**ONLINE APARTMENT RESERVATION BOOK World Spring MVC Project Documentation**

**Names: SANGWA Angele**

**Id: 23736**

**Deployment of Miss World project**

**1. Functional Requirements:**

**1.1 Signup:**

The system should allow users to create an account by providing necessary information, including username, password, email, and other relevant details.

Usernames should be unique, and the system should validate the uniqueness during the signup process.

Passwords should meet certain complexity requirements, such as minimum length, and should be securely stored using encryption techniques.

Upon successful signup, the system should send a confirmation email to the user for account verification.

**1.2 Login:**

Users should be able to log in to their accounts using their credentials (username and password).

The system should validate the user's credentials and grant access only to authorized users.

Invalid login attempts should be tracked and limited to prevent brute-force attacks.

**1.3 CRUD (Create, Read, Update, Delete):**

The system should provide functionality for creating, reading, updating, and deleting various entities related to the apartment reservation book, such as room.

Users with appropriate privileges should be able to perform these actions on the respective entities.

Appropriate validation should be applied to ensure the integrity and consistency of the data.

**1.4 Pagination:**

In scenarios where large amounts of data are involved, such as displaying a list of room the system should implement pagination.

The system should allow users to navigate through pages and limit the number of records displayed per page.

**1.5 Sorting:**

The system should provide sorting options for different attributes of entities, such as sorting room by their name.

Users should be able to specify the sorting order, such as ascending or descending.

**1.6 Elimination:**

The elimination process may occur in multiple rounds, and the system should update the status of eliminated room accordingly.

**1.8 Email:**

The system should send emails to users for various purposes, such as account confirmation, password reset.

The email functionality should be integrated with an email service provider or SMTP server for reliable delivery.

**2. Non-Functional Requirements:**

**2.1 Security:**

The system should enforce secure authentication and session management to protect user accounts from unauthorized access.

Passwords should be stored securely using industry-standard hashing algorithms.

User input should be validated and sanitized to prevent common security vulnerabilities, such as SQL injection or cross-site scripting (XSS) attacks.

**2.2 Performance:**

The system should be able to handle a large number of concurrent users and maintain optimal performance.

Database queries should be optimized to ensure efficient retrieval and manipulation of data.

Caching mechanisms should be implemented, where applicable, to reduce the load on the server.

**2.3 Usability:**

The user interface should be intuitive, user-friendly, and responsive.

Proper feedback messages and error handling should be provided to guide users and assist them in completing tasks successfully.

The system should follow established design standards and conventions to enhance usability.

**2.4 Scalability:**

The system should be designed to accommodate future growth and increased user load.

It should be possible to add additional servers or resources to handle increased demand without significant downtime or performance degradation.

**2.5 Reliability:**

The system should be highly reliable and available, minimizing the chances of system failures or downtime.

Proper error handling and logging mechanisms should be in place to assist

with debugging and troubleshooting.

**2.6 Compatibility:**

The system should be compatible with different web browsers, ensuring a consistent user experience across platforms.

Compatibility with different devices, such as desktops, laptops, tablets, and smartphones, should also be considered.

**2.7 Maintainability:**

The system should be modular and well-organized, allowing for easy maintenance and future enhancements.

Proper documentation and code comments should be provided to facilitate understanding and modification of the system by other developers.

**2.8 Integration:**

The system should integrate with external services, such as email providers or SMTP servers, for sending emails.

Integration with a database management system (DBMS) should be established to store and retrieve data efficiently.

**Project Scope**

Develop a Miss World Spring MVC application with signup, login, CRUD, pagination, sorting, voting, elimination, and email functionalities.

Create a user-friendly interface for contestants, judges, and administrators to manage the Miss World pageant.

**Timeline:**

Phase 1: Requirements Gathering and Design (3 days)

Phase 2: Implementation of Core Functionality (1 week)

Phase 3: Testing and Bug Fixes (1 day)

**Resources:**

Project Manager: Responsible for overall project coordination, task allocation, and monitoring progress.

Developer: Skilled in Java, Spring MVC, HTML, CSS, and JavaScript for system implementation.

