**SUMMER INTERNSHIP 2023**

**Project Report**

**Members:**

**Akshay Malik (202212008)**

**Lakshay Malik (202212051)**

**Project Title: Faculty Appointment System**

* **Introduction**

The Faculty Schedule Appointment System by students based on MERN Stack is a web application that empowers students to schedule appointments with their faculty members in an organized and efficient manner. This system aims to facilitate seamless communication between students and faculty, allowing students to seek guidance, discuss academic matters, and resolve queries effectively.

**Key Features:**

* **Student Authentication:** Students will have their user accounts, which they can access by registering or logging in securely. This ensures that only authorized students can schedule appointments.
* **Faculty Profile:** The system will maintain profiles for each faculty member, including their availability, office hours, contact information, and specialization.
* **Appointment Scheduling:** Students can view faculty availability through a user-friendly calendar interface and request appointments based on open slots.
* **Real-Time Notifications:** The system will send automated notifications to students to confirm their appointment requests, inform them of any changes or cancellations, and remind them of upcoming appointments.
* **Appointment Management:** Faculty members can review and manage their appointment schedule, accepting or rescheduling appointments as needed.
* **User Dashboard:** Students will have a personalized dashboard displaying their upcoming appointments, past appointments, and any pending requests.
* **Privacy and Security:** Ensuring data privacy and security is crucial. The system will implement robust security measures to protect user information.

**Technology Stack:**

* **Front-end:** React.js will be used to build the user interface, providing a smooth and responsive experience for students.
* **Back-end:** Node.js and Express.js will handle server-side logic and API development.
* **Database:** MongoDB will store and manage the application's data efficiently.
* **User Authentication:** Implement JSON Web Tokens (JWT) for secure and reliable student authentication.
* **Real-Time Notifications:** Socket.io can be utilized to enable real-time communication and instant notifications.
* **State Management:** Redux or React Context API will manage the application's state and data flow.

**Benefits:**

* **Efficient Communication:** The system streamlines the process of scheduling appointments, leading to improved communication between students and faculty members.
* **Time-Saving:** Students can quickly view faculty availability and secure appointments without the need for manual coordination.
* **Improved Faculty-Student Interaction:** Regular appointments foster stronger faculty-student relationships and a more conducive learning environment.
* **Flexibility:** Students can access the system at any time, making it convenient to schedule appointments according to their availability.
* **Functional and Non-Functional Requirements**

**Functional Requirements:**

* **User Registration and Authentication:**

Students can register with the system using their email or university ID.

* **Faculty Profile Management:**

Faculty members can create and update their profiles with details such as name,

contact information, office hours, and areas of expertise.

* **Appointment Scheduling:**

Students can select available time slots and request appointments with specific

faculty members.

* **Real-Time Notifications:**

Both students and faculty should receive real-time notifications for

appointment requests, confirmations, and cancellations.

* **Appointment Management:**

Faculty members can view their appointment schedule, including upcoming

and past appointments.

* **User Dashboard:**

Students should have a personalized dashboard displaying their appointment

history and pending requests.

**Non-Functional Requirements:**

* **Security:**

The system should implement robust security measures to protect user data

and prevent unauthorized access.

* **Performance:**

The application should be responsive and provide quick response times, even

during peak usage periods.

* **Scalability:**

The system should be designed to accommodate a growing number of users

and appointment requests without compromising performance.

* **User Interface (UI) and User Experience (UX):**

The UI should be intuitive and user-friendly, making it easy for students and

faculty to navigate and use the system.

* **Availability:**

The system should have high availability to ensure students can schedule

appointments whenever needed.

* **Compatibility:**

The application should be compatible with various web browsers and devices

to cater to different users' preferences.

* **Data Integrity:**

The system should maintain data integrity, ensuring that appointment

schedules and user information remain accurate and consistent.

* **Error Handling and Logging:**

The system should handle errors gracefully and provide clear error messages

to users.

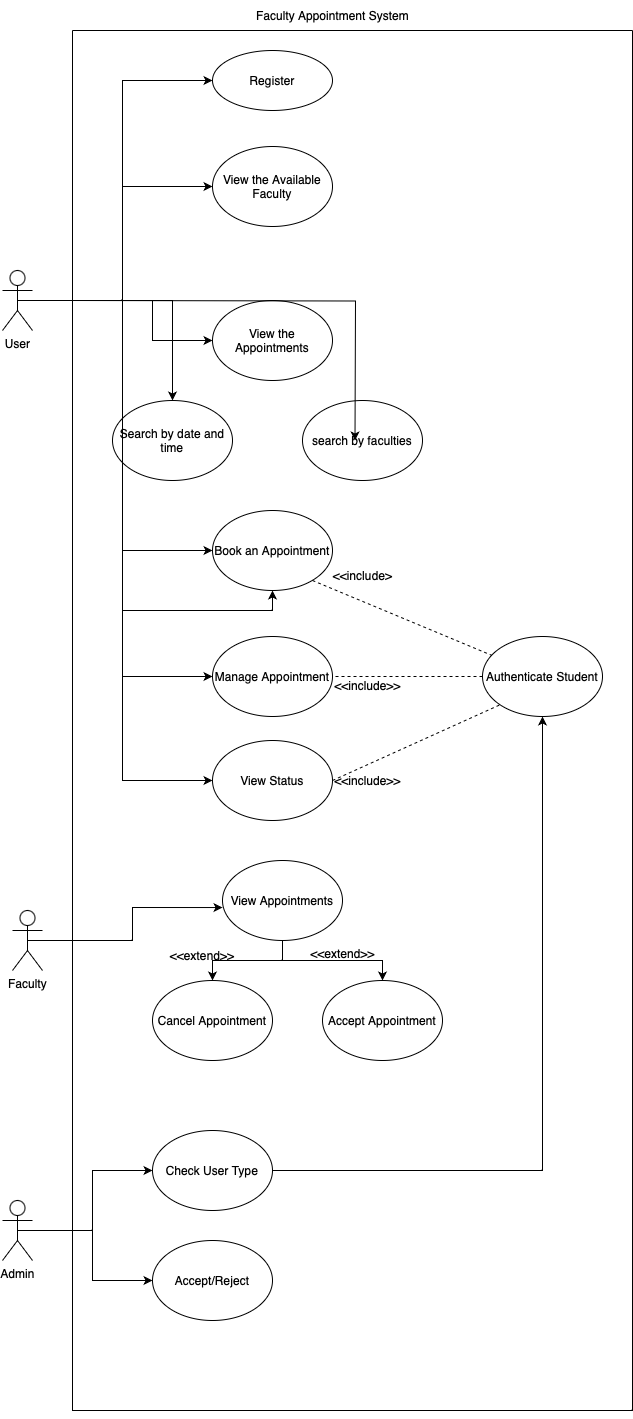
* **Data Privacy and Compliance:**

The system should adhere to data privacy regulations and protect user

information from unauthorized access or disclosure.

