Mithibai College of Arts, Chauhan Institute of Science & Amrutben
Jivanlal College of Commerce and Economics
(AUTONOMOUS) Affiliated to University of Mumbai
NAAC Reaccredited 'A' Grade, CGPA: 3.57 (February 2016)

A

Project Report On

RentEase: Ultimate App for House Rental

Project Report submitted for

Bachelor of Computer Science

Submitted by

Tahir Momin

Under the guidance of

Mrs. Jayshree Ravi

&

Mrs. Neelam Jain

YEAR: 2023-2024

Mithibai College of Arts, Chauhan Institute of Science & Amrutben Jivanlal College of Commerce and Economics
(AUTONOMOUS) Affiliated to University of Mumbai
NAAC Reaccredited 'A' Grade, CGPA: 3.57 (February 2016)

CERTIFICATE

This is to certify that the project titled RentEase is successfully done by

Mr. TAHIR MOMIN in fulfillment of **Bachelor Of Computer Science** under the **University of Mumbai,** Mumbai through the Mithibai College Of Arts, Chauhan Institute Of Science & Amrutben Jivanlal College Of Commerce & Economics (Autonomous) Mumbai carried out by him under our guidance and supervision.

Head Of Department	**External Examiner**	**Internal Guide**

Date:

Department of Computer Science

Mithibai College of Arts, Chauhan Institute of Science & Amrutben
Jivanlal College of Commerce and Economics
(AUTONOMOUS) Affiliated to University of Mumbai
NAAC Reaccredited 'A' Grade, CGPA: 3.57 (February 2016)

ACKNOWLEDGEMENT

I avail this opportunity to express my sincere and deep gratitude to many who are a factor in helping me gain the knowledge and experience during the project and throughout the course.

I have great pleasure in presenting this project. The completion of this project is not merely due to only my own efforts but also due to the guidance given by our professors.

I am thankful to our project guide Mrs. Neelam Jain Ma’am & Mrs. Jayshree Ravi Ma’am for their support. I also thank H.O.D., Principal, Vice Principal and respected faculty members for their kind support and help throughout the entire course.

I am also highly indebted to our project team leader Mrs. Jayshree Ravi Ma’am for his continuous support & guidance without which this project couldn’t have been in reality.

Finally, I express my deep regards to all of those who stretch their helping in the execution of my project.

Table of Content

1. Project Introduction

Introducing "RentEase" - The Ultimate House Rental App

RentEase, the definitive solution for all your house rental needs, serves both tenants on the quest for their perfect homes and property owners in search of reliable renters. This groundbreaking platform redefines property management, making it not only efficient but also accessible to all parties involved.

Effortless Property Search: The task of finding the ideal rental property can often seem like a daunting endeavor, but RentEase transforms it into a seamless experience. The app boasts an extensive database of rental properties, ranging from cozy apartments to spacious houses and upscale condos. Its user-friendly interface ensures that users can effortlessly navigate through these options, guaranteeing they discover accommodations that perfectly align with their preferences and requirements.

For Tenants: RentEase offers tenants an experience that's not only transparent but also streamlined. Once users identify a property of interest, they can dive into comprehensive listings featuring high-resolution photos, virtual tours, and detailed property descriptions. The app simplifies the process of scheduling viewings, engaging in direct communication with property owners or managers, and submitting rental applications—all within a secure and integrated platform. Rest assured, RentEase takes privacy seriously, safeguarding personal information while facilitating seamless communication.

For Property Owners: Managing rental properties has never been easier. RentEase simplifies the entire process, saving property owners precious time and effort. Property owners can effortlessly create and manage listings, effectively showcasing their property's unique features through uploaded photos to attract potential tenants. The app also empowers property owners to receive and review rental applications, conduct background checks, and communicate with prospective renters—all within a seamless and integrated platform. Additionally, RentEase offers secure payment processing, making it convenient to collect rent and handle financial transactions with unwavering confidence.

RentEase is steadfast in its mission to create a seamless and secure rental experience for all parties involved. The ultimate goal is to simplify the rental process while providing greater control over housing needs. RentEase stands as the unequivocal solution for those seeking a hassle-free rental experience.

2. Preliminary Analysis

RentEase is not just another mobile application; it represents a visionary project that seeks to redefine the way individuals engage with the ever-evolving house rental market. Our core mission is to craft a user-centric platform that reimagines and enhances the entire house rental experience, placing accessibility, security, and efficiency at the forefront. Beyond the conventional rental features that most platforms offer, RentEase aims to integrate advanced functionalities that encompass robust user authentication, dynamic property listing management, effective communication tools, streamlined rental application processes, secure payment solutions, and responsive user support.

Our journey commences with an in-depth analysis of the housing market, revealing a compelling demand for a comprehensive house rental application. This demand is fueled by several key factors, including shifting housing trends characterized by a burgeoning population of renters. In today's fast-paced world, consumers prioritize convenience, security, and trust in their rental transactions. Furthermore, the omnipresence of mobile applications in modern life underscores the need for a dedicated, feature-rich app tailored specifically to the intricate world of house rentals.

RentEase is poised to introduce a suite of key features within the application, each meticulously designed to cater to the diverse needs of both tenants and landlords. For tenants, the application provides a seamless and transparent rental experience. It empowers them to explore detailed listings, schedule viewings, communicate directly with property owners or managers, and securely submit rental applications. Privacy is paramount, and RentEase ensures that personal information remains protected throughout the process.

For property owners, RentEase streamlines the property management process, allowing them to create and manage listings with ease, conduct background checks, receive and review rental applications, and securely manage financial transactions. The platform also offers a secure payment processing system, simplifying the collection of rent and financial management.

RentEase relies on cutting-edge technologies such as Flutter, Dart, Node.js, Express, and SQLite3 to create a robust and scalable foundation. These technologies have been meticulously chosen to ensure that the platform not only withstands the test of time but also accommodates growth and efficiently manages data.

To further enhance the project's prospects for success, we will conduct a comprehensive feasibility study. This study will meticulously assess the technical, operational, and financial aspects, providing us with invaluable insights into potential challenges, risks, and opportunities. It ensures a well-informed approach to development and launch.

Our stakeholders encompass a diverse group, including tenants, landlords, our dedicated development and technical team, investors, financial supporters, and regulatory authorities responsible for overseeing compliance with local housing and rental regulations.

In conclusion, RentEase stands poised to revolutionize the house rental market by offering a secure, convenient, and innovative platform. Through meticulous planning, a thorough feasibility study, and an unwavering commitment to meeting user needs, RentEase aspires to become a prominent player in the realm of house rental platforms. By simplifying the rental process and enhancing accessibility, RentEase seeks to elevate the experiences of tenants and landlords alike in the dynamic world of house rentals. Our journey is marked by a commitment to innovation, efficiency, and user satisfaction, making RentEase more than just an app; it's a transformative force in the rental market.