Database Administrator: Responsible for database design, optimization, and maintenance.

UI/UX Designer: Create an intuitive and visually appealing user interface.

Quality Assurance (QA) Team: Conduct testing, identify bugs, and ensure system stability.

Infrastructure Team: Provide servers, network setup, and system deployment support.

**Deliverables**:

System requirements document

Wireframes and UI design mockups

Fully functional Apartment World Spring MVC application

Test cases and test reports

User documentation and training materials

**Milestones:**

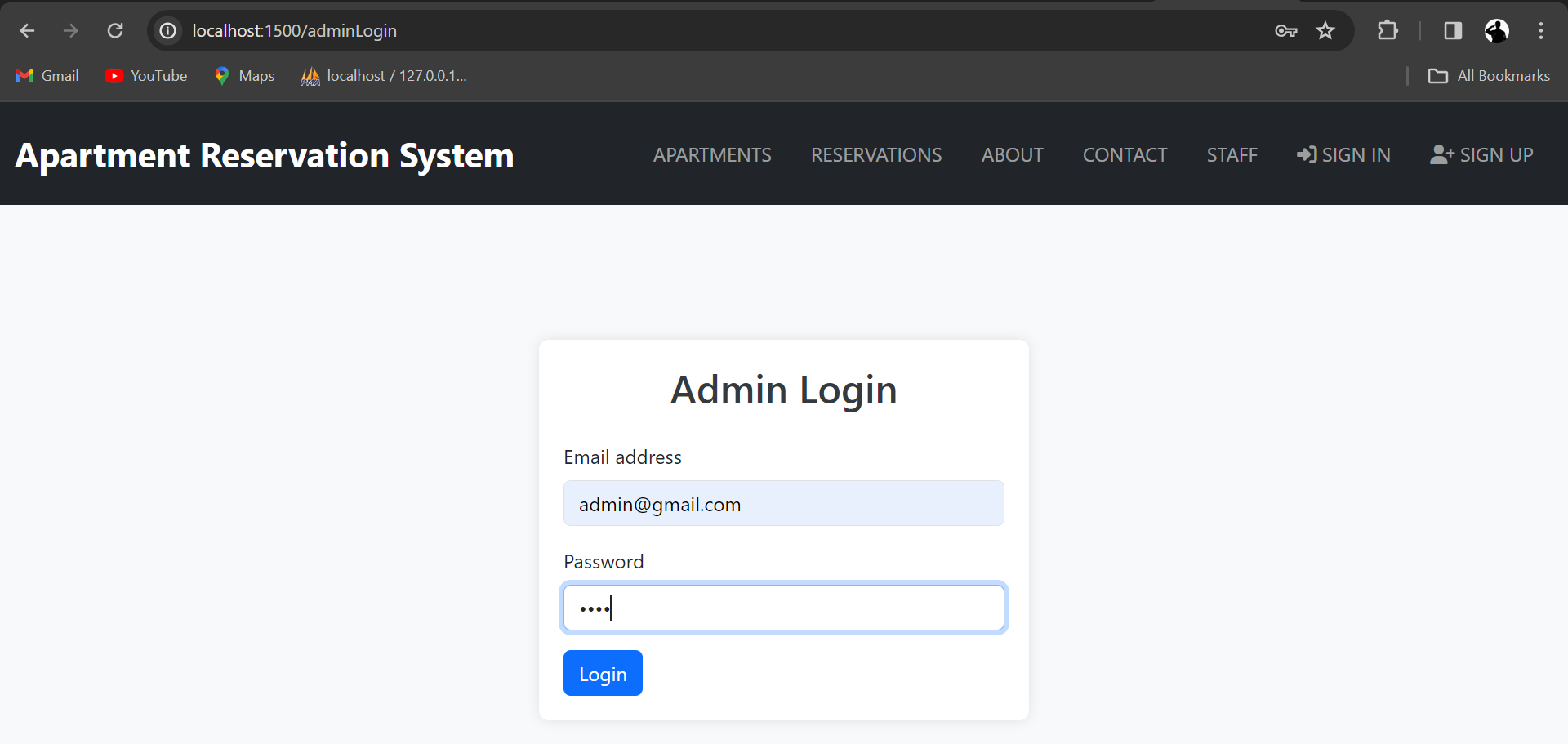
Milestone 1: Requirements Gathering and Design Approval