* **Methodology/Process**
* A new User have to Login or Existing User Have to register.
* After Sign in User Can see the Various Faculty With their Specialization and their Appointment Timing.
* If user is a Faculty Then he/she can Apply for Faculty From the Dashboard Menu APPLY FOR FACULTY
* After Applying then Admin Got the Notification , and he accept or reject the request After Verification.
* If he accepted as a faculty from admin side, then the faculty can update his profile anytime and Students can Apply for appointment to this faculty.
* Faculty can also see what appointment he/she receive and according he/she can choose to accept or reject as per availability.
* User request for Appointment to faculties as per the timings.
* **Design (class, sequence, package, activity etc.)**

**Use Case Diagram**



* **Coding (APIs, Framework)**

const express = require("express");

const colors = require("colors");

const morgan = require("morgan");

const dotenv = require("dotenv");

const connectDB = require("./config/db");

const path = require("path");

//dotenv config

dotenv.config();

//mongoDB connection

connectDB();

//rest object

const app = express();

//middlewares

app.use(express.json());

app.use(morgan("dev"));

//routes

app.use("/api/v1/user", require("./routes/userRoutes"));

app.use("/api/v1/admin", require("./routes/adminRoutes"));

app.use("/api/v1/faculty", require("./routes/facultyRoutes"));

//\*\*\*\*\*\*\*\*\*\*\*DEPLOYMENT \*\*\*\*\*\*\*\*\*\*\*

//Static Files

app.use(express.static(path.join(\_\_dirname, "./client/build")));

//creating Route

app.get("\*", function (req, res) {

res.sendFile(path.join(\_\_dirname, "./client/build/index.html"));

});

//port

const port = process.env.PORT || 8080;

//listen port

app.listen(port, () => {

console.log(

`Server running in ${process.env.NODE\_MODE} Mode on Port ${process.env.PORT}`

.bgCyan.white

);

});

* **Testing**

Testing is a crucial part of the software development process to ensure that the Faculty Schedule Appointment System based on the MERN (MongoDB, Express.js, React.js, Node.js) stack functions correctly and meets the desired requirements. We will discuss different types of testing and how to approach testing for the system.

* Unit Testing:
* Unit testing focuses on testing individual components and functions in isolation.
* Write tests for API routes, database operations, and frontend components.
* Integration Testing:
* Integration testing verifies the interaction between different

components or modules.

* Test the interaction between the frontend and backend, ensuring

they communicate correctly.

* Mock the backend responses for frontend tests.
* End-to-End (E2E) Testing:
* E2E testing validates the system as a whole, simulating user interactions

and testing the system's behavior in a real-world scenario.

* Use tools like Cypress or Puppeteer for E2E testing.
* Write tests that cover different user scenarios, such as

scheduling appointments, canceling appointments, and viewing

appointment history.

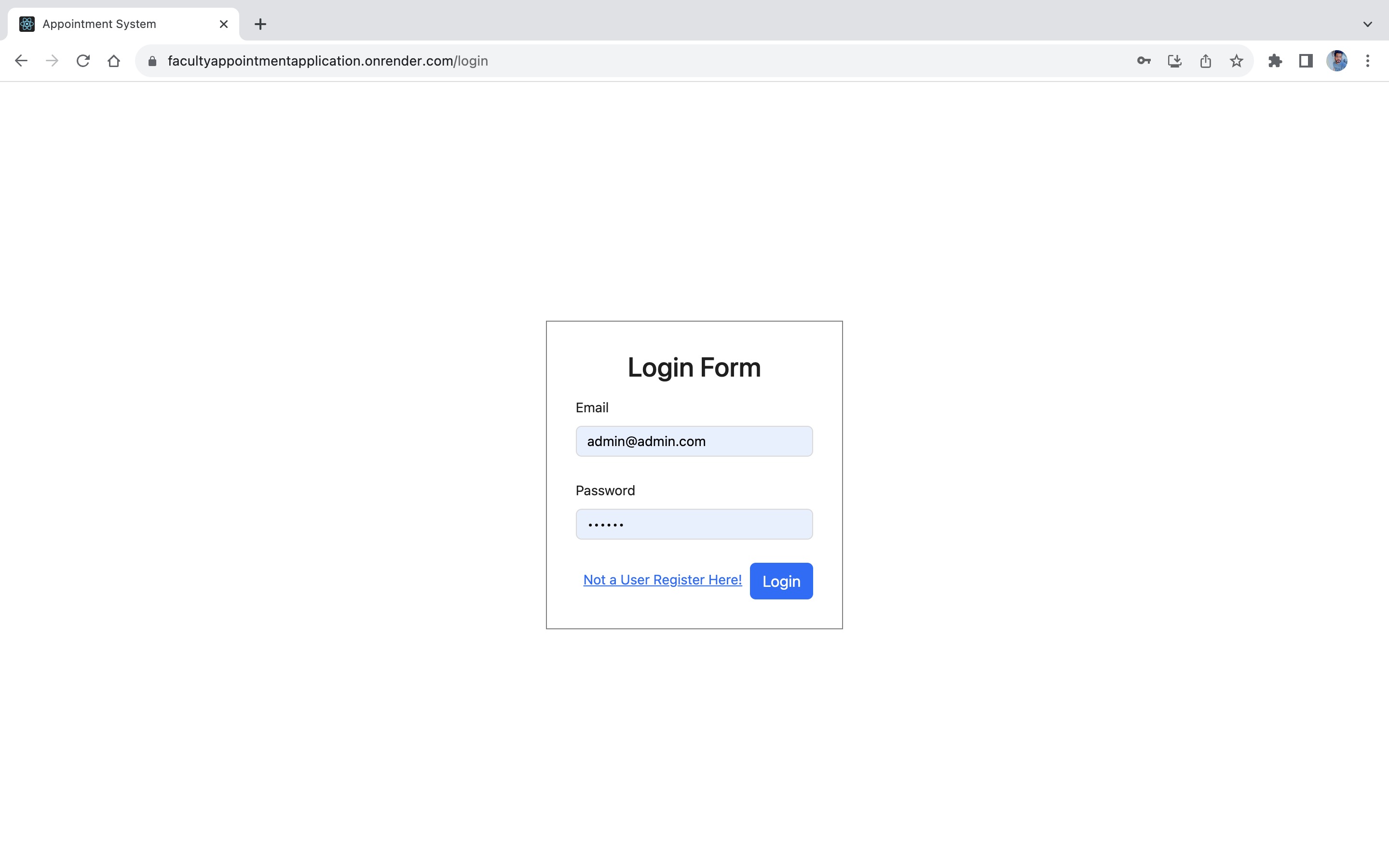
* API Testing:
* Test the API endpoints using tools like Postman or Newman to ensure they return the expected responses for various input scenarios.
* Test edge cases, validation, and error handling for API routes.
* Load Testing:
* Conduct load testing using tools like Artillery or JMeter to evaluate system performance under various load conditions.
* Verify that the system can handle multiple concurrent users and appointments without performance degradation.
* Security Testing:
* Perform security testing to identify vulnerabilities in the system.
* Ensure that user authentication and authorization are correctly implemented to prevent unauthorized access.
* Check for possible injection attacks, XSS, CSRF, etc.
* User Acceptance Testing (UAT):
* Involve actual users (faculty and students) to perform UAT on the system.
* Collect feedback and address any usability issues or improvements.
* Error and Exception Handling Testing:
* Test the system's error and exception handling mechanism to ensure proper error messages are displayed and logged.
* Verify that the system gracefully handles unexpected scenarios.
* Performance Testing:
* Measure the system's performance under different workloads and ensure it meets performance requirements.
* **Snapshots (GitHub Link, Live link, Demo etc.)**

