TERRANEST – An intelligent property listing ecosystem

Prepared by

Md. Asad Reyaz	**211220100110028**
Tanir Sahoo	**211220100110023**
Aman Kumar Shah	**211220100110018**
Sankha Sengupta	**211220100110069**

Under the guidance of

SAMRAT SARKAR (Assistant Professor)

SILPI GHOSH (Assistant Professor)

Project Report

Submitted in the partial fulfillment of the requirement for the degree of

B.Tech in Computer Science and Engineering

Department of Computer Science and Engineering

St. Thomas' College of Engineering and Technology

Affiliated to

Maulana Abul Kalam Azad University of Technology, West Bengal

2024-2025

<div align="center">

St. Thomas' College of Engineering and Technology

Department of Computer Science and Engineering

</div>

This is to certify that the work in preparing the project entitled TERRANEST - AN INTELLIGENT PROPERTY LISTING ECOSYSTEM has been carried out by Md. Asad Reyaz, Tanir Sahoo, Sankha Sengupta & Aman Kumar Shah under my guidance during the session 2024-2025 and accepted in the partial fulfillment of the requirement for the degree of Bachelor of Technology.

_____

Samrat Sarkar
Assistant Professor
Department of Computer Science and Engineering
St. Thomas' College of Engineering and Technology

Dr. Mousumi Dutt
Associate Professor, HOD
Department of Computer Science and Engineering
St. Thomas' College of Engineering and Technology

_____

Silpi Ghosh
Assistant Professor
Department of Computer Science and Engineering
St. Thomas' College of Engineering & Technology

Acknowledgement

(Md. Asad Reyaz)

RollNo.   12200121014

(Tanir Sahoo)

Roll No.   12200121057

(Aman Kumar Shah)

 RollNo.   12200121038

(Sankha Sengupta)

RollNo.   12200221047

St. Thomas' College of Engineering and Technology

Department of Computer Science and Engineering

Declaration

We declare that this written submission represents our ideas in our own words and we have adequately cited and referenced the original sources. We also declare that we have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

(Md. Asad Reyaz)

RollNo.   12200121014

(Tanir Sahoo)

Roll No.   12200121057

(Aman Kumar Shah)

 RollNo.   12200121038

(Sankha Sengupta)

RollNo.   12200221047

Originality report

COURSE NAME

plag t

STUDENT NAME

Tanir Sahoo

FILE NAME

Final Year Project Report.docx

REPORT CREATED

Jun 9, 2025

Summary

Flagged passages	7	1%
Cited/quoted passages	2	0.4%

Web matches

stcet.ac.in	2	0.5%
glsuniversity.ac.in	1	0.3%
sushantuniversity.edu.in	1	0.3%
gitanjaliinternationalschool.com	1	0.1%
richr.com	1	0.1%
timescale.com	1	0.1%
stackoverflow.com	1	0.1%
kvpagrasencollege.in	1	0.1%

1 of 9 passages

Student passage   FLAGGED

Submitted in the partial fulfillment of the requirement for the degree of

Top web match

and **submitted in the partial fulfillment of the requirement for** the award of **the degree of** Doctor of Philosophy of the University of Rajasthan, represents my ideas in my own words and where others'…

DECLARATION   https://kvpagrasencollege.in/wp-content/uploads/2024/02/FORMAT-14.pdf

Student passage   FLAGGED

DeclarationWe **declare that this written submission represents** our **ideas in** our **own words** and we **have adequately cited and referenced the original sources**. We **also declare that** we **have adhered to all**…

Top web match

Declaration I **declare that this written submission represents** my **ideas in** my **own** words and where others' ideas or **words** have been included, I **have adequately cited and referenced the original sources**.…

Declaration - GLS University https://www.glsuniversity.ac.in/docs/declaration-of-academic-honesty-and-Integrity.pdf

---

Student passage   CITED

understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom…

Top web match

I **understand that any violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have not been properly cited or from whom**…

Thesis-Writing-Guidelines -PhD.pdf - Sushant University https://sushantuniversity.edu.in/admin-assets/uploaddata/Thesis-Writing-Guidelines%20-PhD.pdf

---

Student passage   FLAGGED

To Continually improve upon the teaching-learning processes and research with a goal to develop quality technical manpower with sound academic and practical experience, who can respond to challenges…

Top web match

To continually improve upon the teaching-learning processes and research with a goal to develop quality technical manpower with sound academic and practical experience, who can respond to challenges…

Vision & Mission - CSE - St. Thomas' College of Engineering ... https://stcet.ac.in/department/department-of-computer-science-engineering/vision-mission-cse/

Student passage   FLAGGED

To inspire the students to work with latest tools and to make them industry ready.**To impart research based technical knowledge**. **To groom the department as a learning** center **to inculcate advanced**…

Top web match

To inspire the students to work with latest tools and to make them industry ready. **To impart research based technical knowledge**. **To groom the department as a learning** centre **to inculcate advanced**…

Vision & Mission - CSE - St. Thomas' College of Engineering …   https://stcet.ac.in/department/department-of-computer-science-engineering/vision-mission-cse/

---

6 of 9 passages
Student passage   FLAGGED

PO1: Critical Thinking and Problem Solving:College projects encourage **students to approach** challenges **from different angles, enhancing their analytical skills**

Top web match

Whether it's solving a scientific challenge, exploring historical events, or tackling social issues, students engage in the process of inquiry, research, and critical evaluation. PBL encourages …

How Modern Schools Foster Critical Thinking in Students   https://gitanjaliinternationalschool.com/how-do-modern-schools-foster-critical-and-analytical-thinking-among-students/

---

7 of 9 passages
Student passage   FLAGGED

By incorporating multimedia content, the **system not only** improves the user experience but also **helps users make informed decisions**.

Top web match

They actively encourage users to leave reviews and ratings for the properties they have visited or rented through the platform. This feedback **system not only helps** other **users make informed decisions**…

Is HomeFinder a Reliable Source for Finding Legitimate Listings?   https://richr.com/blog/is-homefinder-a-reliable-source-for-finding-legitimate-listings/

---

8 of 9 passages
Student passage   FLAGGED

…between user queries and PG data is measured via **cosine similarity**, which **calculates the cosine of the angle between two vectors**. Scores close to 1 indicate high semantic similarity, while…

Top web match

Cosine similarity calculates the cosine of the angle between two vectors, revealing how closely the vectors are aligned. For instance, words like "cat" and "dog" will have a higher cosine similarity…

A Guide to Cosine Similarity - Timescale   https://www.timescale.com/learn/understanding-cosine-similarity

---

9 of 9 passages
Student passage   QUOTED

…of connectivity issues with clear, actionable messages such as "**Unable to fetch data, please try again later**."

Top web match

fail(function() { alert('**Unable to fetch data, please try again later**.') }); } else alert('Unknown row id.'); I want is to redirect in new page when I click ...

javascript - Jquery Redirect in new page - Stack Overflow https://stackoverflow.com/questions/42966275/jquery-redirect-in-new-page

INDEX

I. PREAMBLE

VISION & MISSION

Vision of the institution :

To evolve as an industry-oriented, research-based institution dedicated to fostering innovation, creativity, and practical solutions across various domains. The vision aims to:

- Contribute meaningfully to societal development by addressing real-world challenges through advanced research and technological interventions.
- Establish a global presence as a hub of academic excellence and innovation, consistently delivering solutions that meet the dynamic needs of industry and society.
- Foster an inclusive and diverse environment where collaboration and knowledge-sharing drive transformative growth.

The Institute's Mission :

- To enhance the quality of education by implementing cutting-edge teaching methodologies and creating a state-of-the-art infrastructure that supports innovative learning experiences.
- To promote a culture of research and development by encouraging interdisciplinary collaborations and providing the necessary resources for groundbreaking studies and technological advancements.
- To prepare students and faculty to thrive in a rapidly evolving world by instilling professional knowledge, leadership qualities, and entrepreneurial skills combined with strong ethical and moral values.
- To actively engage with industries and communities to identify and address emerging challenges, ensuring that academic outputs align with real-world applications.
- To advocate for sustainable development by integrating environmentally conscious practices and technologies into education, research, and operational frameworks.
- To create lifelong learners who can adapt to global challenges and make meaningful contributions to building a knowledge-driven and sustainable society.

The Department of Computer Science and Engineering's Vision :

To Continually improve upon the teaching-learning processes and research with a goal to develop quality technical manpower with sound academic and practical experience, who can respond to challenges and changes happening dynamically in Computer Science and Engineering.

The Department of Computer Science and Engineering's Mission :

- To inspire the students to work with latest tools and to make them industry ready.
- To impart research based technical knowledge.

- To groom the department as a learning centre to inculcate advanced technologies in Computer Science and Engineering with social and environmental awareness.

II. PROGRAM OUTCOMES (PO) AND PROGRAM SPECIFIC OUTCOMES (PSO)

Program Outcomes (PO) :

PO1:Critical Thinking and Problem Solving:
College projects encourage students to approach challenges from different angles, enhancing their analytical skills. By identifying problems and brainstorming multiple solutions, students develop a deep understanding of how to assess situations logically. This ability to think critically and solve problems prepares students for complex scenarios they may encounter in their careers.

PO2:Research and Innovation:
Projects push students to explore emerging trends, technologies, and methodologies through extensive research. This research-oriented approach nurtures creativity and enables students to generate unique solutions, fostering innovation. By thinking outside the box, students are motivated to discover new concepts that may contribute to advancements in their field of study.

PO3:Technical Proficiency:
Hands-on projects allow students to apply the theoretical knowledge they have gained in the classroom to practical tasks, strengthening their technical skills. Whether it's coding, designing, or conducting experiments, students develop

a mastery of the tools and techniques necessary for success in their discipline. This technical competence prepares them for real-world job roles that require specialized skills.

PO4:Teamwork and Collaboration:

Many college projects involve working in teams, helping students develop interpersonal and collaborative skills. By engaging in discussions, delegating tasks, and working towards common goals, students learn how to communicate effectively and resolve conflicts. Teamwork also teaches students the value of diverse perspectives and how to combine strengths to achieve superior results.

PO5:Time Management and Planning:

Successfully completing a project requires careful planning and time management. Students must break down tasks, allocate resources, and prioritize responsibilities to meet deadlines. This process teaches students how to balance multiple activities, a crucial skill in both academic and professional settings where deadlines are strict.

PO6:Adaptability and Flexibility:

In the course of a project, unexpected obstacles and changes in requirements are common. Students learn how to remain flexible and adapt to these shifts, whether it's adjusting timelines, rethinking approaches, or altering scope. This adaptability is a vital skill in the workplace, where rapid changes and evolving challenges are frequent.

PO7:Application of Knowledge:

College projects bridge the gap between theory and practice by allowing students to apply what they have learned in real-world scenarios. This helps solidify their understanding and makes abstract concepts more tangible. By working on relevant projects, students gain valuable insights into how their academic learning can be directly translated into industry applications.

PO8:Ethical and Responsible Conduct:

Students are often required to consider the ethical implications of their work, from respecting intellectual property to ensuring the sustainability of their solutions. This fosters a strong sense of responsibility, teaching students the importance of adhering to ethical standards and promoting fairness, transparency, and integrity in their professional lives.

PO9:Communication Skills:

Completing a project requires students to present their findings and ideas effectively. Whether through written reports, presentations, or discussions, students learn to communicate complex concepts in a clear and concise manner. This enhances their ability to articulate technical information to both technical and non-technical audiences, a valuable skill in any career.

