Najm Masri, Amer Abu Amriah, Sanjana Nagwekar, Muhammad Rizki Miftha Alhamid

CS 157A - Introduction to Database Management Systems

Professor Ramin Moazeni

May 10, 2024

Tables of Contents

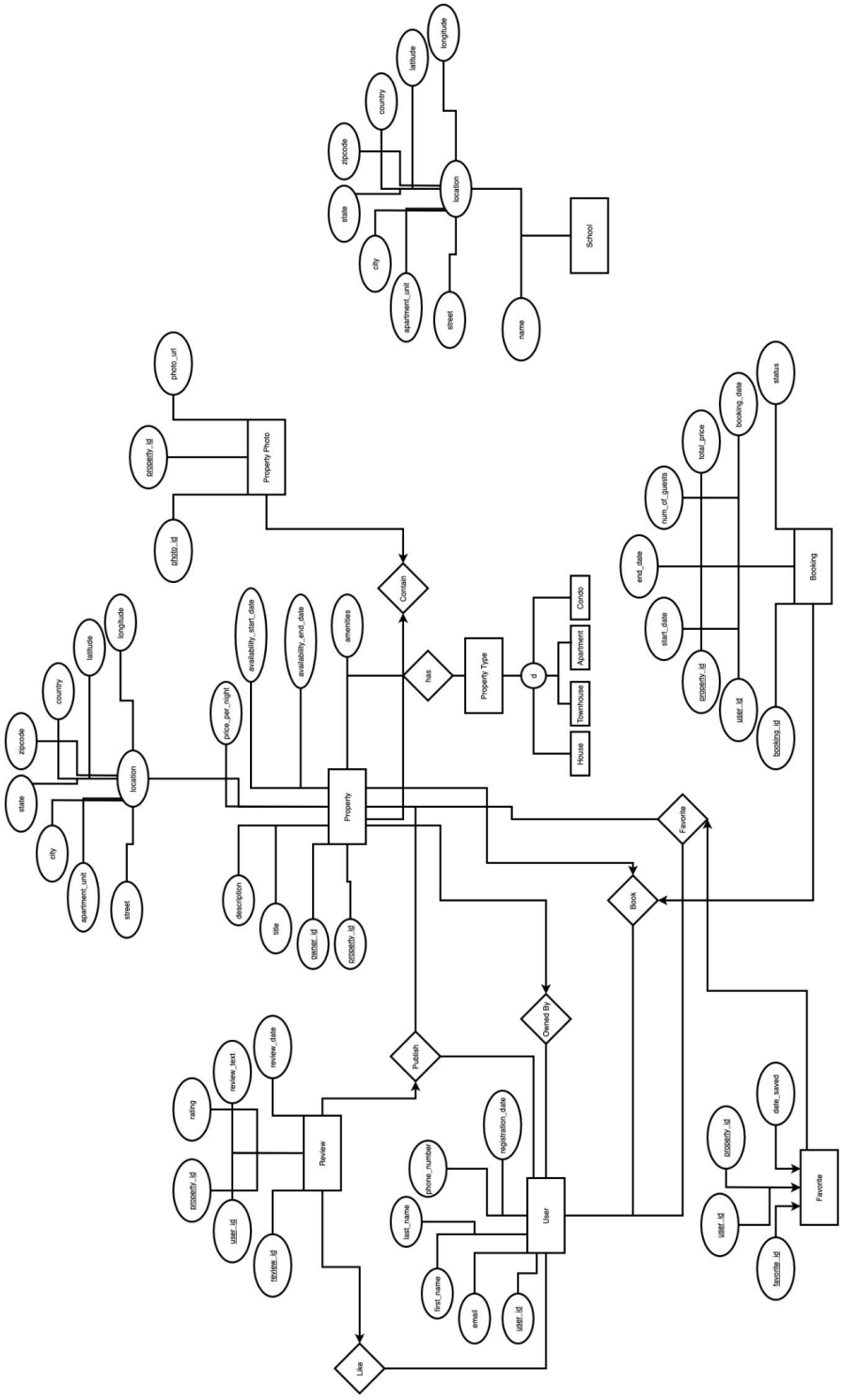
Goals and Description of DormDash

DormDash's goal is to simplify the search for student housing by addressing the challenges students face when seeking affordable housing near their academic institutions. Our application optimizes the search and booking process through a student-focused platform that offers tailored listings such as apartments and vacation homes close to universities. Aimed at enhancing user convenience and reducing the complexity of securing student accommodations, DormDash not only simplifies finding a place to stay but also improves the overall experience by focusing on the specific needs and preferences of students.

Application/Functional Requirements and Architecture

The application features a robust backend built with Node.js and Express, secured with bcrypt and JSON Web Tokens (JWT) for authentication purposes. Our frontend utilizes React.js, ensuring a responsive and interactive user interface, while CSS and JavaScript enhance the visual and functional aspects. The architecture also includes a MySQL database to manage user data, listings, and transactions efficiently. For development and testing, tools like Postman and GitHub are employed to maintain high standards of code integrity and project collaboration. This comprehensive setup ensures that DormDash not only meets but exceeds the functional requirements necessary for providing an exceptional service to its users. The platform supports essential functionalities such as logging in/out, detailed searching and filtering by school name and availability, adding the listings to the users favorites page, and booking the listings.