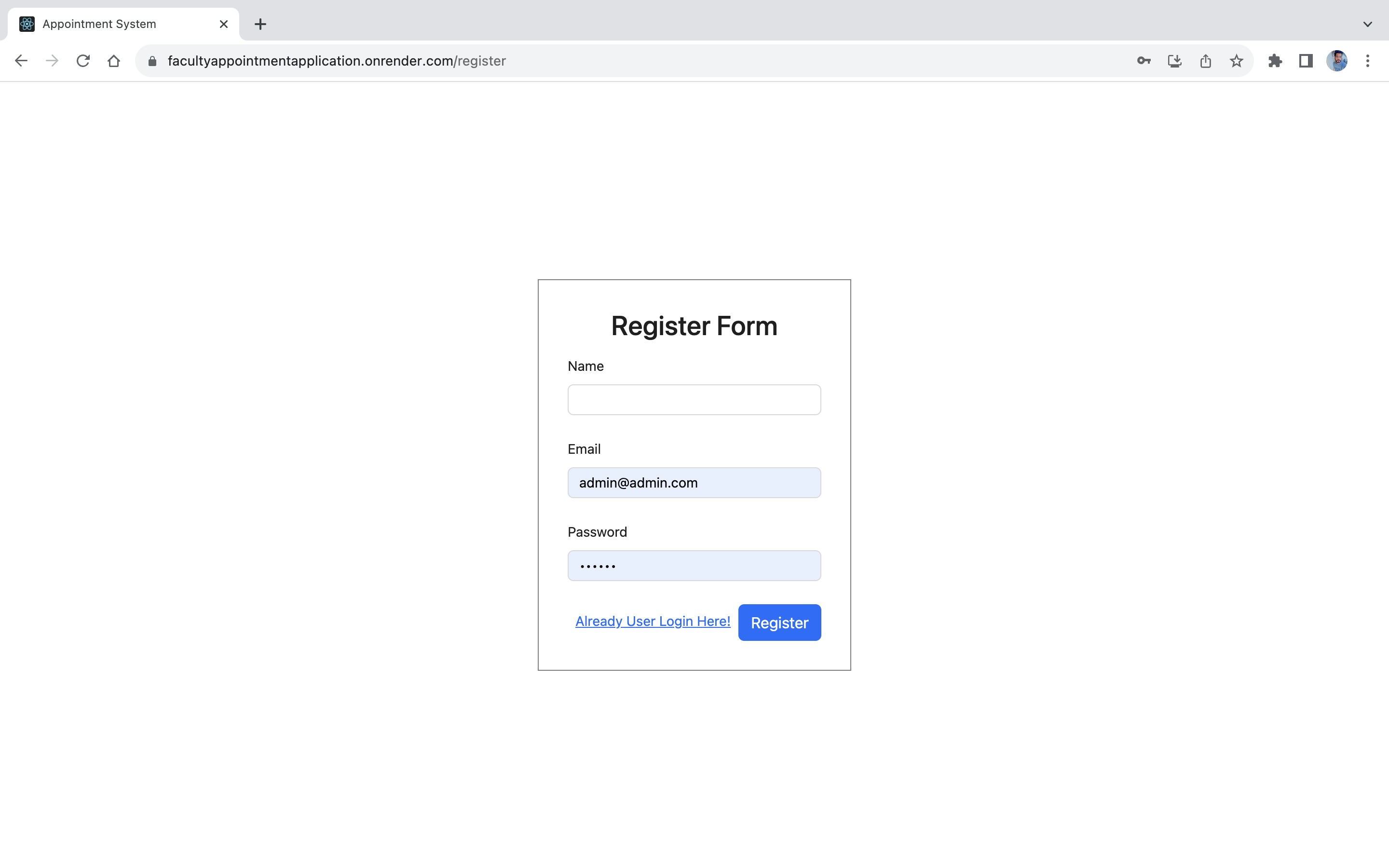
We Deploy our Project on Render.

