TRIBHUVAN

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Project Proposal

On

The Dorm Den: A MERN Stack Hostel Web Application

Submitted By:

Arahanta Pokhrel (THA076BCT009)

Bikrant Bidari (THA076BCT013)

Sameer Shrestha (THA076BCT039)

Sudeep Kaucha (THA076BCT044)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu Nepal

February, 2023

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Minor Project Mid-Term Report

On

The Dorm Den: A MERN Stack Hostel Web Application

Submitted By:

Arahanta Pokhrel (THA076BCT009)

Bikrant Bidari (THA076BCT013)

Sameer Shrestha (THA076BCT039)

Sudeep Kaucha (THA076BCT044)

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Computer Technology.

Under the Supervision of

Er. Praches Acharya

February, 2023

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the Institute of Engineering, Tribhuvan University for the inclusion of minor project during Bachelor's in Computer Technology.

We would like to extend our sincerest gratitude to Mr. Praches Acharya for unwavering support and guidance throughout our project. His invaluable insights and direction have been instrumental in helping us doing the project successfully. We are also thankful to department for their guidance and support throughout this project. Their valuable insights and feedback were instrumental in helping us to complete this report to the best of our abilities.

Finally, we would like to acknowledge the support and encouragement of our family and friends, who have always believed in us and supported us in some tough decision along the way.

Thank you all for your contributions to this project.

Sincerely,

Arahanta Pokhrel (THA076BCT009)

Bikrant Bidari (THA076BCT0013)

Sameer Shrestha (THA076BCT039)

Sudeep Kaucha(THA076BCT044)

ABSTRACT

Every year there is a huge influx of students in Kathmandu valley from all over the country for their higher studies, as such, most of them are seeking a place to accommodate. For students, hostels or dormitory are best choice as they can get place to stay as well as daily foods. So, hostels are one of the basic infrastructures for our students who are the backbone and future of our country Even after all these years, finding a proper and clean hostel is hassle for students so our project The Dorm Den seeks to be a one stop solution to find a suitable hostels for all the students. The Genuity of information and reviews will be guaranteed in our application since only verified owner can add info and only verified students will be able to leave a review (thus frauds and fake reviews can be avoided). We will be using MERN stack for the development of our web application, which will have an efficient, user friendly and attractive interface to interact with the data. Our project will not only be a way for student to find a good place to stay but also a good portal for hostel owners to highlight their business.

Keywords: Express, Hostel, MongoDB, NodeJS, REACT.

Table of Contents

ACKNOWLEDGEMENT i

ABSTRACT ii

List of Figures v

List of Tables vi

LIST OF ABBREVIATIONS vii

1 INTRODUCTION 1

1.1 Background 1

1.2 Motivation 1

1.3 Problem Definition 2

1.4 Objectives 2

1.5 Scope and Applications 3

2 LITERATURE REVIEW 4

2.1 Previous works 4

2.2 Recommendation of hostels 4

2.3 Places autocomplete and Geo encoding 5

2.4 Geospatial Information Processing in MongoDB database 6

3 REQUIREMENT ANALYSIS 7

3.1 Project Requirements 7

3.1.1 Hardware Requirements 7

3.1.2 Software Requirements 7

3.2 Feasibility Analysis: 7

4 METHODOLOGY 9

4.1 System Block Diagram 9

4.1.1 React as Frontend 10

4.1.2 MongoDB 10

4.1.3 Express 10

4.2 Use Case Diagram 11

4.3 Entity Relation (ER) Diagram 12

4.4 Description of System Development 13

4.4.1 Data Collection 13

4.4.2 UI design 13
4.4.3 Host Database 13
4.4.4 Develop Frontend 13
4.4.5 Test and Deployment 13
5 IMPLEMENTATION DETAILS 15
5.1 Geospatial Datatype 15
5.1.1 Introduction: 15
5.1.2 Implementation detail: 15
5.2 Pigeonmap API: 16
6 RESULTS AND ANALYSIS 18
6.1 Features and Functionality 18
6.1.1 Backend 18
6.1.2 Frontend 25
6.2 Performance Analysis 26
6.3 Security Analysis 26
7 REMAINING TASKS 27
8 REFERENCES 28
APPENDICES 29
Project Schedule 29
Project Budget 29

List of Figures

Figure 1 System Block Diagram......9
Figure 2 Use Case Diagram......11
Figure 3 Entity Relation Diagram......12
Figure 4 : Register a user......18
Figure 5: Login User Received......20
Figure 6: Reset token sent......21
Figure 7: Token received in mail......21
Figure 8 : Setting new password......22
Figure 9: Register hostel......23
Figure 10: Review Hostel......24
Figure 11 : Registration Page......25
Figure 12: Register Hostel Page......25
Figure 13 : Search Page......26

List of Tables

Table 1: Project Schedule 29
Table 2: Project Budget 29

LIST OF ABBREVIATIONS

BDUF	Big Design Upfront
ERD	Entity Relation Diagram
HCI	Human Computer Interaction
MERN	MongoDB Express React Node
RDMS	Relational Database Management System
SSPL	Server-Side Public License
UCD	User-centered Design
UI	User Interface

1 INTRODUCTION

1.1 Background

Altogether there are 1,440 campuses and 460,826 students currently in Nepal as per data collected in year 2077/78 [1] where majority of students prefer to choose to go away from home to study at a larger university as large universities often offer a wider range of academic programs and extracurricular activities than smaller colleges, which can provide students with more opportunities to explore their interests and develop new skills. Attending a large and well-known university can be a prestigious accomplishment, and may open future opportunities such as scholarships, internships, and job offers.

Living in a hostel can be a good option for these students who are looking for a cost-effective, social, and convenient place to live while studying rather than renting a place, room, or dorm.

Our application, a hostel booking app aims to help users to find hostels in their destination and book a bed or a room quickly and easily, without the need to call or visit the hostel in person. This can save students time and effort and make their stay more comfortable and manageable. On top of that, we also are aiming to provide hostels with a convenient way to manage their reservations and occupancy, and potentially attract more guests through our application.

1.2 Motivation

Motivation for our application comes from several factors, one possible motivation is the need to make it easier for people to find and book accommodations without the need to call or visit the hostel in person.

Hostels are generally cheaper than renting a place, especially if the student is sharing a room with others. This can make hostels a more affordable option for students who are on a tight budget. Using our application, students can save their time and effort and use their resources more effectively. Hostels are often located in central areas, making them a convenient option for students who want to be close to campus. So, searching suitable places near to central areas becomes more easier for students, filtering through seas of hostels using our application. Students may prefer additional services and amenities, such as a shared kitchen, laundry facilities, and internet access provided by hostels which can be easily skimmed through by filtering hostels based on features and amenities. Hostels also typically offer a variety of room types, such as dorms, private rooms, which can accommodate different budgets and preferences of students in contrast to renting a place.

1.3 Problem Definition

When booking a hostel offline, it can be difficult to access detailed information about the hostel, such as its location, amenities, and policies. This can make it challenging for students to compare different hostels and make an informed decision about where to stay.

Hostels can fill up quickly, especially during peak seasons. When booking offline, it may be difficult to determine the availability of beds or rooms at the hostel, and students may have to call or visit the hostel in person to check availability and make a reservation.

An online hostel booking application like ours can help alleviate these challenges by providing users with easy access to detailed information about hostels and their availability and allowing them to make reservations quickly and easily.

1.4 Objectives

- Provide a convenient and efficient way for students to find and book hostels.

- Offer detailed information about hostels, including their location, amenities, policies, and availability.

- Offer hostels with a simple and effective way to manage their reservations and occupancy.

1.5 Scope and Applications

Hostel Booking application like ours can be used by students for faster and efficient finding and booking hostels in various locations, and by hostels to manage their reservations and occupancy.

The application provides detailed information about hostels, including their location, amenities, policies, and availability. Users could search for hostels based on their location, budget, and other criteria, and compare different options to find the best hostel for their needs.

Hostels could use the application to manage their reservations and occupancy, and to update information about their hostel, such as the availability of beds or rooms, policies, and amenities. This could help hostels attract more guests and manage their reservations more efficiently.

The scope and application of a hostel booking application would be to provide students and hostels with an easy and convenient way to find and book accommodations.

2 LITERATURE REVIEW

2.1 Previous works

There have not been many previous works done in the field of hostel portal for students in Nepal, but something like our work can be seen in many Properties sales site or Hotel booking websites.

Previous work on the MERN stack has primarily focused on creating scalable and efficient web applications with a user-friendly interface. One such project is a hotel booking website developed using the MERN stack (from official React examples), which provides a platform for booking hotel rooms online. The website was developed using MongoDB for database management, Express for server-side routing, React for the front-end interface, and Node.js for server-side scripting. The website was well received by users, who appreciated its user-friendly interface, fast loading times, and the ability to book rooms quickly and easily. ("Hotel Booking App using the MERN Stack with TypeScript & Redux," ReactJSExample.