Conceptual Diagram



https://drive.google.com/file/d/1VZr4PdoCcqG8vqQUUSksLNBf8oiCJDnc/view?usp=sharing

Relational Schema

Entities:

1. **User** (user_id, email, password, first_name, last_name, phone_number, registration_date)
2. **Property** (property_id, owner_id, title, description, location, price_per_night, amenities, availability_start_date, availability_end_date)
    - Foreign Key(owner_id) references User
3. **Property Type**(type_id, name)
4. **House**(type_id)
    - Foreign Key(type_id) references Property Type
5. **Apartment**(type_id)
    - Foreign Key(type_id) references Property Type
6. **Townhouse**(type_id)
    - Foreign Key(type_id) references Property Type
7. **Condo**(type_id)
    - Foreign Key(type_id) references Property Type
8. **PropertyPhoto** (photo_id, property_id, photo_url)
    - Foreign Key(property_id) references Property
9. **Favorite** (favorite_id, user_id, property_id, date_saved)
    - Foreign Key(user_id) references User
    - Foreign Key(property_id) references Property
10. **Booking** (booking_id, user_id, property_id, start_date, end_date, num_guests, total_price, booking_date, status)

- Foreign Key(user_id) references User
- Foreign Key(property_id) references Property

11. **Review** (review_id, user_id, property_id, rating, review_text, review_date)
    - Foreign Key(user_id) references User
    - Foreign Key(property_id) references Property
12. **School**(school_id, name, location)

Relationships:

1. **User Owns Property**(user_id, property_id)
    - Users to Properties (One-to-many) users can own multiple properties.
    - Foreign Key(user_id) references User
    - Foreign Key(property_id) references Property
2. **User Makes Booking**(user_id, booking_id)
    - Users to Bookings (One-to-many) users can make multiple bookings.
    - Foreign Key(user_id) references User
    - Foreign Key(booking_id) references Booking
3. **Property Contains Photo**(property_id, photo_id)
    - Properties to Photos (One-to-many) each property can have multiple photos
    - Foreign Key(property_id) references Property
    - Foreign Key(photo_id) references Property Photo
4. **User Marks Favorite**(user_id, property_id)
    - Users to Favorites (Many-to-many) users can save multiple properties as favorites
    - Foreign Key(user_id) references User

- Foreign Key(property_id) references Property

5. **Property Has Favorite**(property_id, favorite_id)
    - Property to Favorites(One-to-many). Each property can have multiple favorites.
    - Foreign Key(property_id) references Property
    - Foreign Key(favorite_id) references Favorite
6. **Property Has Booking**(property_id, booking_id)
    - Properties to Bookings (One-to-many) a property can have multiple bookings
    - Foreign Key(property_id) references Property
    - Foreign Key(booking_id) references Booking
7. **Property Has Review**(property_id, review_id)
    - Properties to Reviews (One-to-many) a property can have multiple reviews.
    - Foreign Key(property_id) references Property
    - Foreign Key(review_id) references Review
8. **User Writes Review**(user_id, review_id)
    - Users to Reviews (One-to-many) a user can write multiple reviews
    - Foreign Key(user_id) references User
    - Foreign Key(review_id) references Review
9. **User Likes Review**(user_id, review_id)
    - Users to Reviews (One-to-many) a user can like multiple reviews
    - Foreign Key(user_id) references User
    - Foreign Key(review_id) references Review
10. **Property Is Type**(property_id, type_id)
    - Property to Property Type (One-to-One) a property has a property type

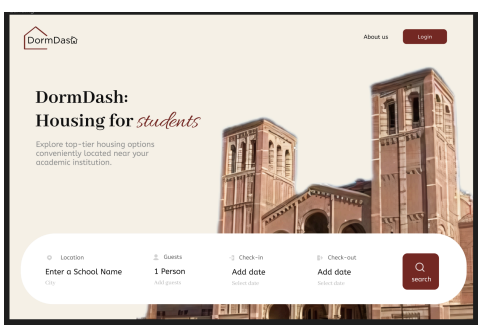
- Foreign Key(property_id) references Property
- Foreign Key(type_id) references Property Type

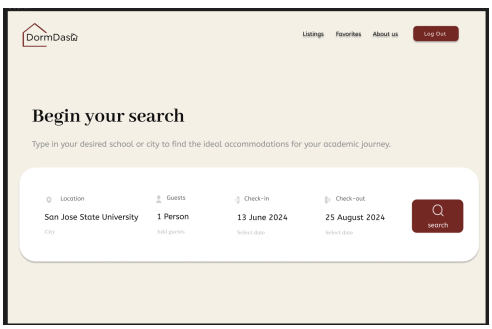
Major Design Decisions

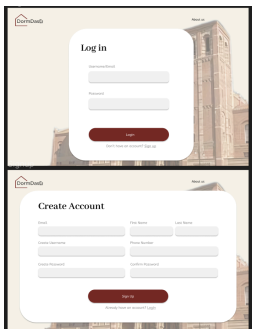
To develop our web-based application, several major design decisions were made to ensure that all functional and non-functional requirements were met. These decisions involve user experience, system architecture, and security in order to improve the overall performance of our application.

