
HumaraGhar - A Revolutionary real-estate application.

*A project report submitted in partial fulfillment of the requirements for the
award of the degree of*

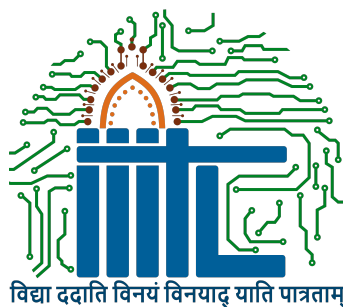
B.Tech. in Information Technology

by

**Ayush Kumar
(LIT2020024)**

**Akshay Bhatnagar
(LIT2020016)**

under the guidance of
Dr. Mainak Adhikari



**Indian Institute of Information Technology, Lucknow
November 2023**

© Indian Institute of Information Technology, Lucknow 2023.

Declaration of Authorship

We, **Ayush Kumar (LIT2020024)** and **Akshay Bhatnagar(LIT2020016)**, declare that the work presented in “**HumaraGhar**” is our own. We confirm that:

- This work was completed entirely while in candidature for B.Tech. degree at Indian Institute of Information Technology, Lucknow.
- Where we have consulted the published work of others, it is always cited.
- Wherever we have cited the work of others, the source is always indicated. Except for the aforementioned quotations, this work is solely our work.
- We have acknowledged all major sources of information.

Signed:

(Ayush Kumar)

(Akshay Bhatnagar)

Date:

CERTIFICATE

This is to certify that the work entitled "**HumaraGhar**" submitted by **Ayush Kumar and Akshay Bhatnagar** who got their name registered on **Jul 2020** for the award of B.Tech. degree at Indian Institute of Information Technology, Lucknow is absolutely based upon their own work under the supervision of **Dr. Mainak Adhikari**, Department of Computer Science, Indian Institute of Information Technology, Lucknow - 226 002, U.P., India and that neither this work nor any part of it has been submitted for any degree/diploma or any other academic award anywhere before.

Dr. Mainak Adhikari
Department of Computer Science
Indian Institute of Information Technology, Lucknow
Pin - 226002, INDIA

Acknowledgements

We are pleased to have successfully completed the project **HumaraGhar**. We thoroughly enjoyed the process of working on this project and gained a lot of knowledge doing so. We would like to take this opportunity to express our gratitude to Dr. Arun Mohan Sherry, Director of IIIT Lucknow, for permitting us to utilize all the necessary facilities of the institution.

We are immensely grateful to our respected and project supervisor **Dr. Mainak Adhikari** for their valuable help and guidance. We are indebted to them for their invaluable guidance throughout the process and their useful inputs at all stages of the process.

We also thank all the faculty and support staff of Department of Information Technology, IIITL. Without their support over the years, this work would not have been possible. We are also grateful to our colleagues and friends who provided us with the necessary resources and support whenever needed. Lastly, we would like to thank our family for their unwavering support and encouragement throughout the project.

Lucknow
November 2023

Ayush Kumar
Akshay Bhatnagar

ABSTRACT

The modern dynamics of housing and shared living arrangements have evolved significantly, paving the way for innovative solutions to streamline the process of finding roommates and renting accommodations. This project introduces a comprehensive room renting application that revolutionizes the traditional approach to housing searches by integrating collaborative features and personalized matchmaking.

The core feature of this application is its versatility, offering users the flexibility to either form teams to explore rental options collectively or seek individual accommodations. The unique selling proposition (USP) lies in its capability to facilitate seamless roommate searches for those seeking cohabitation or simply looking to fill vacancies in existing spaces.

Key functionalities include a sophisticated matchmaking system leveraging user preferences and profiles to pair compatible roommates and predicting accurate renting price suggestions based on neighbourhood and amenities provided. The application utilizes Next.js for its frontend, harnessing its capabilities for dynamic and responsive web interfaces. Meanwhile, Supabase, a robust PostgreSQL-based backend, powers the application's database, ensuring efficient data management and scalability.

Through a user-centric design approach and a user-friendly interface, this application aims to redefine the room renting experience, fostering connections, and simplifying the often daunting task of finding suitable accommodations or roommates.

Contents

1	Introduction	1
1.1	Background and Context	1
1.2	Need	1
1.3	Problem Statement	2
1.4	Objectives	2
1.5	Scope	3
2	Literature Review	4
2.1	Market Trends in India	4
2.1.1	Real Estate Market Trends	4
2.1.2	Rental Market Trends	5
2.1.3	Recent Developments and Future Projections	6
2.2	Why we chose the PropTech sector?	6
2.3	Co-Living & Renting Together	9
2.3.1	Framework of Analysis	9
2.3.2	Owning vs Renting a house	10
3	Methodology	12
3.1	Development Approach	12
3.2	Iterations and Sprints	12
3.3	Tools and Technologies	14
3.3.1	Programming Languages and Frameworks	14
3.3.2	Development Tools	14
3.3.3	Database Management:	15
3.4	Development Process	15
3.4.1	Setup	15
3.4.2	Database Structure and Functionality	18
3.4.3	Rent Agreement Generation	21
3.4.4	AI-Driven Price Prediction Model	22

4	Simulation and Results	31
4.1	Login and Signup	31
4.2	Onboarding Flow	33
4.3	Dashboard	34
4.3.1	Find Roommate	34
4.3.2	Find Room	34
4.3.3	Add Listing	37
4.3.4	Team Management	38
4.3.5	Rent Agreement Generation	39
4.3.6	Admin Panel	40
5	Conclusion and Future Work	41

Chapter 1

Introduction

1.1 Background and Context

In the evolving landscape of urban living, the challenges associated with securing suitable accommodations transcend the boundaries of traditional housing searches. Our endeavor to develop a room renting application stems from the personal experiences of our team members—recent bachelor graduates navigating their entry into professional realms in new, bustling cities. The initial hurdle of securing affordable living spaces without an established network of friends or acquaintances drove us to conceive a solution that redefines the norms of housing searches.

As young professionals transitioning from academic settings to bustling urban environments, the lack of a supportive network to share the costs and experiences of renting became a significant obstacle. This personal experience serves as the driving force behind our initiative, inspiring us to craft a comprehensive solution that addresses not only the financial burden but also the social and logistical challenges associated with finding suitable accommodations and compatible roommates.

1.2 Need

The contemporary housing market poses challenges for individuals transitioning from academia to professional spheres, particularly in unfamiliar urban landscapes. There exists a pressing need for an inclusive, efficient, and user-centric platform that accommodates both solo renters and those seeking collaborative living arrangements or compatible roommates. The absence of such a tool creates inefficiencies and hurdles in finding afford-

able, compatible, and convenient housing solutions.

1.3 Problem Statement

The challenge lies in bridging the gap between traditional housing approaches and the modern needs of individuals seeking varied accommodation options. The absence of a centralized platform that caters to both the individual renter and those interested in collaborative living leads to prolonged searches, incompatible living situations, and financial strains.

1.4 Objectives

1. **Comprehensive Housing Searches:** Develop a platform that accommodates both individuals seeking independent rentals and those forming teams for collective exploration of rental spaces, ensuring inclusivity and convenience.
2. **Efficient Roommate Matchmaking:** Create a robust matchmaking system that caters not only to those in search of compatible roommates but also to individuals seeking solo accommodations, ensuring tailored matches.
3. **AI-Driven Rent Price Suggestions:** Utilize machine learning models trained on diverse rent data to suggest optimal rent prices for listed properties, empowering users with informed pricing strategies.
4. **Owner Dashboard & Property Management:** Enable property owners to manage their listings through a dedicated dashboard, facilitating tasks like sending rent reminders, generating contract templates for rent agreements, and ensuring seamless property management.
5. **Controlled Chat Feature:** Implement a secure and controlled chat function, allowing users to interact and negotiate further deals only with authorized and interested parties, enhancing user security and convenience.
6. **User-Centric Design:** Prioritize user experience by delivering an intuitive interface that simplifies the search for accommodations or roommates, irrespective of the individual's preferences.

7. **Addressing Financial and Social Barriers:** Alleviate the challenges associated with securing affordable living arrangements and establishing connections in new cities, catering to diverse housing needs.

1.5 Scope

The project aims to develop a Next.js-based web application utilizing Supabase, a PostgreSQL-based backend, to create a scalable, dynamic, and responsive platform. The focus lies in providing a holistic solution catering to both individuals seeking independent rentals and those in search of compatible roommates, fostering a sense of community and ease in the process of finding suitable accommodations. Leveraging machine learning models, the platform will suggest rent prices for listed properties, while a controlled chat feature will enable secure interactions between authorized users, further enhancing the platform's utility and user experience.

Chapter 2

Literature Review

Summarizing and analyzing existing research, studies, and relevant publications that relate to our project.

2.1 Market Trends in India

2.1.1 Real Estate Market Trends

1. **Urbanization and Population Growth:** India's rapid urbanization continues to drive demand for housing. With a growing population and increasing urban migration, the need for affordable and accessible housing remains a significant trend. [4]
2. **Shift in Rental Preferences:** There has been a shift in preferences among the younger population towards rental accommodations due to mobility for jobs, lower commitment, and financial flexibility. This shift is particularly notable in metro cities and urban hubs.
3. **Co-living and Co-working Spaces:** Emerging trends show a rise in demand for co-living spaces and co-working environments, especially among millennials and young professionals. These spaces offer a sense of community, shared amenities, and cost-effectiveness.

The rise of co-living and co-working spaces is not just a trend; it's a significant shift in urban lifestyles. In densely populated cities like Mumbai, Delhi, and Bengaluru, these concepts are addressing the challenges of affordable housing and the need for flexible, productive workspaces. Moreover, co-living offers attractive returns—2-4 times higher than the traditional residential yield of 2-3

per cent—leading to higher investors’ interest in actively pursuing options in the market to create flexible co-living facilities. [5]

4. **Tech Integration in Real Estate:** Technology adoption in real estate has seen substantial growth. Digital platforms for property searches, virtual property tours, and online rent payment systems have gained popularity, enhancing convenience for both landlords and tenants.
5. **Government Initiatives:** Various government initiatives like the Pradhan Mantri Awas Yojana (PMAY) and Smart Cities Mission aim to provide affordable housing and improve infrastructure, influencing the real estate landscape and rental market dynamics.