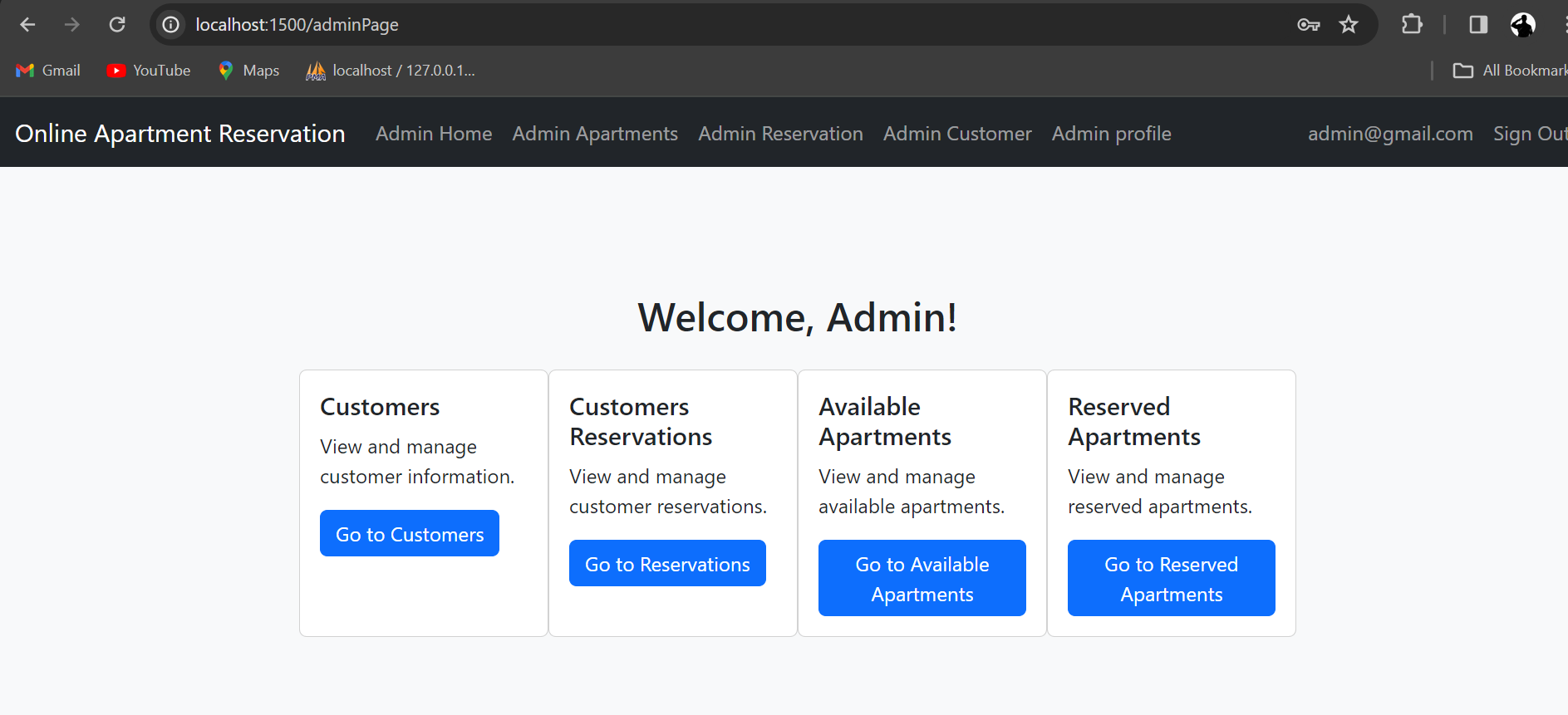
Milestone 2: Core Functionality Implementation Completion

