**Housing Initiative Project**

A Project Report

Presented to

CMPE-272

Fall, 2023

By

Naga Varun Bathina, Mohana Satyanarayana Moganti, Supriya Katragadda, Kevin Ho

12/06/2023

Copyright ⓒ 2023

Naga Varun Bathina, Mohana Satyanarayana Moganti, Supriya Katragadda, Kevin Ho

ALL RIGHTS RESERVED

**ABSTRACT**

**University Housing**

By Naga Varun Bathina, Mohana Satyanarayana Moganti, Supriya Katragadda, Kevin Ho

Housing is an age-old problem that continues to forever haunt society, and for postsecondary educational institutions, it is no different. These institutions create their own miniature societies, and although they provide their own housing services, these services are often faced with significant overhead and limitations due to a variety of factors. The goal of the Housing Initiative project is to tackle these limitations. We aim to provide a fully integrated housing solutions platform for these institutions so that they can focus more on the actual services that they provide to their students. We hope that by consolidating everything into one easy to manage platform, the actual services will improve, which will help improve not just the quality of student residential life, but also the housing problem in society.

**Acknowledgements**

We would like to acknowledge the hard work that the team has done thus far to make this project a reality.

**Table of Contents**

**Chapter 1 Introduction**

1.1 Project goals and objectives

1.2 Problem and motivation

1.3 Project application and impact

1.4 Project results and deliverables

1.5 Market research

1.6 Project report structure

**Chapter 2 Project Background and Related Work**

2.1 Background and used technologies

2.2 State-of-the-art technologies

2.3 Literature survey

**Chapter 3 System Requirements and Analysis**

3.1 Domain and business requirements

3.2 Customer-oriented requirements

3.3 System function requirements

3.4 System behavior requirements

3.5 System performance and non-functional requirements

3.6 System context and interface requirements

3.7 Technology and resource requirements

**Chapter 4 System Design**

4.1 System architecture design

4.2 System data and database design

4.3 System interface and connectivity design

4.4 System user interface design

4.5 System component API and logic design

4.6 System design problems, solutions, and patterns

**Chapter 5 System Implementation**

5.1 System implementation summary

5.2 System implementation issues and resolutions

5.3 Used technologies and tools

**Chapter 6 System Testing and Experiment**

6.1 Testing and experiment scope

6.2 Testing and experiment approaches

6.3 Testing report

**Chapter 7 Conclusion and Future Work**

7.1 Project summary

7.2 Future work

**List of Figures**

Figure 1. Use Case Diagram of Housing Initiative………………………………………………16

**List of Tables**

Table 1. Market share of major companies in industry……………………………………………9

**Chapter 1 Introduction**

**1.1 Project goals and objectives**

The Housing Initiative project is a vision of the future where housing doesn’t have to be such a big problem. In this future, we imagine housing as a process that can be as simple as a series of swipes. Although housing has been made simple in this future, university housing still presents a special use case. As such, the goal of the Housing Initiative project is to build a product that acts as a housing solutions platform targeting university students. Our objective is to recruit universities of all kinds to adopt and employ our software platform for their housing solution needs. We envision a future where our product has been widely adopted to make vast improvements to the housing situation in universities.

**1.2 Problem and motivation**

The problem that this project tackles is the difficulties of finding and obtaining housing–specifically for university students. Within SJSU alone, housing is a very relatable issue. As referenced in [1], in a 2021 SJSU Cares survey, 11.2% of student participants had experienced homelessness while 41.5% had experienced housing insecurity. This statistic is heartbreaking. Because we believe that everyone deserves the right to be housed, we were motivated to build a solution that addresses this problem. The needs of this project are to implement an easy and accessible way for university students to find and obtain housing and for universities to manage their housing solutions. To accomplish this, we will need to offer a seamless integration process for universities to connect to the platform with their own respective endpoints. With this done, the completion of this project will result in technical contributions such as a software as a service model with an intuitive interface that simplifies and improves the quality of university housing solutions.

**1.3 Project application and impact**

The applications for this project are mainly to combat the housing crisis within our society. However, we also note that housing solutions offered by universities can be lack-luster because of all the endpoints they have to manage. This is also where our project comes into play because as an integrated platform, we will handle all the connections to the different services that the universities offer. We anticipate that our project will positively impact society by improving the process for finding and obtaining housing, thereby improving the housing situation. By doing the difficult work of consolidating everything together, we anticipate that universities will be able to eliminate significant overhead and spend more time improving their housing solution services. When the overall quality for residential life of university students becomes better, we also anticipate students to perform better and thus increase the quality of their academic contributions.

**1.4 Project results and deliverables**

The results of this project will include the main software application code that will enable university students to search and apply for housing. The platform will also include the other side–providing functionality for universities to integrate their own housing solutions. By the end of this project, we will have finished the deliverables, which contains the code base as well as the project report and presentation slides for the prototype. The prototype of the project mainly consists of the system platform where the university housing solution services are streamlined and consolidated together into one convenient and efficient application.

**1.5 Market research**

Although the market for property management software solutions is very competitive with many already well established companies, there does not seem to be any that are solely focused on student housing. Several major property management software companies that do include student housing in their services are AppFolio, Entrata, and RealPage. Table 1 shows their respective market shares, which was obtained from reference [2]. Additionally, after a thorough analysis of the products, many of them seem to behave similarly, with only a unique branding or a few unique features distinguishing each of them from the rest.