2.1.2 Rental Market Trends

1. **Rise in Rental Yields:** Despite fluctuations, rental yields have been stable or rising in certain areas, attracting investors and encouraging property owners to engage in the rental market.
2. **Demand for Flexible Rentals:** There’s an increasing demand for flexible rental options, including short-term leases and furnished accommodations, particularly from young professionals and students.
3. **Emergence of PropTech Solutions:** Proptech startups are introducing innovative solutions for property management, tenant screening, and rent collection, streamlining processes for both landlords and tenants.
4. **Localized Rental Dynamics:** Rental markets vary significantly across different cities and regions in India. For instance, metropolitan areas like Mumbai and Delhi exhibit different rental patterns, pricing structures, and demand-supply dynamics compared to tier-II or tier-III cities.
5. **Rental Regulations and Policies:** Rental laws and regulations, such as the Rent Control Act and local tenancy laws, significantly impact the rental market. Understanding these regulations is crucial for both landlords and tenants. [1]

2.1.3 Recent Developments and Future Projections

1. **Post-Pandemic Impact:** The COVID-19 pandemic has influenced the rental market, causing temporary shifts like increased demand for spacious homes, a surge in remote work leading to altered location preferences, and a focus on hygiene and safety in rental spaces.
2. **Technology and Data Analytics:** Continued integration of technology, including AI-driven property searches, blockchain for secure transactions, and data analytics for market predictions, is expected to redefine the rental landscape, making it more efficient and transparent.
3. **Sustainability and Green Spaces:** Growing awareness of sustainability and environmental concerns is influencing rental choices, with preferences for eco-friendly properties and communities on the rise.
4. **Policy Changes:** Anticipated policy changes or amendments in rental laws, especially concerning tenancy agreements and rent control, could significantly impact the market dynamics in the coming years.

2.2 Why we chose the PropTech sector?

The story of proptech startups in India took shape in the mid-2000s after the entry of Info Edge-owned 99acres and Times Internet-owned Magicbricks, Quikr-owned CommonFloor and PropTiger (REA India)-owned Makaan.com. Following this, the story revolved around Housing.com, a once-celebrated startup which then faced a series of mishaps. In the past five years, the proptech segment has evolved manifold and has seen startups in brokerage tech led by Square Yards, construction tech led by Infra.Market, AI, AR, VR, IoT, SaaS, and other spaces.

The Indian real estate sector is predicted to reach \$1 trillion in market size by 2030, up from \$200 billion in 2021, and contribute to 13% of the country's GDP by 2025. And looks like tech companies in this space have a role to play as well. As per data compiled by Fintrackr, proptech startups have mopped up nearly \$2.4 billion between January 2021 and March 2023. This comprises 39 growth stage companies raising \$2.25 billion and 69 early stage startups raising \$145 million. If we take previous data, then proptech startups have managed to raise \$2.9 billion since January 2020.



Figure 2.1: Key Highlights

Funding in proptech startups peaked in 2018 with \$1.28 billion. Even as the trend continued in 2019, the impact of lockdown can be seen in 2020 when the fundraise plunged to less than \$500 million. This again saw a revival in 2021, only soon to get impacted by an overall slowdown in the funding environment in 2022 and 2023. While pre-Covid era was dominated by the likes of OYO, co-working space providers, the post-Covid period saw massive funding in construction and building material focused startup such as Infra.Market, real estate rental startup NoBroker, home decor and interior startups Livspace and HomeLane. [2]

In this section, we have highlighted top funded startups in proptech and their capital efficiency ratio based on their financial performance in FY22.

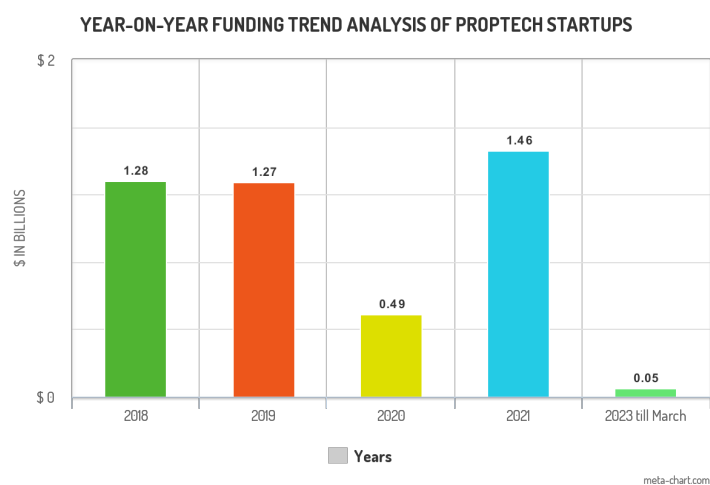


Figure 2.2: Funding trend analysis

Table 2.1: Capital Efficiency of top PropTech Startups

Name	Revenue(FY22)	Overall Funding	Capital Efficiency
IndiQube	₹6236 Cr	₹342 Cr	2.00
WeWork India	₹351.4 Cr	₹1300 Cr	1.03
Awfis	₹257 Cr	₹760 Cr	0.50
NestAway	₹57.8 Cr	₹828 Cr	0.07
Stanza Living	₹115 Cr	₹1672 Cr	0.07
NoBroker	₹116 Cr	₹2743 Cr	0.06

Table 2.2: Top Revenue Generating Real Estate Focused Companies

Name	Revenue(FY21)
Square Yards	₹245.7 Cr
Anarock	₹182 Cr
99acres	₹173.8 Cr
Housing.com	₹88 Cr
PropTiger	₹50.57 Cr
NoBroker	₹166.5 Cr
Quikr Homes	₹60.7 Cr
MagicBricks	₹166 Cr

2.3 Co-Living & Renting Together

We live in a globally connected world and this has led to the real estate sector experiencing disruption led by nomadic millennials, who are re-defining the meaning of 'living' and 'working'. The concept of 'shared economy' has just started to unfold in India and the days ahead look much more exciting. Unlike earlier when 'ownership' was fundamental to success in life, today 'sharing' has taken the centre stage.

2.3.1 Framework of Analysis

JLL Research conducted a comprehensive demand survey targeting millennials across the top seven cities of Mumbai, Delhi NCR, Bengaluru, Hyderabad, Chennai, Kolkata and Pune. The key objective of this assessment was to study their behavioural patterns for owning and renting houses.

The nomads of today are becoming a major force driving the Indian housing market—an estimated 42% constituted millennials¹ in 2018 across the top seven cities in India. It is relevant to note that millennials will continue to form a major proportion of the country's working population, growing to nearly 41% of the workforce in 2023. *Nearly 40% of India's millennial workforce are migrants*

Migrant millennial workforce prefers to rent

With millennials driving housing demand, there is a marked change in demand patterns. Gen Y has a different take on home ownership from their parents. Earlier generations moved to the peripheral locations to fulfil their dreams of owning a home, but millennials refuse to compromise.

While selecting an accommodation, connectivity to their workplace, convenience and security are the factors on top of their decision making

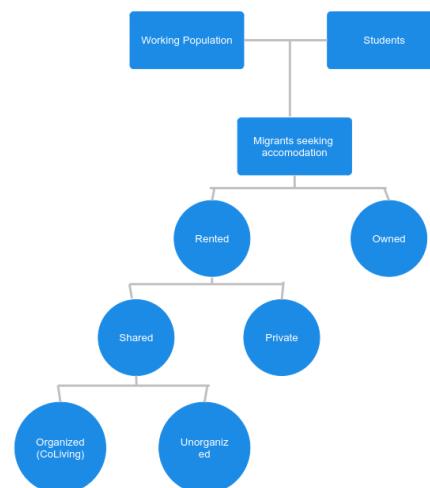


Figure 2.3: Framework Analysis

tree and not ownership of the property. Moreover, a migrant workforce prefers rented apartments because of the uncertainty attached with the duration of their stay as well as cost savings in renting as against purchasing accommodation.

The outcome-an increased demand for rental housing.

2.3.2 Owning vs Renting a house

Home ownership has been long considered as a basis to measure financial and social security in India. However, there has been a discernible change in the mind-set and the behaviour of the recent generations. This is attributable to the following:

1. High cost of housing in most gateway cities and lower returns from buying a house (EMI to rent ratio is typically 2-3 times in most cities).
 2. Change in the nature of work - short term nature of assignments warranting higher mobility and flexibility.
 3. Delayed marriage and child rearing, less inclination to block funds and rather spend on travel, food and leisure (considered to be discretionary in the past).
- 93% of the migrant respondents who were single stayed in rented accommodation across the top 7 cities
 - 60% said that they didn't plan or were unsure about buying a house in the future
 - Budget constraints and limited flexibility were cited as the key reasons for not owning a house

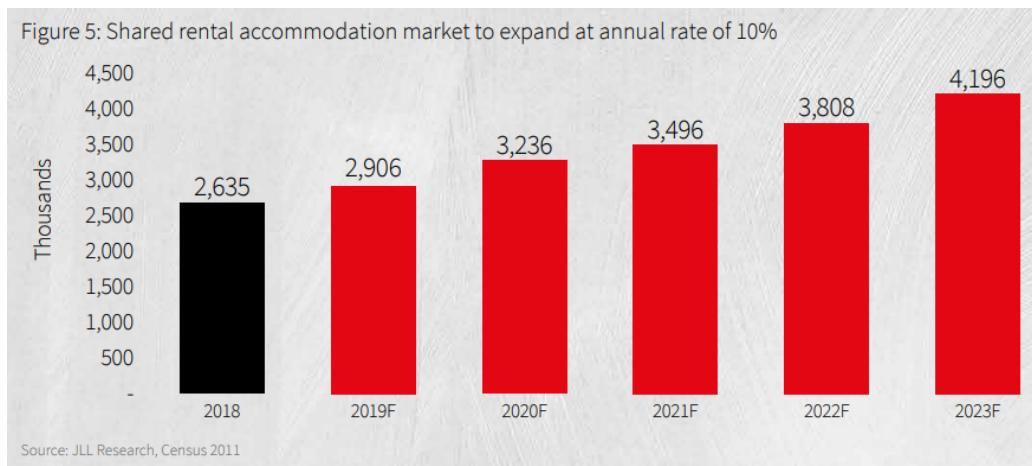


Figure 2.4: Funding trend analysis

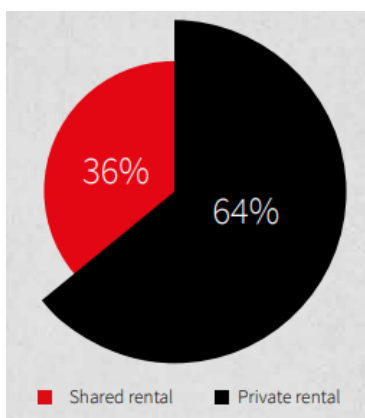


Figure 2.5: Living Preferences

Here's Why Millennials Find Shared Rental Apartments Most Viable:

- Cities like Delhi, Pune, Hyderabad, Bengaluru, Mumbai, Chennai and Kolkata attract millennials for education as well as employment
- Millennials consider proximity to workplace or education institute as most important while choosing accommodation
- Renting private apartment in commercial or educational hub beyond financial means of most millennials
- With housing rent typically accounting for 25-30% of average monthly income in urban India, rental costs get apportioned in case of shared accommodation [3]

Chapter 3

Methodology

3.1 Development Approach

While starting the project, we had two methods or ideologies to go by either following a waterfall model or an agile model. The waterfall model is a sequential design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) while the agile model is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. Both of these models have their own advantages and disadvantages.