2.1 Limitations of the Present System

House rental apps, though popular, have certain limitations that have prompted the development of RentEase, a new house rental app. In this section, we will delve into these limitations, addressing aspects such as user experience, data management, features, and more.

1. User Experience:

House rental apps often suffer from complex user interfaces and navigation, making it challenging for users to find and secure suitable rental properties. The search and filtering options may not be intuitive, resulting in a frustrating user experience. Furthermore, communication with property owners or managers can be cumbersome.

2. Limited Property Listings:

Many house rental apps have a limited inventory of available properties. Users may find that the app does not cover a broad geographic area or lacks listings in specific neighborhoods. This limitation restricts users' choices and may lead to missed opportunities.

3. Data Management:

Users often face difficulties in managing their rental history and lease agreements within house rental apps. These apps may not provide effective tools for storing and organizing essential documents and information related to their rental properties, leading to a cluttered and disorganized experience.

4. Lack of Customization:

Some house rental apps lack customization options for user profiles and preferences. Users may not be able to personalize their search criteria adequately, resulting in recommendations and property listings that do not align with their specific needs and preferences.

5. Transaction Handling:

Handling rental transactions, including rent payments, security deposits, and lease renewals, can be challenging within some apps. Users may find it cumbersome to manage financial aspects of their rental agreements, leading to potential confusion and payment delays.

6. Limited Communication Features:

Communication features within house rental apps may be limited, hindering effective interaction between tenants and property owners or managers. Users may face difficulties in scheduling property viewings, reporting maintenance issues, or resolving disputes within the app.

7. Inadequate Tenant Screening:

Some apps do not provide comprehensive tenant screening tools for property owners and managers. This limitation may result in challenges in selecting reliable tenants, potentially leading to issues with rental payments and property maintenance.

8. Integration Challenges:

Integrating with other services, such as property management software or payment gateways, can be cumbersome within certain house rental apps. This limitation affects workflow automation and can result in manual data entry and repetitive tasks for users.

9. Security:

Security concerns, including data privacy and protection, can be a significant limitation. Users may worry about the safety of their personal and financial information within the app, leading to trust issues.

These limitations highlight the need for a comprehensive and user-friendly house rental app like RentEase. By addressing these challenges and offering a more streamlined and feature-rich platform, RentEase aims to provide an enhanced house rental experience for both property seekers and landlords, improving efficiency and convenience in the rental process.

2.2 Proposed System

RentEase is poised to be a groundbreaking house rental app that stands as a solution to the limitations encountered in existing rental platforms. The proposed system aims to reimagine the way users search for rental properties, interact with property owners and managers, and manage their rental journey. It is designed with a holistic approach to prioritize user experience, data management, and seamless communication.

1. User-Centric Design:

RentEase's commitment to user experience starts with a user-centric design philosophy. The proposed system meticulously crafts its user interface to be not only visually appealing but also highly intuitive. Every element on the screen is thoughtfully placed to optimize usability. This design approach ensures that users, regardless of their technological proficiency, can navigate the app effortlessly. The menus are streamlined, and calls to action are clear, providing an accessible and enjoyable experience. It focuses on minimizing the learning curve, making the app welcoming and efficient for everyone.

2. Intuitive Search and Filtering:

The proposed system elevates the search and filtering functionalities to a new level of intuitiveness and effectiveness. RentEase employs advanced algorithms that delve into user preferences, location, budget, and property features to provide tailored search results. A user-friendly filter system is a cornerstone of this approach. Users can refine their searches with ease, ensuring they find properties that align perfectly with their needs. Furthermore, intelligent recommendations based on user behavior enhance the house-hunting journey, presenting users with properties they might have otherwise overlooked.

4. Comprehensive User Guidance:

RentEase is dedicated to providing users with comprehensive and standardized guidance throughout their rental process. Whether users are submitting rental applications or seeking to understand complex lease agreements, the proposed system offers clear explanations and step-by-step instructions. Access to a wealth of resources, including FAQs, video tutorials, and legal guides, empowers users to navigate every stage of their rental journey with confidence. This commitment to user guidance minimizes confusion, reduces errors, and ensures users are well-informed every step of the way.

5. Mobile Responsiveness and Accessibility:

The proposed system recognizes the significance of mobile accessibility in the modern age. RentEase is meticulously optimized to provide users with a consistent and smooth experience, regardless of their chosen device. The system prioritizes mobile responsiveness, with a focus on fast loading times, user-friendly touch interfaces, and seamless transitions between devices. This ensures that users can access RentEase with the same efficiency and comfort whether they are using smartphones, tablets, or desktops.

6. Data Management Excellence:

RentEase's commitment to data management excellence sets it apart. The proposed system offers users the ability to effortlessly organize and store their rental history, lease agreements, and essential documents within the app. It leverages advanced cloud-based storage solutions that prioritize data security, accessibility, and reliability. Users can also harness the power of robust data analytics tools, which provide valuable insights into their rental portfolios. These insights enable informed decision-making and optimized property management, ensuring that RentEase isn't just about finding rental properties but also effectively managing them.

In conclusion, the proposed system for RentEase is more than a house rental app; it is a transformative platform that reimagines the house-hunting and property management experience. By championing user experience, intuitive search and filtering, efficient communication, comprehensive user guidance, mobile accessibility, and data management excellence, RentEase aims to set a new standard in the industry. The proposed system is driven by a deep commitment to deliver convenience, empowerment, and peace of mind to both house hunters and property owners. This vision makes RentEase the preferred choice for all rental needs, marking a significant leap forward in the rental platform landscape.

This extended description provides a more comprehensive view of how RentEase intends to reshape the rental platform landscape, prioritizing user experience, data management, and seamless communication for all users.

2.3 Feasibility Study

2.3.1 Economic Feasibility:

The economic feasibility study for RentEase involves assessing the financial viability of the proposed rental property app. The following key considerations are taken into account:

- Market Analysis: Thorough market research is conducted to understand the demand for rental properties in the target locations. This includes analyzing rental trends, vacancy rates, and the preferences of potential renters. Understanding the competition in the market and identifying gaps that RentEase can fill is essential to determine the app's market potential.
- Revenue Generation: Several revenue generation strategies are explored to ensure the app can sustain itself financially. This may include charging landlords a fee for listing their properties on the platform or offering premium features for a subscription fee. Additionally, RentEase may consider partnerships with real estate agents or property management companies for a commission on successful rental agreements facilitated through the app.
- Cost Analysis: A detailed cost analysis is conducted to determine the expenses associated with app development, maintenance, and operations. This includes software development costs, server hosting, marketing, customer support, and any other ongoing expenses. The goal is to identify cost-effective solutions and budget constraints while maintaining the app's quality and user experience.
- Return on Investment (ROI) Analysis: The ROI is calculated to estimate the app's profitability over time. This involves comparing the projected revenue with the total investment to determine if the app can generate a positive ROI within a reasonable timeframe.