We found the best match possible to our project with Nepal Homes a large-scale property sales site from PropTech Nepal Private Limited, it is very user friendly, good optimization for site loading and proper data management. It also provided location search facilities and sorting on based on price, which we had to replicate in our project too. They seem to be using React as frontend, but we could not find about their entire stack as it was not publicly available and we did not get any response from them either. But we could learn about proper system flow through their websites.

2.2 Recommendation of hostels

For unbiased recommendation of hostels in our project we will be using reviews of authenticated students' feedback (authentication through user ID), but doing so we would face a major hurdle i.e., sparse number of reviews/feedbacks in the beginning.

For example, if hostel A had 100 reviews, among them 90 are 5 stars and remaining are 4 stars (making their average less than 5) and hostel B had 5 reviews, all being a 5-star rating (making their average exactly 5 stars). This would pose a huge problem when sorting based on reviews.

To overcome this, we will be using true Bayesian estimate, which has been previously used by IMBD to make their top 250 movies list. [2]

Weighted rating (WR) = (v / (v+m)) * R + (m / (v+m) *c

Which can be mathematically simplified to:

Rating = (R *v + C * m) / (v +m)

Where the meaning of symbols is,

R = average for the movie (mean)

v = number of votes for the movie

m = minimum votes required to be listed in the Top N (This can be thought as tuning parameter)

C = the mean vote across the whole report (The average rating across the whole database)

Using this we would create a ranking list for every hostel in the database and use that for unbiased recommendation.

2.3 Places autocomplete and Geo encoding

For projects like this where (work is done based on location) place autocomplete and Geo encoding is done to get proper data to work on.

Location autocomplete is an API (Application Programming Interface) service that suggest users location based on real time input of the users, this will help to correct any spelling mistakes and provide the correct location to the application.

And for the location-based search we need exact coordinates of the searched location, for that Geo encoding was used in previous projects. This takes the name of street, city, state, or country and returns a coordinate for that spot, which is then passed to the geospatial library so that it can filter all the objects using this location as a reference point.

2.4 **Geospatial Information Processing in MongoDB database**

The amount of personal location data is forecast to increase by 20% every year, and location-aware information occupies a substantial proportion of the data generated every day: 2.5 quintillion bytes [3] . How to store, manage and query geospatial data has been the focus of research and problems that must be solved.

New geospatial application requires more flexible data schema, a faster query response time, and more elastic scalability than traditional spatial relational database currently have. The most communal problem occurrence has been seen when streaming requests from client to servers suddenly increase, which might cause response delay or even server unavailability. To solve this very scalability problem, a scalable framework was proposed back in MongoDB to implement elastic deployment for geospatial information sharing with the client users growing in number [4]. In this framework, MongoDB is chosen because it is a distributed database and supports a flexible storage schema suitable for massive map tile storage.

Several studies have found that RDBMS have some disadvantage in terms of big data storage and query in some specific areas, such as in streaming request or large data access environment in geospatial applications [5] .

MongoDB provides two types of geospatial index types: geohash for 2d sphere and 2d. Within 2d sphere relevant queries are implemented through the calculation of index on Earth-like 2d sphere. Within 2d index, queries are calculated through a calculation of geometries on a two-dimensional plane. Four topological query operations are provided in MongoDB for geospatial data: $geoIntersects, $geoWithin, $near, and $nearSphere.

In a quantitative comparison of geospatial big data processing between the PostGIS and MongoDB databases, MongoDB had some advantages with its “within” and “intersection” queries and in terms of its response time for loading big geospatial data. [6]

3 REQUIREMENT ANALYSIS

3.1 Project Requirements

3.1.1 Hardware Requirements

The hardware requirements for a hostel searching website are minimal and can easily be met by a standard laptop or desktop computer. A reliable internet connection is necessary, but otherwise the website does not require any specialized hardware. It can be listed as

• A computer (laptop or desktop) with sufficient processing power, memory, and storage to run the web browser and support the application.

3.1.2 Software Requirements

We need following software to accomplish our project

- Visual Studio Code: A free and open-source code editor
- Postman: An API development and testing tool.
- Figma: Popular UI design and collaboration tool.
- MongoDB: A cross-platform document database
- Express: A back-end Web application framework
- React: A JavaScript library for building user interface
- Node.js: A cross-platform JavaScript runtime environment
- Web browser for accessing the website
- An operating system that support web browser

3.2 Feasibility Analysis:

- Financial feasibility:

All the libraries we need for the completion of this project are open source. So, the expenditure is minimal for this project. We have to pay for server hosting and acquiring domain name. Also, we have all hardware requirements already so we do not have to buy any additional hardware equipment for the completion of our project.

- Technical feasibility:
  Currently we have a laptop with the following specifications:

Processor: Intel Core i5-8250U @ 1.6GHz

RAM:8 GB

SSD:256 GB

HDD:1 TB

Graphics: NVIDIA, MX130 ,2GB

Network: We have a fast and stable internet connection in our laptop.

For software requirements we have installed all the necessary packages required for the completion of our project

- Operational feasibility:

Security: Measures to protect sensitive data and prevent unauthorized access to the application.

User Acceptance: User acceptance of the application based on ease of use, performance, and functional requirements.

Maintenance: Cost and effort required to maintain and upgrade the application over time.

Integration: Ability to integrate with existing systems and data sources.

Scalability: Ability to handle increased demand for the application.

Reliability: Ability of the application to perform consistently and without failures.

4 METHODOLOGY

4.1 System Block Diagram

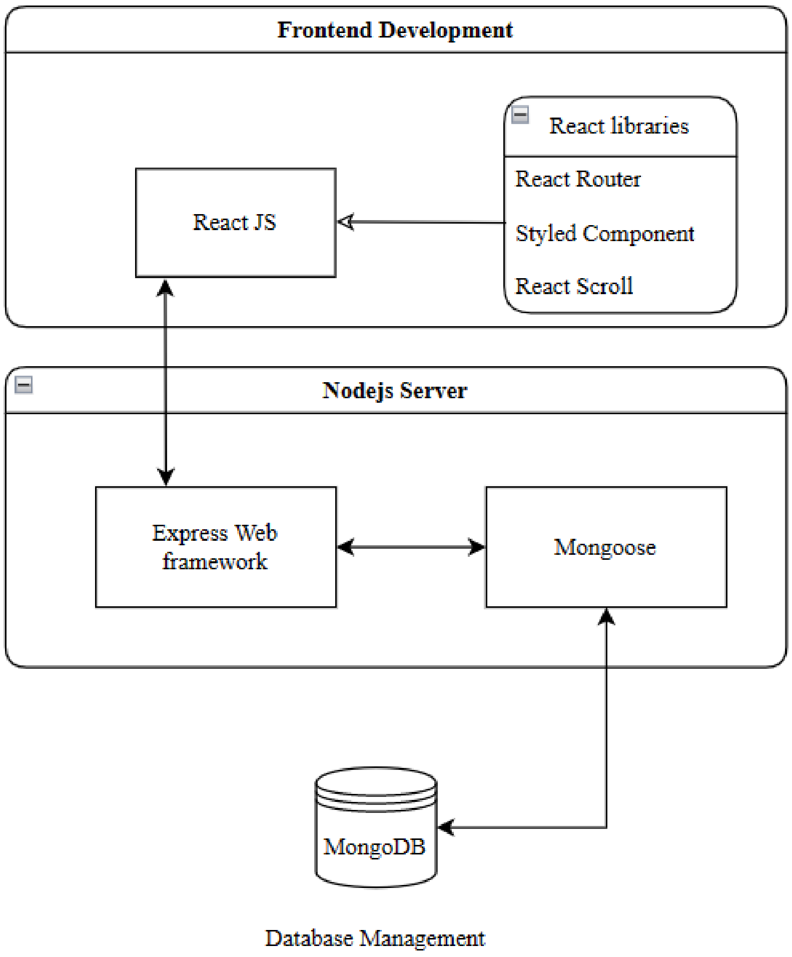


Figure 1 System Block Diagram

We will be using MERN stack for this web application, it stands for MongoDB Express React Nodejs. Our Data would be hosted in mongo, which is a NoSQL database system. These data are presented in an attractive and efficient manner through react library which is essentially a Frontend oriented script. Express will work as a bridge between our frontend and backend development, something like a waiter who works between a customer and cooks.