After careful consideration, we decided to go with a Hybrid model which is a combination of both waterfall and agile model. Initially, we followed a Waterfall approach for planning and designing phases of the project. After the planning and designing phase, we transitioned to an agile model for the development phase of the project. This approach helped us to have a clear idea of what we were going to do and how we were going to do it by implementing the features in sprints, continuously monitoring the results, analyzing data, and making adjustments.

3.2 Iterations and Sprints

Our application development journey was organized into distinct sprints, each focusing on pivotal components vital to our renting platform's functionality and user experience.

Sprint 1: Onboarding Experience In this sprint, our emphasis was on crafting a seamless onboarding flow. We concentrated on creating an intuitive and user-friendly registration process, ensuring users effortlessly navigate through initial setup, profile creation, and account authentication.

Sprint 2: Dynamic Dashboard Design Our team dedicated this sprint to developing a dynamic and informative dashboard. Here, the focus was on designing an interface that offers users a comprehensive overview of their rented properties, team collaborations, and reminders, fostering a centralized hub for efficient property management.

Sprint 3: Room Rental & Roommate Finder Logic In this sprint, the spotlight was on formulating and implementing robust logic for room rental and roommate finder functionalities. We concentrated on refining algorithms that power property searches, match roommates based on preferences, and streamline the rental process for individual users and collaborative teams.

Sprint 4: AI-Driven Price Prediction Model A critical phase, this sprint was dedicated to integrating an AI-driven price prediction model. We meticulously crafted and integrated machine learning algorithms to provide accurate rental price estimations, enhancing transparency and aiding users in making informed decisions.

Sprint 5: Team Invitation Management Here, our focus shifted towards creating and implementing the logic necessary to create and manage team invitations effectively. We engineered a system that allows users to form collaborative teams seamlessly, enabling efficient property searches and shared responsibilities among team members.

Sprint 6: Robust Chat System The final sprint concentrated on the development of a robust chat system. We engineered a communication platform that fosters seamless interaction between users, facilitating easy communication within collaborative teams, aiding in decision-making, and enhancing overall user engagement.

These delineated sprints encapsulate the strategic breakdown of our development process, allowing us to iteratively build, test, and refine specific components of our renting application, culminating in a comprehensive and user-centric platform.

3.3 Tools and Technologies

3.3.1 Programming Languages and Frameworks

Next.js Framework with TypeScript: Our renting application was built upon the Next.js framework, utilizing TypeScript for enhanced type safety and code clarity. Next.js, combined with TypeScript, empowered us to develop a robust, scalable, and statically typed application, ensuring improved code quality and developer productivity.

Shadcn UI Components: Shadcn UI components played a pivotal role in expediting the frontend development process. These pre-designed components, coupled with TypeScript support, facilitated rapid UI development, ensuring consistency and aesthetic appeal across the application.

React-Hook-Forms with Zod Validators: For efficient form management and data validation, we integrated React-Hook-Forms with Zod validators. This amalgamation, leveraging TypeScript's typing capabilities, allowed for seamless form handling, validation, and data integrity checks.

3.3.2 Development Tools

VSCode Editor: Visual Studio Code (VSCode) remained our IDE of choice, offering a robust development environment with TypeScript support. Its extensive feature set, including debugging tools and TypeScript linting, facilitated efficient coding practices and collaboration among team members.

Git/GitHub Version Control: Git in conjunction with GitHub served as our version control system, allowing collaborative development and code management. TypeScript's typing features integrated seamlessly with Git, aiding in version tracking and ensuring code consistency.

Vercel for Deployment: Vercel's deployment platform, compatible with TypeScript and Next.js, streamlined our deployment process. This powerful platform facilitated quick and reliable deployment, ensuring that updates were efficiently pushed to production with TypeScript support.

Docker Docker was used to run supabase locally for running supabase-cli scripts to update database migration files.

3.3.3 Database Management:

Supabase: Postgres-Based DB with Buckets for Data and Image Storage; Supabase, integrated with TypeScript for seamless data handling, functioned as our primary database solution. The PostgreSQL-based database, coupled with TypeScript's typing features, provided a secure and scalable infrastructure for storing application data and images through Supabase buckets.

3.4 Development Process

This section will summarize the development process, including how we approached the problems technically.

3.4.1 Setup

The project was initialized using create-next-app with the supabase template.

```
1 npx create-next-app -e with-supabase
```

Cookie-based Auth

We are using Cookie-based Auth in our web application. The Next.js Auth Helpers package configures Supabase Auth to store the user's session in a cookie, rather than localStorage. This makes it available across the client and server of the App Router - Client Components, Server Components, Server Actions, Route Handlers and Middleware. The session is automatically sent along with any requests to Supabase.

```
1 npm install @supabase/auth-helpers-nextjs @supabase/supabase-js
```

Listing 3.1: Installing Auth Helpers

We are then declaring environment variables in .env.local file with following environment variables:

```
1 NEXT_PUBLIC_SUPABASE_URL=your-supabase-url
2 NEXT_PUBLIC_SUPABASE_ANON_KEY=your-supabase-anon-key
```

Listing 3.2: .env.local

The Next.js Auth Helpers are configured to use the server-side auth flow to sign in users into the application. For this we had to setup a Code Exchange route, to exchange an auth code for the user's session, which is set as a cookie for future requests made to Supabase.

We created a callback route handler that performs this exchange at `app/api/auth/callback/route.js`:

```
1 import { createRouteHandlerClient } from '@supabase/auth-  
  helpers-nextjs'  
2 import { cookies } from 'next/headers'  
3 import { NextResponse } from 'next/server'  
4  
5 export async function GET(request) {  
6   const requestUrl = new URL(request.url)  
7   const code = requestUrl.searchParams.get('code')  
8  
9   if (code) {  
10    const cookieStore = cookies()  
11    const supabase = createRouteHandlerClient({ cookies: ()  
      => cookieStore })  
12    await supabase.auth.exchangeCodeForSession(code)  
13  }  
14  
15  
16  // URL to redirect to after sign in process completes  
17  return NextResponse.redirect(requestUrl.origin)  
18 }
```

Listing 3.3: Callback Route Handler

Part of inviting users to our application was having a attractive UI. We achieved this through discussing over design language and then finalising over shadcn's collection for our UI components. Shadcn-ui contains reusable components built using Radix UI and Tailwind CSS.

Dark Mode Support

We are using next-themes package to support dark mode in our application. This package provides a useTheme hook that allows us to access the current theme and toggleTheme function to toggle between light and dark mode. We are using this hook to toggle between light and dark mode in our application.

```
1  npm install next-themes
```

Listing 3.4: Installing next-themes

Creating a theme provider

components/theme-provider.tsx

```
1  "use client"
2
3  import * as React from "react"
4  import { ThemeProvider as NextThemesProvider } from "next-
   themes"
5  import { type ThemeProviderProps } from "next-themes/dist/
   types"
6
7  export function ThemeProvider({ children, ...props }:
   ThemeProviderProps) {
8    return <NextThemesProvider {...props}>{children}</
       NextThemesProvider>
9  }
```

Listing 3.5: Theme Provider

The created theme-provider is added to the root layout and linked with tailwind-css config file to achieve the proper result.

3.4.2 Database Structure and Functionality

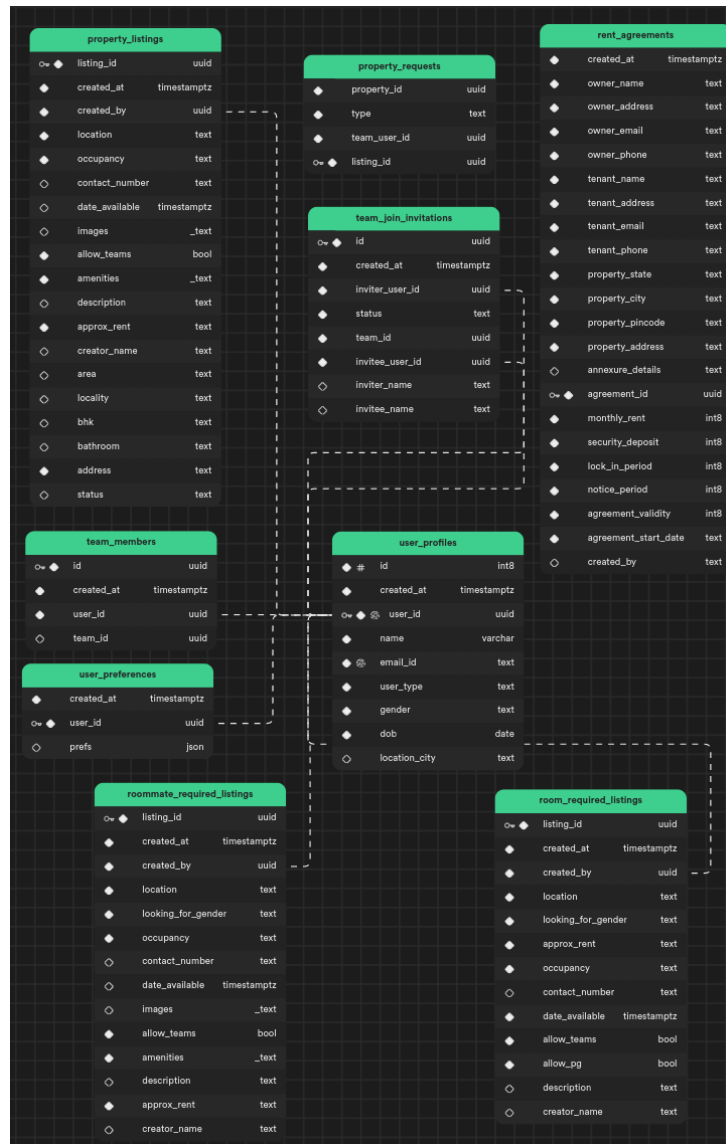


Figure 3.1: Funding trend analysis

This image describes the database structure of our application. We have used supabase's database and storage buckets to store our data. We have used supabase's auth to authenticate users and supabase's storage buckets to store images and other files.