2.3.2 Technical Feasibility:

The technical feasibility study for RentEase involves evaluating the app's technical requirements and capabilities. Key considerations include:

- Technology Evaluation: Assessing the suitability of technologies such as React Native or Flutter for developing a cross-platform mobile app. Evaluating the scalability of the chosen technology to handle an increasing number of users and property listings is crucial.
- Resource Availability: Ensuring that the necessary technical expertise and resources are available to develop and maintain the app. This includes hiring skilled developers, designers, and potentially outsourcing certain tasks if required.
- Stakeholder Engagement: Involving potential users, landlords, and property managers in the technical feasibility study to gather feedback on their needs and preferences. This input helps

shape the app's features and functionality to ensure it meets the expectations of its target audience.

- Data Security and Privacy: Ensuring that the app complies with data protection regulations and implements robust security measures to protect user data and transactions.

2.3.3 Operational Feasibility:

The operational feasibility study for RentEase involves evaluating the practicality of implementing and running the app. Key considerations include:

- User Engagement: Understanding how potential users will interact with the app and ensuring that it is user-friendly and intuitive. Conducting usability testing and gathering feedback from users during the development process is essential.
- Scalability and Performance: Assessing whether the app's infrastructure can handle a growing user base and increasing property listings without compromising on performance and responsiveness.
- Stakeholder Support: Ensuring that landlords, property managers, and renters are willing to adopt the app and actively use it for their rental needs.
- Legal and Regulatory Compliance: Ensuring that the app complies with all relevant laws and regulations related to rental property transactions, data protection, and user rights.

2.4 Stakeholders

Stakeholders play a crucial role in shaping the vision, development, and success of RentEase. Identifying and engaging with these stakeholders is an essential aspect of the preliminary investigation to ensure alignment with goals and expectations.

1. Primary Stakeholders:

Primary stakeholders are individuals or groups who have a direct and immediate interest in the development and success of RentEase. They include:

- Development Team: The development team is at the core of RentEase's creation. This group of software engineers, designers, and project managers is responsible for bringing the app to life. Their technical expertise, creativity, and dedication are pivotal to the app's development and functionality.
- Investors: Investors provide crucial funding and financial support for RentEase. Their interests lie in the app's profitability and long-term sustainability. They may also contribute strategic guidance and business expertise to ensure a return on investment.
- Users: Users are the lifeblood of RentEase. They encompass property seekers (tenants) and property owners or managers (landlords). Users directly benefit from the app's features and usability, making their feedback and satisfaction vital for RentEase's success.

2. Secondary Stakeholders:

Secondary stakeholders are entities or groups that are indirectly affected by or have an influence on RentEase's development and implementation. They include:.

- Industry Associations: Associations related to the real estate and rental industry may influence industry standards and practices. Collaborating with or obtaining insights from these associations can be valuable for RentEase.
- Business Partners: Business partners, such as payment gateway providers or property listing websites, can impact RentEase's functionality and reach. Integration partnerships with these entities may enhance the app's features and accessibility.

3. Stakeholder Roles:

Clearly defined roles and responsibilities for each stakeholder group are essential to ensure effective collaboration and communication throughout the development and implementation of RentEase:

- Development Team: Responsible for app design, coding, testing, and ongoing maintenance, ensuring that RentEase operates smoothly.
- Investors: Provide financial support, strategic guidance, and resources to facilitate app development and growth.
- Users: Actively engage with the app, utilizing its features for property management, search, and rental transactions.
- Industry Associations: Offer insights, networking opportunities, and guidance on industry best practices.
- Business Partners: Collaborate to enhance RentEase's functionality through integrations, expanding its utility and reach.

4. Stakeholder Engagement:

Active and ongoing engagement with stakeholders is critical throughout the development and implementation phases of RentEase:

- Development Team: Regular meetings, feedback sessions, and collaborative discussions to ensure alignment with project goals and milestones.
- Investors: Periodic updates on progress, financial reports, and strategic discussions to maintain transparency and secure continued support.
- Users: Continuous communication channels for feedback, support requests, and feature suggestions to enhance user experience.
- Industry Associations: Participation in industry events, sharing insights, and seeking guidance on market trends and practices.
- Business Partners: Ongoing collaboration to establish and maintain integration partnerships, expanding RentEase's capabilities.

This comprehensive overview identifies the key stakeholders involved in the preliminary investigation of RentEase and their respective roles and engagement strategies. Stakeholder engagement is integral to ensuring that RentEase aligns with expectations and achieves its objectives effectively.

2.5 Technologies Used

The preliminary investigation of RentEase relies on a suite of cutting-edge technologies to assess its feasibility and develop a clear understanding of its potential. Leveraging a diverse set of tools, languages, and platforms, the investigation lays the foundation for RentEase's future development and implementation.

1. Flutter:

Flutter, an open-source UI software development kit, plays a pivotal role in RentEase's preliminary investigation. This versatile framework is employed for creating RentEase's user-friendly and visually appealing mobile app interface. Flutter's cross-platform capabilities ensure that RentEase is accessible across various mobile devices, simplifying development and ensuring a consistent user experience.

2. Dart:

Dart, the programming language powering Flutter, is at the core of RentEase's mobile app development. Dart's robustness, speed, and support for modern programming features facilitate the creation of RentEase's app logic and functionality. It allows for efficient communication between the app's front-end and back-end components.

3. Node.js:

Node.js, known for its non-blocking, event-driven architecture, is chosen to power RentEase's server-side logic. Node.js efficiently handles concurrent requests, making it well-suited for real-time interactions and communication within the app. It plays a critical role in managing user data, facilitating smooth communication, and handling app security.

4. Express:

Express, a minimalist web application framework for Node.js, is utilized to streamline the development of RentEase's server-side components. Its simplicity and robust routing capabilities help create RESTful APIs that enable seamless interaction between the mobile app and the back-end server.

5. SQLite3:

SQLite3 is employed as the database management system for RentEase's preliminary investigation. Its lightweight, serverless, and self-contained nature makes it a suitable choice for the early stages of development and feasibility assessment. SQLite3 is used to store and manage user data, ensuring data integrity and accessibility.