4.1.1 React as Frontend

React.js, more commonly known as React, is a free, open-source JavaScript library. It works best to build user interfaces by combining sections of code (components) into full websites. Originally built by Facebook, Meta and the open-source community now maintain it. We are using React mainly because of its component-based system which Is reusable and easy to maintain. Also React uses virtual DOM which is effectively faster than regular DOM. So, there will improvement in apps performance.

4.1.2 MongoDB

MongoDB is an open-source document-oriented database. The term 'NoSQL' means 'non-relational' i.e., it is not based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data. This format of storage is called BSON (like JSON format). We choose Mongo over relational database because of its flexibility and easy to handle features, also they are easy to expand since they are Schema-less (Adding new feature does not affect old documents and is extremely easy)

4.1.3 Express

Express is a fast and moderate web framework of Node.js. Express can be seen as a layer built on the top of the Node.js that helps manage a server and routes. It was created to make APIs and web development with ease and still is very efficient.

The main reason for choosing this framework is because it is very beginner friendly, Time efficient, fast, and economical.

4.2 Use Case Diagram

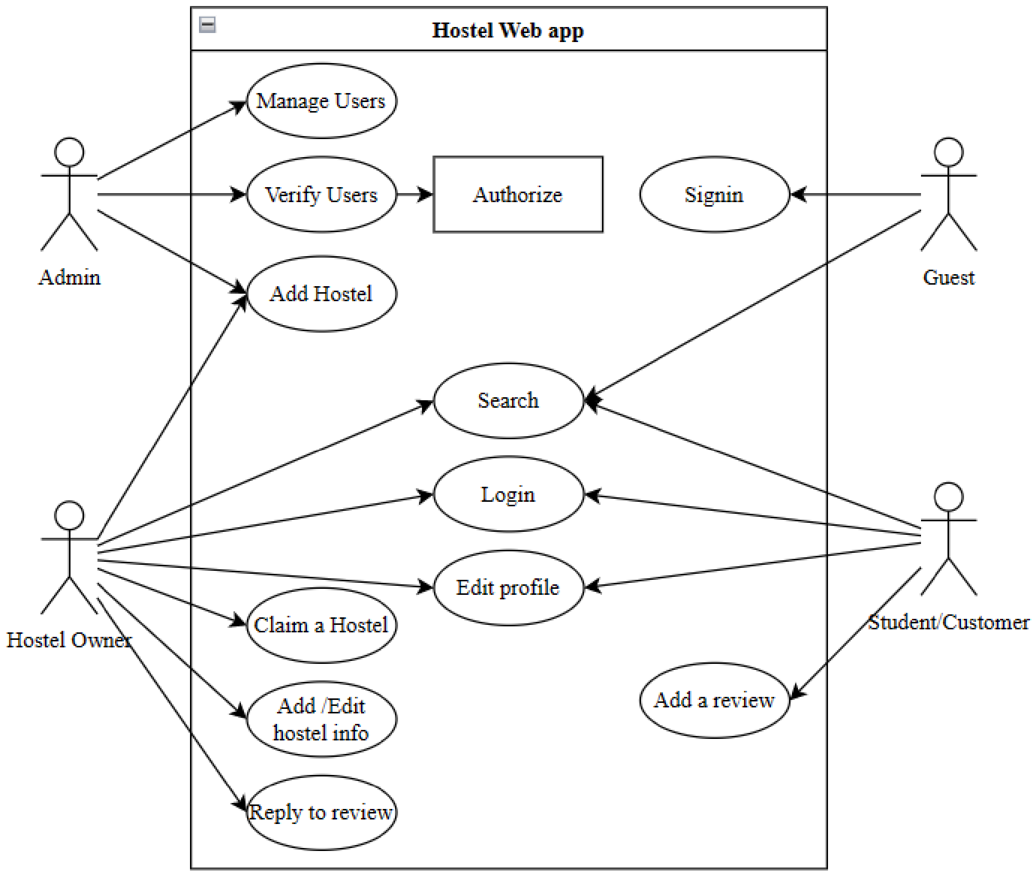


Figure 2 Use Case Diagram

The system's use case diagram, which represents the actors and the system, is depicted in the diagram above. The system's service to the actors is depicted in this diagram. In the system, there are four types of actors: Admins, Guest, Hostel owners and Registered Users. A registered user will be able to search for hostels, write reviews, rate them, and use the recommendation and suggestion services. The user can also make changes to their account. The system administrator will oversee Verifying user documents and moderating any contents as well as add new hostels. Hostel owners can request to add a new hostel or claim one with proper documentation. Guest users are only allowed to search hostels and watch their reviews but cannot give review of their own.

4.3 Entity Relation (ER) Diagram

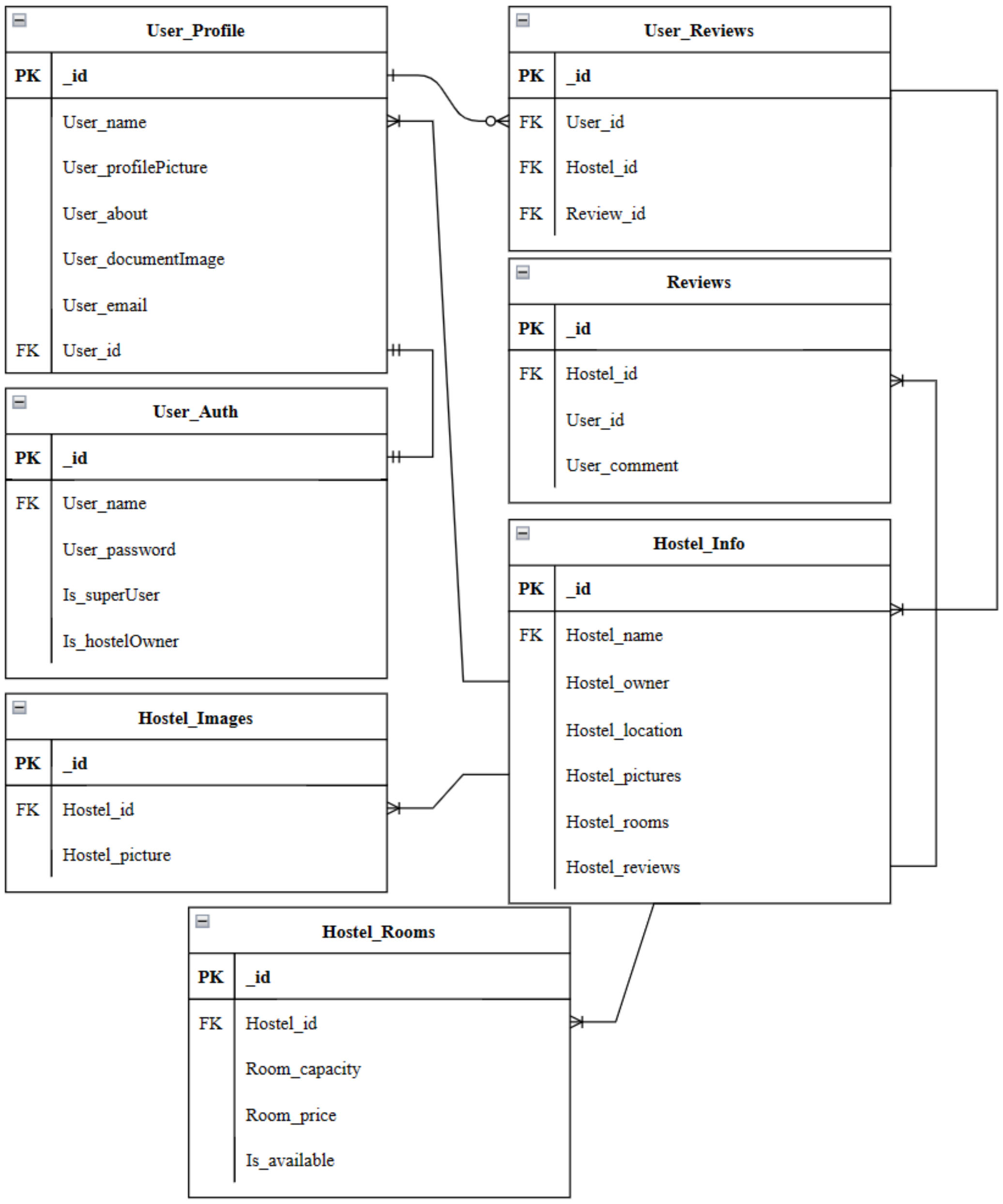


Figure 3 Entity Relation Diagram

ER diagram (entity-relationship diagram) is a diagram which shows entities, attributes of entities and relationship between them. ER diagram of our system is as shown above.

4.4 Description of System Development

4.4.1 Data Collection

For our hostel dataset we will visit different hostel and interview the owner for the required information. This method is effective but quite time consuming and tedious so we would collect data from other sources like google map scrapping as well as data from Nepal hostel association. We want the data to be as relevant as it can be and correct too, this will help enhance user experience in our app and will also fulfill their needs.