Why Supabase and PostgreSQL?

Utilizing Row-Level Security (RLS) Policies

One of the key features that drew us to Supabase is its implementation of Row-Level Security (RLS) policies. RLS policies enable fine-grained control over data access within the database by restricting users' access to specific rows based on defined security policies.

In our room renting application, RLS policies play a pivotal role in ensuring data security and privacy. They allow us to enforce access control mechanisms at the row level, granting users access only to the data relevant to their interactions and permissions. For instance, RLS policies dictate that users can view and interact with property listings, roommate preferences, or matchmaking data only within the scope of their authorized permissions, enhancing data confidentiality and integrity.

Triggers in PostgreSQL

PostgreSQL's robust support for triggers serves as a pivotal element in automating various processes within the database. Triggers enable the execution of predefined actions in response to specific database events, ensuring streamlined and automated workflows.

In our application, triggers play a pivotal role in orchestrating interactions between users and property listings, facilitating personalized recommendations, and triggering roommate matchmaking processes. For instance, when a user expresses interest in a property or updates their preferences, triggers activate functions that analyze compatibility, recommend suitable listings, or initiate matchmaking processes.

PL/pgSQL Functions for Automation

The integration of PL/pgSQL functions within PostgreSQL further amplifies the automation capabilities of our database. PL/pgSQL, a procedural language extension for PostgreSQL, allows the creation of custom functions to automate complex database operations.

These functions empower our system to automate tasks such as automatically moving members to correct teams after they accept or decline an invite request.

Team Invitation Management

Users can browser other users who have signed up for teams and filter them based on the their preferences, the preferences were already stored during onboarding and are used to calculate a score for each user. The score is calculated by comparing the user's preferences with the preferences of the user who is searching for a roommate.

We are mainly taking advantage of two tables in our schema for this purpose, team_members and team_join_invitations. The team_members table contains mapping of users to teams and the team_join_invitations table contains list of all the invitations sent to users to join a team, with a column having the status of the invite as "waiting", "accepted" or "rejected".

We have a postgres trigger in place watching over the value of status and it is triggered when the value of status changes from "waiting" to "accepted".

```
1 CREATE TRIGGER "on_team_invitation_accepted_trigger"
2 AFTER
3 UPDATE ON "public"."team_join_invitations" FOR EACH ROW
4     WHEN (
5         (
6             ("old"."status" <> "new"."status")
7             AND (
8                 "new"."status" = 'accepted'::"text"
9             )
10        )
11 ) EXECUTE FUNCTION "public"."
    handle_add_team_member_after_invitation_accepted"();
```

The trigger calls a function handle_add_team_member_after_invitation_accepted which adds the user to the team and updates the values in team_members table.

```
1 create function "public"."
    handle_add_team_member_after_invitation_accepted" ()
    returns "trigger" language "plpgsql" security definer as
    $$BEGIN
2 UPDATE team_members
3 SET team_id = NEW.team_id
4 WHERE user_id = NEW.invitee_user_id;
5
6 IF NOT FOUND THEN
7 INSERT INTO team_members(user_id, team_id)
8 VALUES (
9     NEW.invitee_user_id,
10     NEW.team_id
11 );
```

```
12 END IF;  
13 RETURN NEW;  
14 END;  
15 $$;
```

The changes are then reflected in the application and the user is added to the team.

3.4.3 Rent Agreement Generation

The implementation of a multi-step form using the useMultiStepForm React hook stands as a cornerstone in our room renting application, specifically facilitating the seamless input of data required for generating rent agreements in PDF format. This client-side process employs the react-pdf package for PDF generation and utilizes file-saver to enable users to download the finalized agreement.

Functionality of the Mutli-Step Form

The useMultiStepForm hook was custom-built to guide users through a structured sequence of input fields necessary for creating a comprehensive rent agreement. Each step of the form collects specific information, such as:

- Agreement details
- Tenant details
- Owner details
- Property details
- Additional details

The structured nature of the multi-step form ensures that users provide all necessary information required to generate a legally binding rent agreement.

Rent Agreement Download

Upon completion of the multi-step form, the entered data is compiled and processed client-side using the react-pdf package. This package dynamically generates a PDF document based on the collected input, encapsulating the terms and clauses specified in the rent agreement.

Subsequently, the file-saver library is utilized to enable users to download the generated rent agreement PDF directly from the client-side interface, providing a convenient and immediate access method for users.

Status of Digital Agreements in Indian Law

In India, the status of digital agreements is governed by the Information Technology Act, 2000, and the Indian Contract Act, 1872. These legal frameworks recognize electronic documents and digital signatures as valid and enforceable, provided they comply with specific requirements, including:

- Authentication of electronic records
- Ensuring the integrity of the information
- Obtaining consent from involved parties

Digital agreements, including electronically signed rent agreements, hold legal validity if they adhere to these stipulations outlined in Indian law.

3.4.4 AI-Driven Price Prediction Model

Context

Housing in India varies from palaces of erstwhile maharajas to modern apartment buildings in big cities to tiny huts in far-flung villages. There has been tremendous growth in India's housing sector as incomes have risen. The Human Rights Measurement Initiative finds that India is doing 60.9% of what should be possible at its level of income for the right to housing.

Renting, also known as hiring or letting, is an agreement where a payment is made for the temporary use of a good, service, or property owned by another. A gross lease is when the tenant pays a flat rental amount and the landlord pays for all property charges regularly incurred by the ownership. Renting can be an example of the sharing economy.

Data Glossary

- **BHK:** Number of Bedrooms, Hall and Kitchen
- **Rent:** Price of Houses/Apartments/Flats.
- **Size:** Size of the House/Apartment/Flat in Square Feet.
- **Floor:** Floor Number of the House/Apartment/Flat
- **Area Type:** Type of the Area (e.g. Super built-up Area, Built-up Area, Carpet Area)
- **Area Locally:** Locality of the House/Apartment/Flat
- **City:** City in which the House/Apartment/Flat is located
- **Furnishing Status:** Furnishing Status of the House/Apartment/Flat (e.g. Fully Furnished, Semi-Furnished, Unfurnished)
- **Tenant Type:** Type of Tenant (e.g. Family, Bachelor, Company Lease, etc.)
- **Bathroom:** Number of Bathrooms in the House/Apartment/Flat
- **Point of Contact:** Contact Number of the Owner of the House/Apartment/Flat

Data Analysis and Visualization

Pairplot of data

```
1 sns.pairplot(rent_data,height=2)
2 plt.show()
```

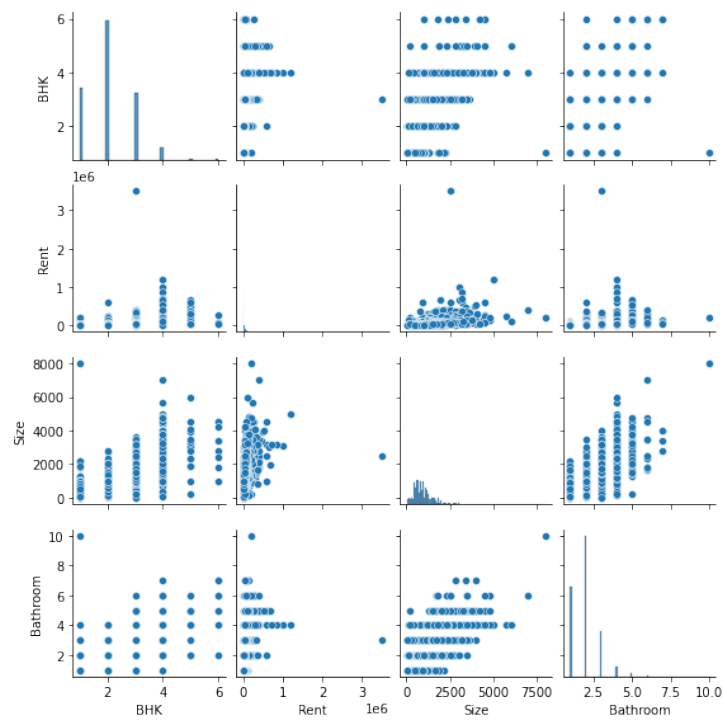


Figure 3.2: Pairplot of data

A univariate and bivariate analysis was done to flush out outliers. [6]

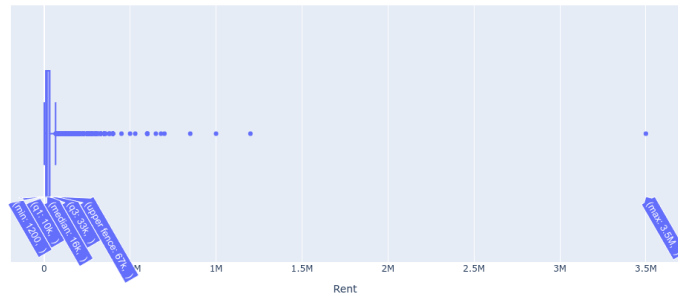
Rent (Our Target Variable)

```

1  fig = px.histogram(rent_data, x='Rent',
2      color_discrete_sequence = px.colors.qualitative.Set3,
3      title="Rent Prices Distribution Histogram")
4  fig.show()
5  fig = px.box(rent_data, x="Rent", title='Boxplot for Rent
6      Prices')
7  fig.show()

```

Boxplot for Rent Prices



Observations: There is one outlier so far out of the inter-quantile range.