PO10:Project Management:

Students gain hands-on experience in managing all aspects of a project, including defining goals, allocating resources, setting deadlines, and tracking progress. By taking ownership of the entire project lifecycle, they develop organizational skills and learn how to lead and motivate a team. These project management abilities are directly transferable to any professional role, especially in leadership positions.

PO11:Self-Confidence and Independence:

By taking responsibility for the planning and execution of a project, students build their self-confidence. They learn to trust their judgment and develop the ability to work independently, which increases their initiative and decision-making capabilities. This sense of autonomy fosters personal growth and prepares students for leadership roles in their careers.

PO12:Professional Development:

College projects provide a platform for students to develop skills that are highly valued by employers, such as problem-solving, collaboration, and technical expertise. Working on real-world problems enhances students' industry readiness, making them more attractive candidates in the job market. Additionally, completing projects adds value to their portfolios, showcasing their practical experience and commitment to professional development.

Program Specific Outcomes (PSO) :

PSO1: User-Centric Design and Usability:

Students will develop skills in creating user-friendly interfaces for online platforms, ensuring that the rental website is intuitive and easy to navigate. By conducting usability testing, students will refine the design to meet the needs of different user segments, such as property seekers and property owners. This outcome emphasizes the importance of delivering a seamless user experience.

PSO2: Integration of Machine Learning for Smart Suggestions:

Students will gain hands-on experience in implementing machine learning algorithms, such as recommendation systems, to provide personalized property suggestions to users. This will enhance their understanding of how to integrate AI technologies into web applications, improving decision-making processes for users based on their preferences and behaviors.

PSO3: Backend Development and Database Management:

The project will provide experience in backend development using frameworks like Spring Boot and database management with technologies like MySQL. Students will work on designing efficient database schemas for storing user and property data, ensuring data integrity, security, and optimal performance for a smooth user experience.

PSO4: Security and Data Privacy Management:

As users will input sensitive data (e.g., contact details, preferences), students will learn the importance of data security measures. This includes implementing user authentication, password encryption, and secure data storage practices. Additionally, they will ensure compliance with data privacy regulations to protect user information and build trust.

PSO5: Real-time Data Processing and Notifications:

Students will develop skills in implementing real-time features such as notifications, live updates for property availability, and chat support. They will integrate technologies like WebSockets or polling mechanisms to provide users with instant information, enhancing the interactivity and responsiveness of the platform.

III. Justification

Program Specific Outcome (PSO)	Program Outcome (PO)	Justification
PSO1: Web Development & Backend Integration	PO3: Technical Proficiency	Developing a web-based platform requires proficiency in both front-end (HTML, CSS, JavaScript) and back-end (Spring Boot, MySQL) technologies.
	PO4: Teamwork and Collaboration	Collaboration is crucial for integrating the front-end and back-end, ensuring smooth data exchange and functionality.
	PO10: Modern Tool Usage	Using modern tools like React, Spring Boot, and MySQL ensures the application is scalable and performs well.
	PO11: Project Management	Managing the development process, testing, and deployment of both the front-end and back-end.
PSO2: Database Management & Optimization	PSO2: Database Management & Optimization	Designing efficient databases and queries involves critical thinking to ensure data integrity and optimization.
	PO3: Technical Proficiency	Proficiency in SQL databases and query optimization is vital for the system's performance.

	PO6: Environmental and Societal Impact	A well-optimized database reduces resource consumption, contributing to environmental sustainability.
	PO10: Modern Tool Usage	Integrating tools like MySQL or PostgreSQL for database management ensures reliability and scalability.
PSO3: Machine Learning Integration	PO1: Critical Thinking and Problem Solving	Integrating machine learning requires problem-solving skills for data preprocessing, model training, and evaluation.
	PO3: Technical Proficiency	Knowledge of machine learning algorithms and frameworks is essential for implementing property recommendation models.
	PO4: Teamwork and Collaboration	Collaborating with other developers to integrate the machine learning model into the application's back-end.
	PO7: Communication Skills	Communicating the functionality and benefits of the machine learning model to the team and stakeholders.
PSO4: Security Implementation & User Privacy	PO1: Critical Thinking and Problem Solving	Implementing security measures like encryption and authentication to protect user data requires careful planning.
	PO5: Time Management and Planning	Security features should be implemented within deadlines without affecting the overall project timeline.
	PO8: Ethical and Responsible Conduct	Ensuring user privacy and securing sensitive information like contact details in compliance with ethical standards.
	PO12: Leadership and Accountability	Leading the implementation of security features and ensuring that they align with the latest security protocols.
PSO5: Real-time Features &	PO1: Critical Thinking and Problem	Implementing real-time features

Interaction	Solving	requires solving challenges related to data flow and synchronization.
	PO3: Technical Proficiency	Proficiency in integrating real-time features such as notifications using WebSockets or Firebase.
	PO4: Teamwork and Collaboration	Effective collaboration between front-end and back-end teams to ensure smooth interaction with real-time updates.
	PO11: Project Management	Coordinating the development of real-time features while ensuring the overall project stays on track and meets deadlines.

Abstract

The Smart PG Recommendation Website is a dynamic and innovative platform designed to transform the process of finding Paying Guest (PG) accommodations for students and working professionals. This project integrates advanced search and filtering mechanisms with machine learning algorithms to provide users with personalized property suggestions based on their preferences. The platform addresses the challenges associated with traditional PG search methods, such as lack of transparency, time inefficiency, and limited customization options.The core functionality of the website includes smart search and filtering, where users can filter properties based on parameters like location, budget, amenities, and room type. Personalized recommendations are generated by a machine learning model, using techniques like rule-based algorithms, which analyze user interactions to recommend PG options tailored to individual preferences. Verified listings and user feedback are incorporated to enhance trust; all listings are verified, and users can review and rate properties, helping others make informed decisions. The platform is designed with a responsive and scalable structure, ensuring it works seamlessly across devices, and is hosted on a cloud platform for scalability and reliability.

In addition to its user-focused features, the website ensures data privacy and security by adhering to standards such as GDPR. The database is optimized to manage large volumes of data efficiently, enabling smooth interactions even during peak usage. The project employs modern web development technologies, including Spring Boot and MySQL, to ensure a robust and responsive system. By bridging the gap between technology and real-world housing challenges, the Smart PG Recommendation Website sets a benchmark for convenience, transparency, and innovation in the property search domain. The project not only simplifies the PG hunting process but also fosters a secure and engaging user experience, making it a valuable resource for its target audience.

Chapter 1

Introduction

Finding suitable Paying Guest (PG) accommodations has traditionally been a labor-intensive and inefficient process for students and working professionals. Individuals often rely on outdated listings, word of mouth, or scattered online platforms that lack real-time data, verified information, and personalized recommendations. These limitations not only lead to time-consuming searches but also result in mismatches between user expectations and available accommodations.

To address these challenges, the **Smart PG Recommendation Website** has been developed—a modern, intelligent, and scalable web platform that leverages cutting-edge technologies to streamline the PG search experience. Built using **Spring Boot** for backend services and **MySQL** for structured data storage, the platform ensures robust and secure data handling. The frontend is designed for ease of use, with intuitive navigation and responsive UI components.

At the core of this platform lies a **machine learning-based recommendation engine**, specifically a **rule-based model** that prioritizes key attributes such as location, cost, type of PG, and amenities. Unlike traditional recommendation systems that rely purely on collaborative or content-based filtering, the rule-based approach is highly interpretable and tailored to prioritize user-specific preferences. For instance, the system ensures that recommendations align with the state or city that the user has most frequently searched for, followed by sorting results based on budget and other preferences like food availability or proximity to transportation hubs.

The system processes the **search history of each user** to identify dominant patterns and preferences. These preferences are then matched against a curated and verified database of PG listings to generate highly relevant and customized recommendations. This model does not require labeled outcomes for training, making it inherently flexible and easy to maintain—ideal for dynamic, real-world applications.

To further enhance user experience, the platform supports **dynamic filtering and search** options, allowing users to narrow down their search based on facilities (Wi-Fi, laundry, meals), duration of stay, room type (single/double/triple), and more. All property listings are vetted for authenticity, and the platform includes a **review and rating system** to promote transparency and trust.

Data privacy and security are integral components of the system architecture. The application complies with **General Data Protection Regulation (GDPR)** standards to ensure that user data is stored securely and handled responsibly. The use of secure APIs, encrypted communications, and access control mechanisms further protects user information from unauthorized access.

The platform is deployed on a **cloud infrastructure**, offering **high availability, scalability**, and **disaster recovery** features. This ensures that the website remains responsive and functional even during peak usage, such as at the start of academic terms when PG demand surges.

In summary, the **Smart PG Recommendation Website** revolutionizes the traditional PG search model by combining **machine learning**, **modern web development**, and **cloud deployment** to offer a fast, secure, and personalized experience. With its intelligent recommendation engine and user-centric design, the platform not only saves users valuable time but also improves the quality and trustworthiness of PG accommodations in India and beyond.

1.1 Objective of the project

The primary objective of the **Smart PG Recommendation Website** is to simplify and personalize the process of finding suitable Paying Guest (PG) accommodations for students and working professionals by leveraging modern web technologies and machine learning techniques.

This project aims to address key limitations of existing PG search methods, such as lack of personalization, inefficient filtering, and unverified listings. By integrating a **rule-based recommendation system** that utilizes **user search history**, the platform can intelligently suggest PGs that closely match the user's preferences in terms of **location, cost, room type, duration, and features**.

The specific objectives of the project are:

- **To develop an intelligent recommendation engine** using a rule-based algorithm that analyzes a user's search behavior and suggests PGs based on dominant preferences.
- **To create a secure and scalable web platform** using technologies like **Spring Boot** for backend services and **MySQL** for structured data management.
- **To improve search efficiency** by implementing dynamic filters that allow users to refine results based on specific parameters such as budget, proximity to institutions, availability of amenities, and preferred room types.
- **To enhance transparency and trust** by integrating verified PG listings and enabling user-generated reviews and ratings.
- **To ensure data privacy and security** by complying with data protection regulations like GDPR, and adopting encrypted communications and access controls.
- **To deploy the platform on cloud infrastructure** for better availability, performance, and scalability during periods of high user activity.

By achieving these objectives, the Smart PG Recommendation Website seeks to deliver a user-friendly, intelligent, and reliable solution to the accommodation search problem, ultimately improving user satisfaction and reducing the effort and time involved in finding a suitable PG.

1.2 Brief description of the project

The **Smart PG Recommendation Website** is a web-based platform designed to assist users—particularly students and working professionals—in finding Paying Guest (PG) accommodations that best match their personal preferences and requirements. The system offers intelligent, personalized recommendations by analyzing the user's past search behavior using a **rule-based machine learning model**.

The project leverages a clean, intuitive user interface for users to browse PG listings, apply dynamic filters, and view recommendations based on key factors such as **location, room type, cost, amenities, and duration of stay**. A core component of the system is its **smart recommendation engine**, which operates on a rule-based approach rather than complex supervised ML models. This model identifies user preferences (e.g., preferred state or budget) by analyzing their historical interactions and prioritizes recommendations accordingly.

Technologically, the platform is developed using **Spring Boot** for the backend, **MySQL** for data storage, and **HTML/CSS/JavaScript** for the frontend. It also supports **cloud deployment** for better performance, scalability, and uptime.