4.4.2 UI design

The user interface (UI) is the space where interactions between human computer system occur. UI is an integral aspect of user experience that mainly consist of two major thing visual design (which conveys the look and feel of a product) and interaction design (which is the functional and logical organization of elements). The goal of UI design is to make an interface which is user friendly, attractive, and efficient to interact, for this we will be using Figma.

4.4.3 Host Database

All the user interfaces would be useless if there is no data to serve to the user, so we will host the data collected in a proper format as shown in ER diagram above, for this we will be using mongoose and the data would be stored in cloud (Mongo Atlas). This will be managed and routed with the help of express (that is any http request to the database would be handled by express).

4.4.4 Develop Frontend

We need to properly code the designed UI, we will do this with the help of REACT along with other NPM react packages like React-Router (for routing), React-Spring, React-Scroll etc. (for animation). In React we can create many reusable components which when combined can form web pages.

4.4.5 Test and Deployment

The final step of application development is testing and deployment, before deploying the app for general use, we would first test the app for any bugs and error or any irregular behavior and finally deploy it. For deployment we will be using Azure cloud service and get a custom domain name for our website

5 IMPLEMENTATION DETAILS

5.1 Geospatial Datatype

5.1.1 Introduction:

Geospatial data type of MongoDB and Mongoose is a powerful feature that enables developers to store and query location-based data. In this report, we will discuss the implementation of the geospatial data type in our hostel booking application to find nearby places using latitude and longitude.

5.1.2 Implementation detail:

- Store location data: To store the location data for each hostel, we created a MongoDB schema that stores the latitude and longitude for each hostel. This data was stored as a geospatial data type, allowing us to query it efficiently. The schema for location data is implemented as:

```
location: {
    type: {
      type: String,
      enum: ['Point']
      required: true
    },
    coordinates: {
      type: [Number],
      required: true
    }
  }
```

- Use Mongoose to interact with MongoDB: To interact with MongoDB, we used the Mongoose library, which provides a convenient and powerful way to interact with MongoDB collections from backend of our application.

- Implement a find: To find nearby places, we implemented a find feature in the hostel booking application that queries the MongoDB collection for hostels within a certain distance from a given latitude and longitude. The query is performed using the MongoDB geospatial operator $near. The snippet of the find feature is implemented as:

```
Hostel.find({
  location: {
    $near: {
      $geometry: {
        type: "Point",
        coordinates: [longitude, latitude],
      },
      $maxDistance: 10000, // this searches for hostel on a
proximity radius of 10km
    },
  },
});
```

- Display nearby places: The query returns a list of hostels within the specified distance, sorted by proximity to the given latitude and longitude. We used this list to display the nearby hostels to the user, allowing them to select a hostel for booking.

The query returns a list of hostels within the specified distance, sorted by proximity to the given latitude and longitude. We used this list to display the nearby hostels to the user, allowing them to select a hostel for booking.

5.2 Pigeonmap API:

Pigeonmap is a JavaScript library that provides a simple and intuitive way to display maps and markers on a web page. The API integrates with popular mapping services such as OpenStreetMap and Leaflet to provide a user-friendly interface for adding and customizing markers on a map.

With Pigeonmap, users can add markers to a map with just a few lines of code. Markers can be customized to include information such as title, description, image, and more. The API also supports adding popups to markers, which can be used to display additional information or multimedia content.

Pigeonmap is designed to be lightweight and fast, making it suitable for use in a variety of applications such as real estate listings, store locations, event locations, and more. It supports a wide range of customization options, including different marker styles and colors, custom icons, and the ability to control the zoom level and center of the map.

6 RESULTS AND ANALYSIS

6.1 Features and Functionality

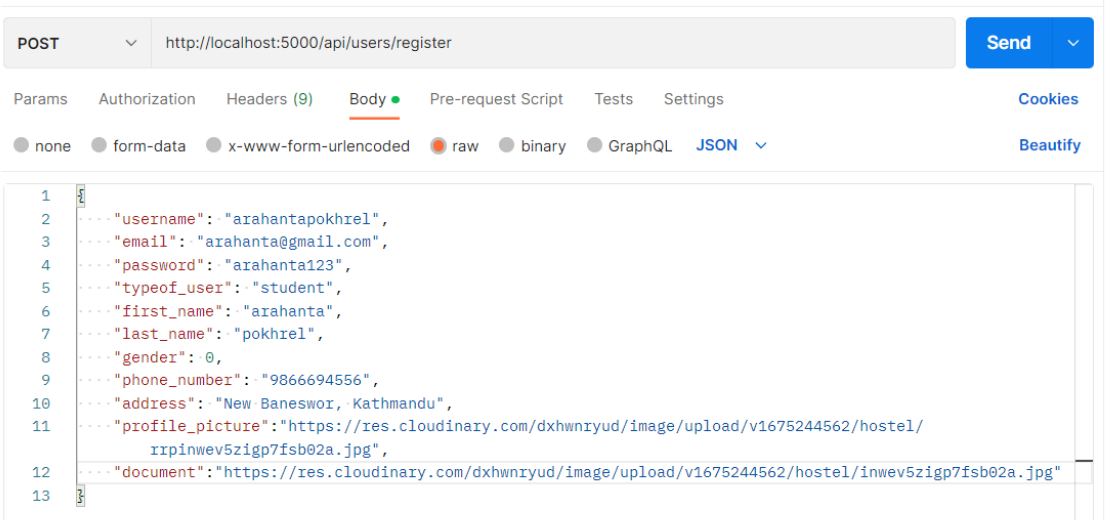
Some of the features and functionalities achieved till now are:

6.1.1 Backend

In our first half of the project, we focused on our backend functionality of website. From authentication using JWT, saving the tokes in cookies, role managing, different server calls for user/hostel registration, updating in database, working with cloudnairy for cloud photo storage (for optimization). We also integrated different APIs for better user friendliness like places autocomplete. We used pigeon map, which is an open-source map APIs for our location-based search and geoapify for GeoEncoding.
We are yet to fully integrate our backend with frontend, so we have shown our request call and response from server using Postman.

We have explained some of the basic functionality implemented in backend down below:

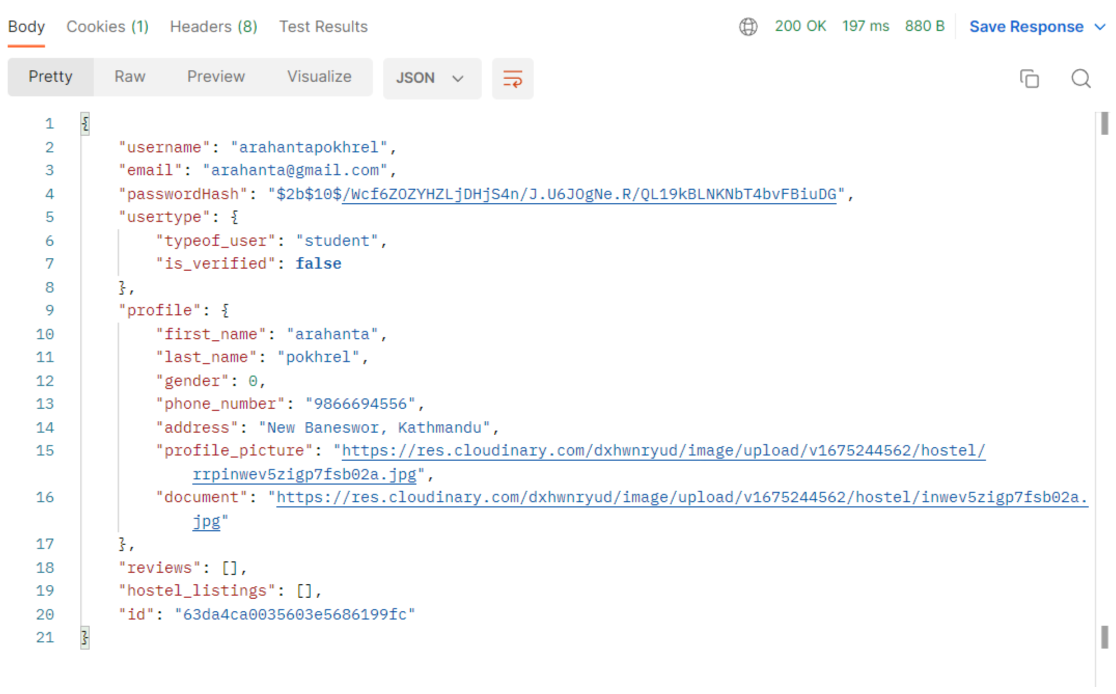
Register a user:



```
http://localhost:5000/api/users/register
```

```
{
    "username": "arahantapokhrel",
    "email": "arahanta@gmail.com",
    "password": "arahanta123",
    "typeof_user": "student",
    "first_name": "arahanta",
    "last_name": "pokhrel",
    "gender": 0,
    "phone_number": "9866694556",
    "address": "New Baneswor, Kathmandu",
    "profile_picture":"https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/
        rrpinwev5zigp7fsb02a.jpg",
    "document":"https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/inwev5zigp7fsb02a.jpg"
}
```