Actions: To remove outlier, as it may affect our assumptions about other variables and analysis

```
1 print(np.where(rent_data['Rent']>2000000))
```

Observations: Outlier's position is at 1837th position in a dataframe.
Deleting the outlier

```
1 rent_data.drop([1837], axis=0, inplace=True)
2
3 fig = px.box(rent_data, x="Rent",title='Boxplot for Rent
  Prices')
4 fig.show()
```

BHK

```
1 rent_data['BHK'].value_counts()rent_data['BHK'].
  value_counts()

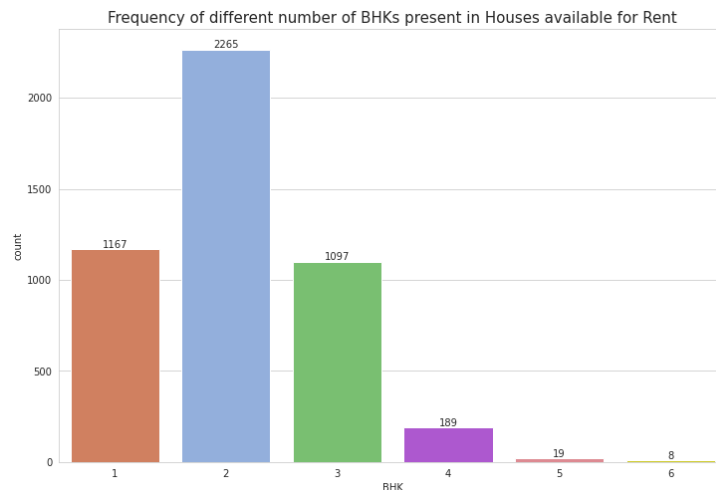
1 sns.set_style('whitegrid')
2 fig,axes = plt.subplots(figsize=(12,8))
3 colors = ['#87ace8','#e3784d', '#6ecc64','#b644e3','#
  eb7c87', '#EAE509']
4
5 ax = sns.countplot(x='BHK',data=rent_data, palette=['#
  e3784d','#87ace8', '#6ecc64','#b644e3','#eb7c87','#
  EAE509'])
6 for container in ax.containers:
7     ax.bar_label(container)
8 plt.title('Frequency of different number of BHKs present
  in Houses available for Rent',fontsize=15)
9 plt.show()
10
11 fig = px.pie(rent_data, names='BHK', height=700, width=
  700, color_discrete_sequence=px.colors.sequential.deep
  , title='Pie Chart for different number of BHKs
  present in Houses available for Rent')
```



```

12 fig.update_traces(textfont_size=15)
13 fig.show()

```



Observations: House with 2 Bathrooms are most common for the houses put up on rent. Houses with 7 and 10 bathroom quite seems inappropriate and not much of use.

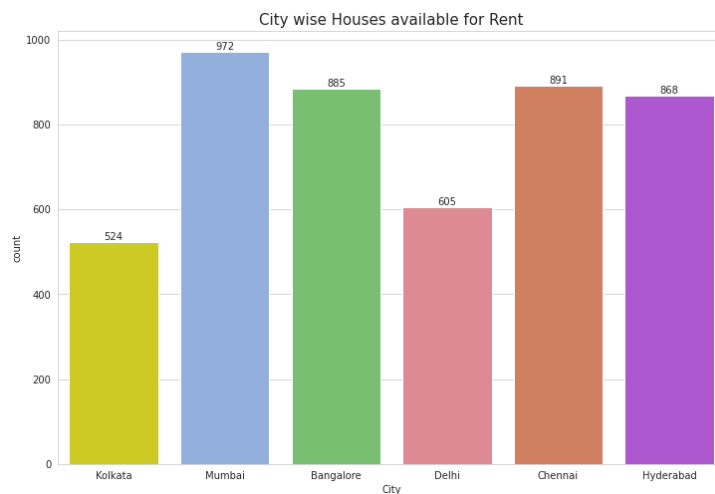
City

```

1 rent_data['City'].value_counts()

1 sns.set_style('whitegrid')
2 fig, axes = plt.subplots(figsize=(12,8))
3 colors = ['#87ace8', '#e3784d', '#6ecc64', '#b644e3', '#
    eb7c87', '#EAE509']
4
5 ax = sns.countplot(x='City', data=rent_data, palette=['#
    EAE509', '#87ace8', '#6ecc64', '#eb7c87', '#e3784d', '#
    b644e3'])
6 for container in ax.containers:
7     ax.bar_label(container)
8 plt.title('City wise Houses available for Rent', fontsize
    =15)
9 plt.show()
10
11 fig = px.pie(rent_data, names='City', height=700, width=
    700, color_discrete_sequence=px.colors.sequential.deep
    , title='Pie Chart for Houses available for Rent in
    different cities')
12 fig.update_traces(textfont_size=15)
13 fig.show()

```



Observations: Mumbai, followed by Chennai and Hyderabad has most number of rented houses, seems like there is very high demand considering the job corporates and other factors.

Liner Regression

A Linear Regression model was used to predict the prices in our application.

The most extensively used modelling technique is linear regression, which assumes a linear connection between a dependent variable (Y) and an independent variable (X). It employs a regression line, also known as a best-fit line. The linear connection is defined as $Y = c + m \cdot X + e$, where 'c' denotes the intercept, 'm' denotes the slope of the line, and 'e' is the error term.

The linear regression model can be simple (with only one dependent and one independent variable) or complex (with numerous dependent and independent variables) (with one dependent variable and more than one independent variable).

Training the model

The model was trained using the following code:

```
1 # Scaling the data
2 y_train= y_train.values.reshape(-1,1)
3 y_test= y_test.values.reshape(-1,1)
4
5 from sklearn.preprocessing import StandardScaler
6 sc_X = StandardScaler()
7 sc_y = StandardScaler()
```

```

8     X_train = sc_X.fit_transform(X_train)
9     X_test = sc_X.fit_transform(X_test)
10    y_train = sc_y.fit_transform(y_train)
11    y_test = sc_y.fit_transform(y_test)

1    print(X_train,y_train)

1    lm = LinearRegression()
2    lm.fit(X_train,y_train)
3    lm_prediction = lm.predict(X_test)
4
5    # Evaluation metrics
6    mae_lm = metrics.mean_absolute_error(y_test,
7                                         lm_prediction)
7    mse_lm = metrics.mean_squared_error(y_test,
8                                         lm_prediction)
8    rmse_lm = np.sqrt(mse_lm)

1    print('MAE:', mae_lm)
2    print('MSE:', mse_lm)
3    print('RMSE:', rmse_lm)

```

using linear regression model we created a function to predict rental prices

```

1    import pickle
2    with open('rent_prices_model.pickle','wb') as f:
3        pickle.dump(lr_clf,f)

```

This is basically utility file where we imported and loaded our model

```

1    import json
2    import pickle
3    import numpy as np
4    import sklearn
5
6    __locations = None
7    __data_columns = None
8    __model = None
9
10   def get_estimated_price(location,sqft,bath,bhk):
11       try:
12           loc_index = __data_columns.index(location.lower())
13       except:
14           loc_index = -1
15       x = np.zeros(len(__data_columns))
16       x[0] = bhk
17       x[1] = sqft
18       x[2] = bath

```

```

19         if loc_index >=0:
20             x[loc_index]=1
21
22         return round(__model.predict([x])[0],2)
23
24     def get_location_name():
25         return __locations
26
27     def load_saved_artifacts():
28         print("loading saved artifacts... start")
29         global __data_columns
30         global __locations
31
32         with open("./artifacts/columns.json",'r') as f:
33             __data_columns= json.load(f)['data_columns']
34             __locations = __data_columns[3:]
35
36         global __model
37         with open("./artifacts/rent_prices_model_1.pickle",'
38             rb') as f:
39             __model = pickle.load(f)
40             print("loading saved artifacts...done")
41
42     if __name__ == '__main__':
43         load_saved_artifacts()
44         print(get_location_name())
45         print(get_estimated_price('Adambakkam',1200,2,3))

```

Flask Server to handle requests from the webapp

```

1     from flask import Flask, request, jsonify
2     import util
3
4     app = Flask(__name__)
5
6     @app.route('/get_location_name',methods=['GET'])
7     def get_location_name():
8         try:
9             print("Before calling util.get_location_name()")
10            locations = util.get_location_name()
11            print("After calling util.get_location_name()")
12            response = jsonify({'locations': locations})
13            response.headers.add('Access-Control-Allow-Origin', '*')
14            return response
15        except Exception as e:
16            return jsonify({'error': str(e)}), 500
17
18    @app.route('/predict_rent_price', methods=['POST'])
19    def predict_rent_price():

```

```

20         try:
21             sqft = float(request.form['sqft'])
22             location = request.form['location']
23             bhk = int(request.form['bhk'])
24             bath = int(request.form['bath'])
25
26             response = jsonify({
27                 'estimated_price': util.get_estimated_price(
28                     location, sqft, bath, bhk)
29             })
30             response.headers.add("Access-control-allow-origin", "*")
31             return response
32         except Exception as e:
33             return jsonify({'error': str(e)}), 500
34
35     if __name__ == "__main__":
36         print("Starting Python Flask Server for Rent Price
37             Prediction Model")
38         util.load_saved_artifacts()
39         app.run()

```

Chapter 4

Simulation and Results

In this section we will going over all the part of our app one by one simulating a user's workflow.

