PROPOSED METHODOLOGY

1. Investigation and Requirement Gathering

Objective: To understand the needs of individuals seeking rooms and roommates and gather insights to design a user-friendly platform tailored to their preferences.

- User Research: Conduct surveys and interviews with students, working professionals, and landlords to understand their requirements, preferences, and common challenges when searching for rooms or roommates. Key areas of interest include budget, location, cleanliness habits, and compatibility criteria.
- Market Analysis: Examine existing platforms for finding rooms and roommates to identify their common features, strengths, and weaknesses. Applications like Roomster, SpareRoom, and Craigslist will be analyzed to determine what makes them effective and where they could improve.

Technical Feasibility

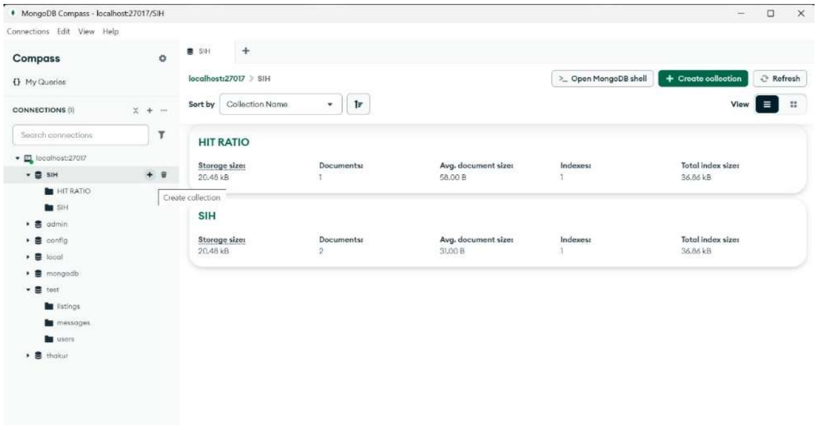
- Review the selected technologies (Node.js, Express.js, MongoDB, HTML, CSS, and Tailwind CSS) to ensure their suitability for building a scalable and secure platform.
- Evaluate how these technologies will be utilized to implement features like advanced search filters, user authentication, and secure data storage while ensuring a seamless user experience**.**

Goal: To provide a data-driven understanding of user needs, allowing the development of a robust, intuitive, and efficient Room and Roommate Finder platform.

        - bio (optional)
        - location
        - preferences (for room seekers)

2. Room Listings Collection:
    - Stores information about rooms available for rent.
    - Fields:
        - roomId (unique identifier)
        - ownerId (reference to the user who owns the room)
        - title
        - description
        - price
        - location
        - amenities (e.g., WiFi, AC, etc.)
        - availableFrom (date)
        - photos (URLs or file references)
3. Applications Collection:
    - Stores applications made by room seekers to room owners.
    - Fields:
        - applicationId (unique identifier)
        - seekerId (reference to room seeker)
        - roomId (reference to room listing)
        - message (optional message from seeker)
        - status (pending/accepted/rejected)
4. Messages Collection (Optional):
    - Stores messages exchanged between room seekers and owners.
    - Fields:
        - messageId (unique identifier)
        - senderId (reference to the user sending the message)
        - receiverId (reference to the user receiving the message)
        - message
        - timestamp

By using MongoDB, the application can handle dynamic and flexible data storage, allowing for future scalability and adaptability.



HIT RATIO

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	1	58.00 B	1	36.86 kB

SIH

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
20.48 kB	2	31.00 B	1	36.86 kB

Authentication and Security:

For authentication, the platform will use JWT (JSON Web Tokens) and bcrypt for secure password hashing, integrated with Node.js. The following components will be included in the authentication process:

1. User Registration:
   - Users (room seekers and room owners) can sign up using an email address and password.
   - Passwords will be securely stored using bcrypt hashing.
   - Upon successful registration, a JWT token will be generated to authenticate users for future requests.
2. Login Process:
   - Users can log in using their email and password.
   - The system will verify the credentials by comparing the entered password with the hashed password stored in the database.
   - If successful, a JWT token is returned for subsequent requests, providing authenticated access to the platform.
3. Password Recovery:
   - Users who forget their passwords can request a password reset link via email.
   - A secure link will be sent, allowing them to reset their password.
4. Session Management:
   - JWT will be used to handle session management. Once authenticated, users will have access to protected routes, such as the ability to view, modify, and apply for room listings, and manage their profiles.
5. Authorization:

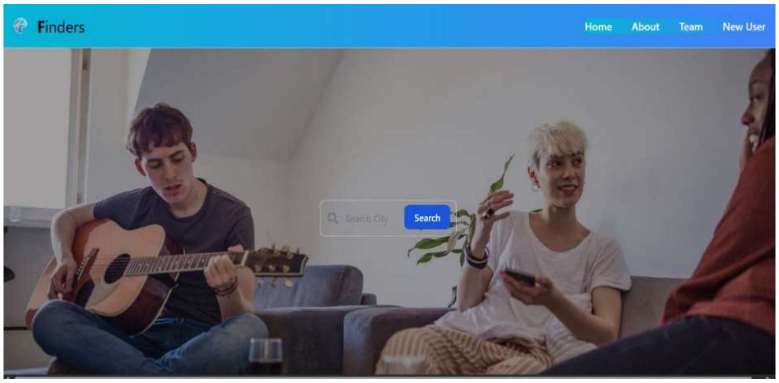
- Roles will be implemented to differentiate between room seekers and room owners. This ensures that only authorized users can access their respective dashboards and perform actions such as creating, modifying, or deleting listings.

3. Development and Implementation

Objective: To develop the core functionality of the Room and Roommate Finder platform, implementing both frontend and backend components, as well as integrating real-time data processing.

Frontend Development:

1. Room Search Interface:
   Develop the room listings page with advanced search filters (location, price range, room type, amenities, etc.) using HTML for structure and Tailwind CSS for styling, ensuring the design is responsive and user-friendly. Room seekers will be able to easily browse through available rooms and filter results according to their preferences.

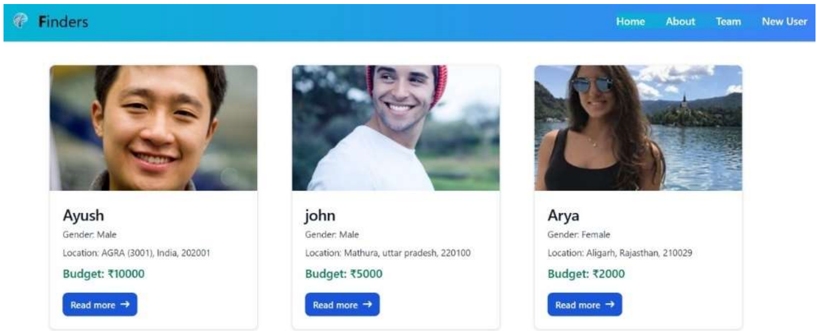


2. Room Application Form:
   Implement a streamlined form for room seekers to apply for rooms directly from the listing page. The form will collect relevant data such as:
   a. Personal details (name, contact, etc.)
   b. Room preferences (location, budget, etc.)
   c. Any additional information or requests
3. The form will also allow users to upload documents like identification or other required files, making the application process efficient and user-friendly.

4. User Dashboard:

   Create a user dashboard where room seekers can:

   a. View and manage their applications (pending, accepted, rejected)
   b. Update their profile information (bio, preferences, etc.)
   c. Track the status of their applications and communicate with room owners
   d. Room owners will also have their dashboard to manage room listings, view applications, and respond to potential tenants.



Ayush
Gender: Male
Location: AGRA (3001), India, 202001
Budget: ₹10000

john
Gender: Male
Location: Mathura, uttar pradesh, 220100
Budget: ₹5000

Arya
Gender: Female
Location: Aligarh, Rajasthan, 210029
Budget: ₹2000

5. Frontend Tech: The user interface will be created using HTML and styled with Tailwind CSS to ensure responsiveness and a modern look.

Backend Development:

1. Real-Time Room Listings:

   The backend will handle storing and managing room listings. MongoDB will be used to store room details, including descriptions, prices, amenities, and available dates. The backend will handle requests for adding new rooms, updating existing listings, and removing rooms.
2. Using Node.js with Express.js, RESTful APIs will be implemented to fetch room listings, add new rooms, and update listings. MongoDB will be connected for storing data and handling real-time interactions.