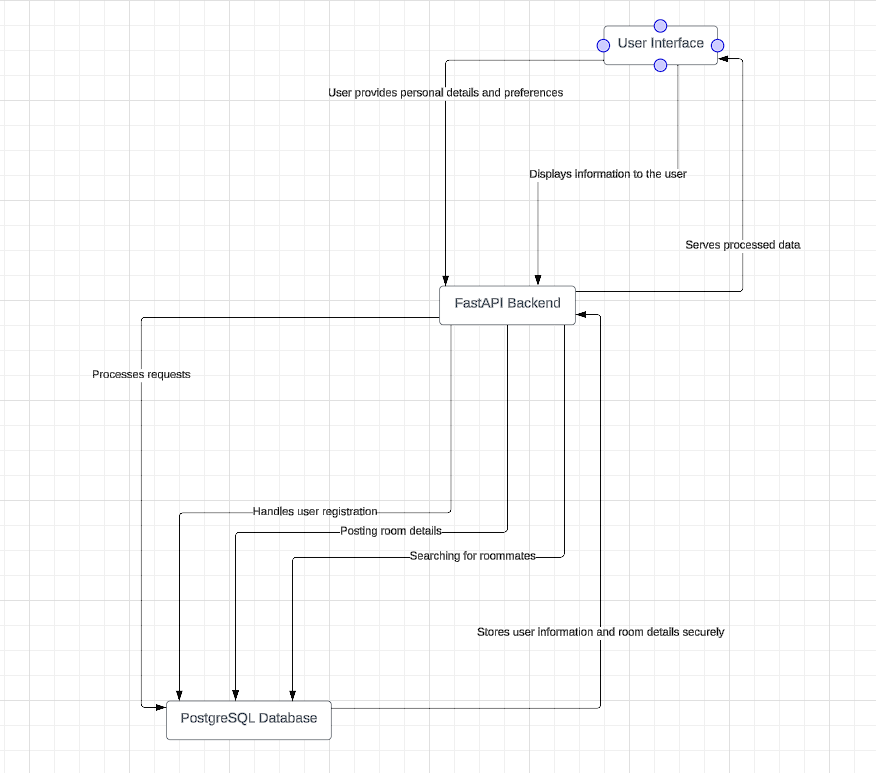
**Project Introduction:**

Our project aims to facilitate the process of finding roommates by providing a user-friendly web portal. Users can register and list their properties, specifying preferences such as location, rent, and occupancy. With comprehensive user profiles and detailed property listings, our platform streamlines the roommate search process efficiently.

****

1. **/profile**:
   * **Method**: POST
   * **Functionality**: This endpoint allows users to register by providing their personal details such as name, gender, city, contact number, and email. Upon successful registration, the user's details are stored in the database.
   * **Input parameters :** entire details that needed to create the profie
2. **/profile/{id}**:
   * **Method**: PUT
   * **Functionality**: This endpoint enables users to update their profile details based on their unique user ID. Users can provide the fields they want to update (name, gender, city, contact number, or email), and the endpoint updates the corresponding fields in the database.
   * **Input parameters**: fields needed to be updated should be entered
3. **/profile/{id}**:
   * **Method**: GET
   * **Functionality**: This endpoint retrieves the details of a user based on their unique user ID from the database.
   * **Input parameters**: enter the id
4. **/add\_listing**:
   * **Method**: POST
   * **Functionality**: This endpoint allows users to add a listing on the portal for finding roommates. Users provide details such as location, what they are looking for, approximate rent, occupancy, who they are, and highlights for the property. This information is then stored in the database.
   * **Input parameters**: fields required for adding the listing
5. **/edit\_listing/{user\_id}**:
   * **Method**: PUT
   * **Functionality**: Users can use this endpoint to update the details of their listing based on their user ID. They can modify fields such as location, what they are looking for, approximate rent, occupancy, who they are, and highlights for the property.
   * **Input parameters**: fields to be updated should be entered
6. **/get\_listing**:
   * **Method**: GET
   * **Functionality**: This endpoint retrieves all listings from the database. It fetches details such as user's name, gender, contact number, listing location, what they are looking for, and who they are.
   * **Input parameters**: no input parameters
7. **/get\_listing\_by\_city**:
   * **Method**: GET
   * **Functionality**: Users can use this endpoint to retrieve listings based on a specific city. They provide the city name as input, and the endpoint fetches listings from that city, including details such as user's name, gender, contact number, listing location, what they are looking for, and who they are.
   * **Input parameters**: city name

**Database Design:**

1. Table: person\_roommate
   * Columns:
     + registration\_id: Unique identifier for each registration.
     + user\_id: Unique identifier for the user associated with the listing.
     + location: Location of the property.
     + looking\_for: Gender preference for roommate.
     + approx\_rent: Approximate rent for the property.
     + occupancy: Occupancy status (e.g., single, multiple).
     + who\_you\_are: Role of the user (e.g., tenant, owner).
     + highlight\_for\_property: Additional features or highlights of the property.
   * Description: This table stores information about property listings, including details such as location, rent, occupancy status, and highlights. Each row represents a unique listing registered by a user.
2. Table: Users
   * Columns:
     + user\_id: Unique identifier for each user.
     + name: Name of the user.
     + gender: Gender of the user.
     + city: City where the user resides.
     + contact\_number: Contact number of the user.
     + email: Email address of the user.
   * Description: This table stores user information, including their name, gender, city, contact number, and email address. The user\_id serves as a unique identifier for each user in the system.
   * Both the tables are connected to each other with user id as foreign key.