**Link:** <https://facultyappointmentapplication.onrender.com>

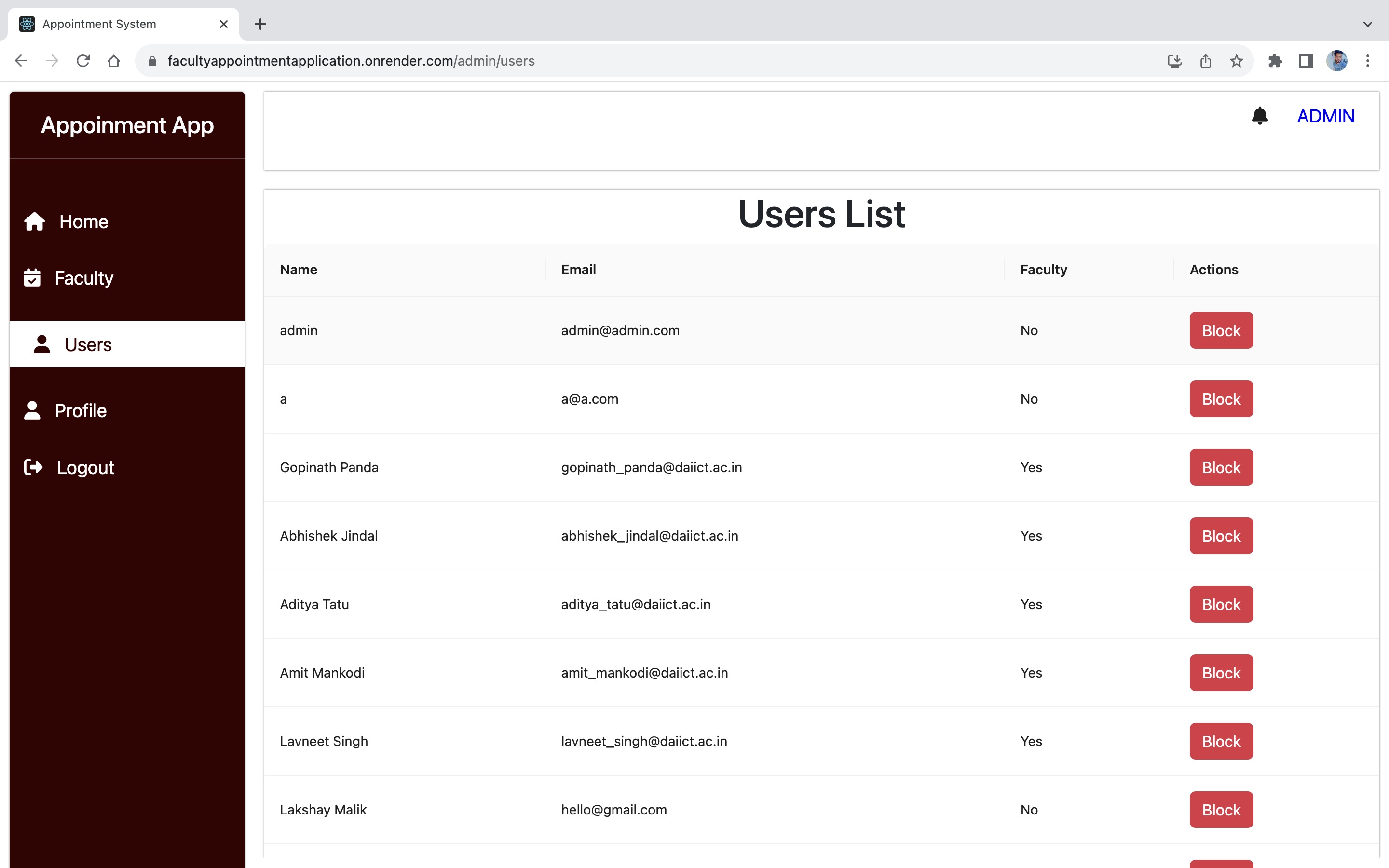
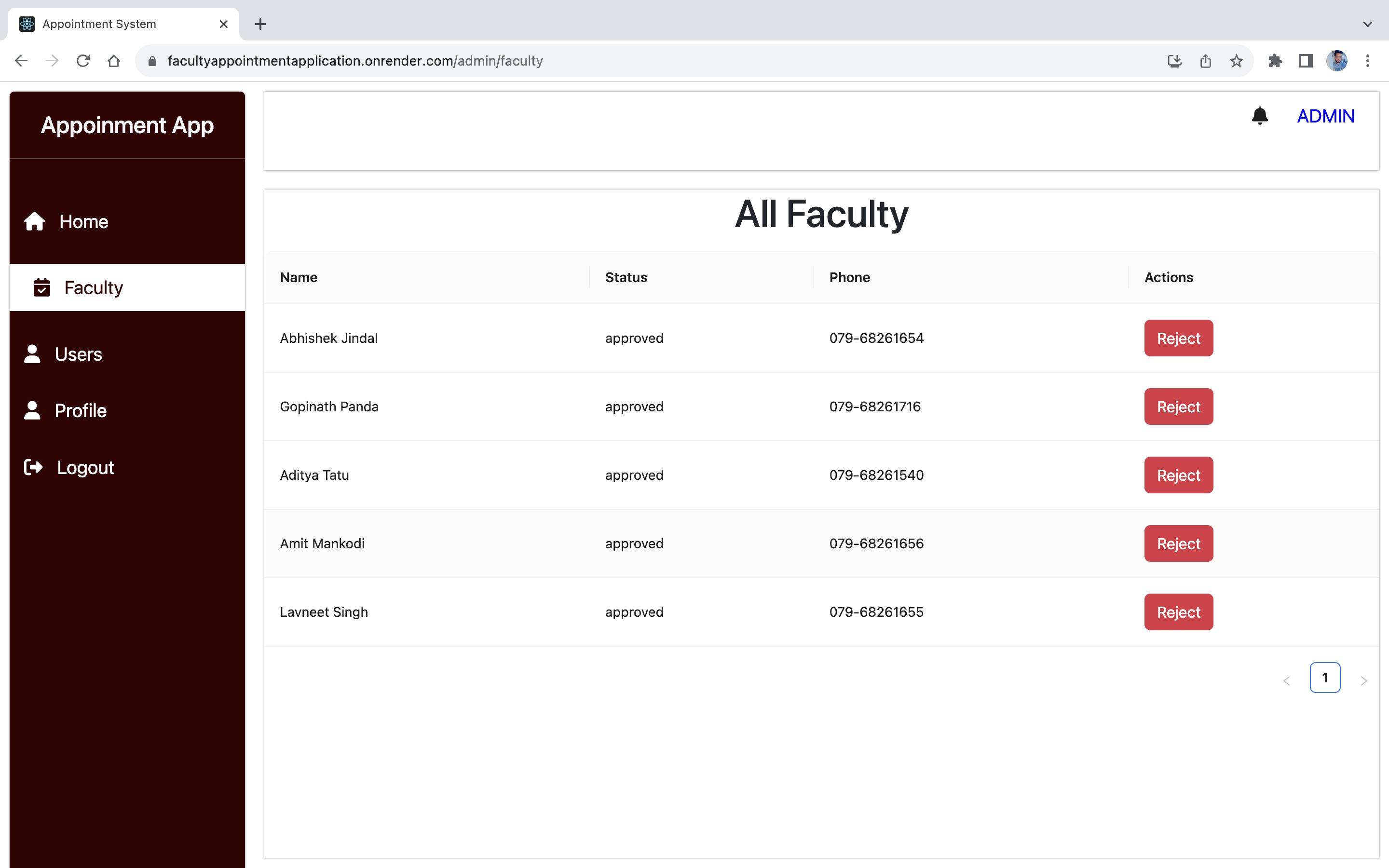
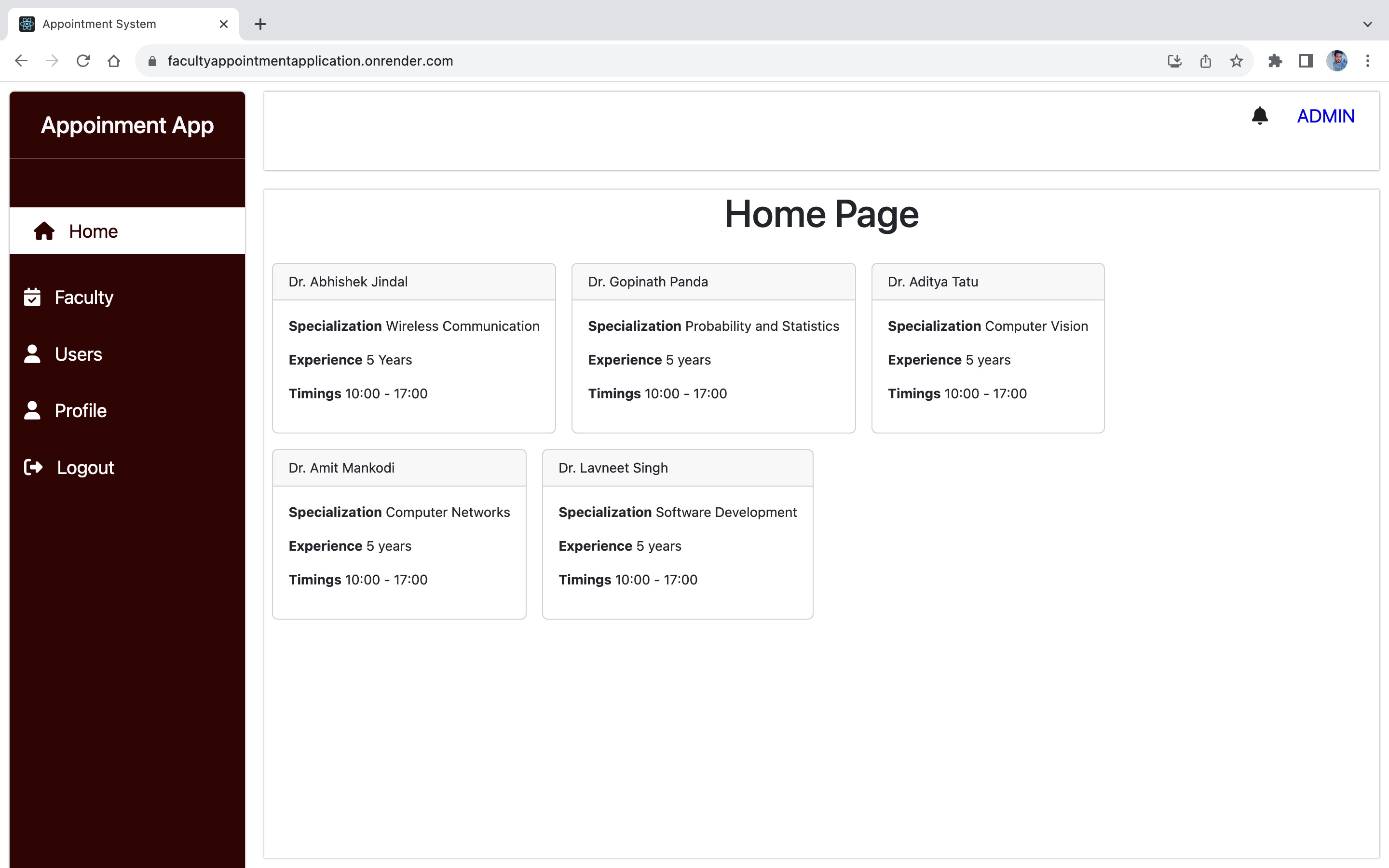
**SnapShots:**