Table 1. Market share of major companies in industry

| Company | Market Share (%) |
| --- | --- |
| AppFolio | 15.68 |
| Entrata | 12.35 |
| RealPage | 8.00 |

**1.6 Project report structure**

The structure of this project report follows the format and outline described in the table of contents. The report begins with an introduction before delving into the background and context in which the project is based on. Then, it goes to the requirements analysis of the project that was conducted at the beginning phases of the project. After discussing the project’s system requirements, the report details the system design for the project and the process that was involved. The next section of the report mentions how this system was implemented, and how the system was tested. The report finally concludes with a summary of the entire process and an assessment with potential future work that can be done.

**Chapter 2 Background and Related Work**

**2.1 Background and used technologies**

As a product that aims to allow universities to seamlessly integrate their housing solutions with our service, there will be numerous technologies and concepts that our project will need to employ to ensure such functionality. Because our application will deal with both university students navigating the housing process and university staff members managing the housing services, it will potentially be dealing with many requests. As such, one of the concepts that our service will require is the concept of scalability in order to be able to keep our service stable and available for usage by all of the different types of users even when the demand increases. To implement scalability into our service, we will use technologies such as MongoDB for our database to efficiently store data. Also, we plan to use React and NodeJS to efficiently process requests and keep the application up and running. Additional knowledge that our project requires includes information related to leasing, payments, and other legal provisions. Armed with knowledge concerning these different fields, we should be able to more easily provide a means of integration between our service and these related aspects of the housing services.

**2.2 State-of-the-art technologies**

While most of the existing products in the student housing software market are similar, many of the companies are able to build their own branding with a few unique features. As one of the major players involved in the student housing software market, AppFolio offers a centralized property management platform with an extensive range of features including data analytics, workflow automation, and artificial intelligence integration. Similarly, Entrata offers an all-in-one management platform marketed as a modern operating system. For a third perspective, RealPage markets its own software product as a solution that will help property managers with everything they will ever need. Although these products all offer an automated way for real estate managers to provide their housing solutions to their student tenants, they each have some additional fluff.

**2.3 Literature survey**

During our research of existing literary research about student housing software solutions, we found that, as referenced in [3], students’ acceptance of residential management software is greatly influenced by their experienced usefulness of such tools. Furthermore, this research found that use of residential management software complemented residents’ experienced usefulness of the tool by improving the entire process from task completion to response time. As such, housing software solutions will have to ensure that they complement the housing services to make them more efficient in order to become successful.

**Chapter 3 System Requirements and Analysis**

**3.1 Domain and business requirements**

To address the unique challenges and requirements of university student housing, we consider the following:

**Business Objectives:** The business objectives that we aim to solve are to enhance the overall efficiency of obtaining housing, to improve student experience and satisfaction, and to reduce administrative overhead for institutions by decreasing the competition for on campus housing and listing all the probable properties in the vicinity of University which accommodate students and are safe for university. Listing a number of options to the students increases the visibility and healthy competition among communities to provide better experience to incoming lease takers.

**Understanding Student Needs:** Incorporating feedback from students to align the system with their preferences facilitates in understanding the students requirements and preferences, lifestyle, and academic requirements. Agglomeration of properties for students helps students to pick their ideal choice by comparing the listed properties. The property owners or leasing office posts Advertisements in the Website posting the properties and prices along with features. Also, the lease undertaker students can post an advertisement in order to secure roommates according to their preferences. The Students can sign-up if for the first time else login to find their desired property.

**3.2 Customer-oriented requirements**

**Student-Centric Design:** The Housing Initiative’s system design focuses on the ease of use for both students and property owners. The Housing Initiative acts as an effective interface for communication between both students and property owners. For a completely fresher student, the application helps in accessibility of available properties and preferences specific to university students' lifestyles. The lease terms, deposits for those lacking SSN, and other conditions should be clearly listed in the interface so that students can quickly access the core details. As a one-stop interface, the application should also allow for students to pay their fees.

**Feedback-Driven Development:** The Housing Initiative should alsoregularly incorporate student feedback with data analytics to enable universities to align their services with actual housing needs and expectations. It is critical to shape the housing solutions according to student needs, so the application should give a clear picture of both the properties and the students’ responses owners as the platform clearly lists both the ends requirements and preferences by taking a look at the data. The reviews are crucial as students believe their seniors and other lease takers in order to estimate the expenses , safety and issues regularly faced like Clogging, issue resolving by property leasing office. This helps in

**Adaptive to Student Trends:** Ensuring the system remains flexible and adaptable to the evolving needs and trends among the student population.

●Student-Centric Design: Designing the system with a focus on ease of use, accessibility, and preferences specific to university students' lifestyles.

●Feedback-Driven Development: Regularly incorporating student feedback to ensure the system aligns with their actual housing needs and expectations.

●Adaptive to Student Trends: Ensuring the system remains flexible and adaptable to the evolving needs and trends among the student population

**3.3 System function requirements**

**Housing Allotment Functionalities:** Developing features for lease application processing which includes login and signup, room assignment, and payment handling tailored to the university housing context. The functionality for the owner of the property to post advertisements regarding vacancies and lease terms who has signup and login functionality as well. The feature of updating and deleting the information regarding advertisement posted regarding property availability according to the lease taken. The list of available accommodations,their features,terms and prices are clearly listed. The images of the houses are displayed in the web application from firebase. The user can filter the available properties based on the features of the property like having 2 bedrooms or 3 baths, or a townhouse unit.