4.1 Login and Signup

The landing page is the first page that the user will see when he/she opens the app. The landing page is shown in

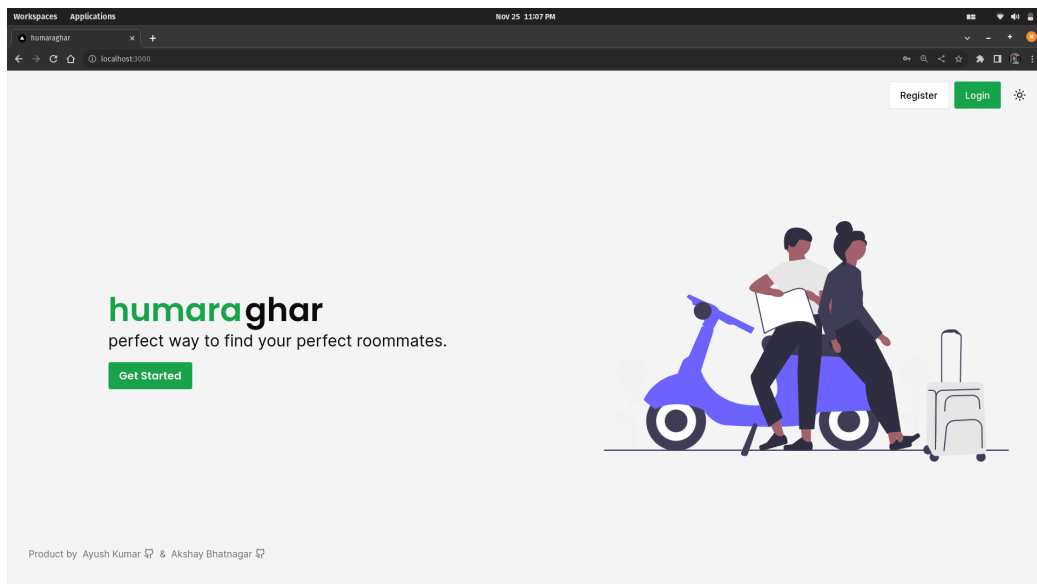


Figure 4.1: Landing Page

The landing page contains a button that will take the user to the sign-up/login page or directly to the dashboard if already signed in.

The sign-up/login page is shown in

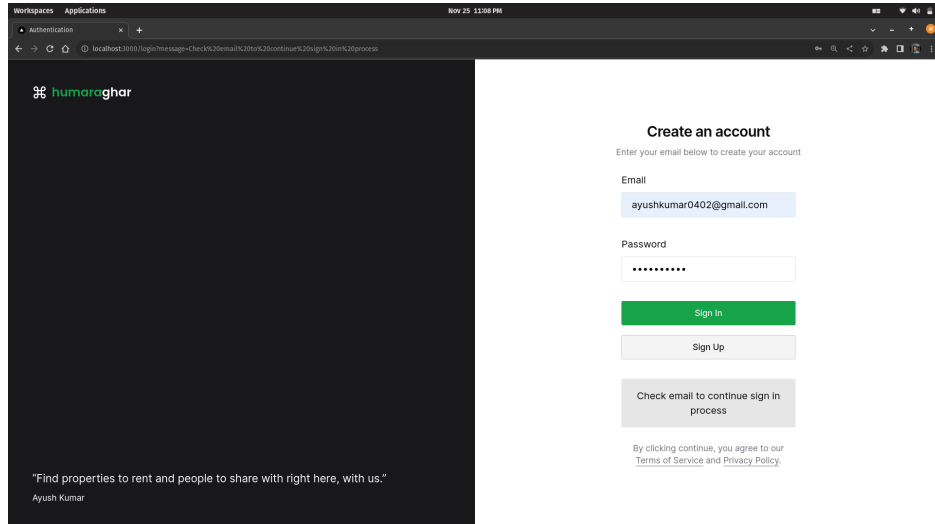


Figure 4.2: Login Page

On entering the correct details, a verification mail will be sent to the user's email address. The user will have to verify his/her email address to be able to login. The verification mail is shown in

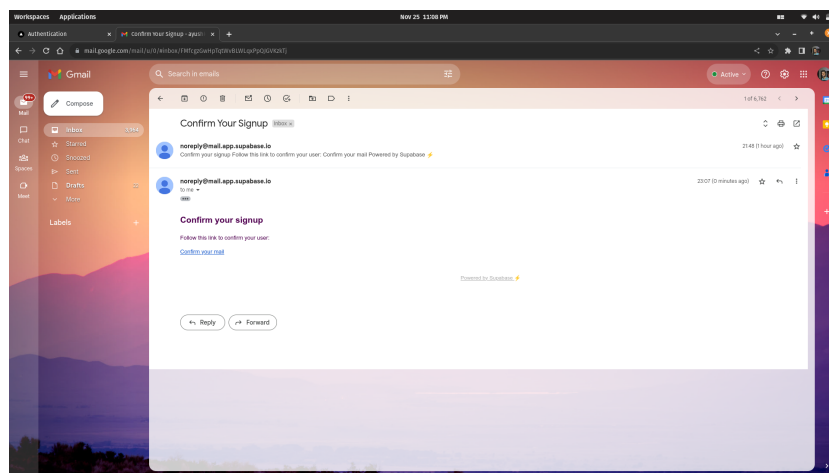
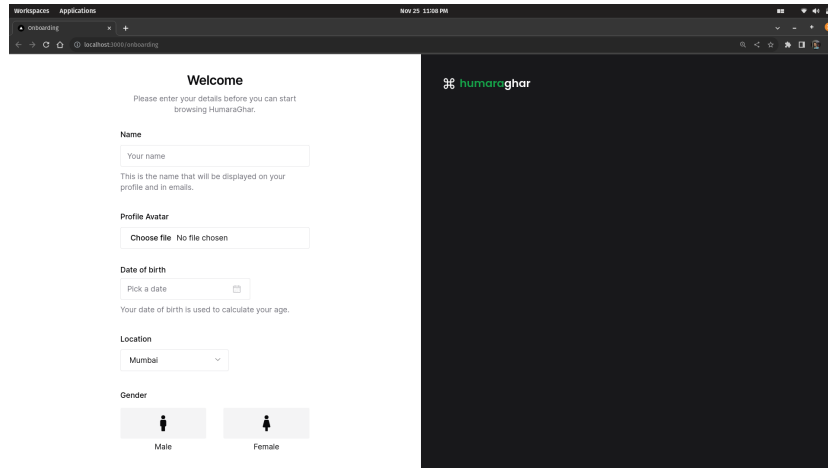


Figure 4.3: Verification Mail

4.2 Onboarding Flow

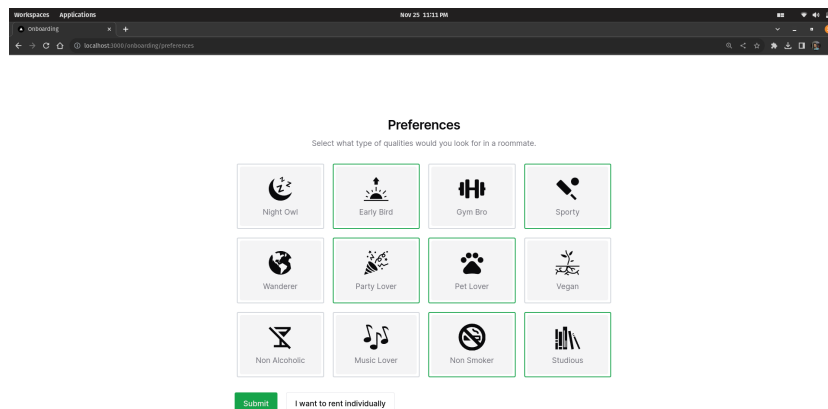
The onboarding flow contains forms containing user's data. The user will have to fill the forms to be able to use the app. The first section of onboarding process aims to collect basic information about the user:



The screenshot shows a web browser window with the URL `localhost:3000/onboarding`. The page has a dark theme. On the left, there's a white form titled "Welcome" with the instruction "Please enter your details before you can start browsing Humaraghar." The form fields are: "Name" (text input), "Profile Avatar" (file upload), "Date of birth" (date picker), "Location" (dropdown menu showing "Mumbai"), and "Gender" (radio buttons for "Male" and "Female"). On the right, there's a dark sidebar with the "humaraghar" logo at the top.

Figure 4.4: Onboarding Page 1

The second section of onboarding process aims to collect information about the user's roommate preferences in case he/she wants to share their room with someone else. The user can skip this section if he/she wants to live alone. This data will be used to calculate match-score between users.



The screenshot shows a web browser window with the URL `localhost:3000/onboarding/preferences`. The page is titled "Preferences" with the instruction "Select what type of qualities would you look for in a roommate." Below this is a 3x4 grid of 12 preference categories, each with an icon and a label: "Night Owl", "Early Bird", "Gym Bro", "Sporty", "Wanderer", "Party Lover", "Pet Lover", "Vegan", "Non Alcoholic", "Music Lover", "Non Smoker", and "Studious". At the bottom, there is a green "Submit" button and a link "I want to rent individually".

Figure 4.5: Onboarding Page 2

4.3 Dashboard

The dashboard is the main page of the app. It contains various sections that will be discussed in detail in the below.

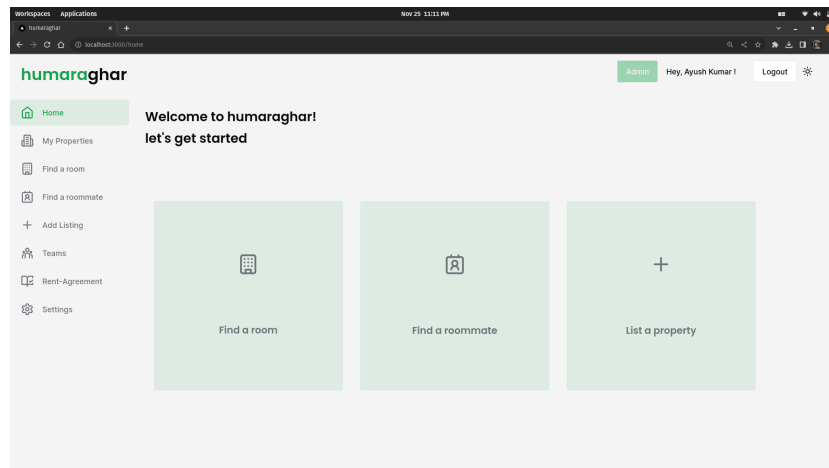


Figure 4.6: Dashboard

The home page has links to redirect user to main parts of the dashboard catered towards finding a roommate and finding a room. The user can also view his/her profile by clicking on the profile icon on the top right corner of the page.