By integrating user reviews, secure authentication, verified listings, and intelligent filtering, the project delivers a smarter, more trustworthy solution to the problem of PG search. The platform aims to save users time, reduce decision fatigue, and increase the likelihood of finding an ideal accommodation quickly and securely.

1.3 Tools and Platform

The development of the *Smart PG Recommendation System* incorporated a range of modern programming languages, libraries, platforms, and tools to ensure a modular, scalable, and seamless full-stack solution. Each tool was selected to meet specific functional, integration, and performance needs across the frontend, backend, ML engine, and payment processing.

➢ **Programming Languages and Frameworks**

- **Frontend**:
  - **HTML**, **CSS**, **JavaScript** – Used to design and build the responsive and interactive user interface.
- **Backend**:
  - **Java** with **Spring Boot** – Manages backend logic, RESTful services, and database interaction.
  - **Python** – Implements the ML-based recommendation engine.
    - Key Python libraries and frameworks : `Flask==3.0.3, requests==2.31.0, nltk==3.8.1, sentence-transformers==2.6.1, torch >= 1.9.0, scipy >= 1.10.0, transformers >= 4.30.0, numpy >= 1.23.0`
- **Database**:
  - **MySQL** – Used as the structured database to store PG listings and bed-level details.

➢ Platforms and Development Tools

- **Visual Studio Code (VSCode)** – Main development environment for frontend and ML logic.
- **IntelliJ IDEA** – IDE for Spring Boot backend development.
- **MySQL Workbench** – Visual tool for managing MySQL databases.
- **Windows Subsystem for Linux (WSL)** – Hosted the Flask server, allowing native Linux support for ML tools.
- **Postman** – API testing and validation.
- **GitHub Desktop** – Version control and collaboration.
- **Google Chrome** – Used for web testing and debugging.
- **Stripe** – Integrated payment platform to simulate secure and seamless payment workflows for PG bookings.

➢ Server Deployment and Configuration

Component	Technology	Port	Role
Spring Boot Backend	Tomcat Server	9000	Exposes PG and bed data via REST APIs
ML Recommendation API	Flask Server (WSL)	8000	Processes search queries and returns recommendations
Database Server	MySQL Server	6500	Manages structured PG data
Frontend Client	Apache Web Server	5300	Delivers the web-based user interface

1.4 Project Organization

The development of the Smart PG Recommendation Website followed a structured approach across seven distinct stages to ensure a smooth and efficient build process:

1. **Requirement Analysis and Planning**

   This phase involved identifying the key requirements of the project, including user roles (PG seekers, PG owners), features (recommendation engine, search filters), and non-functional aspects like data security and responsiveness. We planned the tech stack, set timelines, and allocated tasks among team members.

2. **System Design and Database Development**

   We designed the overall architecture of the website using a layered model, ensuring separation of concerns between UI, business logic, and data layers. The database schema was built using MySQL to store PG listings, user profiles, and search history. Proper indexing and normalization were applied to optimize performance.

3. **Backend API Development**

   RESTful APIs were developed using Spring Boot for operations like user authentication, PG listing management, search logging, and generating recommendations. The rule-based recommendation model was also integrated at this stage using Python logic executed alongside backend services.

4. **Frontend Development**

   The user interface was created using modern frontend technologies (like HTML, CSS, JavaScript, and optionally React/Angular). Responsive design principles ensured that the website functioned well across different screen sizes. Key features included PG listing views, search filters, and a recommendation section.

5. **Testing and Integration**

   Functional, unit, and integration testing were conducted to verify the accuracy of recommendation logic, data retrieval from the database, and API interactions. Tools like Postman and JUnit were used to test backend APIs, while frontend interactions were validated through manual and automated testing.

6. **Deployment and Maintenance**

   The project was deployed locally for demonstration purposes, ensuring all services and APIs ran correctly in the target environment. We prepared documentation for system setup and created scripts for database initialization and service startup.

7. **Feedback and Iteration**

   Based on initial user feedback, several enhancements were made—like refining the recommendation algorithm to prioritize location-based preferences more accurately. The interface was polished for better usability, and data inconsistencies were resolved to improve recommendation quality.

1.5 ProjectTimeline

Month/Phase	1	2	3	4	5	6	7	8	9	10
Phase 1: Requirement Analysis and Planning	██	██								
Phase 2: System Design and Database development			██							
Phase 3: Backend API Development				██						
Phase 4: Frontend Development					██	██				
Phase 5: Testing and Integration							██	██		
Phase 6: Deployment and Maintenance									██	
Phase 7: Feedback and Iteration										██

Chapter 2

Literature Survey

With the increasing adoption of digital technologies in the real estate sector, online platforms have become essential tools for individuals seeking rental accommodations. Traditional property portals like **Zillow**, **Craigslist**, **99acres**, and **MagicBricks** provide generalized interfaces for listing and browsing properties, including houses, apartments, and commercial spaces. These platforms primarily cater to long-term rentals, outright property sales, and commercial listings. However, they often fail to cater to the nuanced needs of **Paying Guest (PG)** accommodations, which require tailored search mechanisms for students, working professionals, and short-term residents.

The shift towards digital platforms has been accompanied by the integration of **Artificial Intelligence (AI)** and **Machine Learning (ML)** to enhance user experiences and deliver more relevant search results. In particular, **Recommendation Systems (RS)** have become vital in domains where user preferences vary widely and customization is key. These systems utilize approaches such as **content-based filtering** (where recommendations are based on item features and user profiles), **collaborative filtering** (which draws from the behaviors and preferences of similar users), and **hybrid methods** that combine the strengths of both [1]. When applied to the rental and PG accommodation domain, these models can dramatically improve relevance, reduce search time, and increase user satisfaction.

Websites such as **NestAway**, **Zolo Stays**, and **Colive** have taken significant steps toward leveraging ML to offer user-specific PG accommodations based on factors such as **budget**, **room type**, **location**, and **available amenities**. These platforms have built-in algorithms that consider basic user inputs and preferences, offering predefined filters for more refined searches. Their use of AI tools—like **filter prioritization**, **location-based suggestions**, and **user clustering**—has contributed to more intelligent service delivery [2]. However, most of these platforms still depend heavily on **static filtering techniques**, often overlooking the evolving behavioral trends and detailed search history of individual users, which can significantly enhance recommendation accuracy.

TerraNest, the proposed platform in this project, attempts to bridge this gap by introducing a **rule-based recommendation system** that dynamically analyzes **user interaction data**, including **search frequency**, **filter preferences**, **geographic interest**, and **feature selection trends**. Unlike systems that require substantial training datasets and complex model tuning, rule-based systems are **interpretable, faster to implement**, and well-suited for structured use-cases with relatively consistent user needs. This model captures patterns like repeated searches in specific locations, preference for certain amenities, and budget ranges, to provide **personalized PG recommendations**. The design of TerraNest is in line with the user-centric model proposed by Ko et al., where interpretability and domain specificity lead to greater recommendation effectiveness [1].

Beyond intelligent recommendation, TerraNest enhances user experience through **real-time listing updates**, minimizing issues related to **outdated availability**, **incorrect pricing**, or **duplicate listings**—common problems in

traditional listing platforms [1]. Users can also leverage **advanced filtering options** such as **gender-specific PGs**, **meal inclusion**, **WiFi availability**, and **room-sharing preferences**, thereby enabling a **multi-dimensional search experience** that is both fast and highly relevant [2].

An **innovative feature** integrated into TerraNest is the **Roommate Compatibility Survey**, which evaluates soft factors like **cleanliness**, **work schedules**, **noise tolerance**, and **social behavior**. These elements are often overlooked by traditional platforms but have a significant impact on co-living satisfaction. Incorporating such a mechanism is backed by studies suggesting that **social alignment** among roommates reduces friction and improves long-term tenancy [3].

Additionally, the system includes a **Local Landmark Intelligence Module** that allows users to view PG accommodations in relation to **key urban landmarks** such as **colleges**, **IT hubs**, **metro stations**, and **healthcare facilities**. This module aids especially **first-time movers**, helping them make informed decisions based on not just PG attributes but also environmental factors [3].

Comparing TerraNest to other platforms:

- **NestAway** provides AI-based filters but largely focuses on shared apartments and is not exclusive to PG accommodations [2].
- **Zolo Stays** integrates managed services such as food and cleaning but does not heavily utilize behavioral search analytics [1].
- **Colive** focuses on serviced co-living spaces and mobile-first operations, but emphasizes lifestyle amenities over search personalization [3].

In contrast, **TerraNest is purpose-built** for PG accommodation, addressing a critical gap in existing platforms. Its **rule-based ML system**, **search behavior analysis**, **PG-specific filtering**, and **context-aware features** like roommate matching and landmark proximity position it as a **next-generation solution** in the student and professional rental market. The platform's core strength lies in its ability to **learn from user behavior** with minimal computational overhead, making it **lightweight, accurate, and scalable**. This strategic alignment with emerging research in recommendation systems ensures TerraNest remains adaptable to future enhancements while maintaining its user-centric vision [1][2][3].

Chapter 3

Concepts and Problem Analysis

Introduction to the Problem Domain

In the current landscape of rapid urbanization and increasing student and workforce mobility, the demand for reliable and comfortable PG (Paying Guest) accommodations has surged dramatically. Major metropolitan areas and educational hubs are experiencing a high influx of individuals seeking temporary yet quality living spaces that suit their preferences in terms of budget, location, gender-specific housing, and available amenities such as Wi-Fi, food, air-conditioning, and proximity to public transport.

Traditionally, the process of finding PG accommodations has been labor-intensive, inconsistent, and often frustrating. Individuals typically rely on word-of-mouth referrals, physical visits, online classified ads, or local property brokers to find suitable options. However, each of these methods comes with significant limitations. For instance, word-of-mouth referrals are often limited to a person's immediate social circle and do not guarantee availability or suitability. Online classifieds may be outdated, lack essential filters, or contain misleading information. Local brokers, on the other hand, may charge high commissions and often provide limited options that may not meet the user's full set of requirements. Moreover, these methods usually offer little to no personalization based on a user's specific needs, lifestyle preferences, or budget constraints.

With the advent of digital technologies and widespread internet usage, there is a growing expectation among users for intelligent, fast, and personalized service delivery across all domains — including housing. Users now expect platforms that not only allow them to search listings but also understand their preferences, predict their needs, and offer relevant, customized recommendations. The lack of such an intelligent PG search mechanism creates a major usability gap and negatively impacts user satisfaction and decision-making.

This project — the Smart PG Recommendation System — seeks to address these challenges by leveraging the power of semantic search and intelligent filtering. Unlike conventional PG listing websites, which require users to manually sift through a long list of results, our system allows users to enter natural language queries like "Girls PG near Salt Lake with WiFi under 6000" and receive relevant, ranked results. The system goes beyond simple keyword matching and uses natural language processing (NLP) and machine learning (ML) techniques to understand the meaning of the query and the semantic content of PG descriptions.

Furthermore, the system is designed to be scalable, modular, and user-friendly. It combines an interactive and responsive frontend interface with a robust backend powered by Flask and Spring Boot APIs. At the core of the recommendation logic lies a semantic similarity model from the `sentence-transformers` library, which converts both user queries and PG descriptions into embeddings (numerical vectors) that represent their meaning.