**Integrated Support Features:** We plan to integrate student accommodation features likefunctionalities for student queries, maintenance requests, and emergency contact integration. This helps students to solve all their issues in a single stop web application without accessing several applications for various functionalities.

**Scalable and Flexible Design:** One of the key features of the Housing Initiative isensuring that the system is scalable to accommodate growing student numbers and that the system is flexible enough to adapt to different university settings starting with San Jose State University. The architecture and software requirements are chosen in order to facilitate the ever growing number of students.

**3.4 System behavior requirements**

**Response to Student Applications:** Defining the system that handles and processes student housing applications, including prioritization and allocation logic is so crucial for effectively responding to the applying students. Coordinating about the queries of students regarding the lease terms, security deposits through mails is ensured. As soon as the lease application is started by a student, that particular property is removed from the display catalog and kept as a hold. The

hold is converted into the lease undertaken by that particular primary residence along with other members. The listed accommodations are regularly checked and updated based on the latest additional facilities or terms so that there is no gap between both ends.

**Handling Maintenance and Issues:** In order to ensure smooth user experience, it's very much essential to resolve all the minor and emergency issues of users in time by the leasing’s support team. The discussion forum in the web application is where users can log their issues and communicate using description and pictures. This includes outlining the system's response to maintenance requests and other housing-related issues reported by students.

**System Adaptability:** The prices and terms are adjusted dynamically accordingly to the demand like the start of a new semester.The ability of the system to adapt to different housing scenarios and peak times like opening the new lease houses, and handling heavy traffic for facilitating the new traffic of incoming students.

**3.5 System performance and non-functional requirements**

**Performance Metrics:** The Hpplication is designed in a manner to handle traffic to accommodate requests and Lease applications from students. The update of the properties available and services is real quick and efficient. The system is up throughout the year and designed in a way to handle the traffic during the season like Fall,Spring.

**Security and Data Privacy:** Ensuring the highest standards of data security and privacy, particularly for student personal and housing data is an integral part of Sparta ecosystem. Sensitive information about property posters or users' sensitive information is not compromised.

**User Experience and Accessibility:** Aim of the Sparta is prioritizing a seamless, intuitive user interface that is accessible to all students, including those with disabilities. It consists of simple User Interface with no page taking more than 3-clicks.

**3.6 System context and interface requirements**

**Integration with Properties**: The system of Sparta is aligned in such a way that integrates with existing university localities and the properties nearby for seamless information flow and functionality. The property owners enroll and post their properties as advertisements which are looked through and interested students would follow up by starting their lease application hence a smooth flow is ensured through seamless integration.

**User Interface Design**: Main aim is to simplify the process of Housing allotment which needs a simpler User Interface. Focusing on a user-friendly, intuitive interface tailored for students, enabling easy navigation and housing management is believed to be the core of this initiative.

**Multi-platform Accessibility**: Web Application - Sparta ensures that the system is accessible across various devices and platforms, catering to the diverse tech usage among students making it platform independent.

**3.7 Technology and resource requirements**

**Frontend** : React.JS is utilized to build interactive elements based on UI components.

React.js is commonly used for:

* Single-Page Applications (SPAs): Providing a smooth and interactive user experience.
* User Interface Libraries: Building reusable UI components in large applications.
* Cross-Platform Development:\*\* React Native extends React to build mobile applications.

Minimum Requirements:

1. Node.js:

- Version: 12.0.0 or higher

2. npm (Node Package Manager):

- Version: 6.0.0 or higher (typically comes with Node.js)

**Backend**: **Node.JS, Express.JS**

Node.js is a runtime environment that allows JavaScript code to be executed on the server side. It uses the V8 JavaScript engine from Google Chrome and provides a non-blocking, event-driven architecture. This allows developers to build scalable and high-performance network applications.

Node.js Minimum Requirements:

1. Node.js:

- Version: 12.0.0 or higher

2. npm (Node Package Manager):

- Version: 6.0.0 or higher (typically comes with Node.js)

**Database**: **MongoDB**

MongoDB is a NoSQL database that provides a flexible and scalable solution for storing and retrieving data. It stores data in a JSON-like format called BSON (Binary JSON) and is known for its flexibility, scalability, and ease of use.

Minimum Requirements:

1. MongoDB:

- Version: 4.0 or higher

**Security**: **​​JSON Web Tokens (JWT)**

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way to represent information between two parties. This information can be verified and trusted, making JWTs a secure means of transmitting data between parties.

Structure:

- JWTs consist of three parts: a header, a payload, and a signature. These parts are encoded to form a string, which is the JWT.

```plaintext

Header.Payload.Signature

```

**Chapter 4 System Design**

**4.1 System architecture design**

The system architecture for the Housing Initiative project follows a MERN stack, which stands for MongoDB, Express.js, React.js, and Node.js. This architecture is well-suited for building scalable and efficient web applications.