6. Additional Tools and Technologies:

In addition to the core technologies mentioned above, several other tools and technologies are employed to support the preliminary investigation:

- Version Control (e.g., Git): Version control systems are used to manage and track changes in RentEase's codebase, ensuring collaboration among team members and code integrity.
- APIs and Web Services: Integration with external services and APIs is explored to gather data, enhance functionality, and assess potential partnerships for RentEase.
- Data Analytics Tools: Tools for data analysis and visualization help in understanding user behavior and preferences, contributing to RentEase's feasibility assessment.

The technologies employed in the preliminary investigation of RentEase form a robust foundation for assessing its feasibility and potential. By leveraging Flutter, Dart, Node.js, Express, SQLite3, and a suite of supporting tools and technologies, the investigation explores the capabilities and limitations of RentEase while paving the way for its future development and implementation.

This comprehensive overview highlights the technologies used in the preliminary investigation of RentEase, showcasing how each technology contributes to the assessment and development of the proposed rental management system.

3. Module Description

User Module:

1. User Registration and Login:

- Users can sign up with essential details like name, email, and password.
- Users may also have the option to sign up using their social media accounts for a quicker registration process.
- Registered users can log in using their credentials.

2. Create and Manage Rental Profiles:

- Users can create their rental profiles, specifying their preferences such as location, budget, property type, and other essential details.
- Users can edit and manage their rental profiles based on changing requirements.

3. Search and View Rental Listings:

- Users can search for rental properties based on their preferences, using filters such as location, rent range, property type, and amenities.
- Users can view detailed information about rental listings, including property photos, descriptions, and contact information for landlords.

4. Save Favourite Listings:

- Users can save their favourite rental listings to a personalized "Favourites" for future reference.

6. Apply for Rental Properties:

- Users can submit rental applications to landlords through the app, providing necessary documents and details for consideration.

9. Assistance and Support:

- Users can access customer support through the app to address any issues or queries related to their rental process.

Admin Module:

1. Admin Login:

- Admins log in using their credentials to access the admin panel.

2. User and Property Listing Management:

- Admins oversee user registrations, rental profiles, and property listings to ensure accuracy and compliance with platform guidelines.

- Admins have the authority to approve or disapprove property listings based on verification of property details.

4. Dispute Resolution:

- Admins mediate and resolve disputes between users and landlords, ensuring fair and transparent resolutions.

5. Platform Maintenance and Security:

- Admins ensure the overall stability, security, and maintenance of the app, including software updates and server management.

6. User Support:

- Admins provide support to users and address any technical or operational issues they may encounter.

7. Landlord Verification:

- Admins perform thorough verification checks on landlords to ensure their credibility and legitimacy.

8. Performance Analytics:

- Admins can access performance analytics, including user engagement, rental property popularity, and other key metrics to optimize the platform's performance.

Landlord Module:

1. Landlord Registration and Login:

- Landlords can sign up with essential details like name, email, and password to create their accounts.
- Landlords may also have the option to sign up using their social media accounts for a quicker registration process.
- Registered landlords can log in using their credentials.

2. Property Listing Management:

- Landlords can create and manage their property listings, providing detailed information, photos, and rental terms.
- Landlords can update or remove listings as per property availability.

3. Review Rental Applications:

- Landlords can review rental applications submitted by users and communicate with potential tenants through the app.

6. Rental Agreement and Documentation:

- Landlords can facilitate the rental agreement process, sharing necessary documents and terms with tenants.

7. Rental Payment Management:

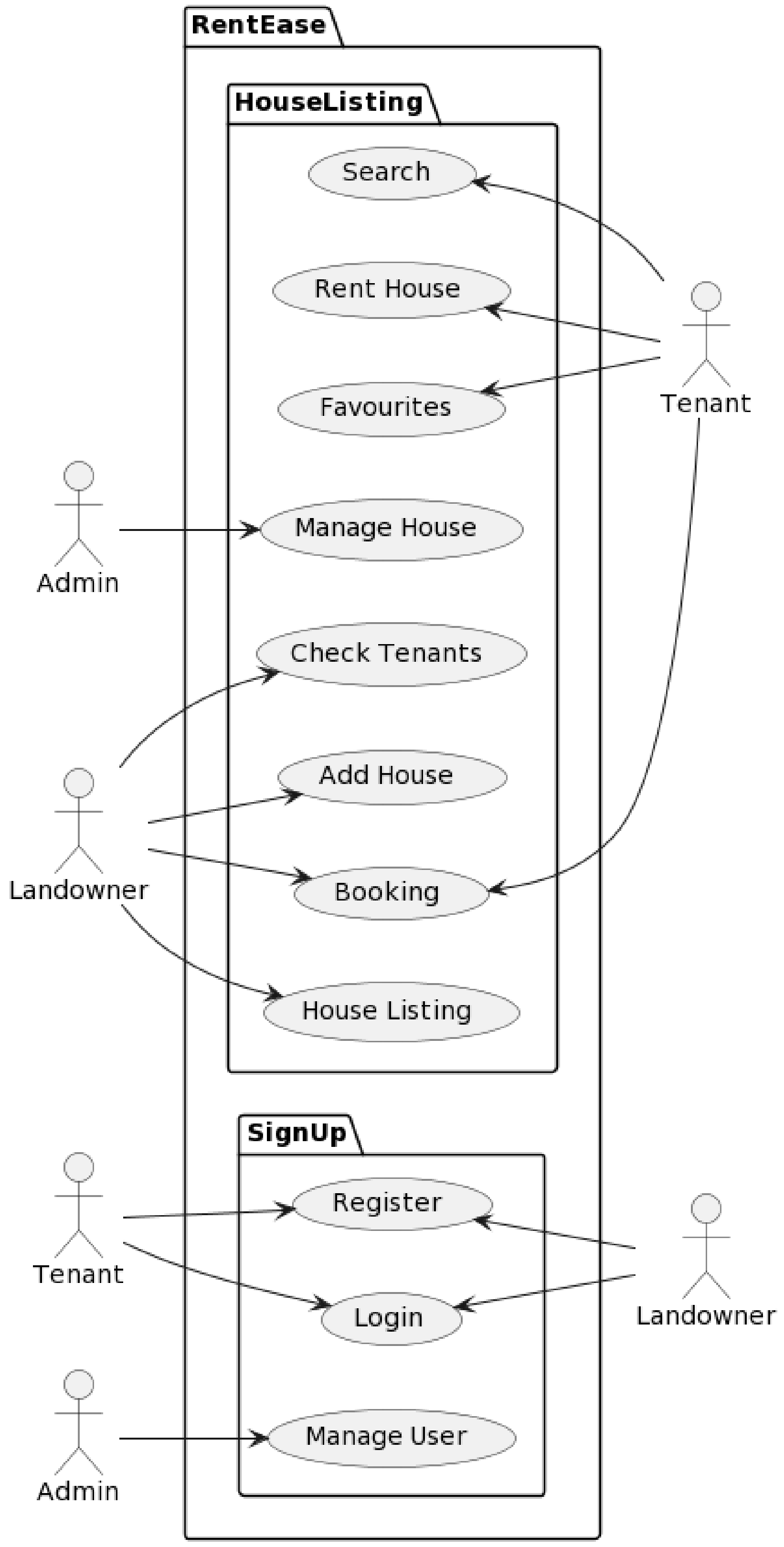
- Landlords can manage rental payments, view payment history, and send payment reminders to tenants.