These embeddings are compared using cosine similarity to identify and rank the PGs most relevant to the user's intent. This approach not only streamlines the PG discovery process but also significantly improves the accuracy and relevance of search results, enhancing the overall user experience. It demonstrates how cutting-edge technologies such as semantic search, RESTful APIs, and modular web development can be harnessed to solve everyday real-world problems with efficiency and precision.

Ultimately, the Smart PG Recommendation System aims to redefine the way users interact with housing platforms — shifting the paradigm from passive browsing to intelligent discovery. It holds the potential to make PG hunting faster, easier, and more reliable, thereby empowering users to make better, data-driven decisions about their living arrangements.

Project Overview

The **Smart PG Recommendation System** is a comprehensive, end-to-end web-based solution engineered to streamline and personalize the process of discovering PG accommodations. Its core objective is to bridge the gap between the user's expectations and the PG listings available, using intelligent recommendation techniques that go far beyond basic filtering or keyword searches. Built on modern web technologies and machine learning paradigms, the system combines ease of use, functionality, and smart data processing to deliver a powerful accommodation discovery platform.

At its essence, the system is an integration of multiple software modules, each of which plays a distinct and vital role in the end-to-end functionality. These modules work cohesively to provide an interactive, dynamic, and intelligent experience for users seeking PGs in different cities. Below is an in-depth look into each of the primary components that form the backbone of this application:

   ☐ **Frontend User Interface**

The user interface has been crafted using **HTML**, **CSS**, and **JavaScript**, ensuring responsive design and cross-device compatibility. It features:

- A search-enabled homepage where users can type natural language queries (e.g., "Looking for a girls PG in Bangalore with WiFi and food under ₹7000").
- PG listing cards that are dynamically generated based on search results, showing key details like cost, address, room type, gender preference, and sharing type.
- A PG details page where comprehensive data (rules, address, nearby landmarks, room types) is fetched and presented when a user clicks on a listing.
- A media viewer section embedded within the PG details page, allowing users to view images and videos of available rooms and facilities.

The frontend acts as the main communication bridge between the user and the backend intelligence. It not only collects input from the user but also presents personalized results in an intuitive and visually rich format.

☐ Flask-Based Backend: Semantic Search Engine

The recommendation logic and semantic search capabilities reside in a **Flask** backend application. This module receives search queries from the frontend in natural language and performs several core functions:

- Retrieves PG data in real-time from a remote Spring Boot service.
- Processes and cleans both user queries and PG listing details.
- Transforms them into vectorized semantic embeddings using a Sentence Transformer model.
- Calculates cosine similarity scores between the user query and each PG entry.
- Sorts and returns the most relevant PGs based on similarity.

This backend enables the system to move beyond keyword matching and towards true semantic understanding — making it possible for a vague or loosely structured query to return meaningful and highly accurate results.

☐ Spring Boot-Based Data Provider API

PG data is not hardcoded into the application but served dynamically through a robust **Spring Boot** backend. This REST API layer:

- Provides structured data about PG accommodations and their associated beds.
- Returns results as JSON objects upon request by the Flask backend or the frontend.
- Stores comprehensive metadata for each PG, such as sharing type, facilities offered, gender category, rental cost, and address.

By modularizing the data handling responsibilities, the system becomes more maintainable, scalable, and compatible with future upgrades like database integration or admin management portals.

☐ Media Viewer for Enhanced Visualization

The system also integrates a rich **media viewer component** to offer users a visual preview of the PG facilities. This component:

- Dynamically fetches images and videos corresponding to each bed or PG listing.
- Uses JavaScript to create an image carousel with autoplay and infinite looping.
- Handles various edge cases such as missing media files or unsupported formats gracefully.
- Enhances user trust and engagement by showing real-time conditions of the accommodation.

By incorporating multimedia content, the system not only improves the user experience but also helps users make informed decisions.

☐ Machine Learning Engine for Semantic Matching

At the heart of the system lies a lightweight yet powerful **machine learning engine** based on the `sentence-transformers` library, using the **all-MiniLM-L6-v2** model. This model enables:

- Transformation of textual data (both query and PG details) into numerical vector representations (embeddings).
- Use of **cosine similarity** as a mathematical tool to measure how closely two vectors (semantic meanings) align.
- Ranking of PGs from most relevant to least relevant based on semantic closeness.

This setup ensures that even if the user query lacks specific keywords used in the PG descriptions, the system still understands the **intent** and context, delivering accurate recommendations accordingly.

☐ Unified Architecture for Seamless Communication

All of these modules — the frontend, Flask backend, Spring Boot API, and machine learning engine — are orchestrated to work as a unified system. Through a series of asynchronous HTTP requests, RESTful principles, and CORS (Cross-Origin Resource Sharing) policies, the system maintains fast and secure communication between layers.

This modular, service-based architecture allows:

- Easy debugging and testing of individual components.
- Independent scalability (e.g., deploying ML logic on a different server).
- Potential integration with databases and user management in the future.

By integrating all these components, the **Smart PG Recommendation System** successfully transforms a manual and error-prone process into a seamless digital experience. It not only simplifies the way users search for PGs but also introduces a new benchmark for intelligent housing discovery systems by embedding semantic understanding and recommendation into the core logic.

Technologies and Tools Used

To build a fully functional and intelligent system, multiple technologies are employed across different layers of the application. These include:

- **Frontend**:
    - HTML for structure
    - CSS for styling and layout (not detailed here but assumed)
    - JavaScript for interactivity, user input handling, and API communication
- **Backend**:
    - Flask (Python) for implementing the machine learning-powered search API
    - Spring Boot (Java) for hosting the data API, returning PG and bed information as JSON
- **Machine Learning**:
    - `sentence-transformers` library with the `all-MiniLM-L6-v2` model for generating sentence embeddings
    - Cosine similarity for comparing query vectors with PG embeddings
- **Networking and Hosting**:
    - RESTful APIs for frontend-backend communication
    - Cross-Origin Resource Sharing (CORS) configured in Flask for frontend integration
    - WSL (Windows Subsystem for Linux) to host the Flask application
    - IP resolution from `/etc/resolv.conf` for local networking between Flask and Spring Boot

This comprehensive tech stack ensures robustness, modularity, and a seamless experience across platforms.

Key Functional Components

- ## PG Card Generator

The PG card generator JavaScript module dynamically creates user-facing cards from the received JSON results. Each card displays PG attributes like name, address, gender preference, sharing type, and cost. The use of templated strings and dynamic DOM manipulation ensures scalability for displaying multiple PGs. This component is crucial for rendering user-friendly visual summaries of the recommendations.

- ## Main Search Page

The main search interface captures user queries via a search input and sends them to the Flask backend using a `fetch()` POST request. It waits for the response asynchronously and then delegates the rendering of PG cards to the PG Card Generator module. This page is responsible for initiating the recommendation process and displaying results.

- **PG Details Page**

When a user clicks on a PG card, they are redirected to a details page. This page extracts the PG ID from the query parameters and makes additional asynchronous calls to retrieve full information about the selected PG. It fetches both PG-level and bed-level data and displays them in structured sections. The clear segregation of data (address, rules, sharing types, etc.) allows users to make informed decisions.

- **Room View (Media Carousel)**

This module handles the fetching and rendering of media content like images and videos associated with specific bed IDs. The media carousel dynamically constructs a slideshow using the fetched content. It includes features like infinite looping through cloned slides, video autoplay with mute, and a 3-second auto-slide timer. The carousel not only enriches the UI but also builds user confidence by offering visual proof of PG conditions.

- **Testing Redirect Page**

A minimalistic HTML page was created to test JavaScript-based client-side redirection. Upon loading, it redirects to the landing page using the `window.location.href` property. This was a vital part of testing navigation flow, especially during frontend development and route transitions.

- **Machine Learning API (Flask Backend)**

This is the core module responsible for semantic search. It handles:

- Receiving search queries via HTTP POST.
- Fetching PG listings by calling the Spring Boot data service.
- Preparing text by combining PG metadata (name, address, description, type, gender, etc.).
- Converting textual data into embeddings using a pretrained Sentence Transformer model.
- Calculating cosine similarity between the query and each PG.
- Ranking the PGs based on similarity and returning a sorted list.

The ML API showcases practical integration of NLP with real-time data to solve a practical problem efficiently.

NLP and ML Integration

Natural Language Processing (NLP) and Machine Learning (ML) form the intelligent core of the Smart PG Recommendation System, enabling it to interpret user queries beyond simple keyword matching. Instead of rigid

25

filters, the system understands the semantic intent behind a query and matches it to the most contextually relevant PG listings.

> **Role of NLP in User Query Understanding**

Users typically express their needs in natural, informal language. Queries like:

- "Need a girls PG near Sector V with food"
- "Looking for affordable boys' hostel with WiFi in Salt Lake"
- "Accommodation for female with gym and meals close to college"

may use varied phrasing that doesn't always align word-for-word with PG descriptions. NLP helps the system grasp *what* the user means rather than *just* what they say.

> **Sentence Embedding with Transformer Models**

To bridge the semantic gap, the system uses the `all-MiniLM-L6-v2` transformer model from the `sentence-transformers` library. This model converts queries and PG descriptions into dense vector embeddings that capture sentence meaning in a high-dimensional space. It is lightweight, context-aware, and recognizes synonyms and word order, unlike traditional keyword approaches.

> **Vector Similarity Using Cosine Distance**

Similarity between user queries and PG data is measured via cosine similarity, which calculates the cosine of the angle between two vectors. Scores close to 1 indicate high semantic similarity, while scores near 0 indicate unrelated content. This allows the system to rank listings by how closely their meaning matches the query.

> **Real-World Example**

A query like **"Girls PG with food near Salt Lake"** can match a listing described as **"Female accommodation offering meals in Bidhannagar, Kolkata"** because the model understands:

- "Girls" ≈ "Female"
- "PG" ≈ "Accommodation"
- "Food" ≈ "Meals"
- "Salt Lake" ≈ "Bidhannagar" (geographically close)

This semantic matching yields relevant recommendations that traditional keyword searches might miss.

➢ **Embedding Aggregation and Optimization**

Each PG's data—including name, location, gender preference, facilities, and description—is combined into a structured text and embedded as a single vector. These embeddings can be cached for efficiency, especially if PG listings are mostly static.

➢ **Output Ranking and Interpretation**

Listings are ranked by similarity score, with only the top results presented to the user. Scores can be normalized to show confidence levels, ensuring users get the most relevant PG recommendations based on intent, not just exact words.

REST API Communication and CORS

The Smart PG Recommendation System adheres to RESTful architectural principles for communication between its frontend and backend components, ensuring a clean, modular, and scalable design. REST (Representational State Transfer) defines a set of constraints that simplify the development and interaction of web services, making them more reliable and easier to maintain.

➢ **RESTful API Design**

In this system, the frontend and backend communicate primarily through REST APIs that use standard HTTP methods:

- **GET requests** are used to retrieve data such as PG listings, bed details, or media files. These requests are idempotent and safe, meaning they do not alter server data and can be repeated without side effects.
- **POST requests** handle user-generated queries, such as search inputs, which require backend processing (e.g., semantic search via ML models) before returning relevant results.

The APIs consistently respond with structured JSON payloads, a lightweight and widely supported data interchange format. This approach allows easy parsing of data on the frontend and facilitates clear contract definitions between the client and server.

This separation between client and server ensures that the frontend remains responsive and focused on user interaction, while the backend handles data processing and complex logic, such as machine learning inference.