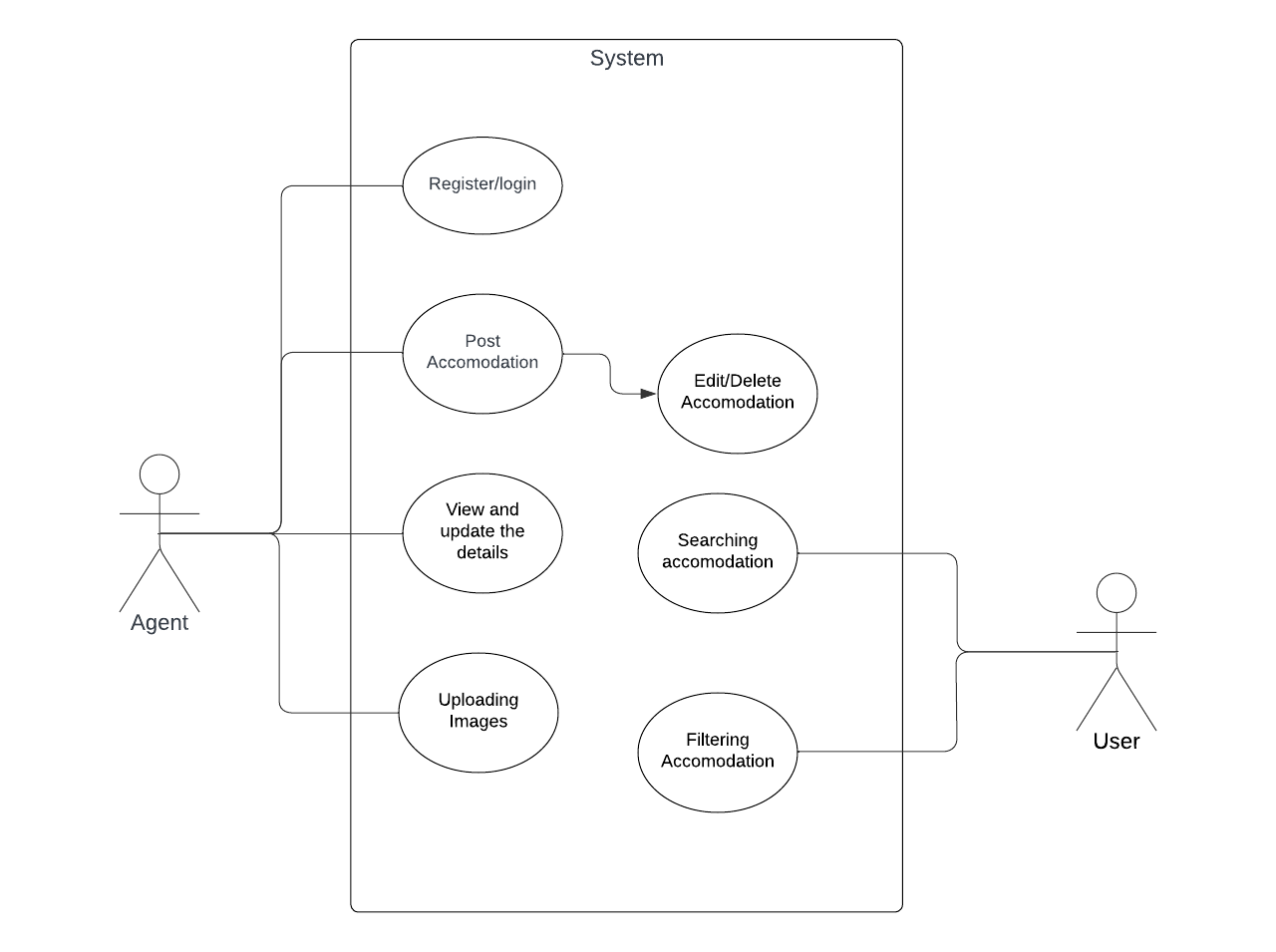


Fig 1: Use Case Diagram of Housing Initiative

**4.2 System Data and Database Design:**

1. Collections:

- Student Collection: Stores information about registered students.

- Property Collection: Contains details about available properties posted by property owners.

- Lease Collection: Records lease applications submitted by students.

- Review Collection: Stores reviews provided by students for different properties.

2. Data Relationships:

- Students and Lease: One-to-Many relationship (A student can submit multiple lease applications).

- Property and Lease: One-to-Many relationship (A property can have multiple lease applications).

3. Indexes:

- Indexes on critical fields for efficient data retrieval.

- Ensure optimal performance for search and query operations.

4. Database Security:

- Implement access controls and authentication mechanisms to secure the database.

- Regularly update MongoDB to address security vulnerabilities.

**4.3 System interface and connectivity design**

1. User Interface (UI):

- Designed using React.js for a user-friendly and intuitive experience.

- Supports multi-platform accessibility, ensuring compatibility with various devices.

2. API Endpoints:

- Utilizes Express.js to define RESTful API endpoints.

- Endpoints for user registration, property listing, lease application, and other relevant functionalities.

3. Data Flow:

- Frontend communicates with the backend through API calls.

- Data is transmitted securely using JWT for authentication and authorization.

4. Third-Party Integrations:

- Firebase integration for storing and retrieving property images.

- Google Maps API for displaying property locations and nearby amenities.

5. Scalability:

- System designed to be scalable to accommodate a growing number of students and properties.

- Load balancing and efficient resource utilization for scalability.

This system design ensures a robust, scalable, and secure architecture for the Housing Initiative project, addressing the unique challenges and requirements of university student housing. The seamless integration of frontend, backend, and database components contributes to an efficient and user-centric housing solution platform.