10. Assistance and Support:

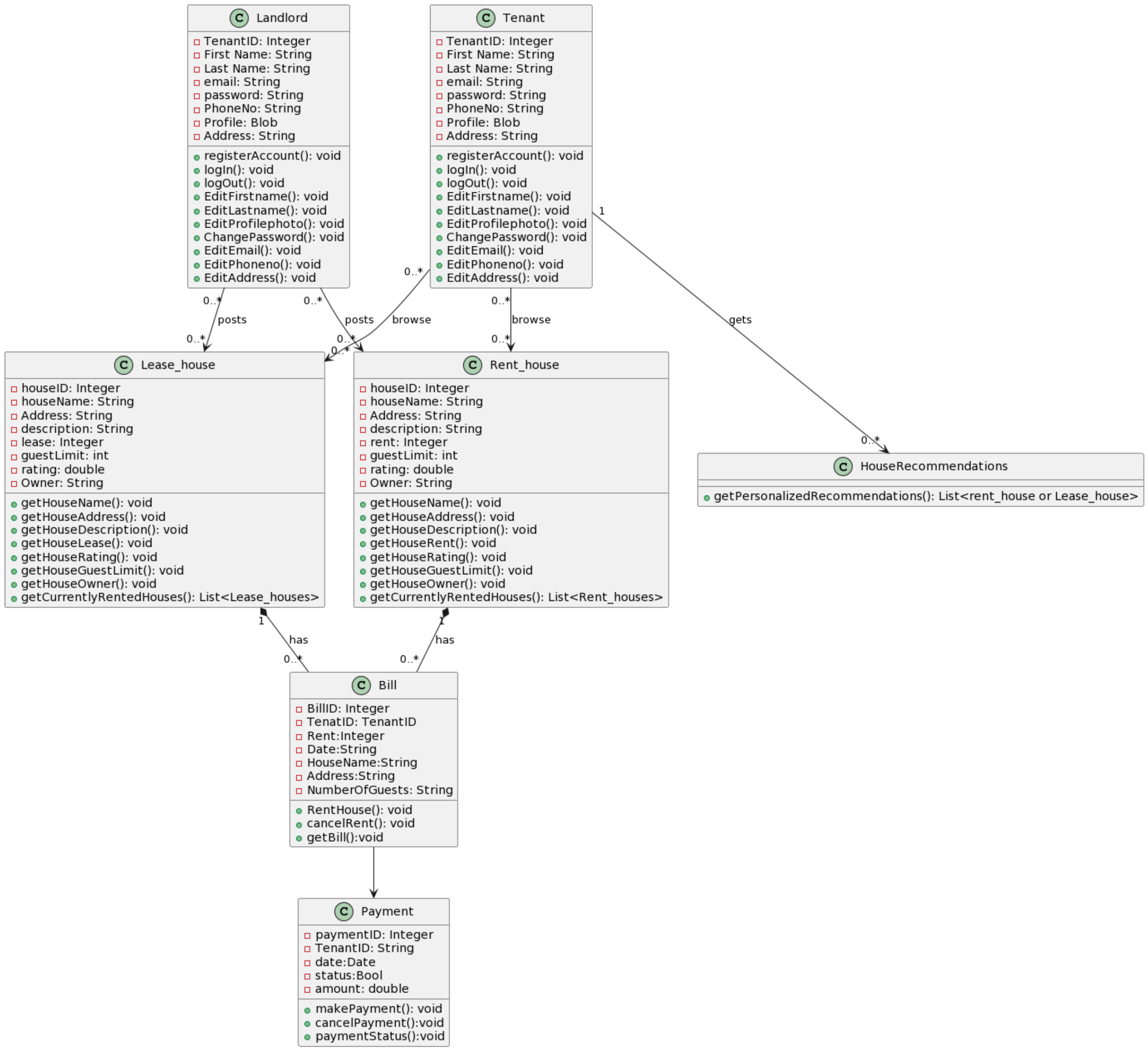
- Landlords can access customer support through the app for any queries or assistance needed during the rental process.