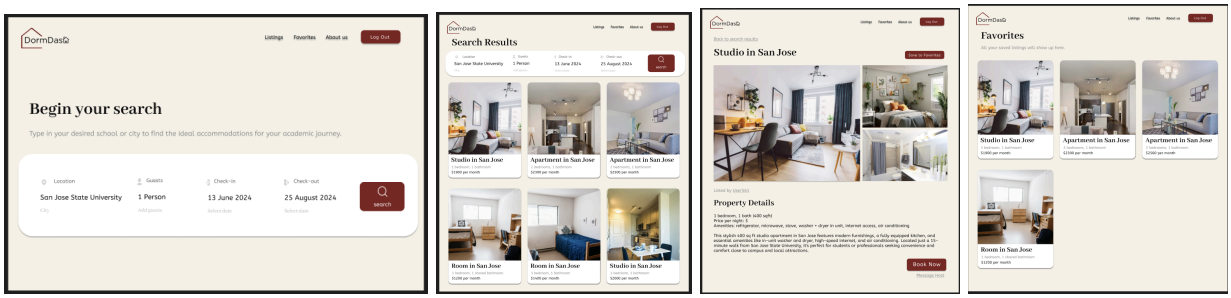
Before starting to develop our application, our team utilized Figma to create an interactive design prototype of our application, DormDash. This tool enabled us to create a cohesive and functional design across all pages. Through Figma, we developed interactive prototypes that not only simulated the user experience but also provided us insight into all aspects of the application that need to be implemented. This process was crucial for creating a consistent design with reusable components that can be used as a style guide and help maintain a coherent look and feel throughout the platform.

The following images illustrate our initial Figma design prototype for each page of our application.









When designing our application, our team prioritized creating an intuitive and easy-to-navigate user interface that would cater primarily to college students seeking housing near academic institutions. The design approach was minimalist and focused on allowing users to easily search for properties and save or book them in their user accounts. The interface was designed to be fully responsive, ensuring a seamless experience across different devices and screen sizes. The layout of property listings and details is structured to highlight essential information first, such as price, description, and availability. This hierarchy aids users in making quick and informed decisions. The front end of our application was developed using HTML, CSS, and JavaScript, to follow the Figma design template to create user interface components. We also utilized local storage for saving user login and search data, enhancing the user experience by making these elements persist between sessions without requiring server calls.

By adhering to these design principles, DormDash aims to offer a reliable, efficient, and user-friendly platform for booking short-term rentals, meeting the diverse needs of students and users.

Implementation details

The DormDash application was implemented using a tech stack comprising Node.js, Express, and MySQL for the backend, delivering a robust server-side architecture for handling HTTP requests, business logic, and database interactions. The frontend was developed using HTML, CSS, and vanilla JavaScript to create a dynamic and responsive user interface.

Backend Architecture:

- **Node.js & Express**: The backend server was built on Node.js with Express as the web framework. Express handled routing, middleware integration (such as authentication and error handling), and served as the core application logic layer.
- **MySQL Database**: MySQL was used as the relational database management system to store and manage data related to users, properties, bookings, reviews, and other application entities. The database schema was designed to ensure data integrity and support efficient querying for search and filtering functionalities.
- **Authentication & Security**: User authentication was implemented using JSON Web Tokens (JWT) and bcrypt for password hashing. JWTs were generated upon successful login and used to authenticate subsequent requests to protected endpoints, ensuring secure access control.
- **API Endpoints**: The backend exposed RESTful API endpoints to interact with the frontend. Endpoints were defined for user authentication (login, registration), property listings (search, filter), bookings, favorites, and reviews. Each endpoint handled specific CRUD (Create, Read, Update, Delete) operations related to its corresponding resources.

Frontend Architecture:

- **HTML/CSS/JavaScript**: The frontend was developed using HTML for structure, CSS for styling, and JavaScript for interactivity. JavaScript was utilized to make asynchronous HTTP requests to the backend API and dynamically update the UI based on server responses.