4.3.1 Find Roommate

The find roommate page contains a list of all the users that have listing themselves as looking for a shared room, so anymore already having a place to live but looking for a roommate to split the expenses can find them. Alternatively, you can also team-up among people looking for a room and search for an empty flat together. The user can filter the list based on location and preferences.

4.3.2 Find Room

The find room page contains a list of all the users that have listing themselves as looking for a roommate, (already having a place to live). The user can filter the list based on location and preferences again. The rooms which are not shared are also listed here. The user can also view them.

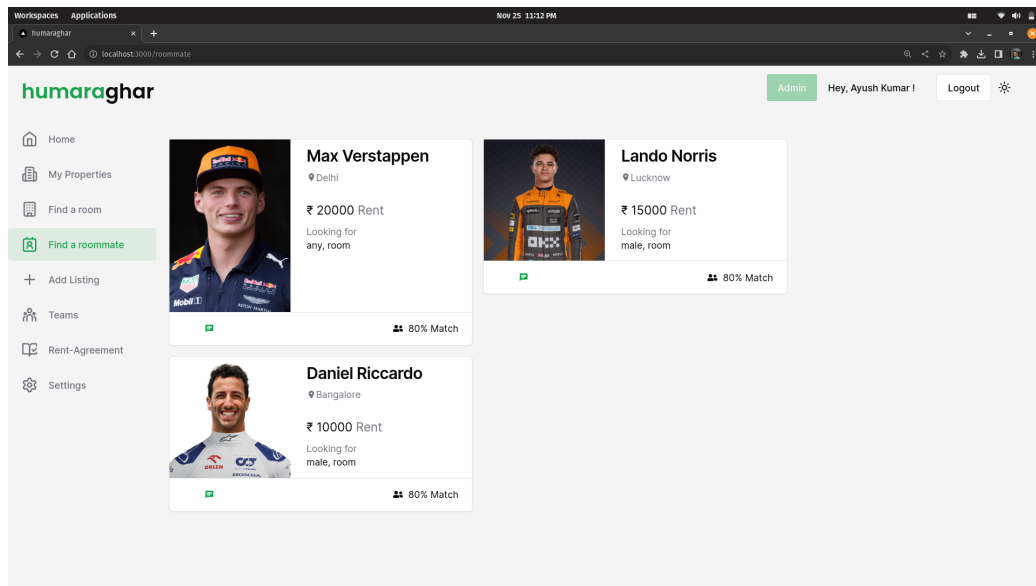


Figure 4.7: Find Roommate

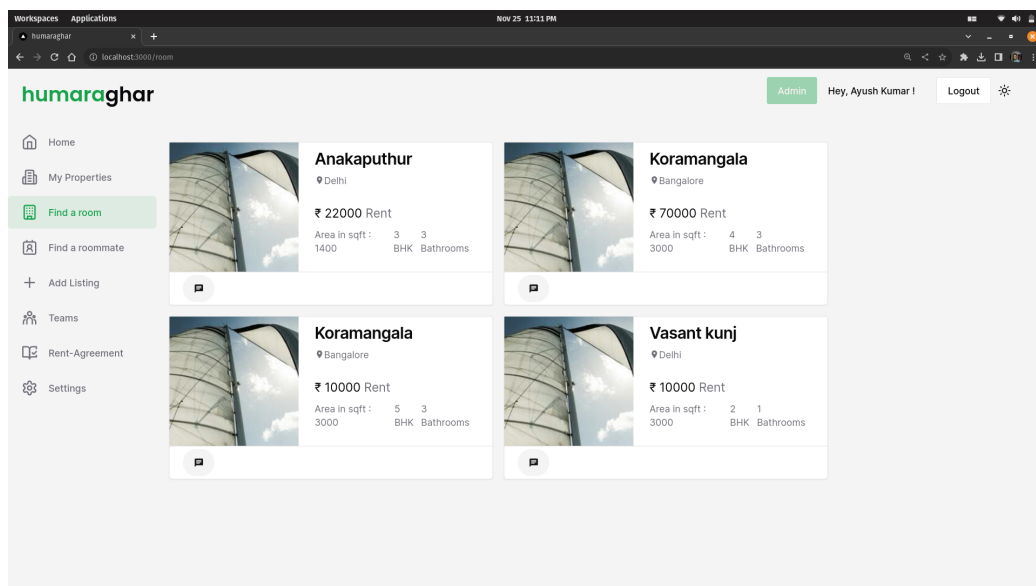


Figure 4.8: Find Room

The user can either view the detailed description of the room or the roommate by clicking the card. Or he/she can directly mark themselves interested in the property by clicking the interested button. It will prompt the user to select if they are interested as a team (if part of any) or individually.

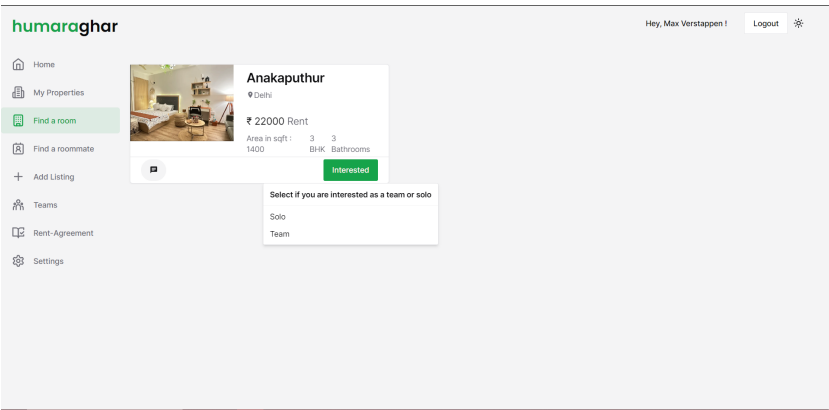


Figure 4.9: Interested As

Clicking as interested would notify the poster and then they can initiate a chat if they are willing to proceed further.

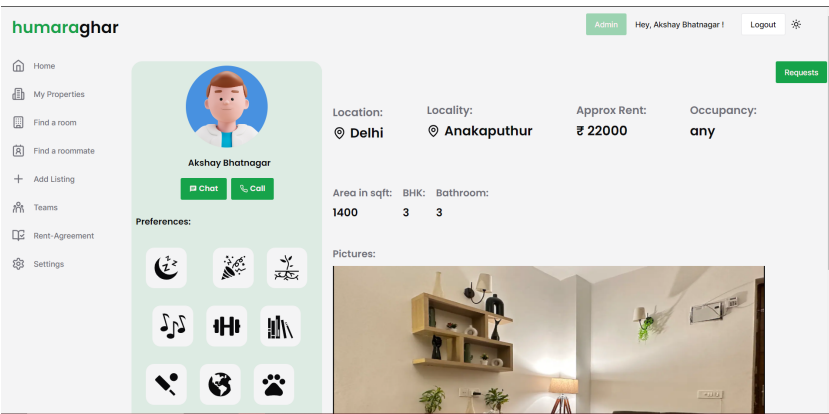


Figure 4.10: Details Page

4.3.3 Add Listing

The add listing page is used to add a new listing to the database. The user can add a listing for wanting to share their room, find a new room or rent their property.

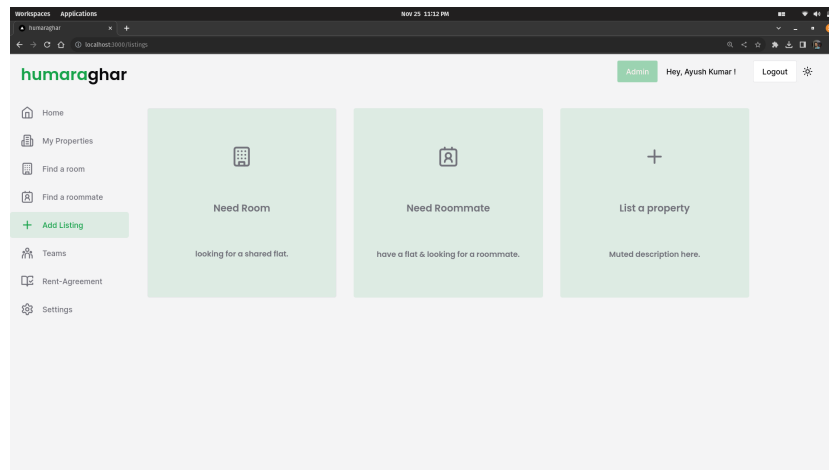


Figure 4.11: Add Listing

Choosing any option will take you to a form where you can fill in the details of the listing. The form for adding a property listing is shown here

A screenshot of the 'Add Room' form in the 'humaraghar' application. The form is divided into two columns. The left column contains fields for 'Address' (with a placeholder 'Address of your property'), 'Location' (a dropdown menu with 'Mumbai' selected), 'BHK' (a dropdown menu with '1' selected), 'Approx Rent' (a text input with 'e.g. 10000'), 'Contact Number' (a text input with '9999999999'), and 'Images' (a dashed box for uploads). The right column contains fields for 'Locality' (a dropdown menu with 'Adambakkam' selected), 'Area in sqft' (a text input with 'e.g. 1250'), 'Number of Bathrooms' (a text input with 'e.g. 2'), 'Occupancy' (radio buttons for 'Single', 'Shared', and 'Any'), and 'Date Available' (a date picker with 'Pick a date'). The sidebar on the left shows the 'Add Listing' option as active. The top right shows the user profile 'Hey, Ayush Kumar!' and a 'Logout' button.

Figure 4.12: Add Room

4.3.4 Team Management

The team management page is used to manage the teams that the user is a part of. The user can view the team that he/she is a part of or create a new team. The user can also view their sent invites and received invites with option to accept or decline them.