4. System Design

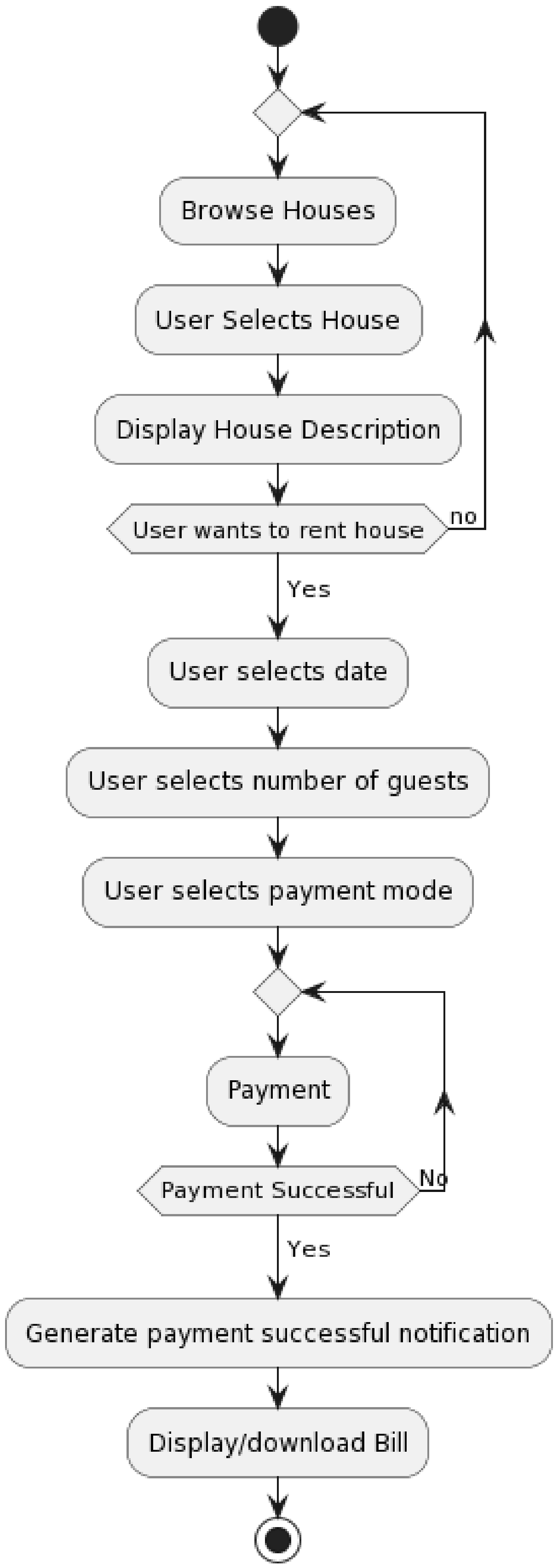
- **Use Case Diagram**



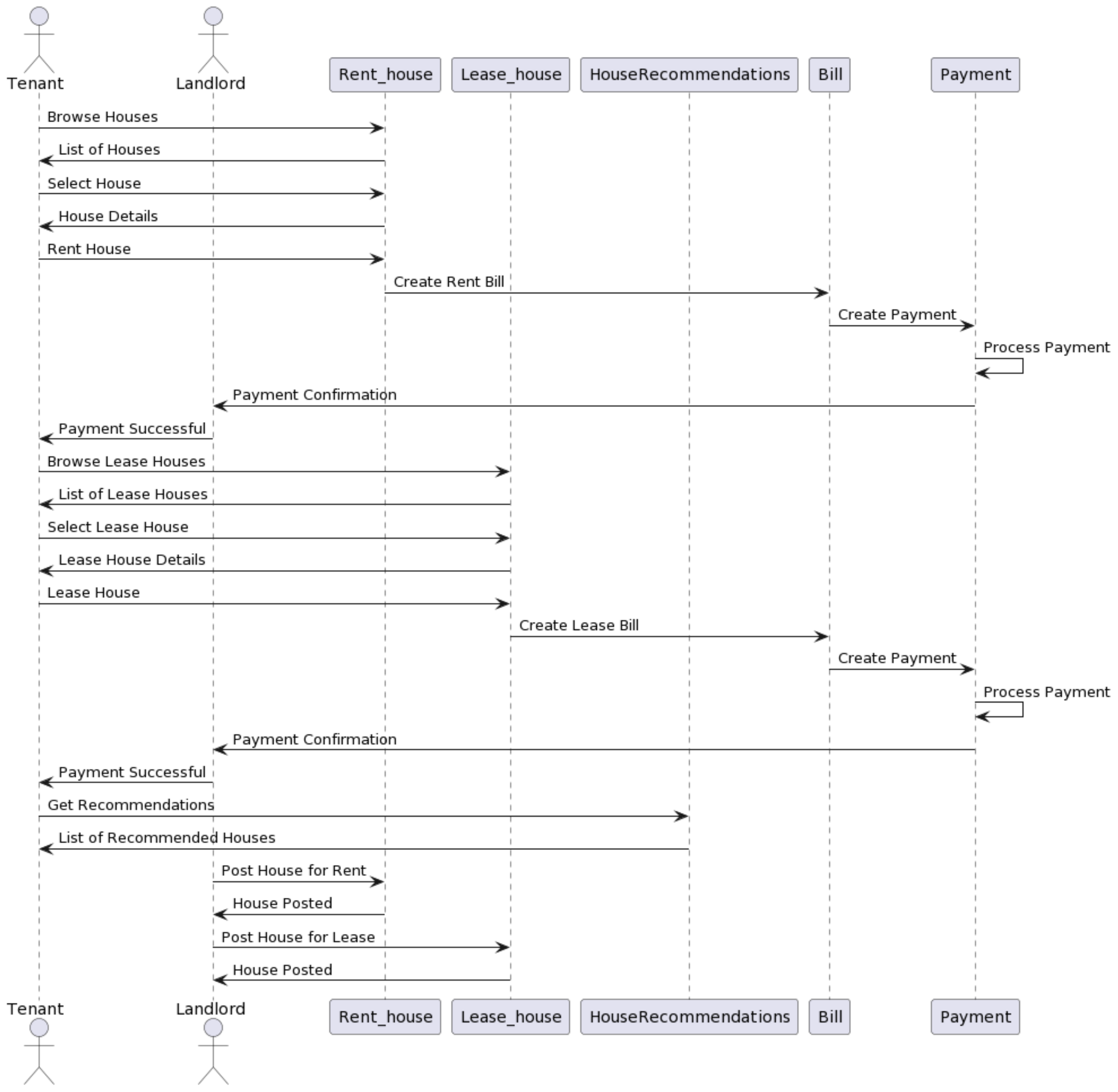
- **Class Diagram**



- **Activity Diagram**



- **Sequence Diagram**



5. Coding

Code For Sign In Page

```
class SignForm extends StatefulWidget {
 @override
 _SignFormState createState() => _SignFormState();
}

class _SignFormState extends State<SignForm> {
 final _formKey = GlobalKey<FormState>();
 TextEditingController emails= new TextEditingController();
 TextEditingController passwords= new TextEditingController();
 String? email;
 String? password;
 bool? remember = false;
 final List<String?> errors = [];

 void addError({String? error}) {
  if (!errors.contains(error))
   setState(() {
    errors.add(error);
   });
 }

 void removeError({String? error}) {
  if (errors.contains(error))
   setState(() {
    errors.remove(error);
   });
 }

 Future<void> getUserData() async {
  final url = Uri.parse('http://'+api+':3000/getuserdata');

  try {
   final response = await http.get(url);

   if (response.statusCode == 200) {
    // Parse the JSON response
    final List<dynamic> userData = jsonDecode(response.body);

    final user = userData.firstWhere(
        (user) => user['email'] == emails.text && user['pass'] == passwords.text,
      orElse: () => null,
    );

    if (user != null) {
      Navigator.push(context, MaterialPageRoute(builder: (context)=>HomeScreen(email:
emails.text,password: passwords.text)));
    } else {
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Invalid email or password. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
   } else {
    print('Failed to retrieve data');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Failed to retrieve data. Please try again.'),
        backgroundColor: Colors.red,
      ),
    );
   }
  } catch (e) {
   print('Error: $e');
   ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text('An error occurred. Please try again later.'),
      backgroundColor: Colors.red,
      ),
    );
  }
}



@override
Widget build(BuildContext context) {
  return Form(
    key: _formKey,
    child: Column(
      children: [
        buildEmailFormField(),
        SizedBox(height: getProportionateScreenHeight(30)),
        buildPasswordFormField(),
        SizedBox(height: getProportionateScreenHeight(30)),

        FormError(errors: errors),
        SizedBox(height: getProportionateScreenHeight(20)),
        DefaultButton(
          text: "Continue",
          press: () {
            if (_formKey.currentState!.validate()) {
              _formKey.currentState!.save();
              getUserData();
              insertData();
            }
          },
        ),
      ],
    ),
  );
}

TextFormField buildPasswordFormField() {
  return TextFormField(
    controller: passwords,
    obscureText: true,
    onSaved: (newValue) => password = newValue,
    onChanged: (value) {
      if (value.isNotEmpty) {
        removeError(error: kPassNullError);
      } else if (value.length >= 8) {
        removeError(error: kShortPassError);
      }
      return null;
    },
    validator: (value) {
      if (value!.isEmpty) {
        addError(error: kPassNullError);
        return "";
      } else if (value.length < 8) {
        addError(error: kShortPassError);
        return "";
      }
      return null;
    },
    decoration: InputDecoration(
      labelText: "Password",
      hintText: "Enter your password",
      floatingLabelBehavior: FloatingLabelBehavior.always,
      suffixIcon: CustomSurffixIcon(svgIcon: "assets/icons/Lock.svg"),
    ),
  );
}

TextFormField buildEmailFormField() {
  return TextFormField(
    controller: emails,
    keyboardType: TextInputType.emailAddress,
    onSaved: (newValue) => email = newValue,
    onChanged: (value) {
      if (value.isNotEmpty) {
        removeError(error: kEmailNullError);
      } else if (emailValidatorRegExp.hasMatch(value)) {
        removeError(error: kInvalidEmailError);
      }
      return null;
    },
    validator: (value) {
      if (value!.isEmpty) {
        addError(error: kEmailNullError);
        return "";
      } else if (!emailValidatorRegExp.hasMatch(value)) {
        addError(error: kInvalidEmailError);
        return "";
      }
      return null;
    },
    decoration: InputDecoration(
      labelText: "Email",
      hintText: "Enter your email",
      floatingLabelBehavior: FloatingLabelBehavior.always,
      suffixIcon: CustomSurffixIcon(svgIcon: "assets/icons/Mail.svg"),
    ),
  );
}

Future<void> insertData() async {
  String profilepic = "user.svg";
  print("hi");

  final url = Uri.parse('http://'+api+':3000/insertcurrent');

  try {
    final response = await http.post(
      url,
      headers: {
        'Content-Type': 'application/json',
      },
        body: jsonEncode({
          'email': emails.text,
          'pass': passwords.text
        }),
      );

      if (response.statusCode == 200) {
        print('Data inserted successfully');
      } else {
        print('Failed to insert data');
      }
    } catch (e) {
      print('Error: $e');
    }
  }

}
```

Singn Up Page :

```
class SignUpForm extends StatefulWidget {
  @override
  _SignUpFormState createState() => _SignUpFormState();
}

class _SignUpFormState extends State<SignUpForm> {

  bool register= false;

  final _formKey = GlobalKey<FormState>();
  TextEditingController emails=TextEditingController();
  TextEditingController password=TextEditingController();