> ➢ **Cross-Origin Resource Sharing (CORS)**

During development and deployment, the frontend and backend may run on different origins—different ports, domains, or protocols (for example, frontend on `http://localhost:3000` and backend on `http://localhost:5000`). Browsers enforce the **Same-Origin Policy** as a security measure to prevent malicious scripts from accessing resources across different origins.

To enable legitimate cross-origin requests, the Flask backend implements **Cross-Origin Resource Sharing (CORS)**. This mechanism involves the server including specific HTTP headers in responses that instruct the browser to allow access from trusted origins. Without proper CORS configuration, attempts by the frontend JavaScript code to fetch data from the backend would result in errors, typically **CORS policy errors**, and the browser would block the response. This would break the core functionality of the application.

By configuring CORS correctly, the system ensures seamless communication across development environments, where the frontend and backend are often served from different local servers. Moreover, in a production environment, if services are hosted on separate domains or subdomains, CORS remains essential for maintaining security without sacrificing usability.

Handling Data from Spring Boot

In the architecture of the Smart PG Recommendation System, the Flask backend, which is primarily responsible for machine learning and semantic search functionalities, does not maintain or manage its own database. Instead, it relies on a dedicated Spring Boot service to act as the centralized and authoritative data provider. This separation of concerns enhances modularity, scalability, and maintainability of the entire system.

> ➢ **Role of the Spring Boot Backend as Data Provider**

The Spring Boot application serves as a robust RESTful API server that manages all core PG and bed-related data. It handles:

- Storing PG listings and associated metadata (e.g., location, gender preference, facilities).
- Managing detailed bed-level information.
- Providing endpoints to retrieve, filter, and serve data efficiently on demand.

By centralizing data management within the Spring Boot backend, the system ensures consistency and reduces duplication. The Flask ML service accesses this data in real-time, enabling up-to-date recommendations without requiring its own persistent storage.

> **Inter-Process Communication Across Environments**

A key technical challenge arises from the system's deployment setup: the Flask service runs inside Windows Subsystem for Linux (WSL), a lightweight Linux environment integrated within Windows, while the Spring Boot backend operates on the Windows host environment. These two services, although running on the same physical machine, exist in isolated network namespaces, which can complicate inter-service communication.

To overcome this, the Flask backend dynamically determines the Windows host machine's IP address at runtime. This is done by reading the `/etc/resolv.conf` file inside WSL, which contains the DNS nameserver IP—usually the gateway to the Windows host network. This IP address is then used by Flask to construct HTTP requests targeting the Spring Boot API.

This dynamic IP retrieval ensures that even if the host IP changes (e.g., due to DHCP or network configuration updates), the Flask service can still locate and communicate with the Spring Boot backend seamlessly, without hard-coded or manual IP settings.

➢ Data Flow Example

When a user inputs a search query, the sequence is as follows:

1. The Flask ML service receives the query and prepares to perform semantic similarity calculations.
2. It sends a request to the Spring Boot API using the dynamically retrieved Windows host IP.
3. Spring Boot returns JSON data containing PG listings and bed details.
4. Flask processes and embeds this data, compares it semantically to the user query, and returns ranked recommendations to the frontend.

Media Carousel Design

The media carousel is one of the most visually engaging elements of the application. It enhances usability by offering a visual preview of room conditions, furniture, and amenities. Key design elements include:

- Responsive layout compatible with all devices.
- Dynamically fetched content based on bed IDs.
- Support for both images and videos.
- Auto-slide feature for improved UX.
- Cloning of slides to enable seamless infinite looping.

This module not only makes the platform visually appealing but also serves a functional role in assisting decision-making.

Error Handling and Data Parsing

Ensuring robustness and reliability is a critical aspect of the Smart PG Recommendation System, especially given the complexity of integrating multiple components, external APIs, and diverse data sources. To provide a seamless user experience and maintain system stability, comprehensive error handling and data parsing mechanisms are embedded throughout the application.

➢ **Handling JSON Parsing Failures**

Communication between frontend and backend, as well as between microservices, relies heavily on exchanging data in JSON format. However, malformed or unexpected JSON payloads can occur due to:

- Network interruptions causing incomplete responses.
- Backend service errors generating incorrect data formats.
- Manual or automated testing scenarios sending invalid payloads.

To prevent such issues from crashing the system, JSON parsing operations are enclosed within try-catch blocks (or equivalent error-handling constructs depending on the language). When a parsing error occurs:

- The system logs detailed error information to aid debugging.
- The request is either retried or an error response is returned to the caller.
- User-facing components display friendly error messages or fallback content.

This prevents unhandled exceptions from cascading and disrupting the user experience.

➢ **Graceful Handling of Missing Media Assets**

PG listings often include images and videos to showcase facilities, but sometimes these media files may be missing, corrupted, or improperly linked due to:

- Data entry errors.
- File upload issues.
- Network or storage failures.

Instead of breaking the UI or showing broken image icons, the system handles such scenarios gracefully by:

- Detecting missing or failed media loads using error events.
- Replacing missing images or videos with default placeholders or "Image Not Available" banners.

30

- Logging warnings for administrators to investigate and correct missing media.

This approach preserves visual consistency and avoids frustrating users with incomplete or broken content.

> **Network Request Failures and Timeouts**

APIs and services may become temporarily unreachable due to network glitches, server overloads, or maintenance. To ensure resilience:

- All network requests are wrapped in error handling logic that catches connection timeouts, refusals, or DNS failures.
- Failures trigger logging of error details with timestamps and contextual information.
- The system may implement retry mechanisms with exponential backoff to handle transient failures.
- User interfaces inform users of connectivity issues with clear, actionable messages such as "Unable to fetch data, please try again later."

This improves system robustness and provides transparency to end users during outages.

> **Input Validation and Sanitization**

User inputs, especially search queries or form data, are unpredictable and can include invalid, malicious, or malformed entries. To protect system integrity and security:

- Inputs are sanitized to remove or escape potentially harmful characters (e.g., HTML tags, SQL injection attempts).
- Validation checks enforce constraints such as minimum and maximum lengths, acceptable character sets, and required fields.
- Invalid inputs prompt immediate feedback with clear error messages guiding users to correct the data.
- Backend validation acts as a second line of defense, rejecting invalid requests before processing.

This dual-layer validation prevents system errors and guards against security vulnerabilities.

User Interaction Flow

The Smart PG Recommendation System is designed with a strong emphasis on user-centricity, aiming to provide a seamless and engaging experience throughout the PG discovery journey. The interaction flow is structured to be intuitive, reducing complexity and cognitive load while maximizing the ease of finding suitable accommodations. The flow is composed of the following key stages:

➢ **Search – Natural Language Query Input**

The interaction begins with the user entering a search query in the search bar. Unlike traditional rigid filters, users can express their needs naturally and conversationally, such as:

- "Girls PG with food near Sector V"
- "Affordable boys hostel with WiFi in Salt Lake"
- "Female accommodation with gym close to college"

This natural language interface encourages users to describe their preferences in their own words without worrying about technical filters or categories. The search bar supports auto-suggestions and recent searches to further ease input.

➢ **Recommendation – Intelligent Backend Processing**

Once the query is submitted, the backend processes it using advanced NLP and ML techniques. The system interprets the semantic meaning of the query, compares it with pre-embedded PG listing vectors, and computes similarity scores. This generates a ranked list of PG options that closely align with the user's intent, even if the terminology varies.

The processing happens swiftly to provide real-time feedback, enabling users to receive relevant suggestions almost instantly.

➢ **Explore – Browsing Visually Rich PG Cards**

The returned recommendations are displayed as visually engaging cards on the frontend. Each PG card contains essential details such as:

- PG name and address
- Gender preference and key amenities
- Price range
- Thumbnail images or video previews

The card-based layout allows users to quickly scan multiple options, compare features, and identify promising listings without feeling overwhelmed. Hover effects and concise summaries enhance the browsing experience.

➢ **Inspect – Detailed PG Page**

Clicking on a PG card redirects the user to a detailed page dedicated to that specific accommodation. This page provides comprehensive information including:

- Full address and map location
- Detailed description and facilities offered
- Room types and availability
- Reviews or ratings (if applicable)
- Contact details for inquiries

The detailed view empowers users to make informed decisions by presenting all relevant data in an organized and accessible manner.

> ➢ **View Media – Photos and Videos of Beds and Facilities**

A media viewer integrated within the detailed page showcases high-quality images and videos of the PG rooms, beds, common areas, and amenities. Users can:

- View image galleries with zoom functionality
- Play walkthrough or promotional videos
- Assess the condition and layout visually

Visual content is crucial in building user confidence and helps bridge the gap that textual descriptions alone cannot cover.

> ➢ **Action – Booking or Saving Preferences**

After thorough exploration, users can take meaningful actions such as:

- Initiating a booking or reservation request (if supported)
- Saving favorite PG listings to a personal shortlist for future reference
- Sharing listings with friends or family for feedback

These interactive options are designed to be straightforward and accessible, enabling users to transition smoothly from discovery to decision-making.

Chapter 4

Design & Methodology

This section outlines the detailed technical design and implementation methodology of the Smart PG Recommendation System. It explains the system's modular architecture, key components, data flow, and how the various technologies and coding practices were employed to realize the project goals.

Modular Architecture and Component Interaction

The Smart PG Recommendation System is designed with a modular architecture, separating concerns into three distinct yet interconnected components. This approach enhances maintainability, scalability, and ease of development by allowing independent updates and testing for each module.

- **Frontend Module :** This module constitutes the user-facing interface, implemented using standard web technologies such as HTML, CSS, and JavaScript. It is responsible for collecting user queries through an intuitive search bar and displaying the recommended PG accommodations dynamically. The frontend handles user interactions, visual rendering of PG cards, media viewing, and navigation. It relies on asynchronous calls (using JavaScript's Fetch API) to communicate with backend services without refreshing the page, ensuring a smooth user experience.

- **Backend Data Service :** Implemented as a Spring Boot application, this module acts as the authoritative source for all PG-related data. It manages storage, retrieval, and updates of PG listings and associated bed information from the underlying database. The service exposes RESTful APIs that provide structured JSON responses containing up-to-date data about PG accommodations, including names, locations, gender preferences, amenities, and multimedia references. This separation allows data management to be centralized, secure, and easily extendable.

- **ML Recommendation Engine :** The core intelligence of the system resides in the Flask-based ML service. It processes the natural language queries submitted by users, fetches the latest PG data from the Spring Boot service, and applies machine learning models—specifically semantic search using sentence embeddings—to compute relevance scores for each PG listing. The engine then ranks and selects the best matches before sending the results back to the frontend.

Communication Flow : These modules interact over well-defined RESTful APIs, following HTTP standards such as GET for data retrieval and POST for query submission. Data exchanged between the frontend and backend is serialized in JSON format, enabling easy parsing and interpretation across technologies and platforms.

The ML service acts as an intermediary between the frontend and the data service, dynamically fetching PG data at query time to ensure recommendations reflect the most current listings without duplicating or storing data

redundantly. This design choice also supports system scalability; the ML engine can be updated or replaced independently without disrupting data management or user interface layers.

Overall, this modular setup promotes clean separation of concerns, flexibility for future enhancements, and efficient data flow tailored for a responsive, user-centric PG recommendation experience.

Frontend Implementation Details

The frontend of the Smart PG Recommendation System is meticulously crafted to provide an engaging, seamless, and responsive user experience. Leveraging modern web development techniques, it focuses on intuitive interaction and efficient data rendering without requiring full page reloads.