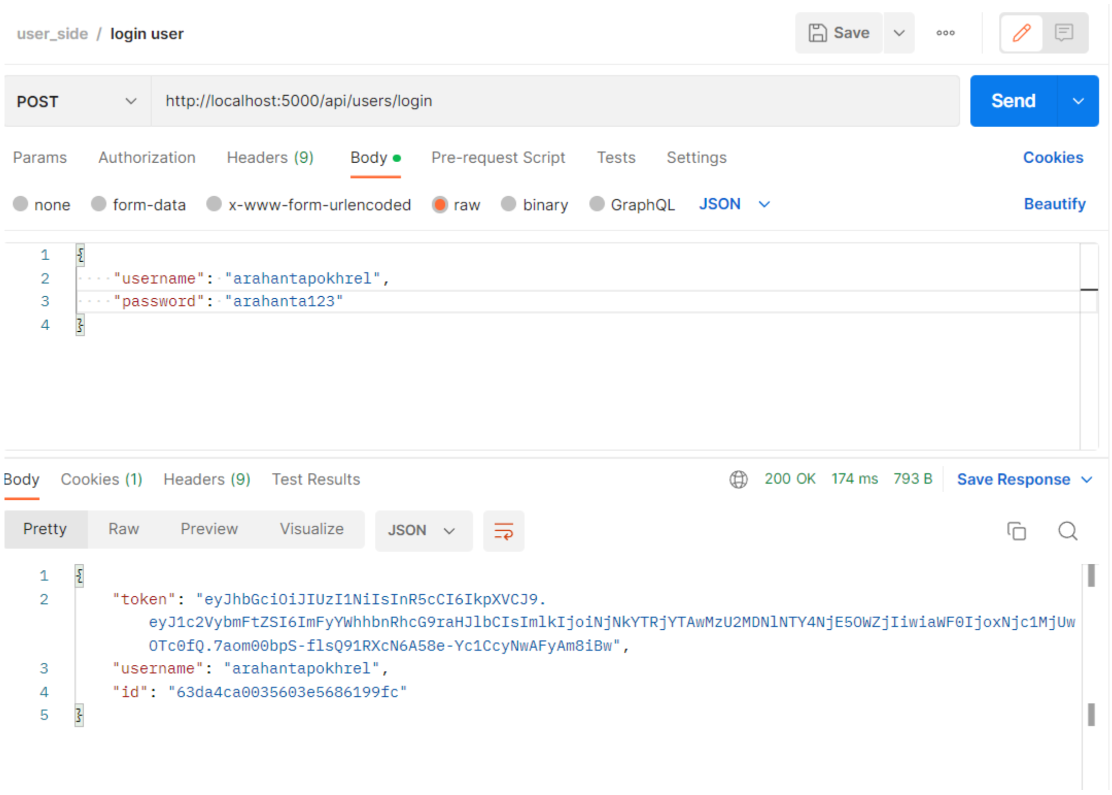
Figure 4 : Register a user



```
{
    "username": "arahantapokhrel",
    "email": "arahanta@gmail.com",
    "passwordHash": "$2b$10$/Wcf6ZOZYHZLjDHjS4n/J.U6JOgNe.R/QL19kBLNKNbT4bvFBiuDG",
    "usertype": {
        "typeof_user": "student",
        "is_verified": false
    },
    "profile": {
        "first_name": "arahanta",
        "last_name": "pokhrel",
        "gender": 0,
        "phone_number": "9866694556",
        "address": "New Baneswor, Kathmandu",
        "profile_picture": "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/rrpinwev5zigp7fsb02a.jpg",
        "document": "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/inwev5zigp7fsb02a.jpg"
    },
    "reviews": [],
    "hostel_listings": [],
    "id": "63da4ca0035603e5686199fc"
}
```

To register user, we make a request to server, where we entered username, password, and other related user info, then Server receives data in key-value pairs, and saves them in database in proper format.

LoginUser:



user_side / login user

POST http://localhost:5000/api/users/login

```
{
    "username": "arahantapokhrel",
    "password": "arahanta123"
}
```

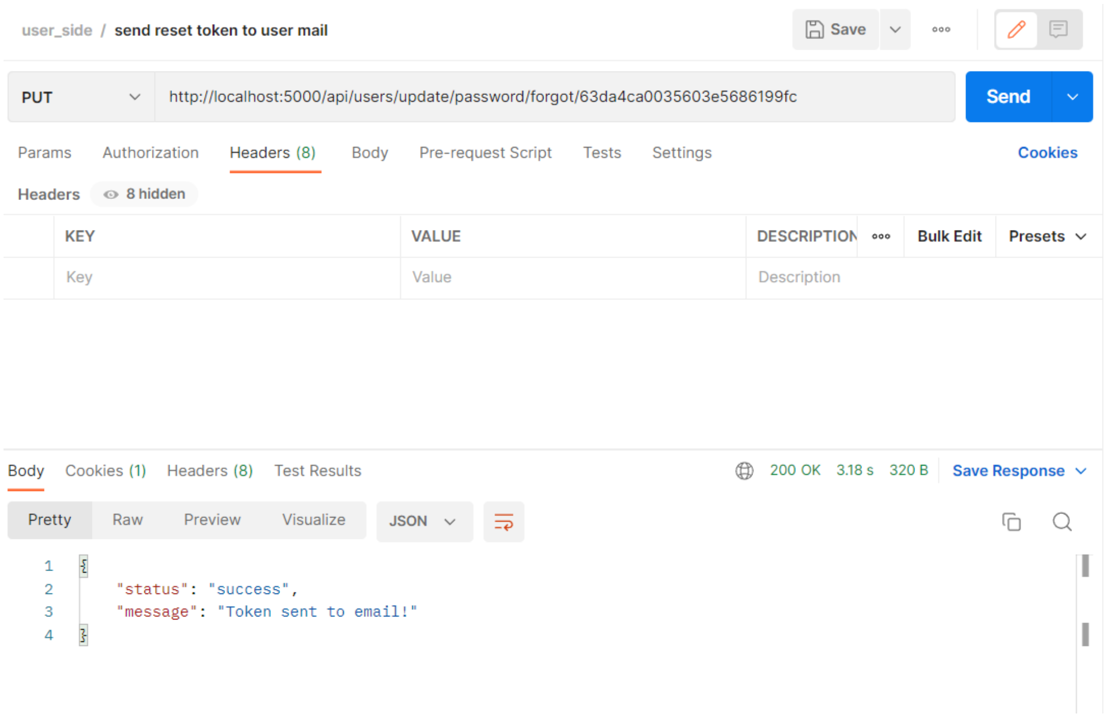
Response: 200 OK 174 ms 793 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ1c2VybmFtZSI6ImFyYWhhbnRhcG9raHJlbCIsImlkIjoiNjNkYTRjYTAwMzU2MDNlNTY4NjE5OWZjIiwiaWF0IjoxNjc1MjUw
        OTc0fQ.7aom00bpS-flsQ91RXcN6A58e-Yc1CcyNwAFyAm8iBw",
    "username": "arahantapokhrel",
    "id": "63da4ca0035603e5686199fc"
}
```

Figure 5: Login User Received

To login onto our system, the user enters their username and password, which is then compared against hashed password saved in our database. Upon successful match, the server returns a token with user id, name, and their role which is than save in cookies of client for any further authentication or authorization check.

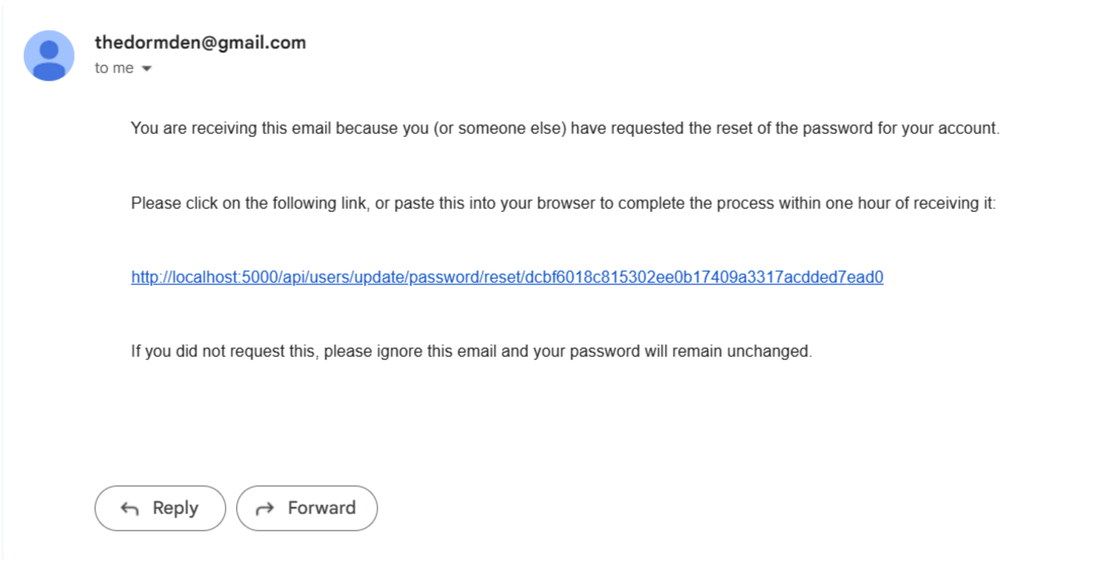
Send reset token to user Email:



```
http://localhost:5000/api/users/update/password/forgot/63da4ca0035603e5686199fc
```

```
{
    "status": "success",
    "message": "Token sent to email!"
}
```

Figure 6: Reset token sent



thedormden@gmail.com
to me

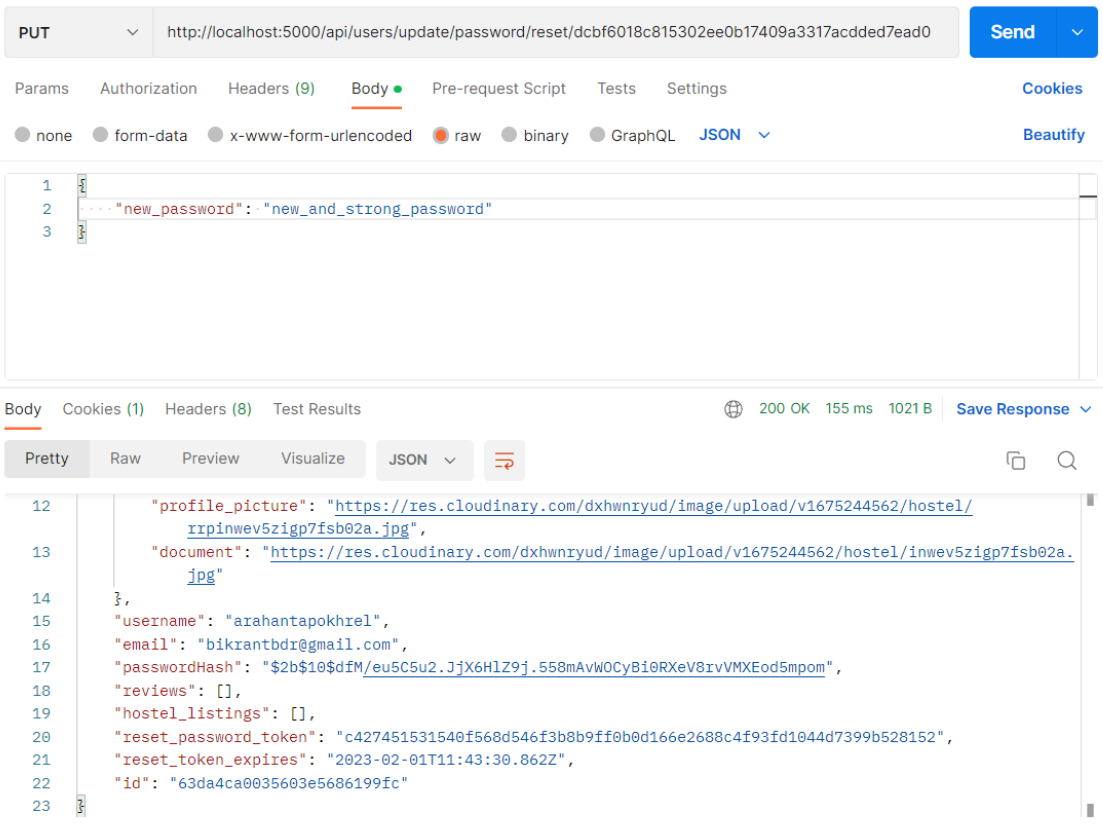
You are receiving this email because you (or someone else) have requested the reset of the password for your account.

Please click on the following link, or paste this into your browser to complete the process within one hour of receiving it:

http://localhost:5000/api/users/update/password/reset/dcbf6018c815302ee0b17409a3317acdded7ead0

If you did not request this, please ignore this email and your password will remain unchanged.

Figure 7: Token received in mail



PUT http://localhost:5000/api/users/update/password/reset/dcbf6018c815302ee0b17409a3317acdded7ead0

```
{
    "new_password": "new_and_strong_password"
}
```

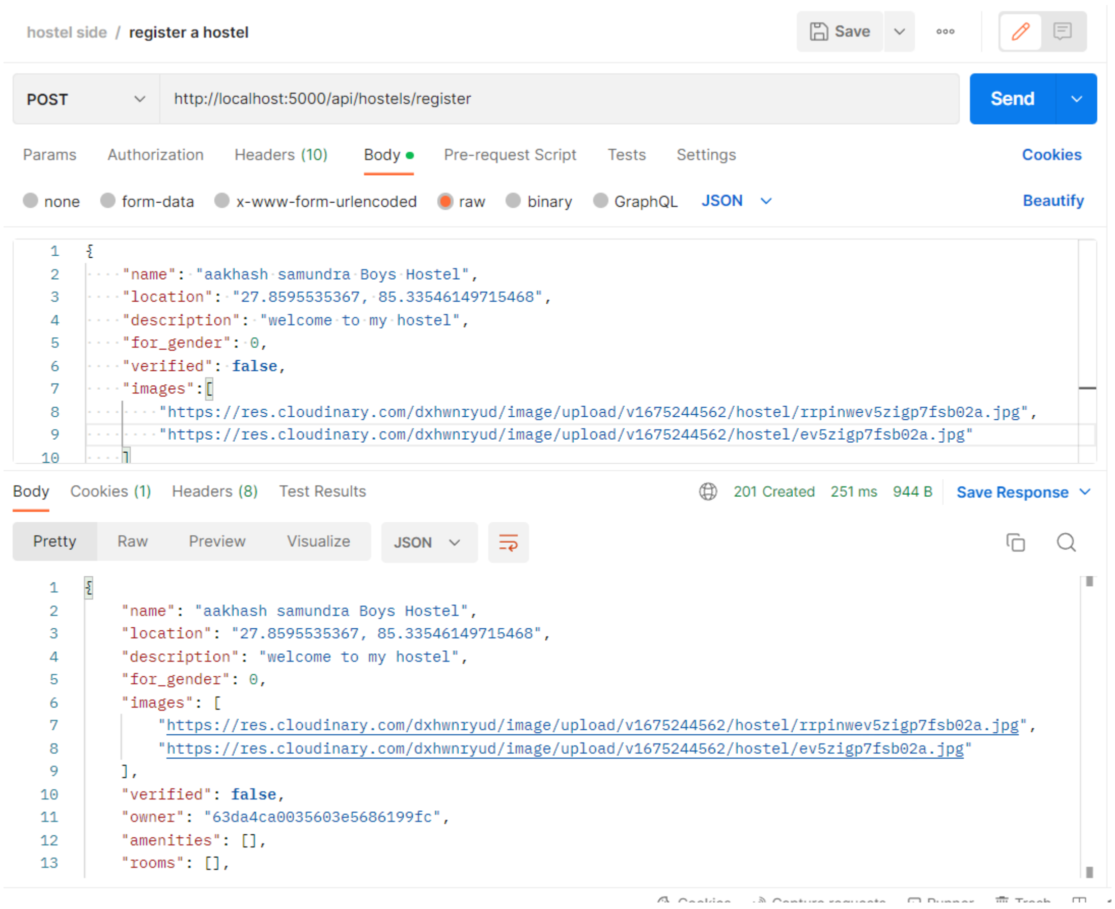
200 OK 155 ms 1021 B