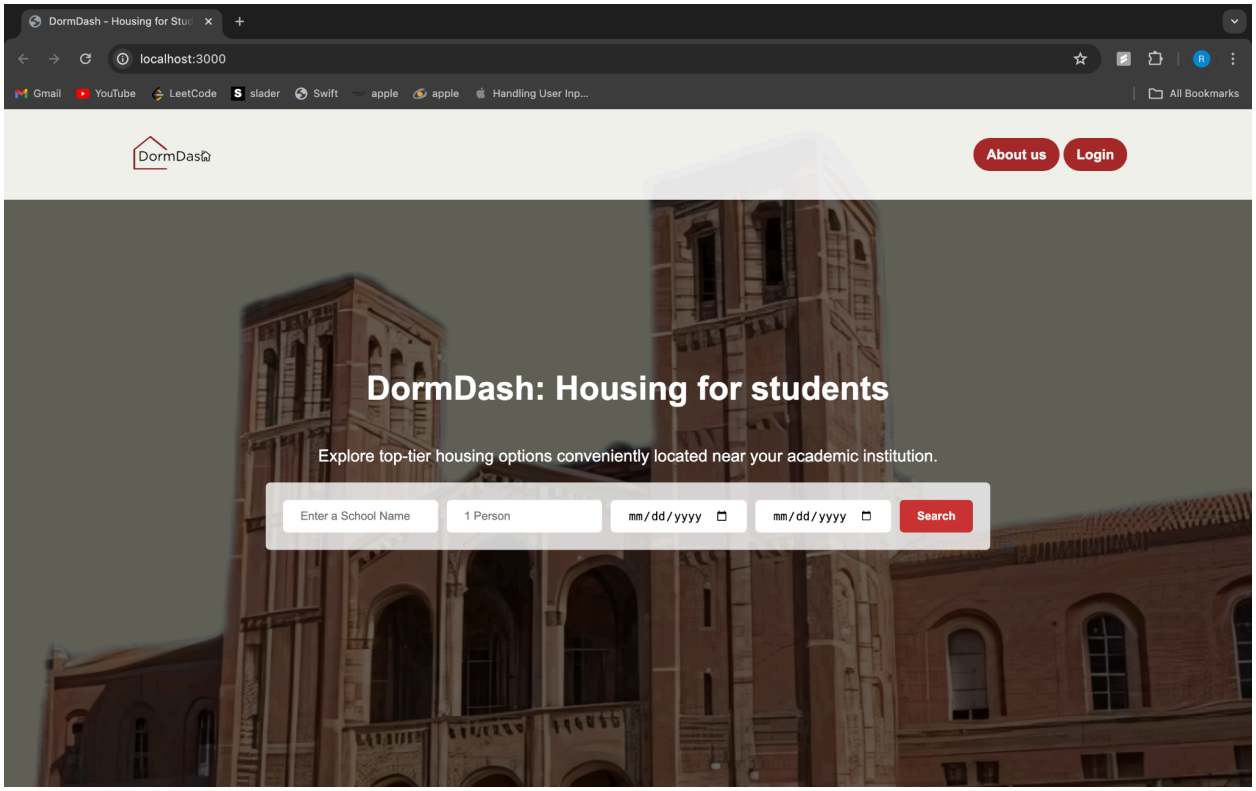
Additional Tools & Workflow:

- **Postman**: Postman was used for API testing and development, allowing developers to send requests, inspect responses, and debug API endpoints during the development and integration phases.
- **GitHub**: GitHub served as the version control system, enabling collaborative development.

