**Neighbourhood Fit**

**Frontend Overview**

* Built with: e.g. **HTML/CSS/JavaScript** (or React, etc.)
* Purpose: Help users search for rental houses based on neighborhood and budget.

**Pages / Components**

**1. Login Page**

* Fields:

Email

Password

* Button:

Login

* Calls API:

POST /api/auth/login

**2. Registration Page**

* Fields:

Full Name

Mobile No

Email

Password

Marital Status

No of Family Members

Profession

Two-wheeler count

Four-wheeler count

* Calls API:

POST /api/auth/register

**3. Home Page**

* Dropdowns:

State

Place

* Shows:
  + List of houses matching the selected place
  + Average rent
  + Safety, schools, hospitals, parks, nightlife info
* Calls API:
  + e.g. GET /api/neighbors

**Example User Flow (Frontend)**

1. User opens login page
2. Logs in successfully
3. Navigates to home page
4. Chooses a state and place
5. Sees houses matching criteria

**Frontend Libraries**

* [e.g. Axios for API calls]
* [e.g. React Router for routing]

**Backend Overview**

The **Neighborhood Fit Project** is a web application designed to help users find houses for rent based on their preferred location and budget.

**Main Features:**

* User registration and login
* Home page where users select:

State

Place

* System displays houses matching the selected location and user’s budget

**API Documentation**

**1. User Registration**

* **Endpoint:** POST /api/auth/register
* **Description:** Registers a new user in the system.
* **Request Example (JSON):**

json

{

"fullName": "Puneeth Vadlamudi",

"mobileNo": "9876543210",

"email": "puneeth@example.com",

"password": "securepassword",

"maritalStatus": "Single",

"noOfFamilyMembers": 3,

"profession": "Software Engineer",

"twoWheelerCount": 1,

"fourWheelerCount": 0

}

* **Successful Response:**

User registered successfully with id X

* **Validation Rules:**

fullName: must not be blank

mobileNo: must not be blank

email: must not be blank, must be unique

noOfFamilyMembers: must be ≥ 1

**2. User Login**

* **Endpoint:** POST /api/auth/login
* **Description:** Authenticates a user with email and password.
* **Request Example (JSON):**

json

{

"email": "puneeth@example.com",

"password": "securepassword"

}

* **Possible Responses:**

Successful login:

nginx

CopyEdit

login successful! Welcome, Puneeth Vadlamudi

Invalid credentials:

nginx

CopyEdit

Invalid Email or Password

**3. Home Page Flow**

* After login, the user lands on the home page.
* The user selects:

**State**

**Place**

* The application queries the backend to fetch available neighborhoods and houses matching:

Selected location

User’s budget

* Houses are displayed with details such as:

Average rent

Safety ratings

Nightlife, schools, parks, hospitals, commute information

**Database Tables**

**users Table**

| **Column** | **Type** | **Notes** |
| --- | --- | --- |
| id | BIGINT | Primary key, auto-increment |
| full\_name | VARCHAR | Required |
| mobile\_no | VARCHAR | Required |
| email | VARCHAR | Required, unique |
| password | VARCHAR | Required |
| marital\_status | VARCHAR | Optional |
| no\_of\_family\_members | INT | ≥ 1 |
| profession | VARCHAR | Optional |
| two\_wheeler\_count | INT | Optional |
| four\_wheeler\_count | INT | Optional |

**Example User Flow**

1. User registers an account.
2. User logs in.
3. On the home page, the user selects a state and place.
4. The system displays houses matching the selection and budget constraints.

**Possible Errors**

* Validation errors during registration:
  + “must not be blank”
  + “must be greater than or equal to 1”
* Database connection issues
* Duplicate email errors
* Invalid login credentials

**Future Enhancements**

* JWT token-based authentication
* User profile updates
* Dynamic search filters for budget, amenities, etc.
* Admin dashboard for managing neighborhoods and houses