  TextEditingController conform_password=TextEditingController();
  bool remember = false;
  final List<String?> errors = [];

  void addError({String? error}) {
    if (!errors.contains(error))
      setState(() {
        errors.add(error);
      });
  }
```

```
void removeError({String? error}) {
  if (errors.contains(error))
    setState(() {
      errors.remove(error);
    });
}


@override
Widget build(BuildContext context) {
  return Form(
    key: _formKey,
    child: Column(
      children: [
        buildEmailFormField(),
        SizedBox(height: getProportionateScreenHeight(30)),
        buildPasswordFormField(),
        SizedBox(height: getProportionateScreenHeight(30)),
        buildConformPassFormField(),
        FormError(errors: errors),
        SizedBox(height: getProportionateScreenHeight(40)),
        DefaultButton(
          text: "Continue",
          press: () {
            if (_formKey.currentState!.validate()) {
              _formKey.currentState!.save();
              getUserData();
              // if all are valid then go to success screen
              if (register==false){
                Navigator.push(context, MaterialPageRoute(builder: (context)=>CompleteProfileScreen(email:
emails.text,pass: password.text,)));
              }
              else{
                ScaffoldMessenger.of(context).showSnackBar(
                  SnackBar(
                    content: Text('Email already registered'),
                    backgroundColor: Colors.red,
                  ),
                );
              }
            }
          },
        ),
      ],
    ),
  );
}
TextFormField buildEmailFormField() {
  return TextFormField(
    controller: emails,
    keyboardType: TextInputType.emailAddress,
    onChanged: (value) {
      if (value.isNotEmpty) {
        removeError(error: kEmailNullError);
      } else if (emailValidatorRegExp.hasMatch(value)) {
        removeError(error: kInvalidEmailError);
      }
      return null;
    },
    validator: (value) {
      if (value!.isEmpty) {
        addError(error: kEmailNullError);
        return "";
      } else if (!emailValidatorRegExp.hasMatch(value)) {
        addError(error: kInvalidEmailError);
        return "";
      }
      return null;
    },
    decoration: InputDecoration(
      labelText: "Email",
      hintText: "Enter your email",
      floatingLabelBehavior: FloatingLabelBehavior.always,
      suffixIcon: CustomSurffixIcon(svgIcon: "assets/icons/Mail.svg"),
    ),
  );
}
TextFormField buildPasswordFormField() {
  return TextFormField(
    controller: password,
    obscureText: true,
    onChanged: (value) {
      if (value.isNotEmpty) {
        removeError(error: kPassNullError);
      } else if (value.length >= 8) {
        removeError(error: kShortPassError);
      }

    },
    validator: (value) {
      if (value!.isEmpty) {
        addError(error: kPassNullError);
        return "";
      } else if (value.length < 8) {
        addError(error: kShortPassError);
        return "";
      }
      return null;
    },
    decoration: InputDecoration(
      labelText: "Password",
      hintText: "Enter your password",
      floatingLabelBehavior: FloatingLabelBehavior.always,
      suffixIcon: CustomSurffixIcon(svgIcon: "assets/icons/Lock.svg"),
    ),
  );
}

TextFormField buildConformPassFormField() {
  return TextFormField(
      controller: conform_password,
      obscureText: true,
      onChanged: (value) {
        if (value.isNotEmpty) {
          removeError(error: kPassNullError);
        } else if (value.isNotEmpty && password.text == conform_password.text) {
          removeError(error: kMatchPassError);
        }
/*
        conform_password.text = value;
*/
      },
      validator: (value) {
        if (value!.isEmpty) {
          addError(error: kPassNullError);
          return "";
        } else if ((password.text != value)) {
          addError(error: kMatchPassError);
          return "";
        }
        return null;
      },
      decoration: InputDecoration(
        labelText: "Confirm Password",
        hintText: "Re-enter your password",
        floatingLabelBehavior: FloatingLabelBehavior.always,
        suffixIcon: CustomSurffixIcon(svgIcon: "assets/icons/Lock.svg"),
      ),
    );
  }

  Future<void> getUserData() async {
    final url = Uri.parse('http://'+api+':3000/getuserdata');

    try {
      final response = await http.get(url);

      if (response.statusCode == 200) {
        // Parse the JSON response
        final List<dynamic> userData = jsonDecode(response.body);

        // Check if the provided email and password match any user record
        final user = userData.firstWhere(
              (user) => user['email'] == emails.text,
          orElse: () => null,
        );

        if (user != null) {
          // Credentials are correct, navigate to the next page (HomeScreen)
          setState(() {
            register=true;
          });
        } else {
          // Credentials are incorrect, show an error message
          register = false;
        }
      } else {
        print('Failed to retrieve data');
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('Failed to retrieve data. Please try again.'),
            backgroundColor: Colors.red,
          ),
        );
      }
    } catch (e) {
      print('Error: $e');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('An error occurred. Please try again later.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }

  }
```