**4.3 System Interface and Connectivity Design:**

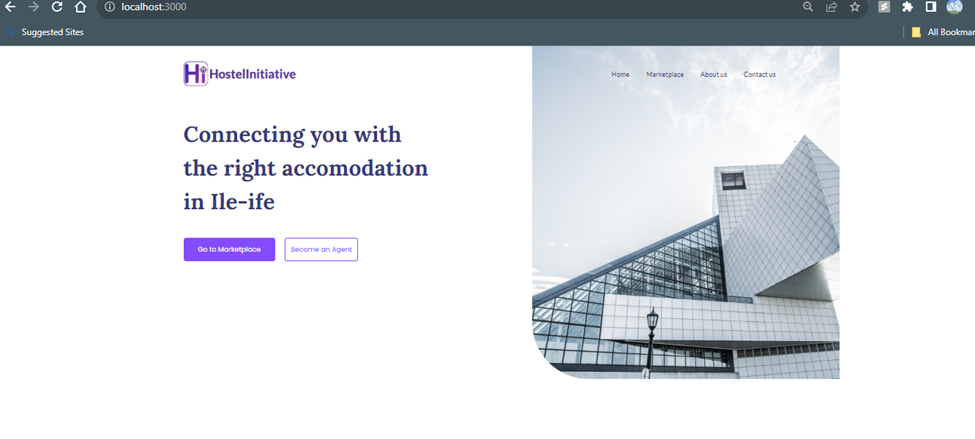
In this section, we'll outline the interface and connectivity design for the Housing Initiative project.

1. User Interface (UI) Design:

- Utilizes React.js for building a responsive and user-friendly interface.

- Follows a modular approach with reusable components for efficiency.

- Implements a single-page application (SPA) design for seamless navigation.



2. API Endpoints and Connectivity:

- Defines RESTful API endpoints using Express.js for communication between frontend and backend.

- Ensures secure data transmission using HTTPS.

- Implements JWT for authentication and authorization.

3. Web Socket Communication:

- Utilizes Web Sockets for real-time communication between the frontend and backend.

- Enables instant updates on lease status and property availability.

4. Multi-Platform Accessibility:

- Ensures the web application is accessible across various devices and platforms.

- Responsive design for optimal viewing on desktops, tablets, and mobile devices.

**4.4 System User Interface Design:**

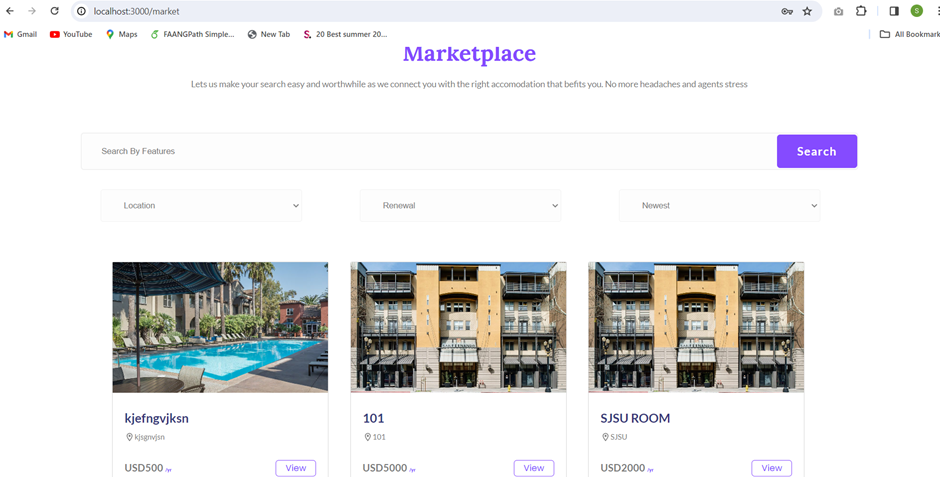
The user interface design focuses on simplicity, ease of use, and accessibility for students, property owners, and administrators.

1. Student Dashboard:

- Displays available properties with filters based on preferences.

- Provides an intuitive lease application process.

- Allows students to track the status of their lease applications.

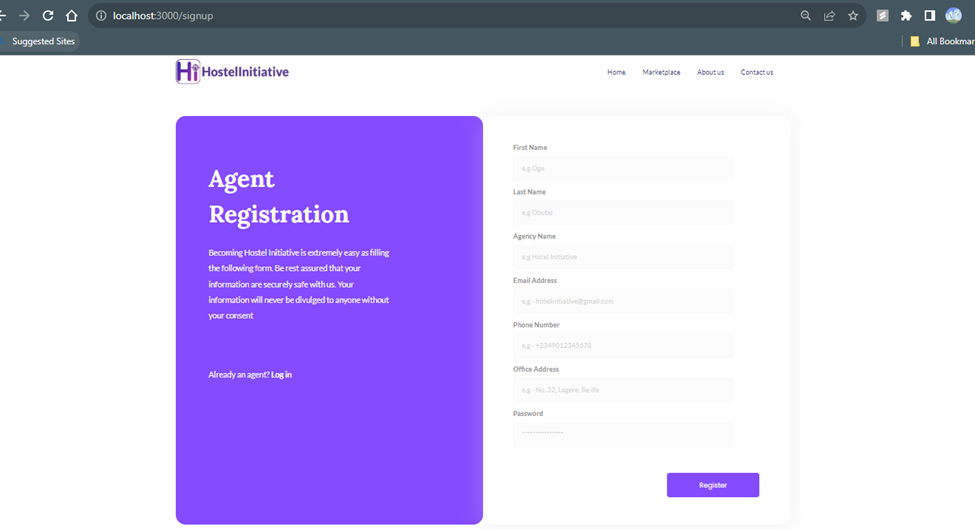


2. Property Owner Dashboard:

- Enables property owners to list available properties.