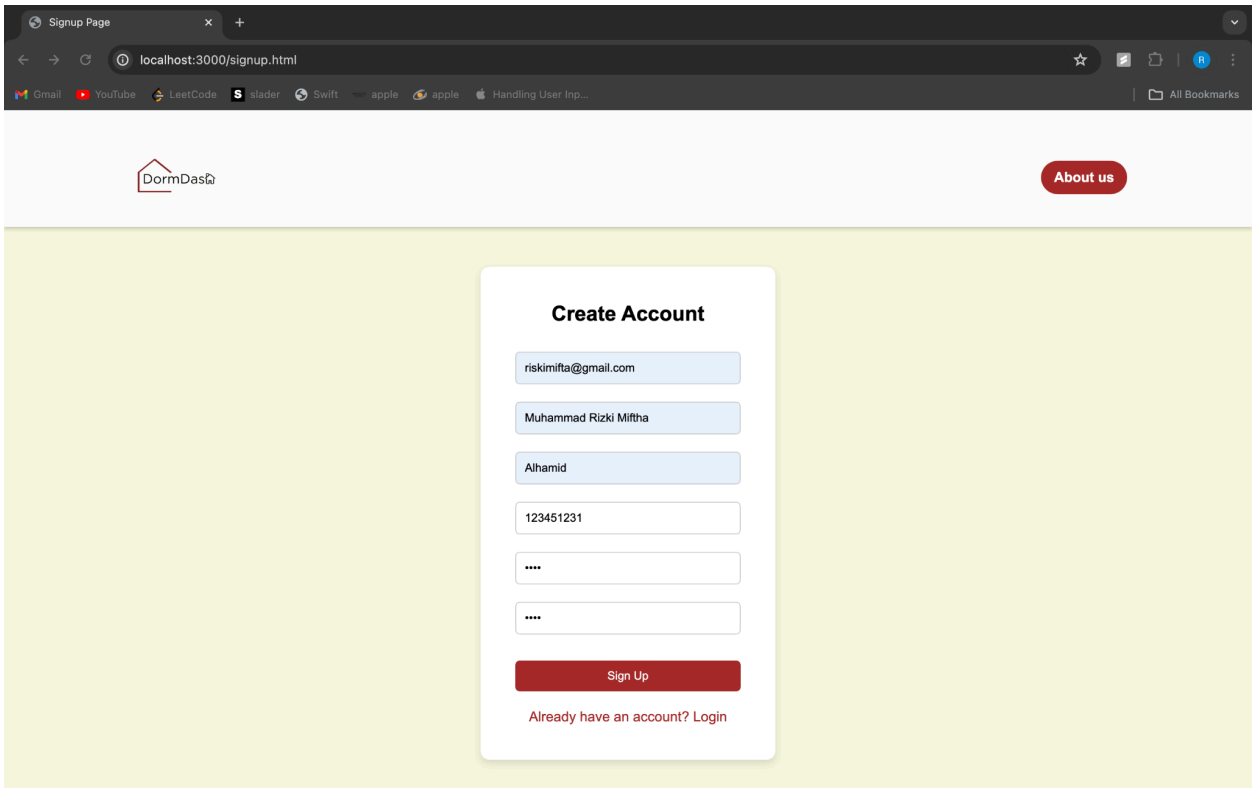
DormDash Demonstration

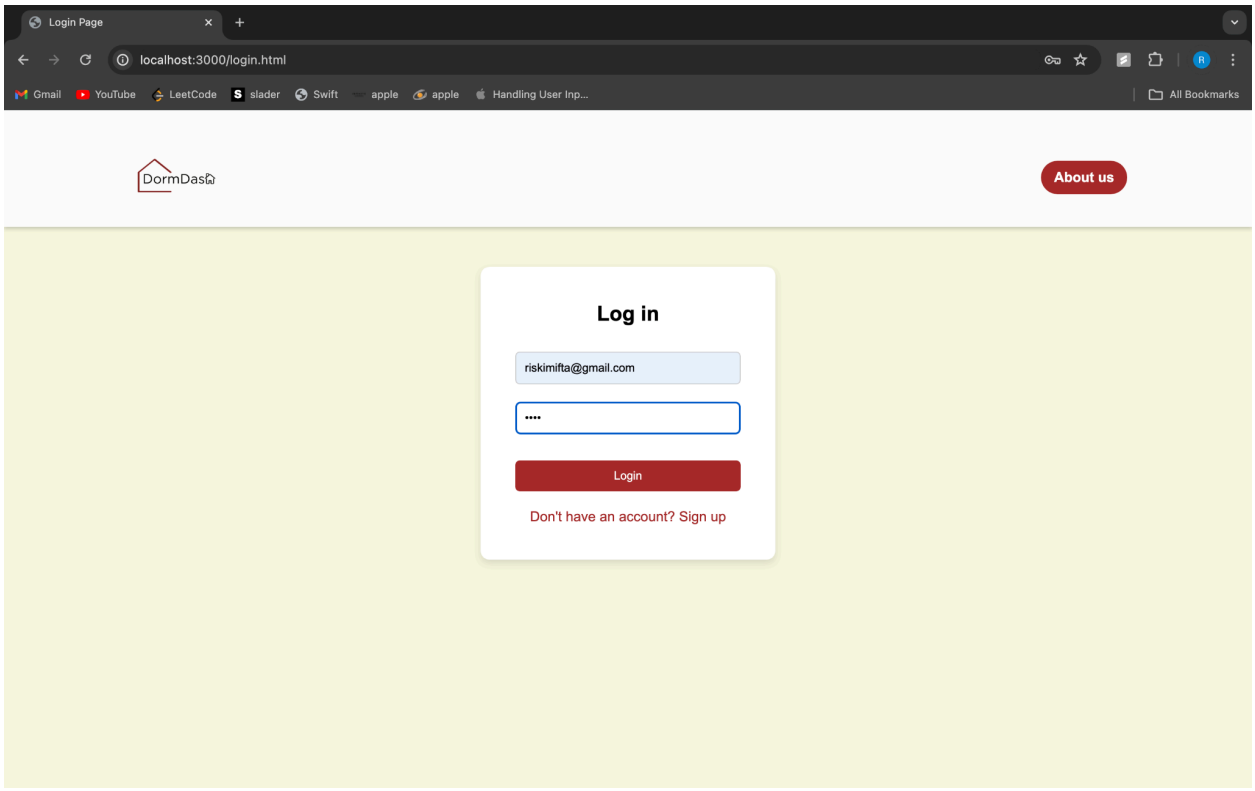
[Link]