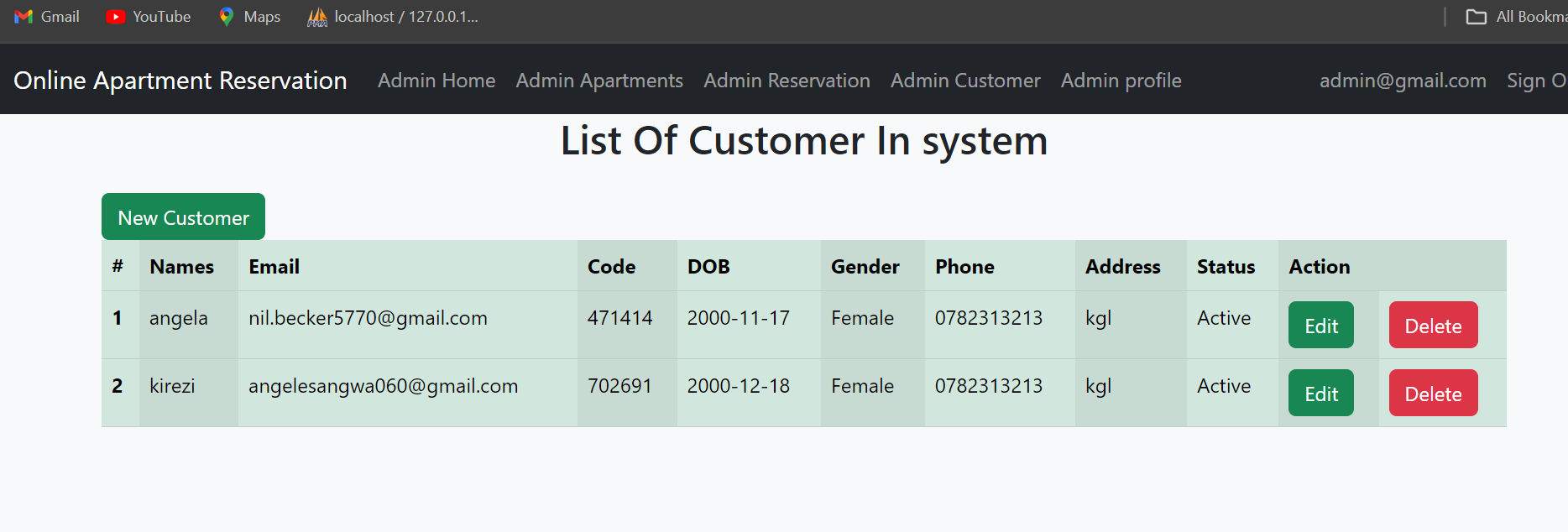
Milestone 3: Testing and Bug Fixes Completed