```
        "profile_picture": "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/
            rrpinwev5zigp7fsb02a.jpg",
        "document": "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/inwev5zigp7fsb02a.
            jpg"
    },
    "username": "arahantapokhrel",
    "email": "bikrantbdr@gmail.com",
    "passwordHash": "$2b$10$dfM/eu5C5u2.JjX6HlZ9j.558mAvWOCyBi0RXeV8rvVMXEod5mpom",
    "reviews": [],
    "hostel_listings": [],
    "reset_password_token": "c427451531540f568d546f3b8b9ff0b0d166e2688c4f93fd1044d7399b528152",
    "reset_token_expires": "2023-02-01T11:43:30.862Z",
    "id": "63da4ca0035603e5686199fc"
}
```

Figure 8 : Setting new password

In case user forgets their password, they can get reset token through their Email address, for this we used nodemailer library, when we click on the link provided in mail we can enter new password, which is than updated in the database.

Register a hostel:


hostel side / register a hostel

POST http://localhost:5000/api/hostels/register

```
{
    "name": "aakhash samundra Boys Hostel",
    "location": "27.8595535367, 85.33546149715468",
    "description": "welcome to my hostel",
    "for_gender": 0,
    "verified": false,
    "images":[
        "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/rrpinwev5zigp7fsb02a.jpg",
        "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/ev5zigp7fsb02a.jpg"
    ]
```

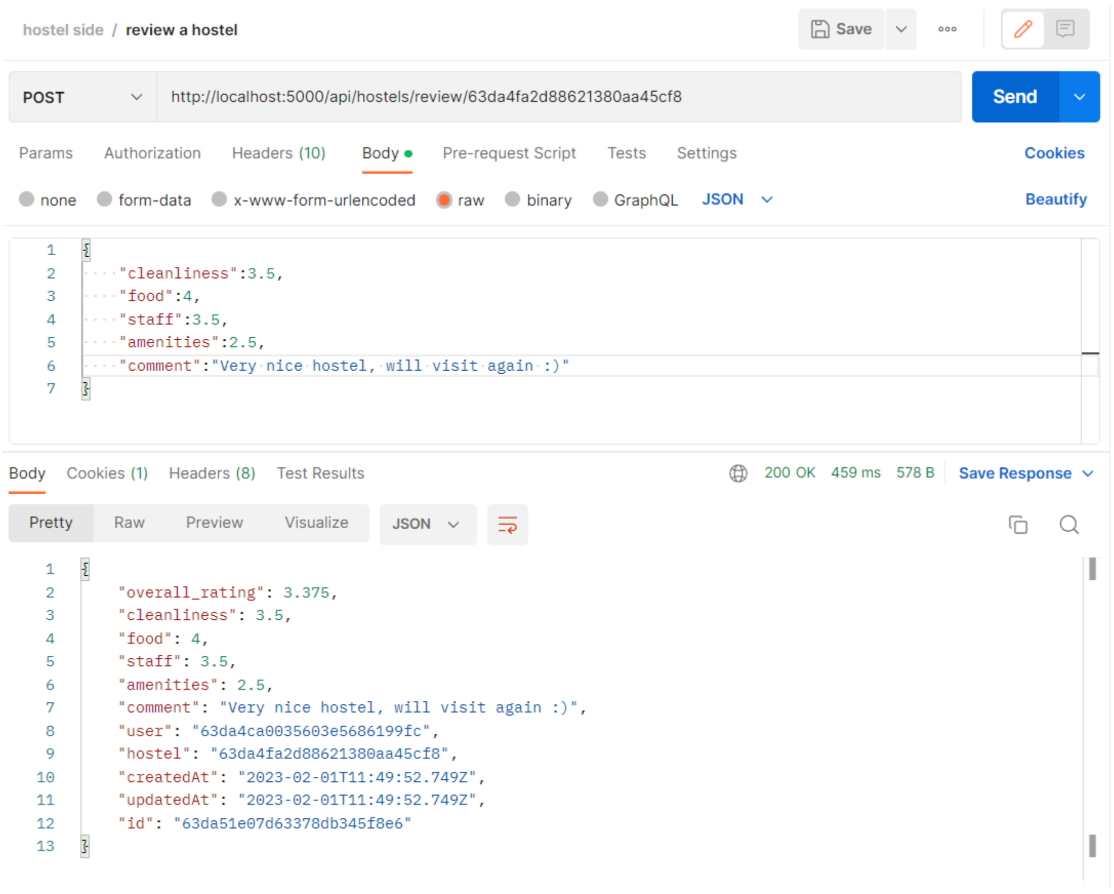
201 Created 251 ms 944 B

```
{
    "name": "aakhash samundra Boys Hostel",
    "location": "27.8595535367, 85.33546149715468",
    "description": "welcome to my hostel",
    "for_gender": 0,
    "images": [
        "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/rrpinwev5zigp7fsb02a.jpg",
        "https://res.cloudinary.com/dxhwnryud/image/upload/v1675244562/hostel/ev5zigp7fsb02a.jpg"
    ],
    "verified": false,
    "owner": "63da4ca0035603e5686199fc",
    "amenities": [],
    "rooms": [],
```

Figure 9: Register hostel

To register a hostel, users must provide some basic information like name, location (through pigeon maps) and their proper registration documentation

Review hostel:



POST http://localhost:5000/api/hostels/review/63da4fa2d88621380aa45cf8

