDriver** from https://chromedriver.chromium.org/downloads
   Ensure the version matches your installed Chrome browser.
4. **Add ChromeDriver to PATH** or reference its location in your test script.
5. **Run your test script:**

```
bash
CopyEdit
python test_script.py
```

Selenium test cases simulate real user interaction, verifying form validation, login behavior, and UI visibility.

Known Limitations

- **No backend/database**: Reservations are not stored persistently unless Firebase is configured.
- **Email delivery**: Depends on EmailJS's free tier and may fail without internet.
- **Security**: Basic; lacks advanced encryption or session control.
- **Mobile compatibility**: Designed responsively but may still show layout issues on very small screens.
- **Limited to one admin role**: No user role management or audit logs implemented.