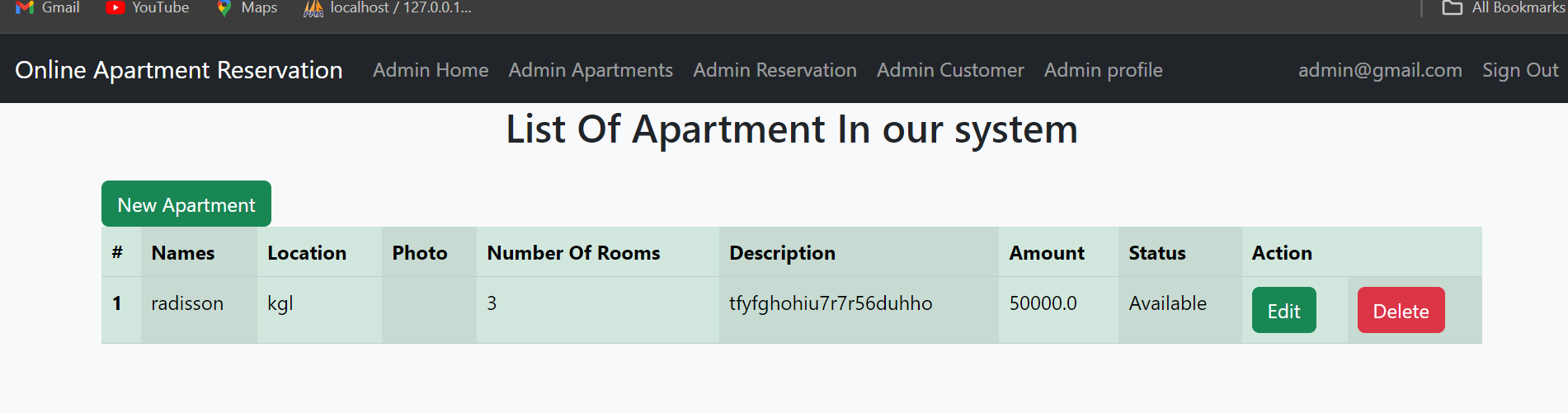
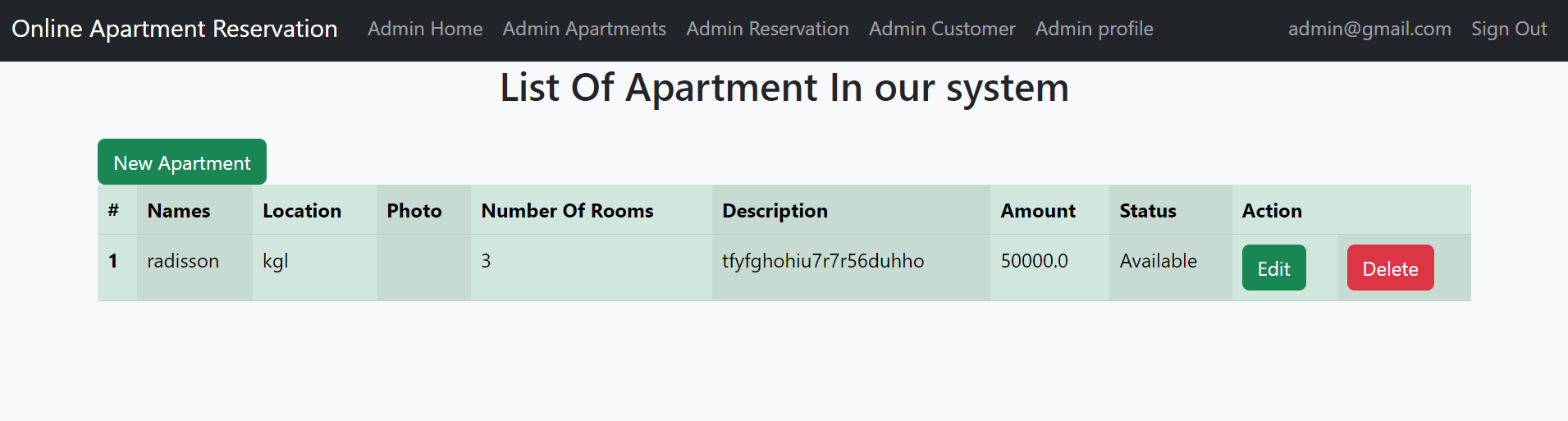
Milestone 4: Successful Deployment and User Acceptance Testing