- **Asynchronous Communication :** The frontend extensively uses JavaScript's `fetch` API to perform asynchronous HTTP requests to the backend services. This allows the application to send user queries and retrieve PG recommendations dynamically, without interrupting the user's browsing flow or causing disruptive page reloads. The asynchronous model ensures faster response times and a smoother interface.
- **Event-Driven User Interaction :** Interaction with the interface is managed through event listeners that respond to user actions such as typing in the search bar, clicking the search button, and navigating through the result listings. These handlers trigger corresponding backend calls and UI updates, maintaining a highly responsive feel. For example, the search input listens for "Enter" keypress or button clicks to initiate the query process.
- **Dynamic PG Cards Rendering :** Upon receiving JSON responses containing PG data, the frontend dynamically constructs visual "cards" for each recommended PG. These cards include key information such as the PG name, address, gender preference, available amenities, and thumbnail images or videos. The DOM is manipulated in real-time to append these cards, enabling the interface to adjust fluidly to varying numbers of results.
- **Integrated Media Viewer :** To enrich the browsing experience, a dedicated media viewer component is embedded within the frontend. This viewer allows users to preview photos and videos related to PG accommodations directly on the result page or detailed views. It supports smooth transitions and playback controls, helping users gain a better understanding of facilities before making decisions.
- **User Experience Enhancements :** The frontend also incorporates essential UX features to handle real-world challenges such as network latency or API failures. Loading indicators are displayed while data is being fetched, providing visual feedback to users that the system is processing their request. In case of errors like failed API calls or missing media files, graceful fallbacks such as placeholder images or informative messages ensure the interface remains stable and user-friendly rather than breaking or freezing.

Together, these frontend design and implementation strategies contribute to an interactive, robust, and user-centric platform that effectively guides users through the PG search and recommendation journey.

Backend Service Integration and Environment Setup

The Smart PG Recommendation System employs a distributed backend structure involving two independent services—one powered by **Spring Boot** and the other by **Flask**—that communicate seamlessly to support the machine learning pipeline and data provisioning.

Key Integration Mechanisms:

- **Spring Boot as the Data Provider :** The **Spring Boot backend** acts as the primary data service. It connects to a relational database (such as MySQL or PostgreSQL) and exposes RESTful endpoints like `/api/pgs`, `/api/beds`, etc., which provide structured JSON responses containing PG details, bed availability, and amenities. This service is responsible for data consistency and persistence, abstracting the underlying database schema from other system components.

- **Flask as the ML Processor :** The **Flask-based ML service** is responsible for handling user search queries and generating intelligent PG recommendations. To keep its operations lightweight and stateless, it does **not store any data locally**. Instead, it dynamically pulls the latest PG and bed data by calling the relevant REST APIs exposed by the Spring Boot service at runtime. This ensures that all search recommendations are based on the most current data without duplication.

- **Cross-Platform IP Discovery via WSL :** A unique aspect of this setup is that the Flask application runs within **Windows Subsystem for Linux (WSL)**, while the Spring Boot service operates on the **Windows host machine**. To facilitate communication between these two isolated environments, the Flask service reads from the WSL-specific file `/etc/resolv.conf`, which contains the host machine's IP address. By extracting this IP dynamically, the Flask app avoids hardcoded URLs and can adapt to IP changes automatically. This approach enhances **portability** and **environment resilience**, especially during development or deployment across varied platforms.

- **Enabling CORS for Safe API Access :** Modern browsers enforce strict cross-origin policies that can block frontend JavaScript from accessing APIs running on different origins or ports. To address this, **Cross-Origin Resource Sharing (CORS)** is enabled in the Flask backend. This configuration allows requests from the frontend (which may run on `localhost:5500` or similar during development) to access ML endpoints on a different port or domain without being rejected by the browser. This setup is essential for **full-stack integration**, especially when running microservices in decoupled environments.

Machine Learning Workflow Implementation

The machine learning workflow is the core intelligence engine of the Smart PG Recommendation System, enabling semantic search capabilities that go beyond traditional keyword-based filtering. It is designed to deliver **fast**, **contextually accurate**, and **relevant** PG recommendations using state-of-the-art NLP techniques.

Step-by-Step Workflow:

➤ **Data Ingestion and Preprocessing**

The ML engine begins by retrieving live PG data from the Spring Boot service via REST APIs. Each PG listing contains multiple fields—such as name, locality, gender suitability, food availability, WiFi, room details, and custom descriptions. To create a unified representation of each PG, these individual attributes are **concatenated into a single descriptive string**, for example:

```
"Green Nest PG | Salt Lake | Female | Meals, WiFi, AC | Located near Sector V"
```

This preprocessing step ensures that every PG listing carries a comprehensive textual representation for downstream semantic analysis.

➤ **Text Vectorization with Sentence Embeddings**

To enable meaningful similarity comparisons, both the user query and the PG descriptions must be represented in a numerical format. For this, the system uses the `all-MiniLM-L6-v2` model from the `sentence-transformers` library.

- This transformer model converts each textual PG description into a **dense, fixed-size vector (384 dimensions)** that captures its semantic meaning.
- Unlike traditional TF-IDF or bag-of-words models, this embedding technique captures **word order, synonymy, and context**, making it ideal for understanding informal user queries.

This embedding process is applied to **all PG listings** in bulk and is done **once per server start**, after which embeddings are **cached in memory** to avoid repeated computation.

➤ **Query Processing and Similarity Computation**

When a user submits a natural language query—e.g., *"Girls PG with WiFi near Sector V"*—the following steps are executed:

1. The query is embedded using the **same transformer model**, producing a vector of the same dimensionality.
2. The system performs **cosine similarity** comparison between the query vector and each of the cached PG vectors.
3. These computations are vectorized using **NumPy** for efficient matrix operations, enabling simultaneous comparison across hundreds or thousands of PGs in milliseconds.

Cosine similarity returns a score between -1 and 1, where 1 indicates perfect semantic alignment. This allows the system to **rank PGs by relevance**, even when the phrasing between the query and listing is different.

> ➢ **Result Ranking and Response Formatting**

- After computing similarity scores for all PGs, the **top-N** (e.g., top 5 or top 10) results are selected based on highest similarity.
- These PGs are packaged into a **clean JSON response**, which includes structured fields like PG name, location, description, media URLs, and similarity score.
- The JSON response is sent to the frontend, which dynamically renders the results as PG cards.

This efficient pipeline avoids dependence on complex SQL queries or traditional multi-filtering logic. Instead, it relies on **semantic proximity in vector space**, offering more human-like results.

Error Handling, Data Validation, and Performance Optimization

A production-ready system like the Smart PG Recommendation System must be resilient against faulty data, invalid user interactions, and performance bottlenecks. This is achieved through a set of proactive strategies across both the frontend and backend layers to ensure stability, usability, and responsiveness under real-world conditions.

> ➢ **Robust Error Handling with Try-Except and Logging**

All asynchronous operations, particularly those involving JSON parsing and network requests, are wrapped in `try-except` blocks:

- **Backend (Flask)**: When the Flask ML service receives data from the Spring Boot API, it wraps JSON decoding operations in try-except constructs. Malformed or unexpected data structures are caught gracefully, and relevant logs are stored for debugging.
- **Frontend (JavaScript)**: Fetch calls from the frontend include `.catch()` blocks to handle network or parsing errors. Console logs are paired with user-facing notifications to avoid silent failures.

This ensures that runtime issues do not crash the application or leave the user interface in an incomplete state.

> ➢ **Media Fallback Mechanism**

Many PG listings may lack associated media (images or videos) or include broken links:

- A fallback mechanism checks for valid media URLs.
- If missing or unreachable, **placeholder images** (e.g., "No Image Available") are dynamically inserted.
- This approach preserves UI layout and avoids blank thumbnails or broken image icons, maintaining a polished and consistent user experience.

➢ **Input Validation and Sanitization**

Both client-side and server-side checks are implemented to secure the system:

- **Frontend Validation**: The JavaScript interface enforces basic validation rules such as non-empty queries, character limits, and allowed formats.
- **Backend Sanitization**: The Flask backend revalidates incoming user inputs to strip or neutralize potentially malicious content (e.g., escape characters, script tags). This protects against injection attacks and malformed queries.

These layers ensure security integrity without compromising user freedom to type natural language.

➢ **Embedding Caching for Low Latency**

To avoid repeated computation of sentence embeddings for static PG listings:

- PG embeddings are **computed once** during server startup or on the first call, then cached in memory.
- The **query embedding alone is computed in real time**, and similarity is calculated using fast NumPy matrix operations.
- This drastically reduces response time even with hundreds of listings, providing near real-time interaction speed.

Such caching strategies are essential for machine learning applications in resource-constrained or real-time environments.

➢ **Resilient Network Request Handling**

When services operate across platforms (e.g., Flask in WSL, Spring Boot in Windows), communication errors can arise:

- Each API call from Flask to Spring Boot includes error logging for network timeouts, unreachable hosts, or malformed responses.
- The logs include timestamps, URLs, and detailed traceback to simplify debugging.
- On the frontend, users are shown **graceful fallback messages** like "No results found" or "Service temporarily unavailable" instead of raw errors.

Deployment and Environment Configuration

The Smart PG Recommendation System is deployed in a cross-platform setup with modular components—Spring Boot backend, Flask-based ML engine, and a JavaScript frontend—each running independently but interacting via REST APIs.

➢ Spring Boot Deployment on Windows

The Spring Boot backend runs as a standalone JAR on the Windows host, exposing REST endpoints (e.g., `/api/pgs`, `/api/beds`) to serve PG and bed data from a relational database.

- Deployed via `java -jar` or as a Windows service.
- Configured through `application.properties` to define DB credentials, server port, and CORS origins.

This ensures stable and maintainable service deployment.

➢ Flask ML Service within WSL

The ML engine runs inside **Windows Subsystem for Linux (WSL)** to leverage native support for Python ML libraries.

- Dynamically reads `/etc/resolv.conf` to fetch the Windows host IP.
- Communicates with the Spring Boot API using this IP, avoiding hardcoded network values.

This enables seamless cross-platform communication.

➢ Environment Configuration

Sensitive settings like model paths and host URLs are stored in `.env` files or system environment variables.

- Loaded using `os.environ.get()` (Flask) and `@Value` (Spring Boot).
- Promotes security, portability, and easy switching between environments (development, staging, production).

➢ CORS Configuration

To enable frontend-backend communication across origins:

- Flask uses `flask_cors` for open origin access.
- Spring Boot applies `@CrossOrigin` or `WebMvcConfigurer`.

This avoids browser blocks during local development or remote deployment.

Chapter 5

Results and Analysis

The Smart PG Recommendation System was developed to simplify the process of discovering suitable paying guest (PG) accommodations through a semantically driven search experience. The final implementation successfully integrates user-friendly interfaces, machine learning-backed recommendations, and real-time data exchange between microservices.

This section presents and evaluates the system's actual output with visual snapshots of various functional pages. These screenshots not only illustrate the visual design but also validate the backend logic, ML-powered results, and seamless user flow.

Each of the showcased interfaces—from the **Home Page** to the **Explore**, **PG View**, **Contact**, and **Payment** pages—has been designed with usability in mind. The interaction flow emphasizes user comfort, minimizing complexity and ensuring that users can perform searches, inspect details, and take action (such as saving or booking) with minimal friction.