- Allows for easy management of property information.

- Provides notifications for lease applications.

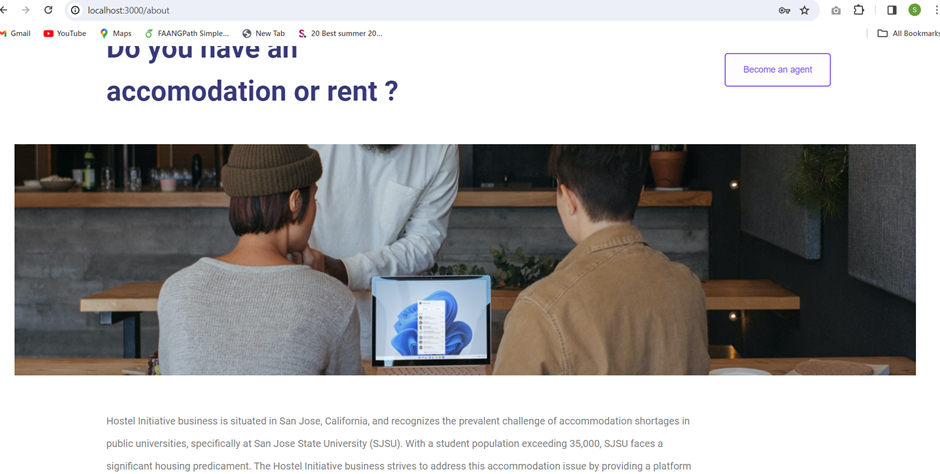


3. Admin Dashboard:

- Facilitates administrators in managing overall system operations.

- Allows for monitoring lease applications and property listings.

- Provides analytics and reporting features.



**4.5 System Component API and Logic Design:**

1. Frontend Components:

- Breaks down UI into React components for modularity.

- Manages state and props for dynamic data rendering.

2. Backend Logic:

- Implements Express.js logic for handling API requests.

- Manages data validation, processing, and communication with the database.

3. Database Interaction:

- Defines MongoDB schema for efficient data storage.

- Utilizes Mongoose ODM for interacting with the MongoDB database.

**4.6 System Design Problems, Solutions, and Patterns:**

1. Problem: Scalability Challenges

- As the user base grows, scalability becomes a concern.

Solution:

- Implements load balancing and horizontal scaling.

- Utilizes cloud services for dynamic resource allocation.

2. Problem: Real-Time Updates

- Ensuring real-time updates for lease status and property availability.

Solution:

- Implements Web Sockets for bidirectional communication.

- Enables instant updates for all connected clients.

3. Design Pattern: Microservices Architecture

- Breaking down the system into smaller, independent services.

Benefits:

- Enhances maintainability and scalability.

- Allows for technology stack flexibility.

4. Design Pattern: Caching Mechanism

- Reducing database load by implementing a caching mechanism.

Benefits:

- Improves response times for frequently accessed data.

- Reduces the load on the database.

These design elements and patterns address various challenges and ensure a robust, scalable, and efficient system for the Housing Initiative project. The diagrams provide a visual representation of the system's architecture, interface, and connectivity.

**Chapter 5 System Implementation**

### **5.1 System Implementation Summary**

**Overview of Project**: Housing Initiative aims to simplify the process of property rental for students by featuring listings, personalized user accounts, and an efficient booking mechanism.

**Implementation Process**: Utilizing MongoDB for data storage, the application manages user information and property details. Express.js was leveraged for developing API interfaces, and React was chosen for its dynamic user interface capabilities. Node.js underpinned the server-side operations. Cloud deployment was chosen for its scalability and broad accessibility.

**Testing Approach**: Comprehensive testing was conducted, including backend API unit tests, frontend tests via React Testing Library, and full-scale integration testing to ensure fluid functionality.

### **5.2 System Implementation Issues and Resolutions**

**Database Connection**: We encountered and subsequently resolved MongoDB connection issues by configuring it accurately and using Mongoose for structured data handling.

**Managing State in React**: The complexity of state management was managed effectively using React’s Context API for global states, and Redux for intricate state scenarios.

**API Integration Challenges**: The integration of RESTful APIs required strategies to maintain data integrity and manage asynchronous operations, accomplished using async/await patterns and Axios for handling API interactions.

**Deployment Obstacles**: Deployment complexities, especially those involving server settings and environmental variables, were tackled using Docker for consistent deployment settings and Heroku for its cloud-based services.

**5.3 Technologies and Tools Employed**

**MERN Stack Constituents**: The application's backbone consisted of MongoDB for database management, Express.js for server and API setup, React for the user interface, and Node.js for backend logic.

**Development Instruments**: Visual Studio Code was the primary development environment, complemented by Git for source control and JIRA for project management.

**Testing Tools**: Backend testing relied on Jest for ensuring API robustness, while frontend testing was done using React Testing Library.