```
{
    "cleanliness":3.5,
    "food":4,
    "staff":3.5,
    "amenities":2.5,
    "comment":"Very nice hostel, will visit again :)"
}
```

200 OK 459 ms 578 B

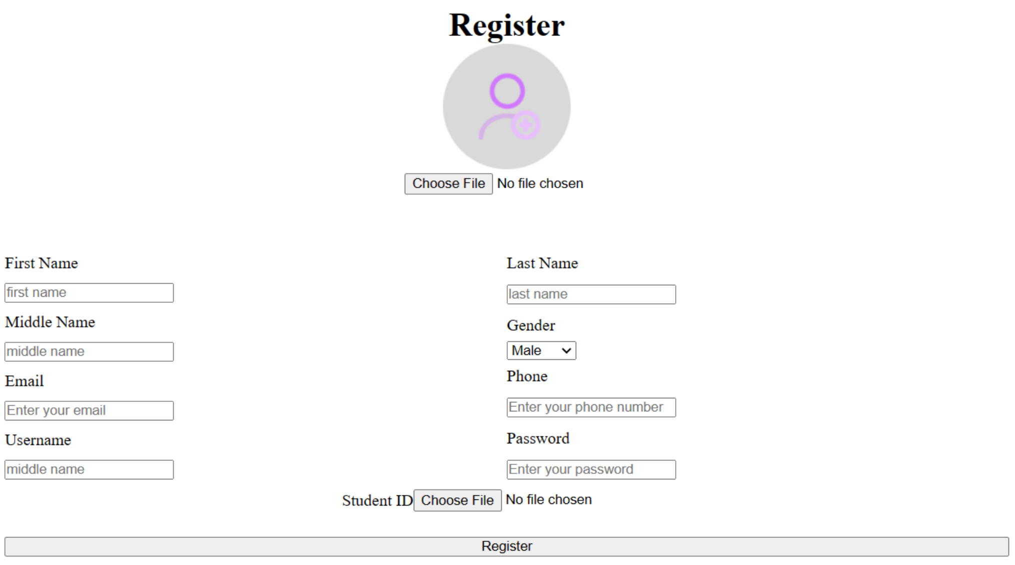
```
{
    "overall_rating": 3.375,
    "cleanliness": 3.5,
    "food": 4,
    "staff": 3.5,
    "amenities": 2.5,
    "comment": "Very nice hostel, will visit again :)",
    "user": "63da4ca0035603e5686199fc",
    "hostel": "63da4fa2d88621380aa45cf8",
    "createdAt": "2023-02-01T11:49:52.749Z",
    "updatedAt": "2023-02-01T11:49:52.749Z",
    "id": "63da51e07d63378db345f8e6"
}
```

Figure 10: Review Hostel

A verified user can leave a review for a particular hostel, our rating system are categorized in four ways: quality of food, behavior of staff, amenities provided and cleanliness, the overall rating is calculated based on provided categories

6.1.2 Frontend

- **Registration Page**



Register

Choose File No file chosen

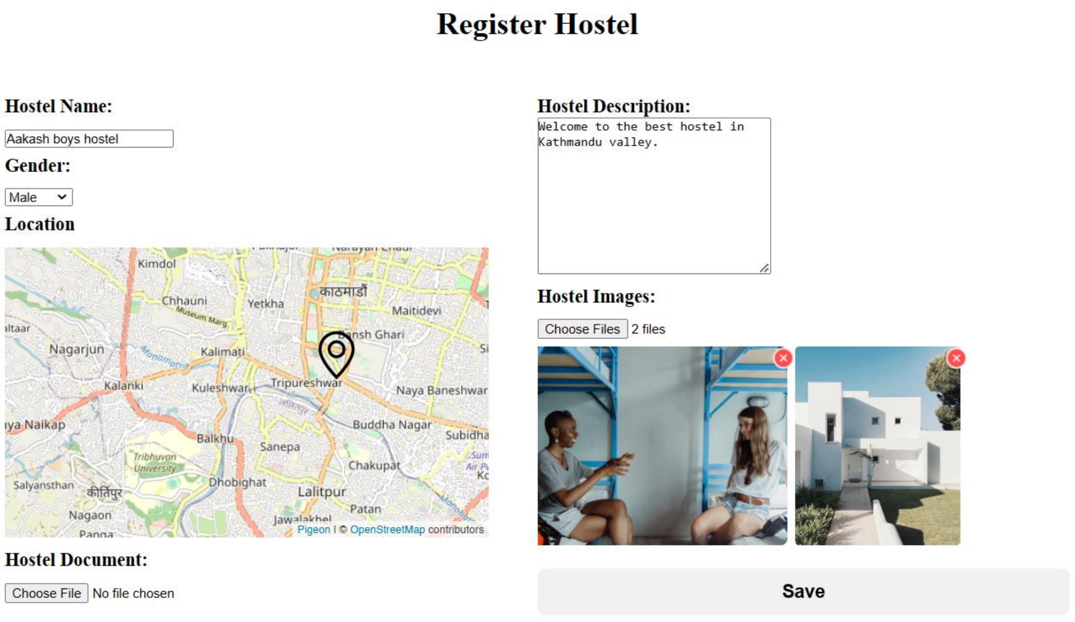
First Name: first name
Last Name: last name
Middle Name: middle name
Gender: Male
Email: Enter your email
Phone: Enter your phone number
Username: middle name
Password: Enter your password
Student ID Choose File No file chosen

Register

Figure 11 : Registration Page

The details of user account is obtained from this page. A basic skeletal structure that takes data and from the user and fetches to the backend is designed.

- **Register Hostel Page**



Register Hostel

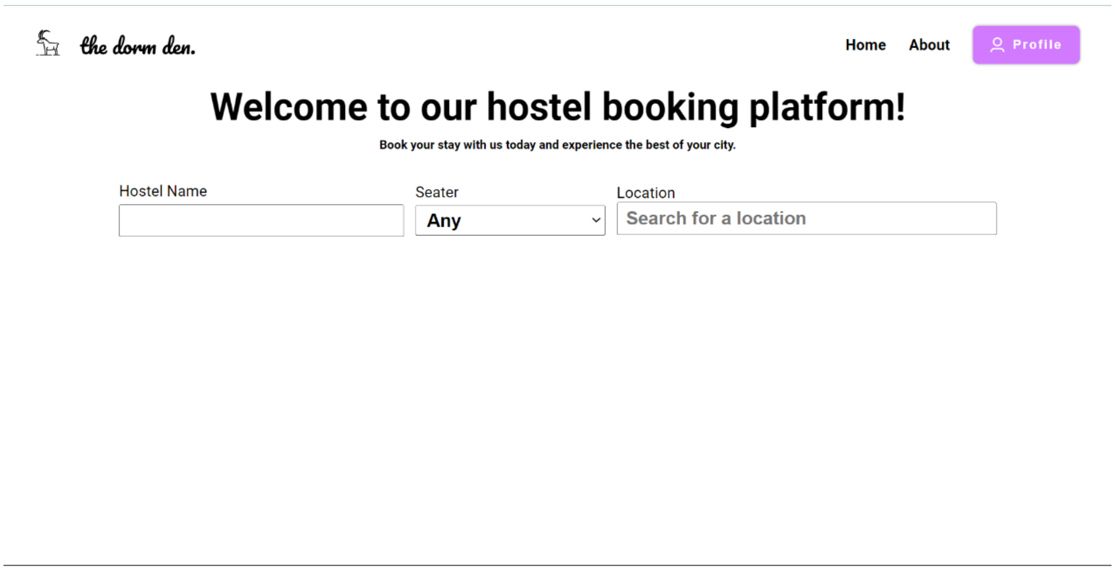
Hostel Name: Aakash boys hostel
Gender: Male
Location
Hostel Description: Welcome to the best hostel in Kathmandu valley.
Hostel Images: Choose Files 2 files
Hostel Document: Choose File No file chosen

Save

Figure 12: Register Hostel Page

The details of hostel is obtained from this page. A basic skeletal structure takes data from the hostel and fetches to the backend is designed.

- **Search Page**



Welcome to our hostel booking platform!

Book your stay with us today and experience the best of your city.

Hostel Name

Seater: Any

Location: Search for a location

Figure 13 : Search Page

This page allows users to search for information about hostels based on their name, seater, and location.

6.2 Performance Analysis

The response time of data retrieval from the backend is 200 to 300 ms, which is fast and efficient for the system. The page load time is efficient, meets industry standards and is consistently within acceptable limits, providing a positive user experience. The web-application can handle a high volume of requests per second with no noticeable lag or delay. The results of resource usage testing indicate that the application uses a minimal amount of memory, CPU, and other resources, providing a fast and responsive user experience.

6.3 Security Analysis

The security of the system is ensured using JWT (JSON Web Token). The token is signed by a secret key on the server, ensuring that the contents of the token cannot be altered during transmission. The client sends the token in the Authorization header with each request, allowing the server to verify the authenticity of the request and identify the user associated with it, without having to perform a database lookup. JWT restricts unauthorized access to any data required and handles the data security issue of the system.

7 REMAINING TASKS

We have completed almost every major components and functionality in our project, some of the remaining tasks are:

1. Dashboard for user profile
2. Admin dashboard area for verifying registration and managing everything
3. Styling, animations, and responsiveness
4. Error handling and fallback states
5. Search Engine Optimization

8 REFERENCES

[1] "Annual Report," University Grants Commission, 2021.

[2] IMDb, "https://www.imdb.com," IMDb, 15 July 2022. [Online]. Available: https://help.imdb.com/article/imdb/track-movies-tv/ratings-faq/G67Y87TFYYP6TWAV#. [Accessed 1 February 2023].

[3] S. Agarwal and S. Rajan, "Performance analysis of MongoDB versus PostGIS/PostGreSQL databases for line intersection and point containment spatial queries," *Spatial Information Research,* vol. 24, no. 6, pp. 671-677, 2016.

[4] B. Cheng and X. Guan, "Design and evaluation of a high-concurrency web map tile service framework on a high," *International Journal of Grid and Distributed Computing,* vol. 9, no. 12, pp. 127-142, 2016.

[5] J. Ferreira, J. Noble and R. Biddle, "Agil. 2007," in *Agile Development Iterations and UI Design*, United States, IEEE Computer Society, 2007, pp. 50-58.

[6] E. Baralis, A. Valle, P. Garza, C. Rossi and F. Scullino, "SQL versus NoSQL Databases for Geospatial Applications," in *In Proceedings of the 2017 IEEE International Conference on Big Data*, Boston, 2017.

APPENDICES

Project Schedule

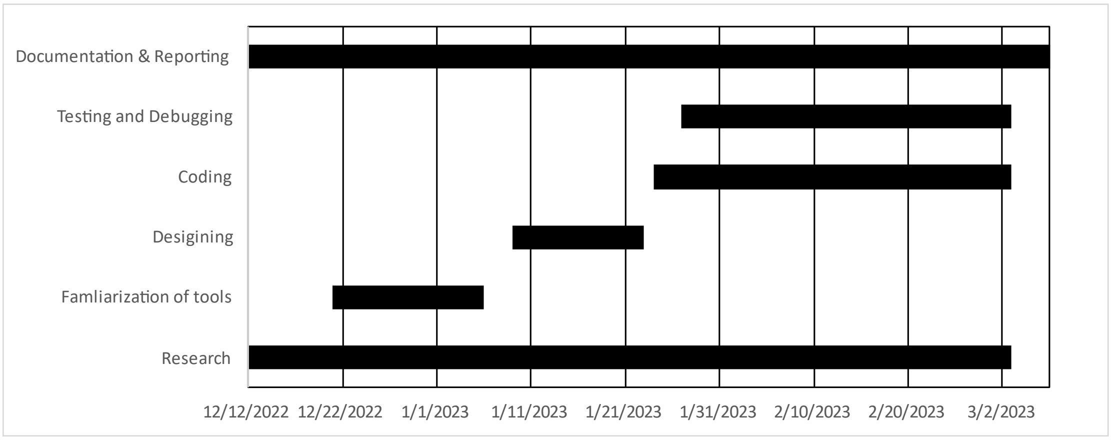


Table 1: Project Schedule

Project Budget

S.N.	Particulars	Price (Rs)	Quantity	Total Price (Rs)
1.	Server Hosting	2000 per month	1	2000
2.	Acquiring a domain name	1500 per year	1	1500
TOTAL				3500

Table 2: Project Budget