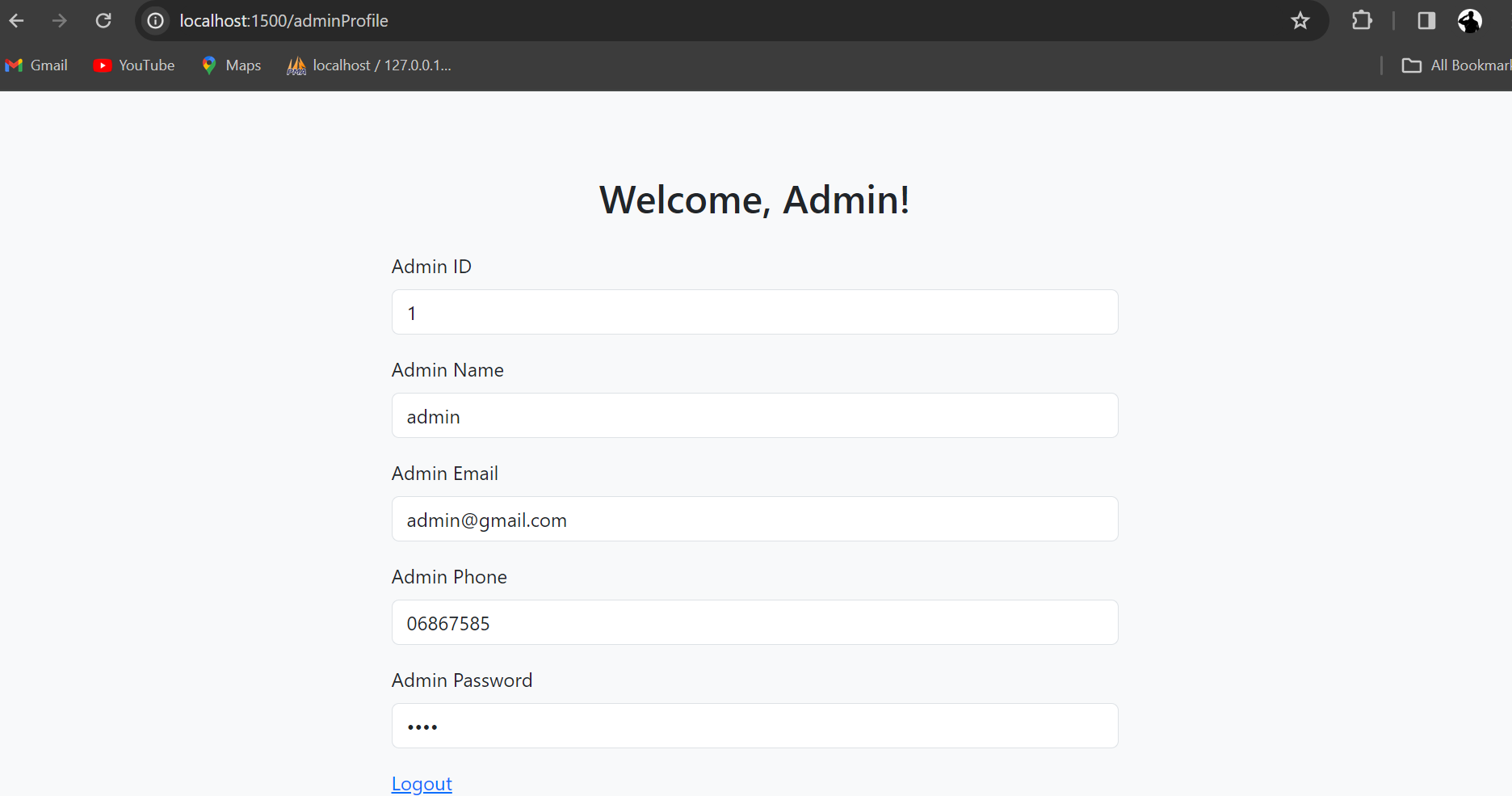
**DATABASE STRUCTURE**

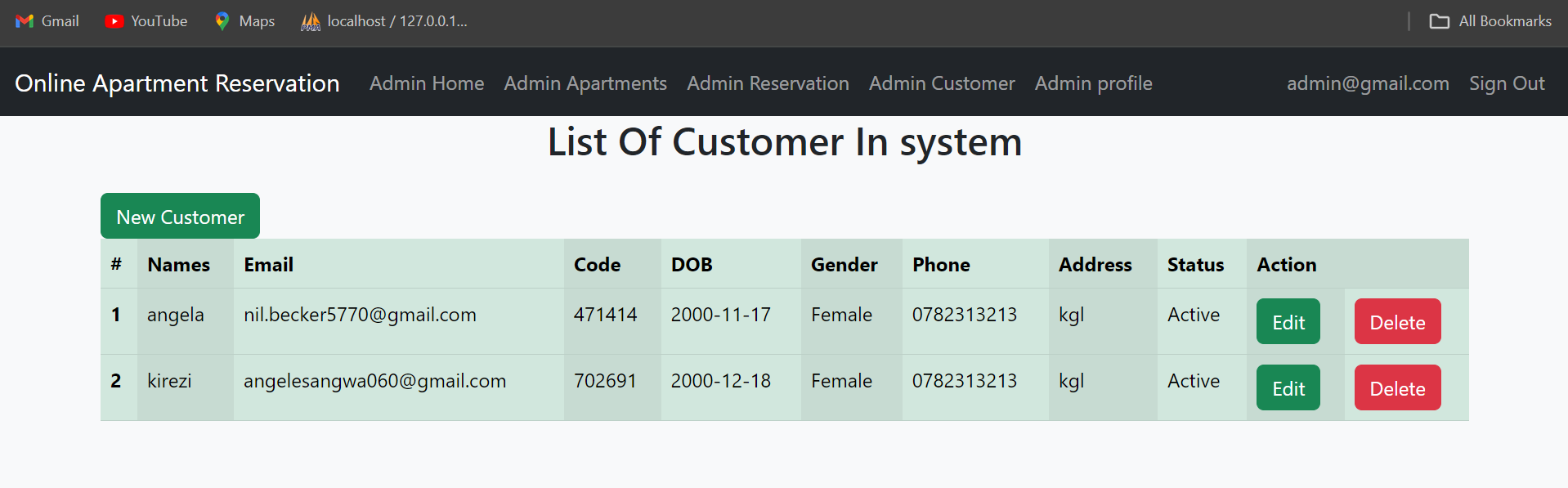