**Supplementary Libraries and Middleware**: Mongoose was crucial for data schema modeling in MongoDB, Passport.js managed user authentication, and Axios streamlined API communication

**Chapter 6: System Testing and Experiment**

This section outlines the objectives, components, methodologies, and metrics used to validate the system's functionality, reliability, and performance. It also addresses risk assessment, identifying potential challenges, and offers a detailed schedule for phased testing and optimization, ensuring a thorough evaluation of the system's capabilities.

**6.1 Testing and Experiment Scope**

In this section, we delineate the comprehensive scope of our system testing and experimentation endeavors. Our primary goal is to validate the functionality, reliability, and performance of the 'Hostel Initiative' system. To achieve this, we employ various methodologies, tools, and metrics.

**6.1.1 Testing Objectives**

The fundamental objective of our testing and experimentation phase is to ensure that the 'Hostel Initiative' system operates flawlessly, meeting user expectations and industry standards. This encompasses:

1. Verification of core functionalities such as user registration, room booking, and payment processing.
2. Assessment of system performance under varying loads.
3. Evaluation of the user experience, including usability, responsiveness, and error handling.
4. Validation of data security and privacy measures.
5. Measurement of user satisfaction and feedback collection.

**6.1.2 Testing Components**

Our system comprises several critical components, each playing a pivotal role in its functionality. The components subject to testing include:

1. User Management: Covers user registration, login, and profile management.
2. Room Booking: Ensures the seamless booking and cancellation of hostel rooms.
3. Feedback System: Assesses the mechanism for collecting and processing user feedback.

**6.1.3 Testing Methodologies**

We employ a combination of manual and automated testing to comprehensively evaluate our system:

1. Postman for API Testing: Postman is utilized to rigorously test API calls, ensuring that endpoints respond accurately and handle various input scenarios.
2. Jest for Unit Testing: Jest is utilized for unit testing, focusing on individual components and functions within the system. This includes testing JavaScript functions, API endpoints, and data handling.

**6.1.4 Test Scenarios**

Our test scenarios encompass a wide range of user interactions and potential system behaviors:

1. User Registration: Testing successful registration, invalid inputs, and error handling.
2. Room Booking: Assessing booking confirmation, cancellation, and availability under different conditions.
3. User Feedback: Ensuring accurate feedback submission and processing.

**6.1.5 Experimentation Metrics**

For experiments aimed at gathering insights and data-driven decisions, we focus on key metrics, including:

1. User Engagement: Tracking user activity, session duration, and feature utilization.
2. Conversion Rates: Measuring the percentage of successful bookings and payments.
3. User Satisfaction: Gathering user feedback and satisfaction scores through surveys and feedback forms.

**6.1.6 Testing Environment**

Our testing environment is set up to mimic real-world conditions:

1. Hardware: Utilizing a variety of devices, including desktops, laptops, and mobile devices.
2. Software: Testing on various web browsers (Chrome, Firefox, Safari) and operating systems (Windows, macOS, Android, iOS).
3. Network Configurations: Evaluating system performance under different network speeds and conditions.

**6.1.7 Expected Outcomes**

Our expected outcomes include:

1. Validation of core functionalities with a minimal error rate.
2. Demonstration of optimal system performance, even under high load conditions.
3. High user satisfaction scores and positive feedback.
4. Confirmation of robust data security measures.

**6.1.8 Constraints and Limitations**

We acknowledge certain constraints and limitations, including:

1. Limited resources and testing time.
2. Challenges in simulating all real-world scenarios.
3. Potential issues with external dependencies (e.g., third-party payment gateways).

**6.1.9 Risk Assessment**

In the Risk Assessment phase, we identify and address potential challenges that may affect the smooth execution of our testing and experimentation. These risks include data security concerns, the possibility of system downtime, variations in user behavior, dependencies on external services, resource limitations, testing environment variability, potential difficulties in collecting user feedback, and the risk of scope creep. By recognizing these risks early, we can proactively implement strategies to mitigate them and ensure the testing process progresses effectively.

**6.1.10 Schedule**

Our project follows a well-structured schedule to maintain progress and meet key milestones. It includes several phases: Unit Testing and API Testing, User Acceptance Testing (UAT), Performance Testing, Security Testing, and Feedback Analysis and Optimization. Each phase has defined objectives, activities, and timelines. The schedule allows us to maintain a clear focus on project goals, adapt to feedback received during testing, and ensure comprehensive evaluation of the 'Hostel Initiative' system's functionality, performance, and security.

#### **6.2 Testing and Experiment Approaches**

In the development of the 'Hostel Initiative', a robust testing approach was essential to ensure the reliability and effectiveness of the service. The project, being a MERN stack application, demanded meticulous testing strategies for both the backend and frontend components.

**6.2.1. API Testing with Postman**: For backend testing, particularly for API validation, Postman was the primary tool. This versatile platform allowed us to simulate client requests and analyze responses from our server. We meticulously tested each API endpoint for various scenarios - including GET requests for room availability, POST requests for booking rooms, and PUT/DELETE requests for updating or canceling bookings. This ensured that all aspects of the server-side logic functioned as intended.

**6.2.2. Unit Testing with Jest**: For more granular testing, especially on the backend, Jest was employed. This popular JavaScript testing framework enabled us to write and execute unit tests that covered individual functions and components of the application. By mocking databases and HTTP requests, we could isolate and test specific pieces of code, ensuring each function performed its intended operation correctly. This level of testing was crucial for validating the business logic of the Hostel Initiative application.