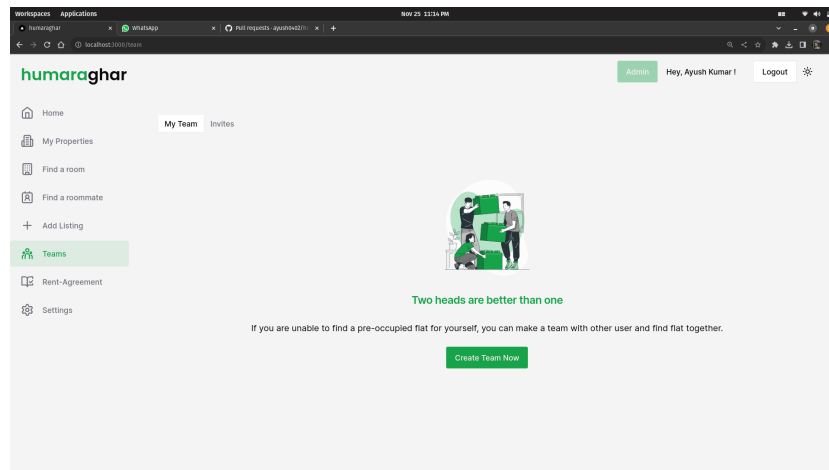


Figure 4.13: Create Team

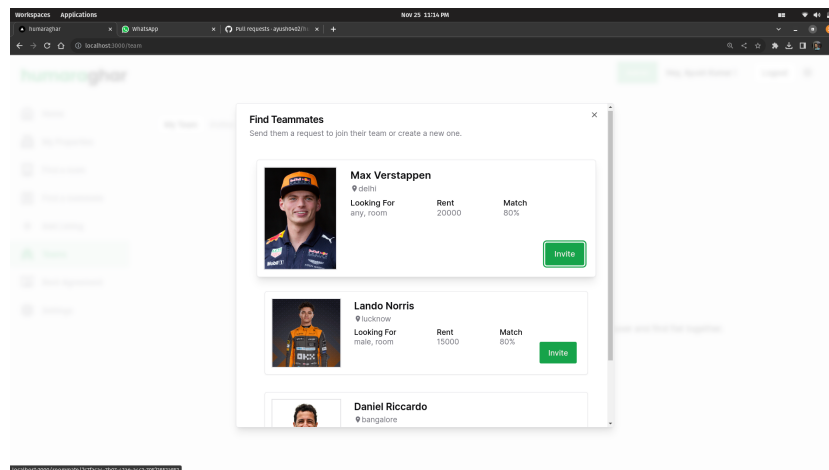


Figure 4.14: Team Invite Popup

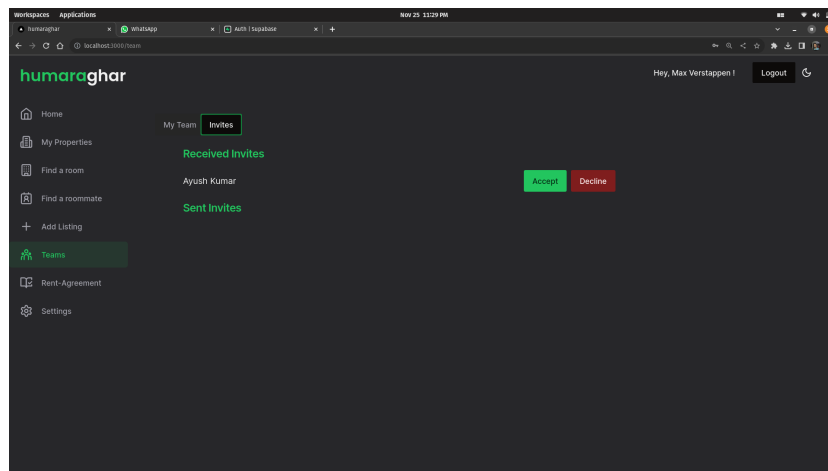


Figure 4.15: Pending Invites

4.3.5 Rent Agreement Generation

The rent agreement generation page is used to generate a rent agreement for the user. The user can fill in the details of the agreement and generate a pdf of the agreement.

Figure 4.16: Rent Agreement Form

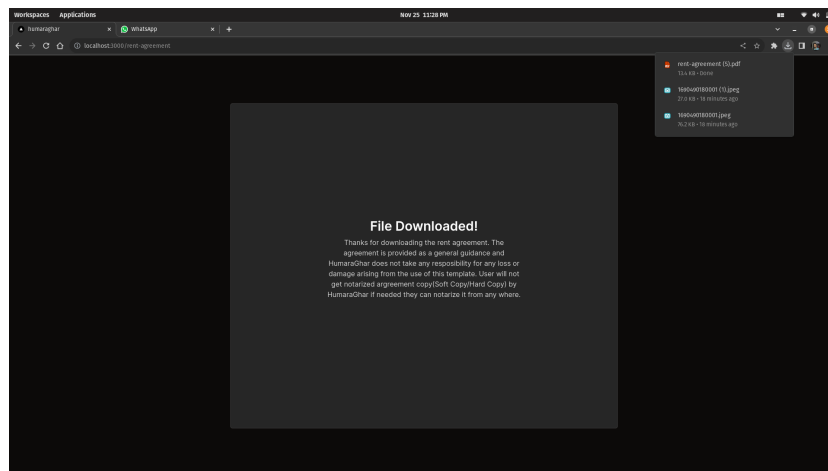


Figure 4.17: Rent Agreement Download

4.3.6 Admin Panel

To stop spam property listings, especially for non-shared rooms, they have to go through a manual acceptance before they are shown to the public. Currently only the project owners have admin access. The admin panel is used to view all the pending listings and accept or reject them.

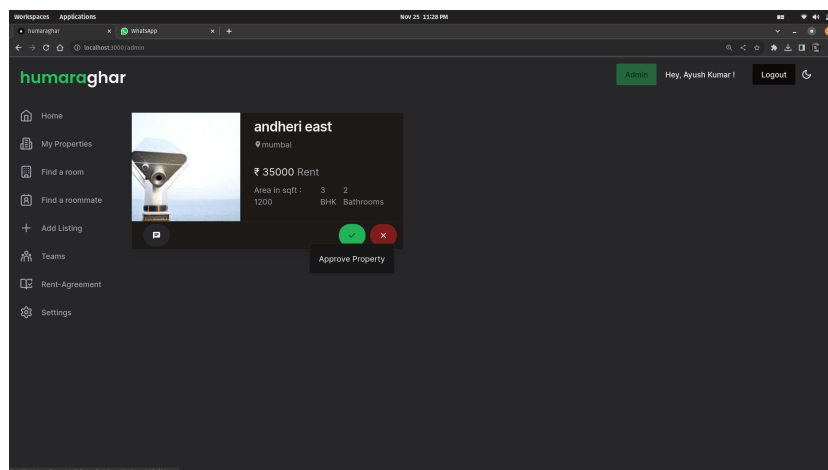


Figure 4.18: Admin Panel

Chapter 5

Conclusion and Future Work

This semester-long project has culminated in the development of a comprehensive room renting application that caters not only to young professionals seeking shared accommodations but also to families and individuals in pursuit of independent renting options. Originating from our own experiences as young graduates navigating new cities, the application's inception aimed to address the challenges faced by various demographic segments in securing suitable accommodations.

The application's unique proposition lies in its versatility, allowing users to form teams for collaborative housing searches, find compatible roommates, or pursue independent renting without sharing. Leveraging sophisticated machine learning models, the platform aids in suggesting optimal rent prices for listed properties, empowering users with informed decisions.

Throughout the developmental phase, a user-centric approach guided the application's design and functionality. By harnessing Next.js for the frontend and Supabase for the backend, the platform ensures a dynamic, scalable, and intuitive experience for a diverse user base.

This project marks a significant step forward in providing solutions beyond the initial challenges faced by bachelor students entering new cities. The application serves as a comprehensive housing solution, addressing the needs of young professionals, families, and individuals seeking varied housing arrangements.

Looking ahead, several areas offer opportunities for further development and expansion:

1. **Refinement of Machine Learning Algorithms:** Continuously improve the machine learning models to enhance the accuracy of rent price suggestions and roommate matchmaking, catering to the diverse needs of families and individuals.
2. **Geographical and Demographic Expansion:** Expand the application's reach to encompass a broader range of cities and regions, considering the specific requirements of families and individuals seeking independent renting options.
3. **Diversified User Experience:** Tailor the user experience to accommodate the distinct preferences of families and individuals, incorporating features like property size specifications, neighborhood suitability, and family-oriented amenities.
4. **Adding support for storage units:** Many users don't want to rent complete residential rooms but only someplace to store their belongings. While in western countries the concept of storage rooms are quite popular, there is still a lot of scope for it in India.
5. **Feedback-Driven Enhancements:** Implement a robust feedback mechanism to gather insights from families and individuals, enabling iterative improvements and customization based on their experiences and requirements.

In conclusion, this project marks a significant achievement in providing a comprehensive room renting application that caters to the diverse housing needs of families, individuals seeking independent renting, and young professionals. As we move forward, the focus remains on continuous improvement and expansion, driven by user feedback and evolving housing trends.

Appendix Title Here

Appendix A: Sample Rent Agreement

You can find the sample rent agreement here.

Appendix B: GitHub Repository of our project

You can find the GitHub repository of our project here.

Appendix C: GitHub Repository of this LaTeX report

You can find the GitHub repository of this LaTeX report here.

Bibliography

- [1] S. HARISH. Public social rental housing in india: Opportunities and challenges. *Economic and Political Weekly*, 51(5):49–56, 2016. ISSN 00129976, 23498846. URL <http://www.jstor.org/stable/44003134>.
- [2] S. P. Harsh Upadhyay. The growth of proptech startups in india post covid: Entrackr report. 1:16–18, 5 2023. URL <https://entrackr.com/2023/05/the-growth-of-proptech-startups-in-india-post-covid-entrackr-report/>.
- [3] J. India. Co-living reshaping rental housing in india. 1:16–18, 6 2019. URL <https://indiahousingreport.in/resources/co-living-reshaping-rental-housing-in-india/>.
- [4] A. Petkar, D. Macwan, and D. Takkekar. Urbanization and its impact on housing. *International Journal of Multidisciplinary Research*, 1:116–121, 11 2012.
- [5] I. Rautela. Co-living segment set for rapid expansion, predicts report. 1:1–2, 5 2023. URL <https://www.thehindubusinessline.com/news/real-estate/co-living-segment-set-for-rapid-expansion-predicts-report/article66823306.ece>.
- [6] P. Verma. House rent prediction: In depth analysis + models. 1:20, 2023. URL <https://www.kaggle.com/code/prashantverma13/house-rent-prediction-in-depth-analysis-models>.