**Login/Signup view:**

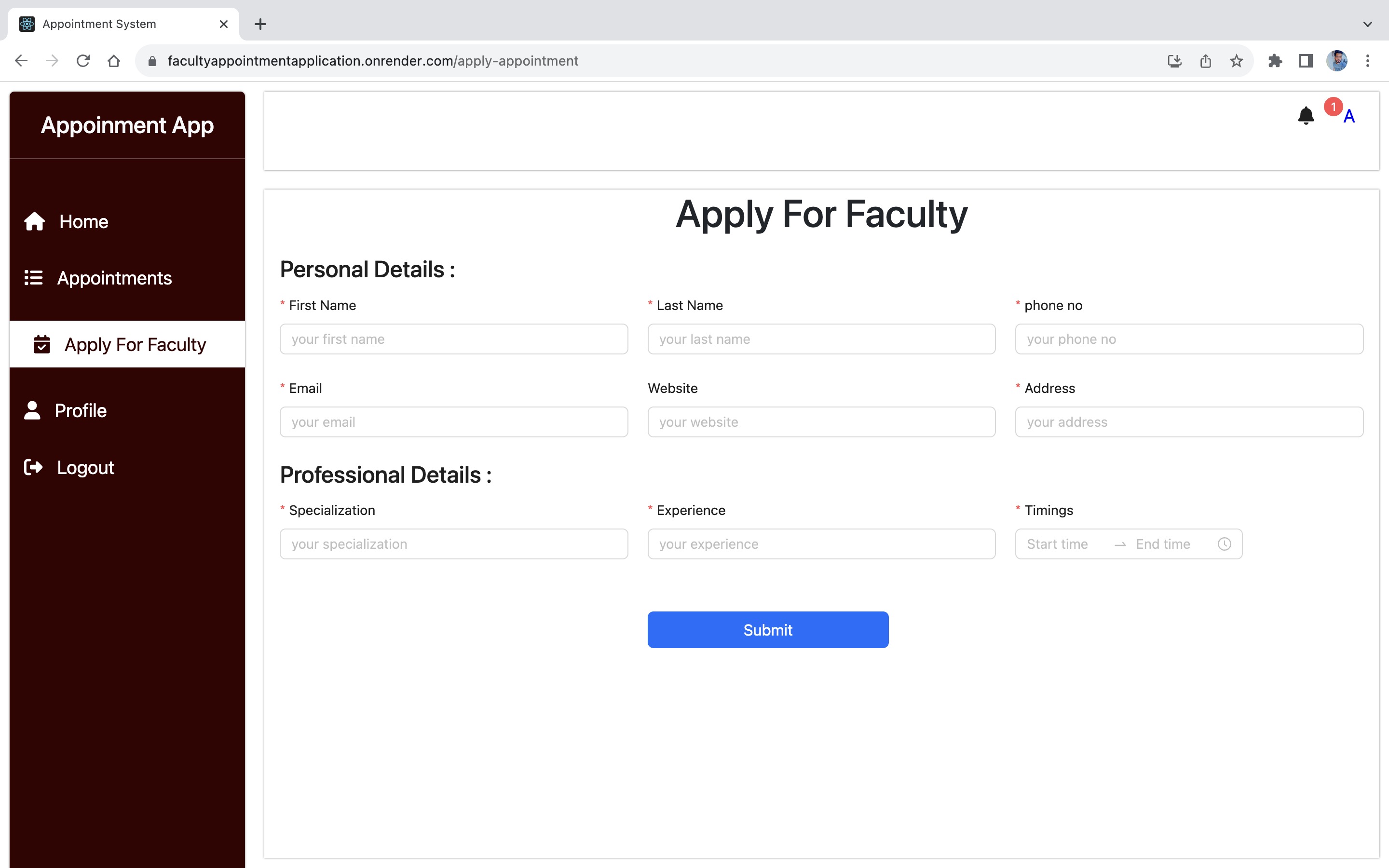
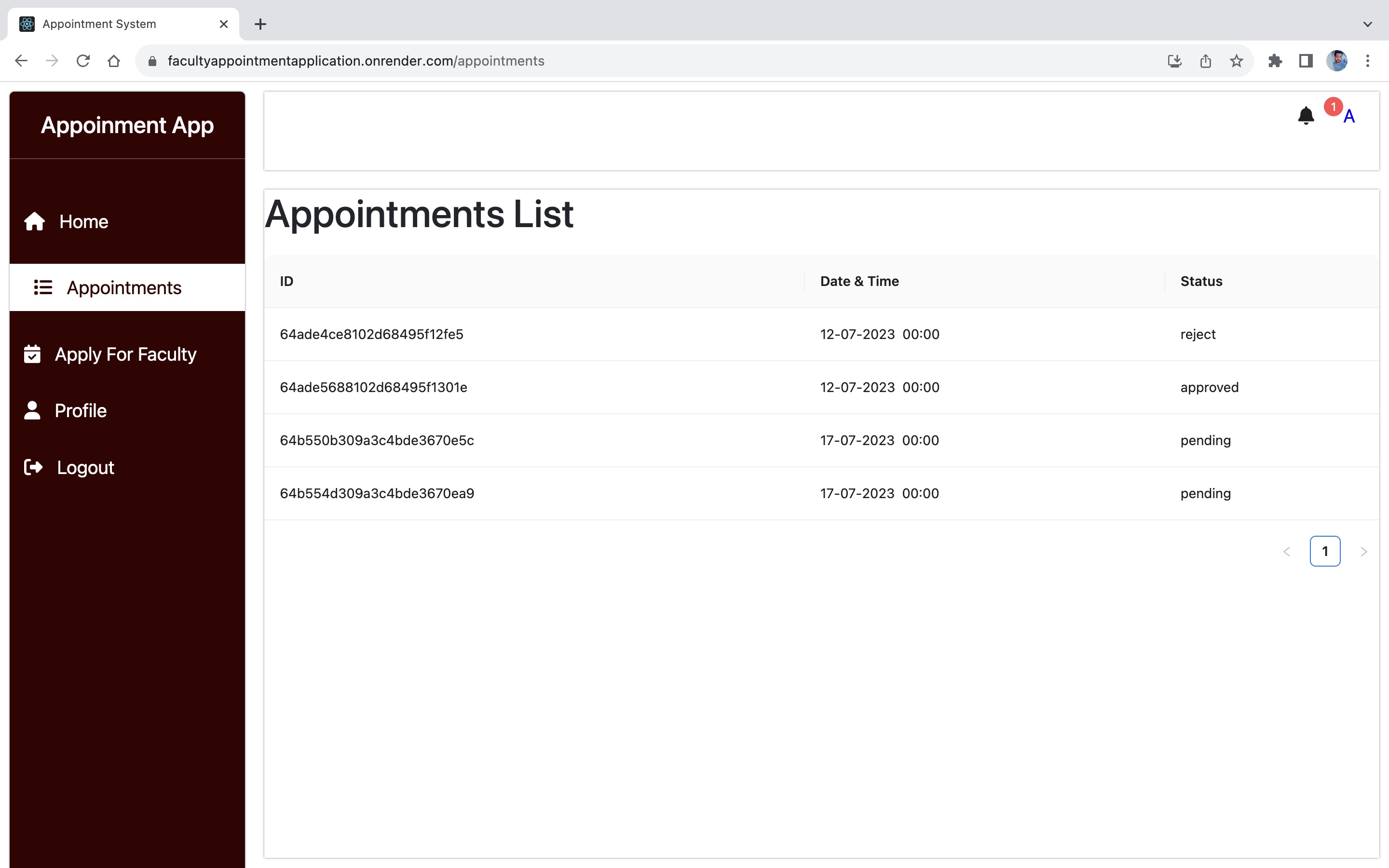