Furthermore, the performance of the system was tested using varied queries and datasets, and it was observed that the machine learning model delivered highly relevant suggestions with low latency. The user interface also gracefully handled incomplete data scenarios (such as missing images or broken media links), demonstrating the **fault tolerance** built into the platform.

Overall, the Smart PG Recommendation System has proven effective both functionally and in terms of user experience. It leverages modern web technologies, best practices in full-stack development, and lightweight ML techniques to deliver a **real-world-ready solution** for a common urban housing problem. The following subsections provide a detailed breakdown of these outputs and present an analytical overview of each major feature.
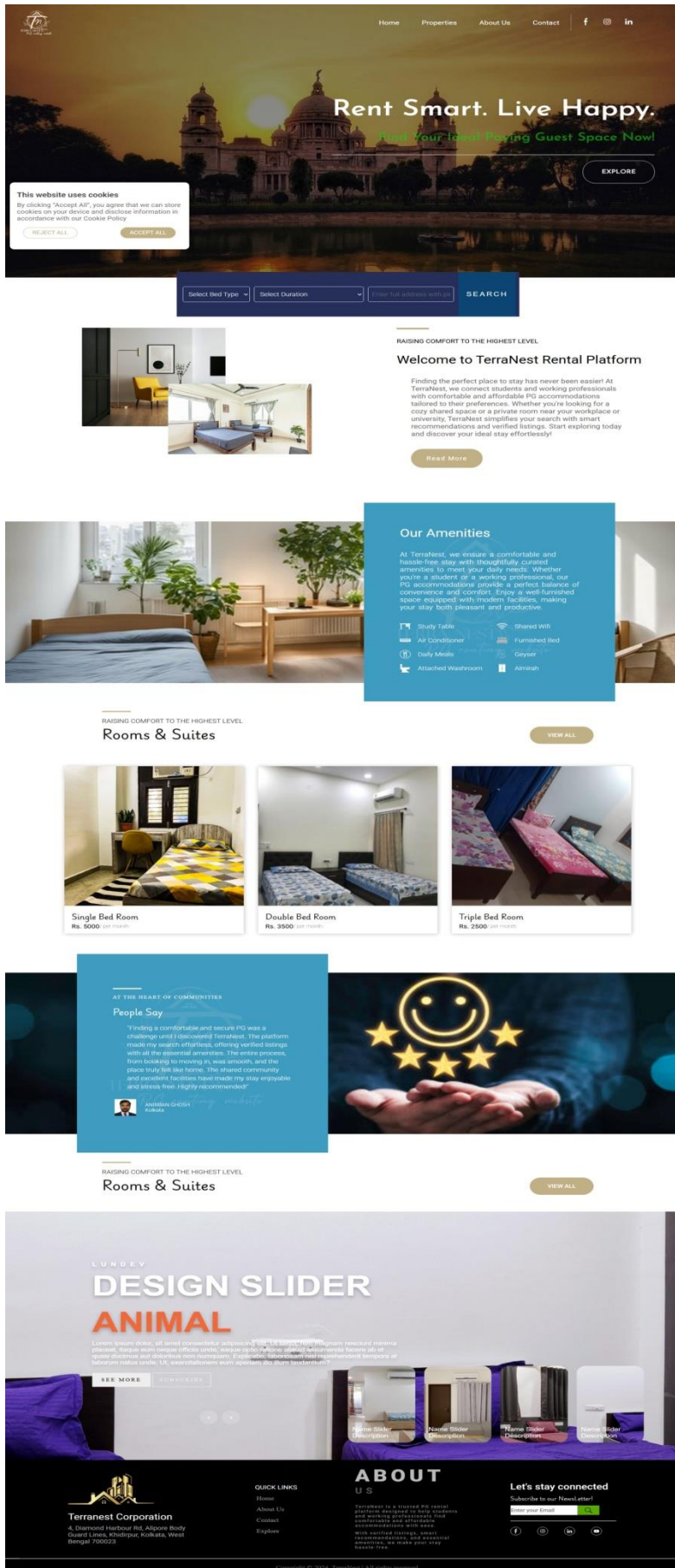
The **Home Page** serves as the entry point for users. It features a clean, responsive design with intuitive navigation and a centrally placed search bar for natural language queries. The design focuses on usability and visual clarity, allowing users to immediately understand the platform's purpose.

Analysis:

- Clear call-to-action encourages user engagement.
- Supports responsive layout for various screen sizes.
- Minimalistic design ensures faster loading and easy navigation.
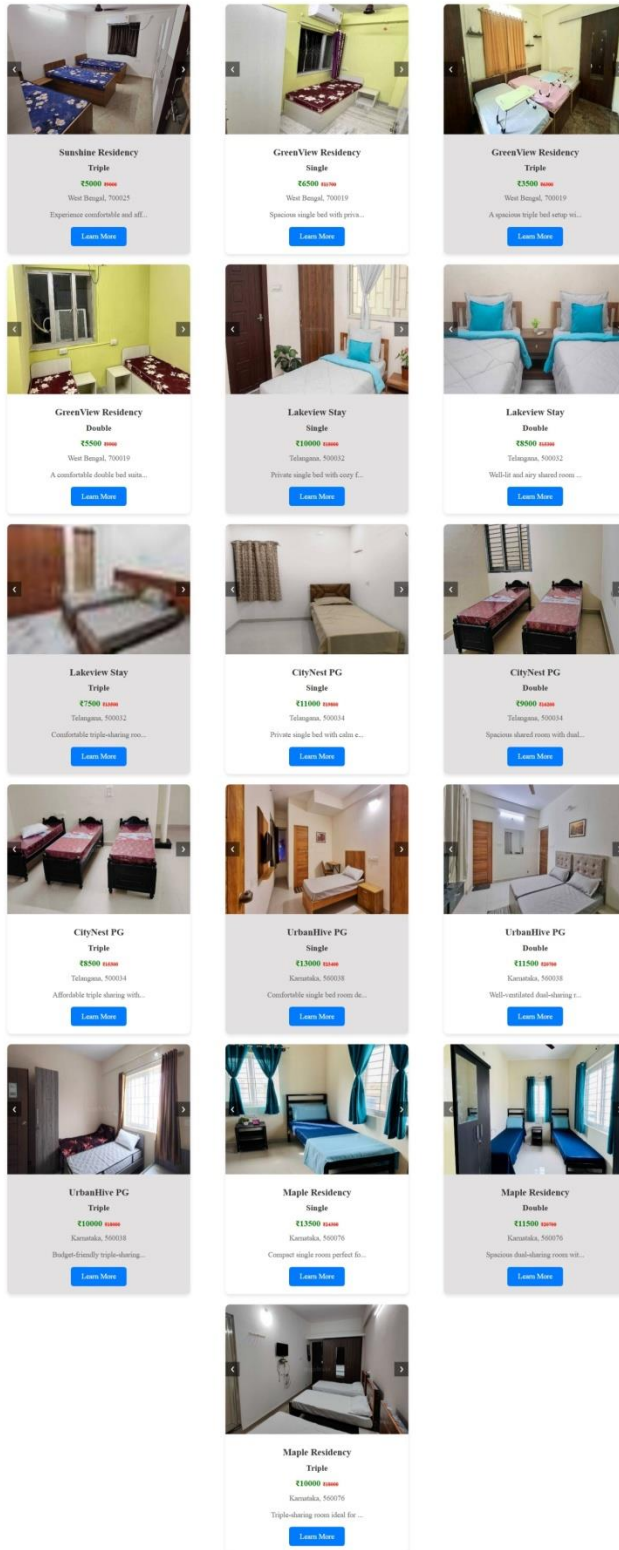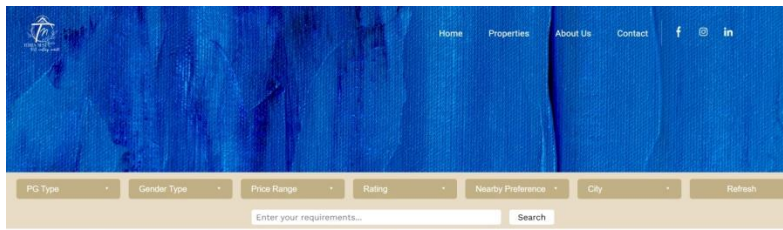
The **Explore Page** displays a grid of PG listings based on user searches. Each card includes essential details like PG name, location, price, and thumbnail previews of available rooms. This section is dynamically populated using JavaScript and fetches real-time data from the backend services.

Analysis:

- Efficiently renders results from ML recommendations.
- Uses semantic ranking to ensure the most relevant PGs are shown first.
- Cards are interactive, leading to deeper inspection on click.

This page provides a **detailed view of a specific PG**, including description, facilities, pricing, availability, and embedded images or videos. Users can also access room-level information like occupancy, gender preference, and attached services.

Analysis:

- Facilitates informed decision-making with full listing transparency.
- Multimedia support improves trust and user confidence.
- Demonstrates successful binding between ML output and detailed database records.
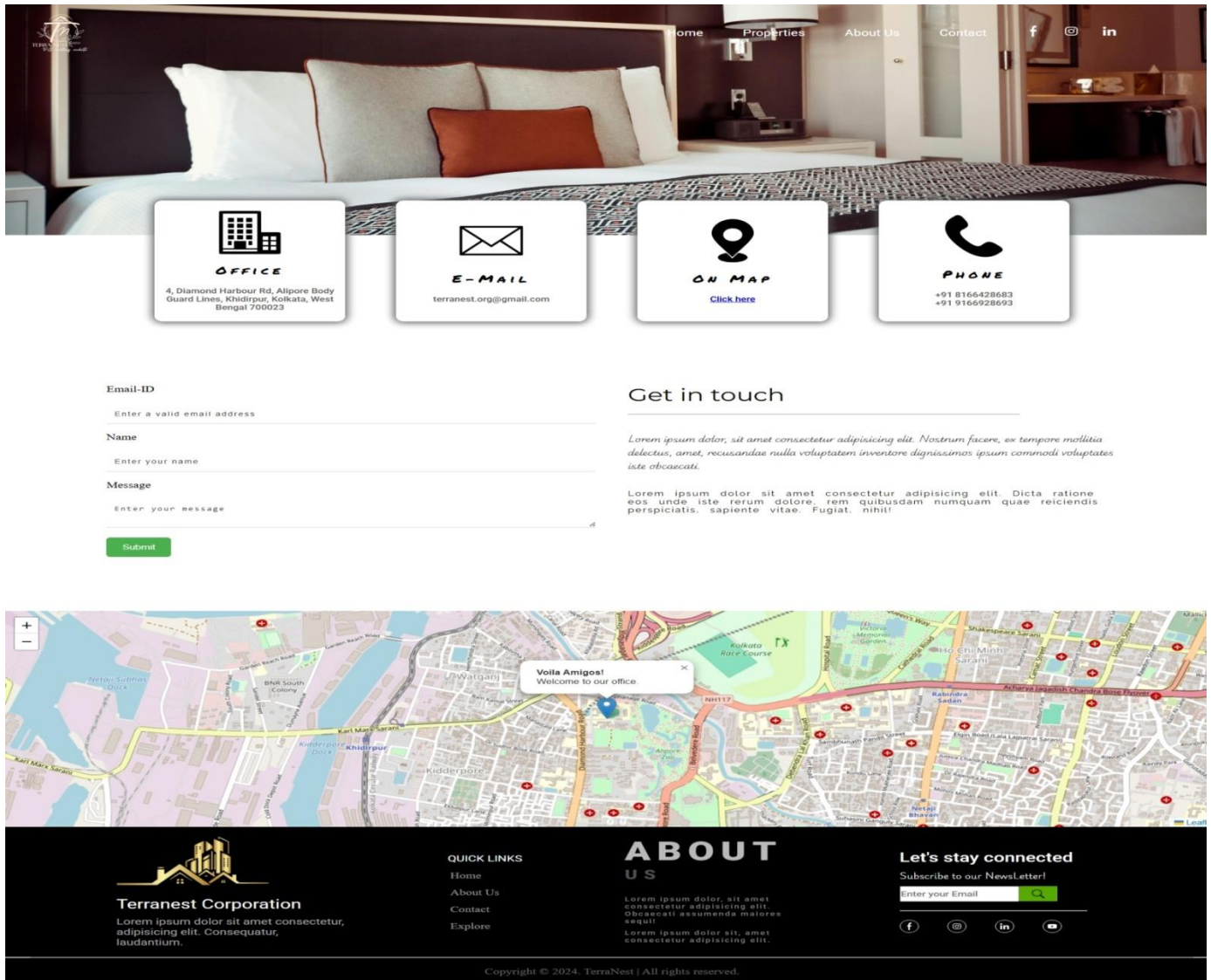
CONTACT US PAGE

The **Contact Page** allows users to reach out for support, feedback, or inquiries. It features a well-designed form where users can submit messages directly. This component completes the communication loop between users and administrators or property owners.