Landing Page

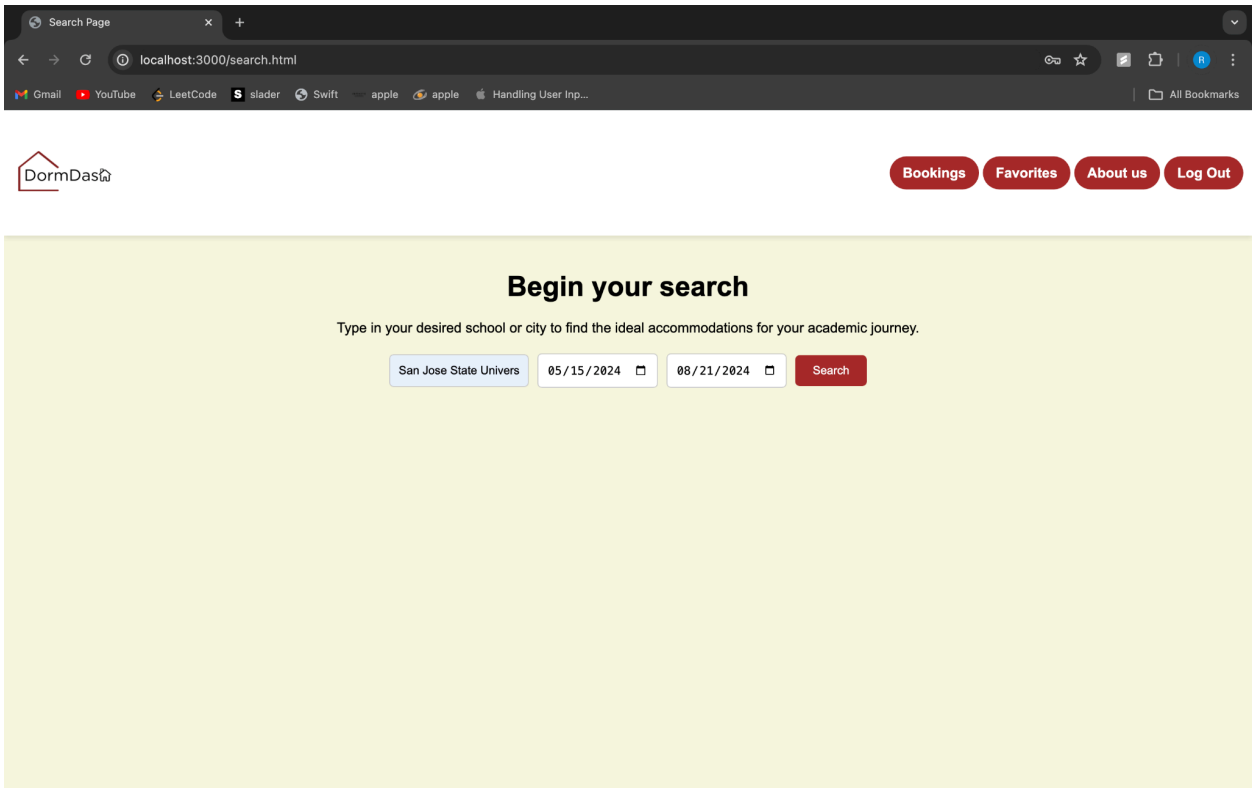


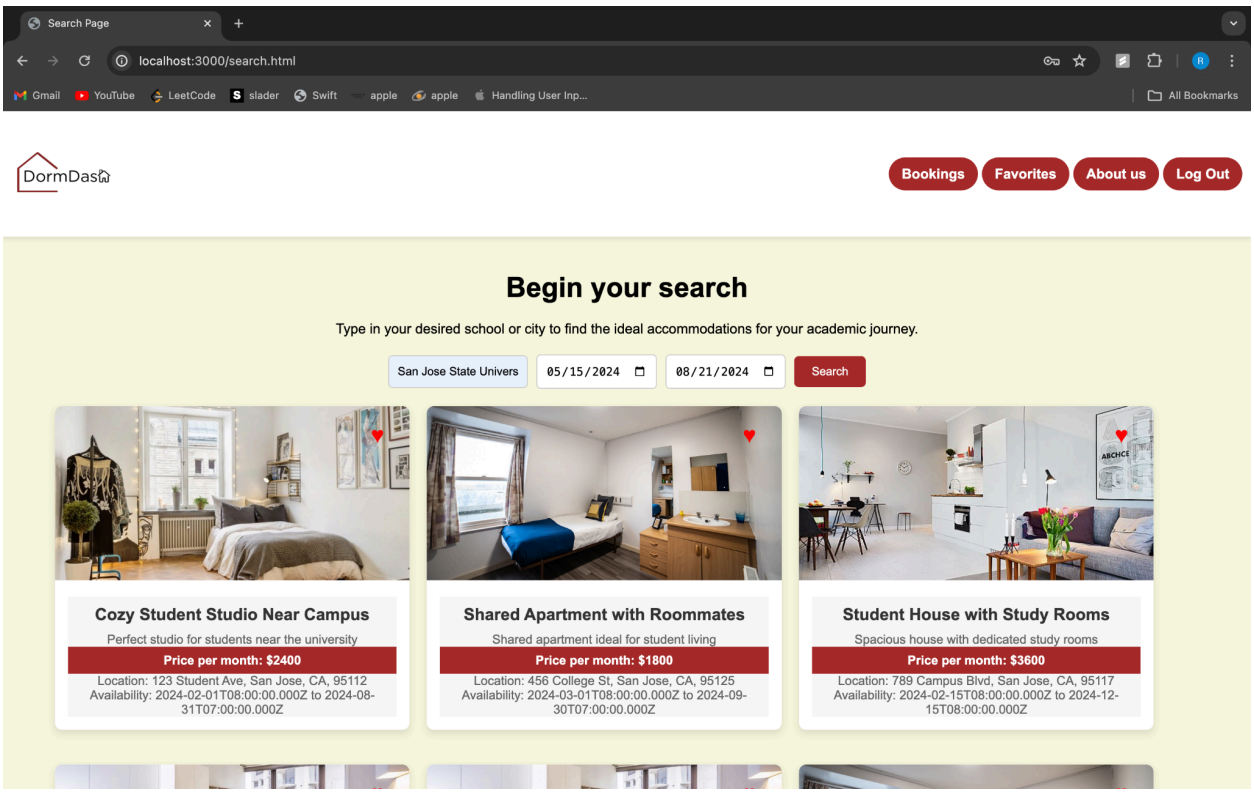
Login & Sign Up Page



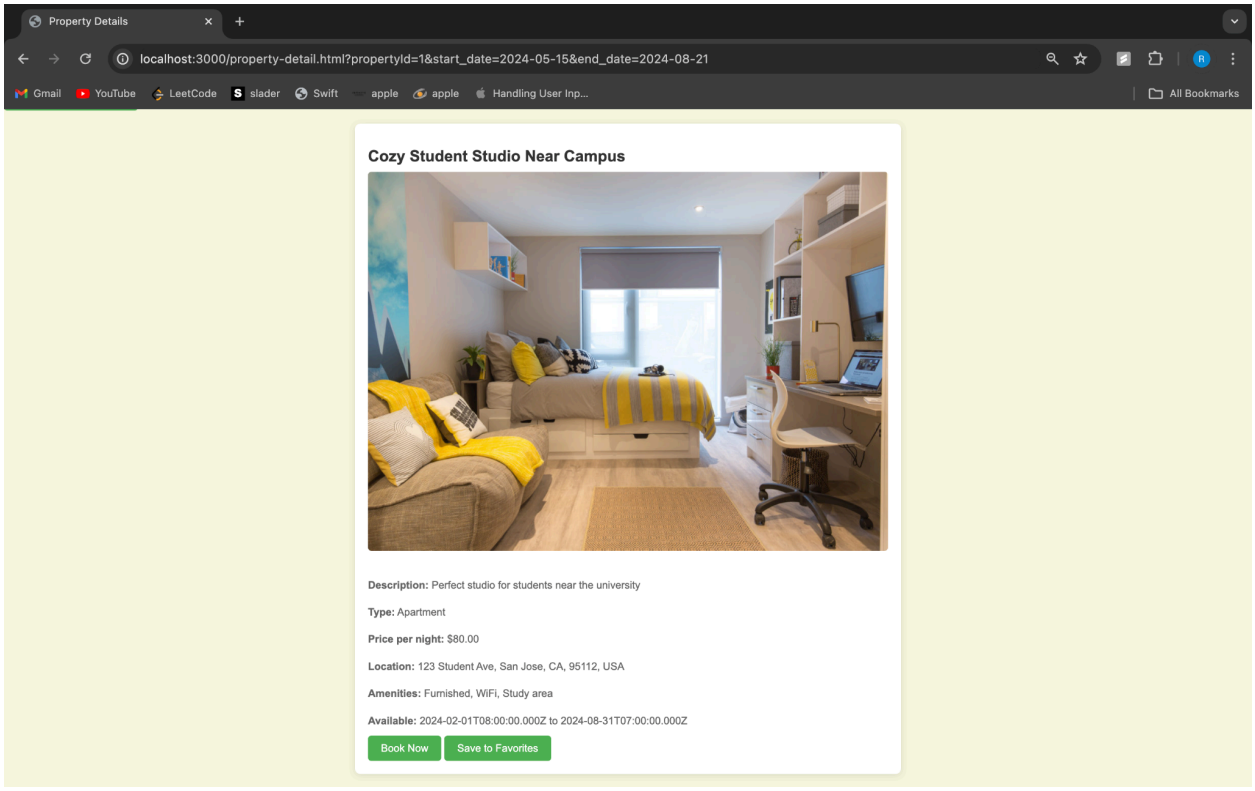


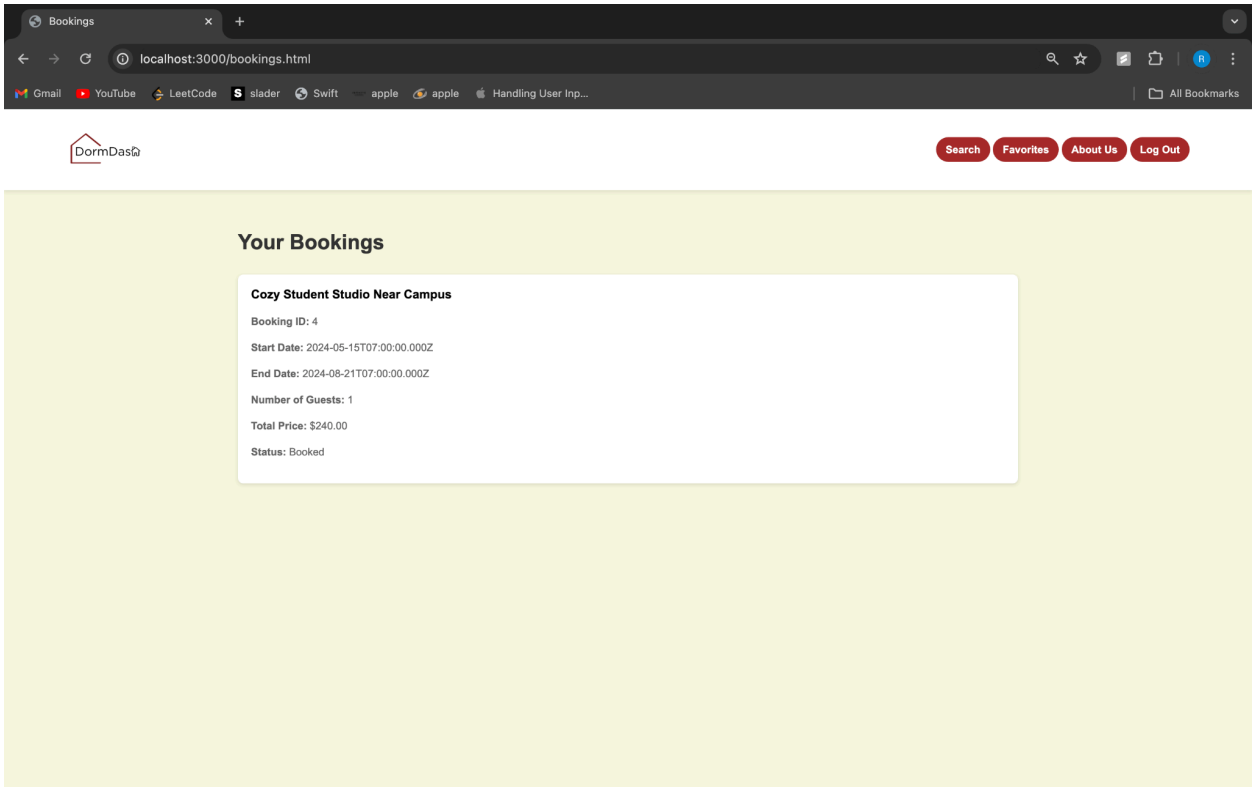
Search Page


Search Page
localhost:3000/search.html
Gmail
YouTube
LeetCode
slader
Swift
apple
apple
Handling User Inp...
All Bookmarks
DormDash
Bookings
Favorites
About us
Log Out
Begin your search
Type in your desired school or city to find the ideal accommodations for your academic journey.
San Jose State Univers
05/15/2024
08/21/2024
Search


Search Page
localhost:3000/search.html
Gmail
YouTube
LeetCode
slader
Swift
apple
apple
Handling User Inp...
All Bookmarks
DormDash
Bookings
Favorites
About us
Log Out
Begin your search
Type in your desired school or city to find the ideal accommodations for your academic journey.
San Jose State Univers
05/15/2024
08/21/2024
Search
Cozy Student Studio Near Campus
Perfect studio for students near the university
Price per month: $2400
Location: 123 Student Ave, San Jose, CA, 95112
Availability: 2024-02-01T08:00:00.000Z to 2024-08-31T07:00:00.000Z
Shared Apartment with Roommates
Shared apartment ideal for student living
Price per month: $1800
Location: 456 College St, San Jose, CA, 95125
Availability: 2024-03-01T08:00:00.000Z to 2024-09-30T07:00:00.000Z
Student House with Study Rooms
Spacious house with dedicated study rooms
Price per month: $3600
Location: 789 Campus Blvd, San Jose, CA, 95117
Availability: 2024-02-15T08:00:00.000Z to 2024-12-15T08:00:00.000Z

Details & Booking Page





Conclusions

DormDash successfully addresses the need for accessible student housing near academic institutions, offering a tailored and scalable platform that facilitates easy connection between students and property owners. This project demonstrated the feasibility and value of a dedicated housing solution that meets the specific needs of students and interns, establishing a strong foundation for future enhancements.