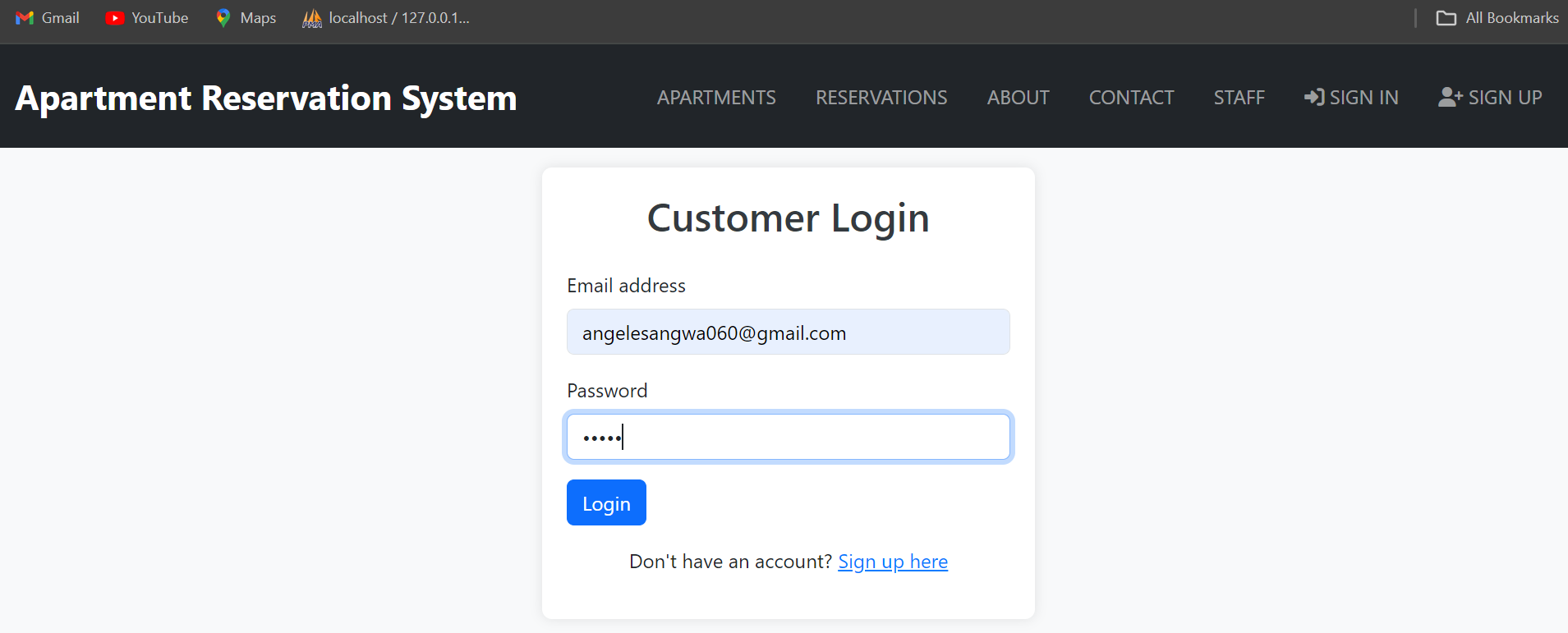
**Admin Dashboard**