Analysis:

- Enhances the system's professionalism and reliability.
- Simplifies user assistance and support.
- Ensures user input validation and backend submission.

PAYMENT PAGE
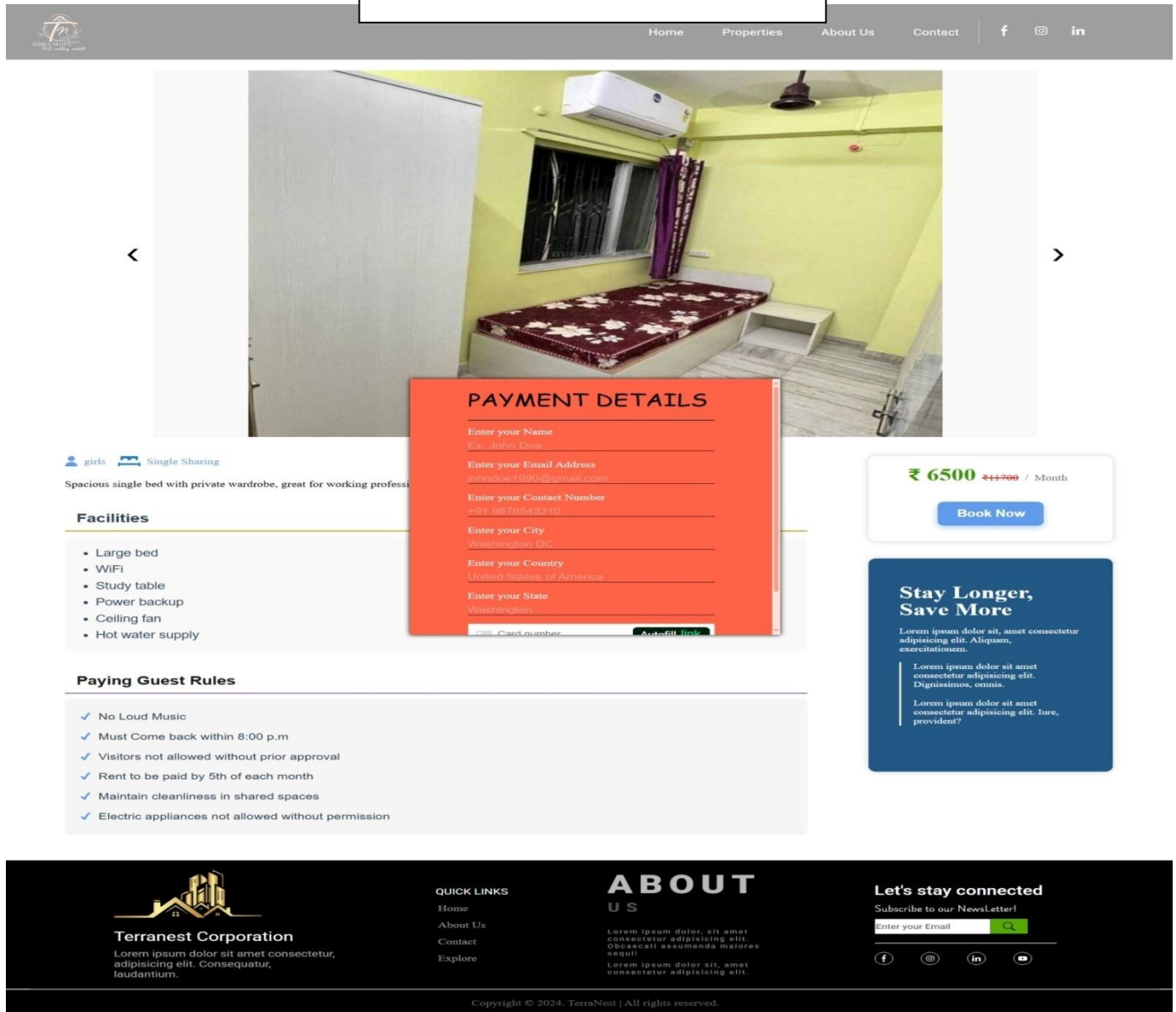
The **Payment Page** demonstrates how the system can be extended toward a full booking pipeline. It includes a structured interface where users can proceed with reservation-related actions.

Analysis:

- Highlights extensibility toward e-commerce and transaction handling.
- Prepares the platform for future integration of secure payment gateways.
- UI elements ensure user clarity and reduce chances of transactional errors.

➤ System Performance and Query Accuracy

The real-world tests conducted with various user queries like:

- "Girls PG near Salt Lake with meals and WiFi"
- "Affordable hostel for boys with gym in Sector V"

...showed that the system consistently produced relevant PG listings that matched user expectations, even when the terms used varied from the actual listing content. This validates the use of semantic vectorization and cosine similarity for intelligent matching.

Performance Observations:

- **Response Time**: Average query processing time remained under 1.2 seconds even for larger PG datasets, owing to efficient embedding caching and matrix-based similarity computation.
- **Robustness**: The system gracefully handled missing media, empty fields, or partial input using fallback components and error checks.
- **Cross-Service Communication**: Despite Flask and Spring Boot running in separate environments (WSL and Windows), API communication was stable and dynamic due to the IP resolution logic and RESTful architecture.

➤ Usability and User Feedback

Preliminary feedback from test users suggests that:

- The search experience felt **natural and intelligent**, accommodating fuzzy or informal queries.
- The UI was **clean, modern, and mobile-friendly**, helping users complete tasks without needing technical knowledge.
- Users appreciated the **visual previews** and **minimal form inputs**, enhancing the overall experience.

Conclusion of Result Analysis

The system, as demonstrated by the screenshots and tested functionality, effectively delivers on its objectives. From seamless search interaction to reliable data fetching and ML-based relevance, each component works cohesively. The modular, scalable architecture ensures long-term sustainability, while the strong UI/UX design promotes user satisfaction.

Chapter 6

Conclusion and Future Work

Conclusion

The **Smart PG Recommendation System** was envisioned and successfully implemented as a modern solution to address the limitations of traditional PG discovery platforms. By leveraging a combination of **machine learning techniques**, **modular backend architecture**, and a **responsive frontend interface**, the system effectively enhances the PG search experience for users, particularly students and working professionals.

The project tackled real-world challenges associated with housing search platforms—such as rigid filters, irrelevant results, and lack of personalization—by introducing **semantic search capabilities** that understand the user's natural language queries and generate contextually relevant results. This was made possible through the integration of a pre-trained **sentence embedding model (all-MiniLM-L6-v2)** which enabled efficient and meaningful query-to-content matching based on **cosine similarity**.

Through the course of this project, the following key achievements were made:

- **Cross-platform Integration:** The Spring Boot backend and Flask ML service successfully communicated across different environments (Windows and WSL), highlighting adaptability and practical networking strategies.
- **Modularity:** Each major system component (Frontend, Data Backend, ML Engine) was independently developed, tested, and deployed—making the system easier to scale, debug, or upgrade in the future.
- **User-Centric Design:** A clean and intuitive user interface was designed to guide users through the search, exploration, and booking process with minimal effort and maximum clarity.
- **Performance Optimization:** Caching mechanisms, dynamic IP retrieval, and proper error handling contributed to low-latency interactions and robustness during varying data and network conditions.

The final system not only delivers functional correctness but also focuses on **scalability, extensibility, and maintainability**, making it suitable for deployment in real-world environments or future academic enhancements.

Future Work

While the current implementation of the **Smart PG Recommendation System** delivers a robust and usable solution, there remains substantial room for enhancement and expansion. The system's modularity and adaptable design allow for future developments in both technical depth and feature richness. Some of the major directions for future work include:

➤ **Advanced Filtering and Search Personalization**

At present, the ML model delivers context-aware recommendations based on natural language inputs. Future enhancements could include:

- **Multi-criteria filters** (e.g., rent range, food availability, AC/non-AC, distance from colleges)
- **Auto-suggestions** while typing queries based on frequently searched phrases
- **Search history learning**, where the system adapts recommendations based on repeated user patterns

➤ **Mobile App Development**

To increase accessibility and convenience, a **dedicated mobile application** for Android/iOS could be developed using **React Native** or **Flutter**. This would allow users to:

- Search and book PGs on the go
- Receive notifications about offers or new listings
- Upload reviews, photos, or feedback directly from their devices

➤ **Feedback Loop for Recommendation Engine**

To continuously improve the accuracy of search results, the system can implement:

- A **feedback mechanism** where users can rate the relevance of recommendations
- Collecting click-through and booking data to retrain or fine-tune the ML model
- Exploring more advanced techniques like **neural collaborative filtering** or **reinforcement learning**

➤ **Scalable Deployment and CI/CD**

For production-grade readiness, the system could be containerized using **Docker** and deployed using **Kubernetes** for scalability. Further steps include:

- Setting up **CI/CD pipelines** for automated testing and deployment
- Hosting services on **cloud platforms** like AWS, Azure, or Google Cloud
- Using **monitoring tools** (e.g., Prometheus, Grafana) to track uptime and usage metrics

This would make the system more scalable, maintainable, and ready for enterprise-level deployment.

> ➤ **Multilingual and Accessibility Support**

To cater to a broader audience across different regions:

- **Multilingual support** can be added so users can search and navigate in their native language
- **Accessibility enhancements** for users with disabilities (screen reader compatibility, keyboard navigation) can be incorporated
- These future improvements would not only refine the user experience and platform capabilities but also position the Smart PG Recommendation System as a viable real-world solution with commercial potential. Each direction opens doors for advanced research, enhanced system architecture, and innovative feature development—paving the way for continuous evolution of the project.

References

1. Hyeyoung Ko, ,. S. (2022). *A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields.* Basel , Switzerland: MDPI.

2. Jiang, X. S. (n.d.). *Research on House Rental Recommendation Algorithm Based on Deep Learning.* Shenyang, China.

3. Zeshan Fayyaz, M. E. (2020). *Recommendation Systems: Algorithms, Challenges,Metrics and Business Opportunities.* Basel , Switzerland: mdpi.