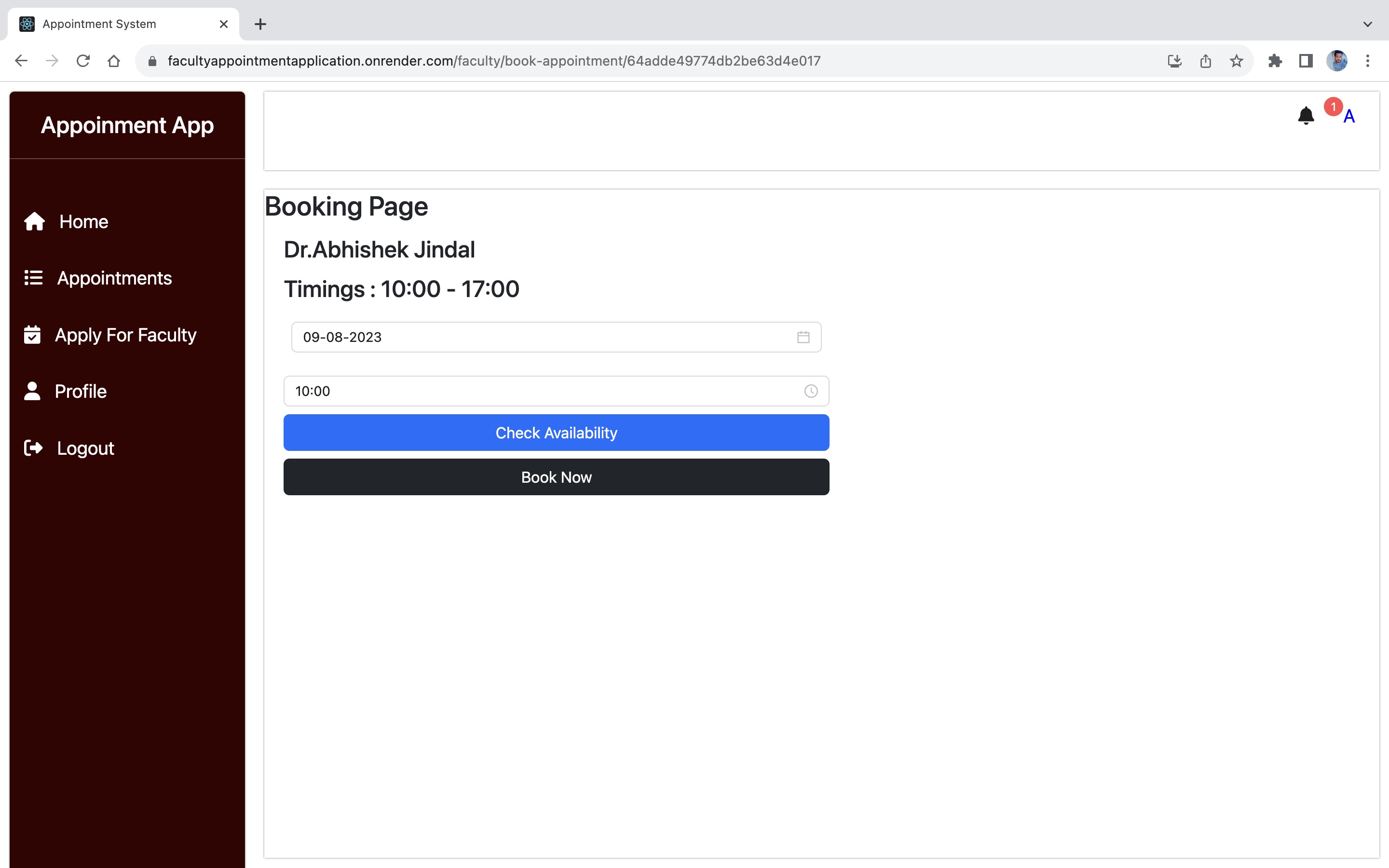
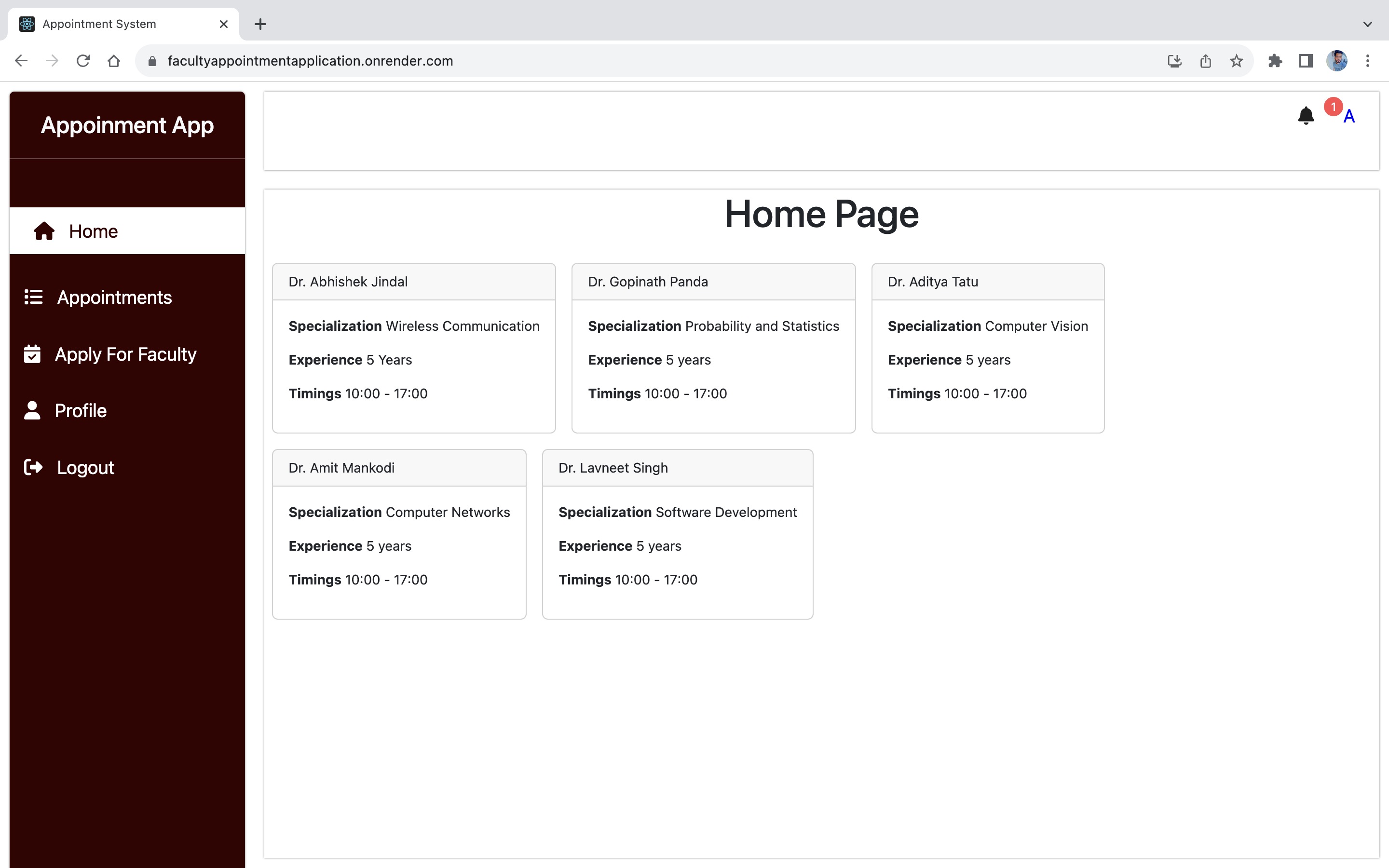