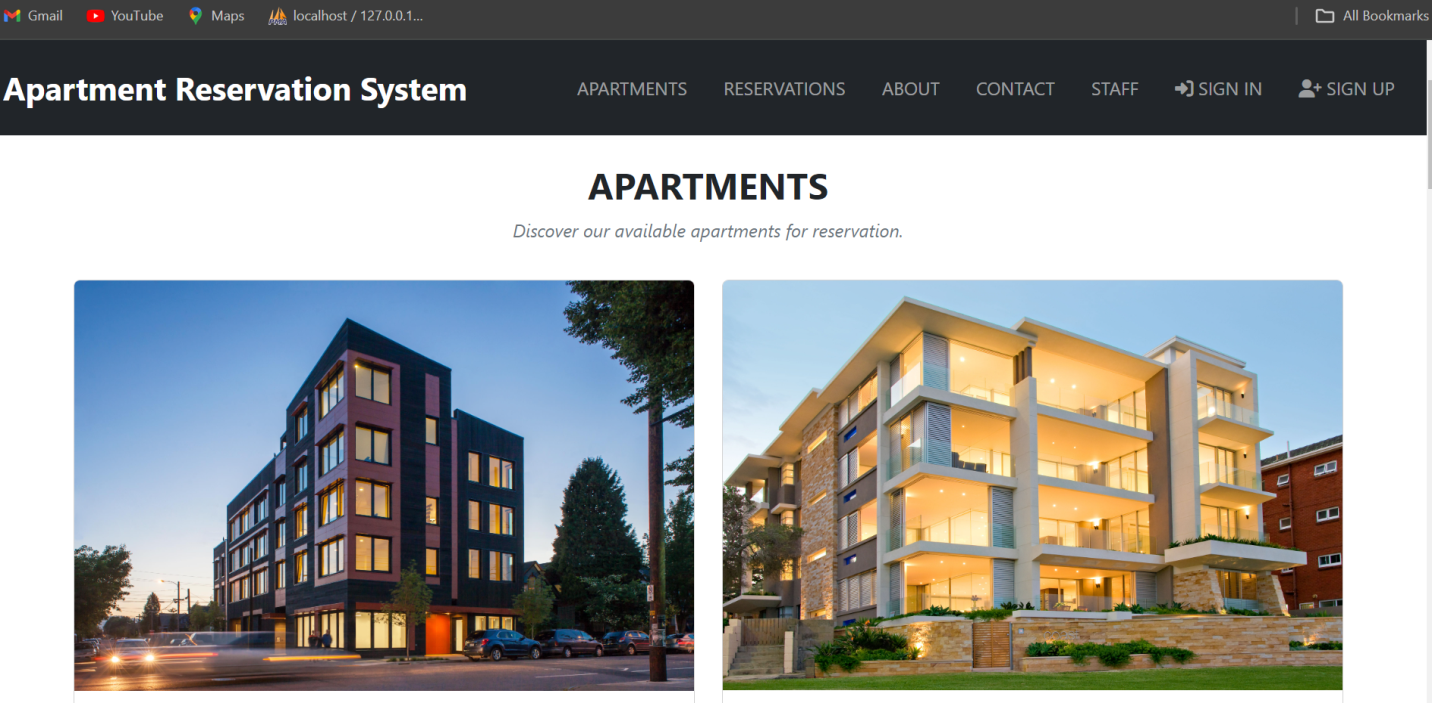
**Delete User**

**HOME**



****

****

****