**6.2.3. Integration Testing**: Beyond unit testing, integration tests were conducted to ensure that different parts of the application worked together seamlessly. This included testing the integration between the MongoDB database and the Express server, as well as the interaction between the React frontend and the Node.js backend.

**6.2.4. Performance Testing**: To assess the application's performance under various conditions, we conducted load testing. This helped us understand how the system behaved under high demand, ensuring that the Hostel Initiative could handle multiple simultaneous bookings without performance degradation.

**6.2.5. User Acceptance Testing (UAT)**: To ensure the application met the user requirements, User Acceptance Testing was conducted with a focus group from the university community. This provided us with valuable feedback on the user experience, usability, and overall functionality of the application.

**6.2.6. Continuous Testing and Integration**: Throughout the development process, we adopted a continuous testing approach, integrating testing into the development lifecycle. This not only helped in identifying issues early in the development process but also ensured that any changes or additions to the codebase did not disrupt existing functionalities.

The combined use of Postman and Jest, along with our comprehensive testing strategy, played a pivotal role in the successful implementation of the Hostel Initiative. This rigorous testing approach ensured that the final product was reliable, user-friendly, and met all functional requirements.

#### **6.3 Testing Report**

**6.3.1. Overview of Testing Approach**:  
In the 'Hostel Initiative' project, testing was a critical component to ensure the robustness and reliability of the system. The testing process was designed to validate every aspect of the application, from individual API endpoints to the overall user experience.

**6.3.2. API Testing with Postman**:  
Postman, a popular tool for API testing, was employed to rigorously test all API endpoints. This process involved:

1. Creating Collections: Grouping related API requests for agents, rooms, and bookings.
2. Testing CRUD Operations: Verifying the functionality of Create, Read, Update, and Delete operations for each endpoint.
3. Validation of Responses: Ensuring that each API request returned the correct status codes, headers, and response bodies.
4. Error Handling: Testing API behavior under various error conditions, such as invalid input or server errors.

**6.3.3. Automated Testing with Jest**:  
Jest was used to perform automated testing, focusing on both backend and frontend components.

1. Backend Testing:
   1. Unit Tests: Writing unit tests for individual functions and database models to validate their expected behavior.
   2. Integration Tests: Ensuring that different parts of the backend worked together seamlessly, particularly focusing on database interactions and middleware.
2. Frontend Testing:
   1. Component Tests: Testing React components in isolation to ensure they rendered correctly and handled props and state updates.
   2. User Interaction Tests: Simulating user actions and verifying the UI's response, such as form submissions and navigation.

**6.3.4. Testing Results and Observations**:  
The testing process uncovered several areas for improvement:

1. A few API endpoints required optimization to handle large data sets more efficiently.
2. Certain user interface components needed better error handling for a smoother user experience.
3. The integration tests helped identify some inconsistencies in the database schema which were rectified.

**6.3.5. Conclusion and Future Testing Plans**:  
Overall, the testing phase significantly enhanced the quality and performance of the 'Hostel Initiative' application. Future testing plans include:

1. Implementing Continuous Integration/Continuous Deployment (CI/CD) pipelines to automate testing and deployment.
2. Expanding test coverage to include more complex user scenarios and edge cases.
3. Exploring performance testing to ensure the application scales well under heavy load.

This comprehensive testing approach has been instrumental in validating the functionality and reliability of the 'Hostel Initiative' project, setting a solid foundation for its deployment and future development.

**Chapter 7 Conclusion and Future Work**

**7.1 Project summary**

After concluding the final stages of this project, we were able to implement the basic functionalities of the service, including user registration and account management for administrators and basic users like customer representatives, other service workers, or students. We also completed the features regarding the actual housing units, such as enabling the upload, editing, and viewing of the different accommodations. While we were able to complete the basic services for registration and housing marketplace exchange, we ended up not having enough time to provide the other features such as giving the users a way to chat with each other.

In the end, our experience working on this project proved valuable. The whole ordeal involved a lot of learning. Through it all, we learned how to conduct market research and analysis before deciding to create a product to enter the market. We also learned how to define the system requirements and how to map them to the design and implementation because different requirements need different designs. It proved difficult to go through this process because we lacked much experience in system design, but with a lot of trial and error, we were able to push past the process. It was especially difficult during the implementation and testing phase because some of the technologies were unfamiliar to us, so we had to take time to process how they worked and understand the concepts.

**7.2 Future work**

For the future, work can be done to improve the project by adding a chatting feature to allow users to discuss the accommodations. It would also be helpful for fellow residents to share information with each other regarding current events, or for administrators to send announcements or other important information to tenants. Additionally, the project could benefit from using services such as AWS to improve performance and scale up even more. A third feature that could be worked on in the future is the ability to access the nearest available grocery market for a housing unit. It would make residential life in the housing accommodations much more convenient if a tenant could have their groceries delivered to their unit.

**References**

“AppFolio - Market Share, Competitor Insights in Property Management.” 6sense, 6sense.com/tech/property-management/appfolio-market-share#

Falter, B., Munoz-Martinez, D., Sexton, V. (2021). “SJSU’s Basic Needs Survey - Spring 2021 Results.” SJSU Cares. https://www.sjsu.edu/sjsucares/about/data.php

Siwale, Maka (2022) "Applying Technology Acceptance Model to measure online student residential management software acceptance," Journal of International Technology and Information Management: Vol. 31: Iss. 2, Article 2.