Lessons Learned

Throughout the development of DormDash, our team encountered several challenges that provided valuable lessons. Collaborative issues, particularly merge conflicts, underscored the importance of maintaining rigorous version control and synchronizing collaborative efforts through tools like Git. Technical hurdles, such as MySQL version compatibility and unexpected database connectivity issues, highlighted the critical need for thorough testing and robust error handling. These experiences emphasized the importance of environment stability and contingency planning in software development.

As we look to the future, DormDash is poised for several strategic enhancements aimed at elevating the user experience and expanding our service offerings. We plan to refine the interactive map feature, providing users with a more dynamic and visually engaging way to explore housing options relative to key landmarks and areas around their academic institutions. This enhancement will not only improve the search experience but also help users make more informed decisions about their housing relative to their daily commute and social needs.

Additionally, we intend to introduce advanced filtering options that allow for more granular searches based on criteria such as proximity to the campus, lease term flexibility, and inclusion of utilities. These filters will cater specifically to the diverse needs of students and interns, providing customized results that align with their unique preferences and budget constraints.

A significant upcoming development is the creation of a comprehensive host dashboard. This tool will enable property owners to manage their listings, bookings, and communications with potential tenants more efficiently, thereby improving response times and service quality. By simplifying the management process, we aim to attract and retain more property owners, which will in turn increase the variety of housing options available on our platform.

Expanding our listings through partnerships with more property owners and educational institutions is another key area of focus. By broadening our inventory across various neighborhoods and cities, DormDash will be able to accommodate a larger segment of the student population, ensuring that more students have access to suitable and affordable housing.

Enhancing the reliability of our reviews and ratings system is also critical. We plan to implement verified reviews to ensure that all feedback is genuine and helpful, thereby helping users make well-informed housing choices based on reliable user experiences.

Together, these improvements will not only enhance the functionality and user-friendliness of DormDash but also solidify our position as a leading housing solution for students and interns. By continually adapting to the needs of our users and staying ahead of

technological advancements, we aim to provide an indispensable tool that makes finding the perfect student accommodation as simple and stress-free as possible.

Final Thoughts

The journey of developing DormDash has been immensely educational and rewarding, laying down a robust blueprint for future growth. As we continue to refine DormDash, we are committed to leveraging the insights gained to enhance functionality, user experience, and overall service quality, ensuring that DormDash remains an essential tool for student housing solutions.