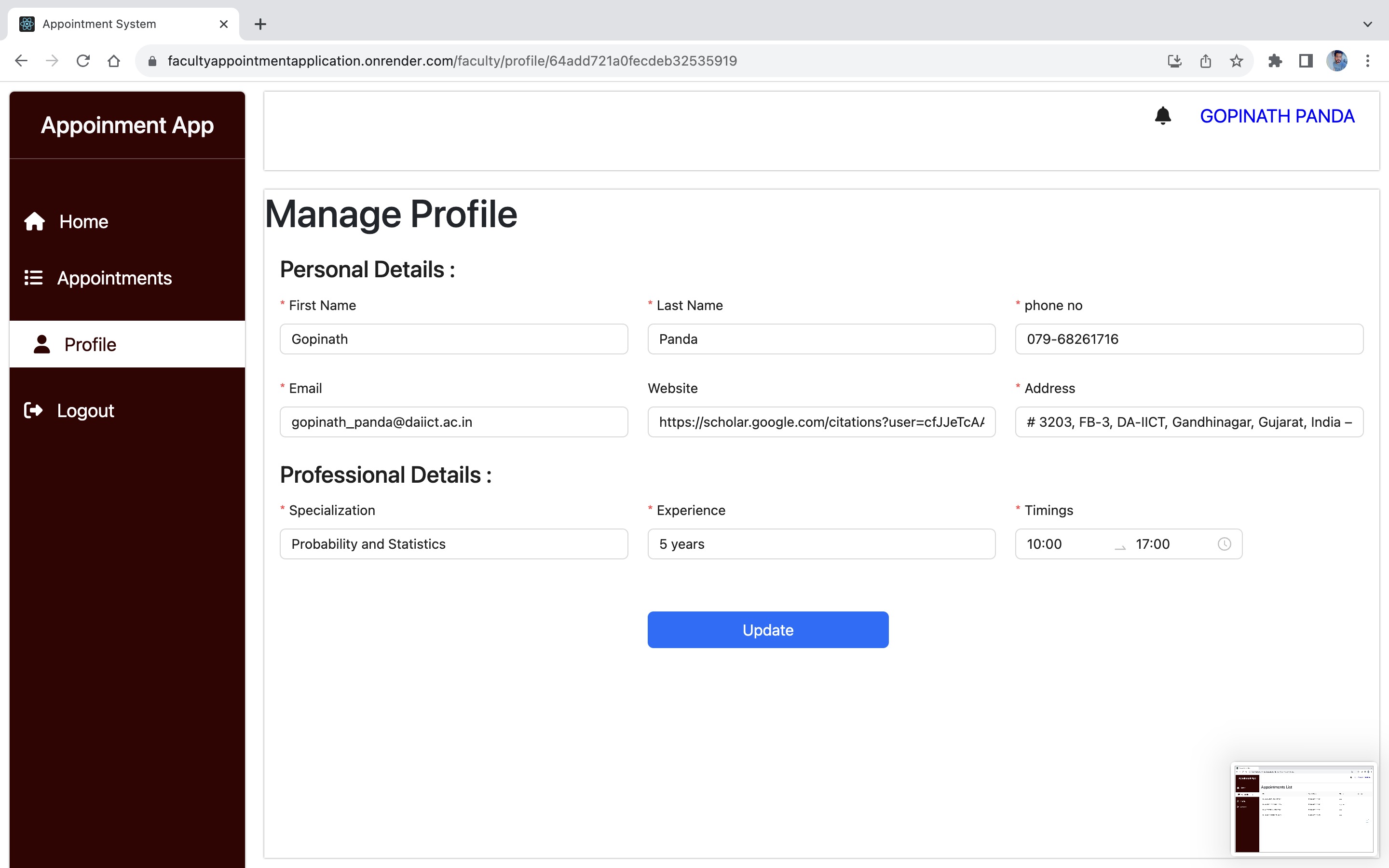
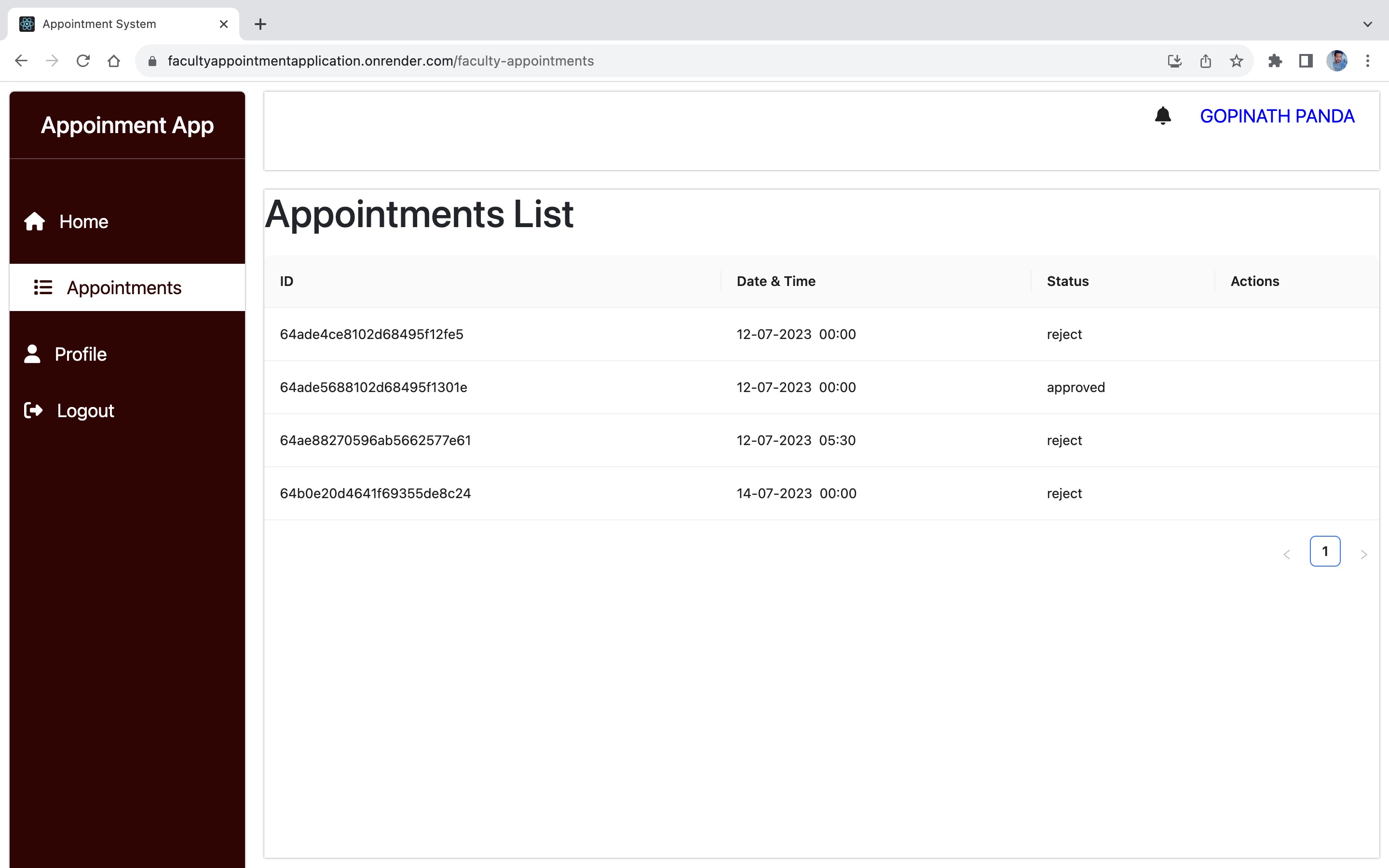
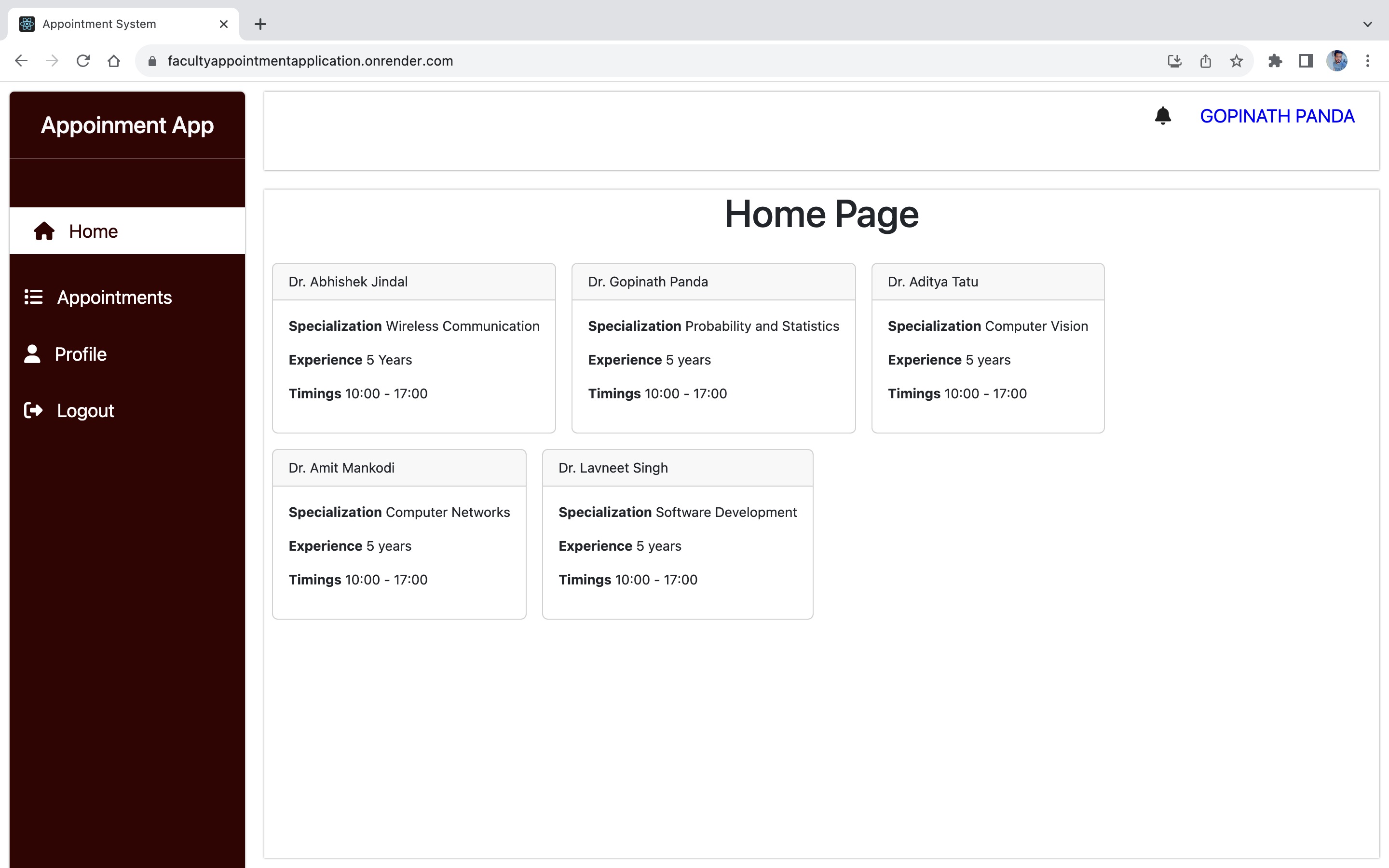
**Admin View:**



**Student View:**



**Faculty View:**



* **Summary**

The Faculty Appointment System based on the MERN stack provides an efficient and user-friendly solution for scheduling appointments between students and faculty members. By leveraging modern web technologies and conducting thorough testing, the system aims to enhance the appointment management process, improving communication and productivity within an academic or professional institution.