EE 547 - B. Franzke - Spring 2023

EE 547 – Applied and Cloud Computing

Revised Project Proposal

[Group 3] Maitreyee Likhite (mlikhite@usc.edu), Naman Rajendra Joshi (namanraj@usc.edu), Sanskar Tewatia (stewatia@usc.edu)

April 14, 2023

Project Title: **Housing Management Application**

Summary and Description

The Housing Management App is a full-stack web application designed for property managers, landlords, and tenants to manage their property and tenancy-related tasks in a streamlined manner. The application consists of a Node.js backend, HTML/CSS frontend, GraphQL for API calls, and MongoDB for database management.

The application will have several features, including:

1) About - showcasing photos of the property, floorplans, amenities, etc.
2) Contact Us - a form where prospective tenants can enter their contact information
3) Login - where users can log in as tenants or as an administrator
4) Payments - where tenants can log their payments, view their account ledger, check their balance sheet
5) Maintenance Requests with Live Updates - where tenants can submit maintenance requests and track their status continuously on-the-go
6) Event Scheduler - where tenants can book community halls or sports rooms for events
7) Notifications - where tenants can receive updates from the administrator about scheduled inspections, cleaning, fumigations, etc.
8) Package Tracking - packages ordered by the tenants can be tracked by the housing management administrator to update estimated delivery dates and notify delivery alerts
9) Roommate Matching - prospective/current tenants can fill a roommate matching form and the system will group users with similar preferences into flatmate groups

The application is designed to simplify and streamline the process of property management by providing a central hub for communication, payments, and maintenance requests. The application also provides important notifications to keep tenants informed about scheduled maintenance, inspections, and other important events. Overall, the Housing Management Application provides an efficient and user-friendly platform for property managers and tenants to manage their property and tenancy-related tasks.

EE 547 - B. Franzke - Spring 2023

1 Proposed Architecture

Our project will consist of a front-end web-application, a backend web server, and resources to asynchronously manage communication between the user/client (a customer browsing for leasing an accommodation in the particular housing facility, or a tenant on lease residing there currently) and the housing management administrator (an employee of the housing facility's management office, responsible for responding and providing information or service for leasing, maintenance, or other management tasks).

We will create the backend server in Node.js and use GraphQL for the API. The users will interact with the API through a front-end client (web-page). We expect the app to include at least the following web pages. We will generate these pages as a combination of static assets and dynamic content using React.

- **About** - For this page, HTML and CSS can be used to design and layout the page. The page will showcase photos of the property, floorplans, amenities, etc.
- **Contact Us** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js can be used to handle the form submission and store the contact information in the MongoDB database. The page will comprise a form where prospective tenants can enter their contact information, such as name, telephone number, and email address.
- **Login** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to handle user authentication and authorization, using a username and password as credentials.
- **Payments** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to handle payment processing, and account management for adding payment methods and storing previous payment statements. Stripe API can be used for payment processing.
- **Maintenance Requests with Live Updates** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to handle the maintenance issue request submission and update the maintenance request status in the MongoDB database. The request can be a description of the issue from the tenant.
- **Event Scheduler** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to handle the event booking and update the event schedule in the MongoDB database. The booking will include details such as date, time slot, and description of the event arrangements required.
- **Notifications** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to send notifications to current tenants about scheduled routine inspections, cleaning, fumigation, pest-control service, etc. Twilio or SendGrid API can be used for sending SMS or email notifications.

EE 547 - B. Franzke - Spring 2023

- **Package Tracking** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to handle the tracking of tenants' packages by the housing management administrator to update estimated delivery dates and notify delivery alerts.
- **Roommate Matching** - For this page, HTML and CSS can be used to design the page, and a backend server with Node.js and GraphQL can be used to allow prospective/current tenants to fill a roommate matching form and the system will group users with similar preferences into flatmate groups.

Figure 1 below depicts the process flow of the housing management application.

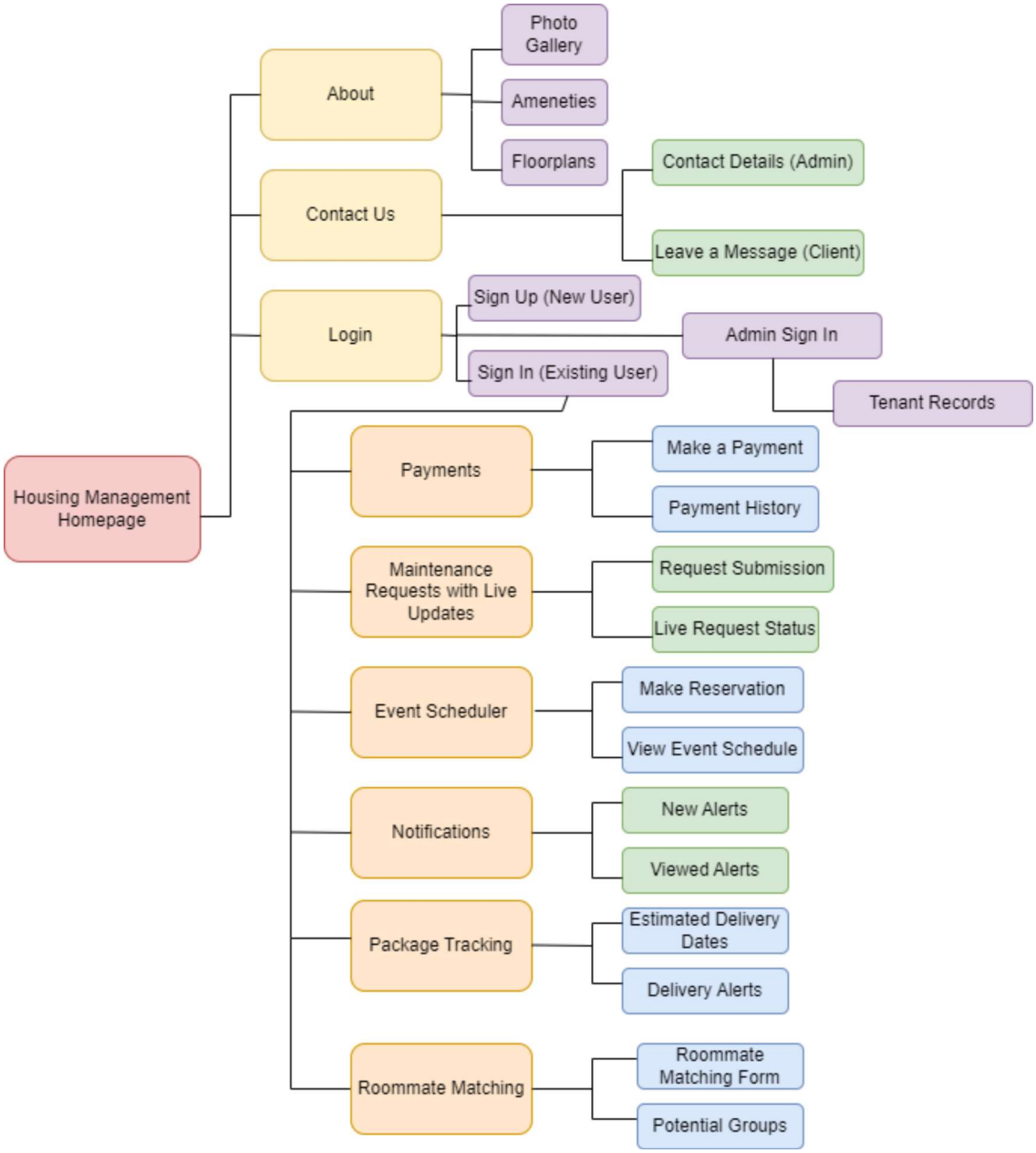

Figure 1: Housing Management Application Workflow

EE 547 - B. Franzke - Spring 2023

2 Likely Outcome and Expected Results

Likely outcome -

- Functionality: The application should function according to the design, including features such as user authentication, payments, maintenance requests, event scheduling, chat functionality, and notifications
- User experience: The application should provide users with an intuitive, user-friendly experience that is responsive, visually appealing, and easy to navigate
- Security: The application should be secure and protect users' personal and financial information. It should have measures to prevent unauthorized access, such as user authentication(encryption) and secure connections.
- Performance: The application should be quick and responsive, not very very expensive computationally.

Possible reasons/aspects of failure -

- Integration issues
- Insufficient resources
- Data Corruption or issues with the database
- Security vulnerabilities

3 References, Tutorials, Codebases, Documentation, and Libraries

1) Node.js:
    a) Documentation: https://nodejs.org/en/docs/
    b) Tutorial: https://www.w3schools.com/nodejs/
2) MongoDB:
    a) Official Website: https://www.mongodb.com/
    b) Documentation: https://docs.mongodb.com/
    c) Tutorial: https://www.w3schools.com/nodejs/nodejs_mongodb.asp
3) GraphQL:
    a) Official Website: https://graphql.org/
    b) Documentation: https://graphql.org/learn/
    c) Tutorial: https://www.howtographql.com/
4) HTML/CSS:
    a) Tutorial: https://www.w3schools.com/html/default.asp
    b) Tutorial: https://www.w3schools.com/css/default.asp

4 Estimated Compute Needs

Our Application can be deployed on AWS using EC2 instances. We can use t2.micro which has 1 vCPU, 1 GB of RAM, and moderate network performance.

EE 547 - B. Franzke - Spring 2023

5 Team Roles

The following is the rough breakdown of roles and responsibilities that we plan for our team:

- Naman: Primary: Responsible for developing the server-side code of the application using Node.js and GraphQL. Secondary: Handle server administration tasks such as setting up and configuring the server, deploying the application.
- Sanskar: Primary: Responsible for developing the client-side code of the application using HTML, CSS, and JavaScript. Secondary: Responsible for designing the user interface and ensuring a consistent and intuitive user experience throughout the application.
- Maitreyee: Primary: Responsible for testing the application, ensuring quality assurance, and deploying the application to AWS. Secondary: Assist with backend development (Node.js and GraphQL).

These responsibilities are subject to change. All team members will work on the final presentation, slides, and report.