Code for Booking Page:

```
class Bill extends StatefulWidget {
  final Map<String,dynamic> property;
  final String? start,end;

  const Bill({super.key,required this.property,required this.start,required this.end});

  @override
  State<Bill> createState() => _BillState();
}

class _BillState extends State<Bill> {

  String email="";
  String pass="";
  String name="";
  int cf = 300;
  late var _razorpay;

  @override
  void initState() {
    // TODO: implement initState
    _razorpay = Razorpay();
    _razorpay.on(Razorpay.EVENT_PAYMENT_SUCCESS, _handlePaymentSuccess);
    _razorpay.on(Razorpay.EVENT_PAYMENT_ERROR, _handlePaymentError);
    _razorpay.on(Razorpay.EVENT_EXTERNAL_WALLET, _handleExternalWallet);
    getcurrentData();
    getUserData();
    super.initState();
  }
```

```
 void _handlePaymentSuccess(PaymentSuccessResponse response) {
   insertData();
   Navigator.push(context, MaterialPageRoute(builder: (context)=>HomeScreen(email: email, password:
pass)));
 }

 void _handlePaymentError(PaymentFailureResponse response) {
   showDialog(
     context: context,
     builder: (BuildContext context) {
       return AlertDialog(
         title: Text("Payment Failed"),
         content: Text("There was an issue processing your payment. Please try again."),
         actions: [
           TextButton(
             onPressed: () {
               Navigator.of(context).pop();
             },
             child: Text("OK"),
           ),
         ],
       );
     },
   );
   print("Payment Fail");
 }

 void _handleExternalWallet(ExternalWalletResponse response) {
 }


 @override
 Widget build(BuildContext context) {
   num tf = int.parse(widget.property['rent']) + int.parse(widget.property['security']) + cf;

   return SafeArea(
     child: Scaffold(
       appBar: AppBar(
         leading: IconButton(
           onPressed: ()=>Navigator.pop(context),
           icon: Icon(
             Icons.arrow_back_rounded,
             color: kblackcolor,
             size: 30,
           ),
         ),
       ),
       body: Column(
             mainAxisAlignment: MainAxisAlignment.center,
             children: [
               Padding(
                 padding: const EdgeInsets.symmetric(
                   horizontal: 10,
                 ),
  child: Card(
    elevation: 8,
    child: Container(
      height: 350,
      width: 530,
      child: Padding(
        padding:
        const EdgeInsets.symmetric(horizontal: 10, vertical: 15),
        child: Column(
          children: [
            Text(
              "Rent Summary",
              style: TextStyle(
                  fontWeight: FontWeight.bold, color: Colors.black),
            ),
            SizedBox(
              height: 10,
            ),
            ListTile(
              contentPadding: EdgeInsets.symmetric(horizontal: 10),
              leading: Container(
                  width: 150,
                  child:
                  Text("House Rent")),
              trailing: Text("₹" + widget.property['rent']),
            ),
            Divider(),
            ListTile(
              contentPadding: EdgeInsets.symmetric(horizontal: 10),
              leading: Text("Maintenance Fee"),
              trailing: Text("₹" + widget.property['security']),
            ),
            Divider(),
            ListTile(
              contentPadding: EdgeInsets.symmetric(horizontal: 10),
              leading: Text("Convenience Fees"),
              trailing: Text("₹" + cf.toString()),
            ),
            Divider(),
            ListTile(
              contentPadding: EdgeInsets.symmetric(horizontal: 10),
              leading: Text(
                "Total Rent",
                style: TextStyle(
                    fontWeight: FontWeight.bold, color: Colors.black),
              ),
              trailing: Text(tf.toString()),
            ),
          ],
        ),
      ),
    ),
  ),
),
SizedBox(
              height: 20,
            ),
            ElevatedButton(
                child: Text("Pay Amount"),
                style: ElevatedButton.styleFrom(
                  backgroundColor: kPrimaryColor,
                  foregroundColor: Colors.white
                ),
                onPressed: () {
                  ///Make payment
                  var options = {
                    'key': "rzp_test_UWNue9UPsKaP3f",
                    'amount': (tf * 100).toString(), //So its pay 500
                    'name': 'RentEase',
                    'description': 'Demo',
                    'timeout': 500, // in seconds
                    'prefill': {
                      'contact': '8097809001',
                      'email': 'rentease@gmail.com'
                    }
                  };
                  _razorpay.open(options);
                })
          ],

      ),
    ),
  );
}

@override
void dispose() {
  _razorpay.clear();
  super.dispose();
}

Future<void> insertData() async {

  final url = Uri.parse('http://' + api + ':3000/insertbooking');

  try {
    final response = await http.post(
      url,
      headers: {
        'Content-Type': 'application/json',
      },
      body: jsonEncode({
        'user_name':name,
        'name':widget.property['name'],
        'address':widget.property['address'],
        'owner': widget.property['owner'],
        'rent': (int.parse(widget.property['rent']) + int.parse(widget.property['security']) + cf).toString(),
        'start': widget.start,
        'end':widget.end
      }),
    );

    if (response.statusCode == 200) {
      print('Data inserted successfully');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Registration successful!'),
          backgroundColor: Colors.green,
        ),
      );
    } else {
      print('Failed to insert data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Registration failed. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('An error occurred. Please try again later.'),
        backgroundColor: Colors.red,
      ),
    );
  }
}

Future<void> getUserData() async {
  final url = Uri.parse('http://'+api+':3000/getuserdata');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final List<dynamic> userData = jsonDecode(response.body);

      final user = userData.firstWhere(
            (user) => user['email'] == email && user['pass'] == pass,
        orElse: () => null,
      );

      if (user != null) {
        setState(() {
          name = user['fname'] + ' ' + user['lname'];
        });
      } else {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('Invalid email or password. Please try again.'),
            backgroundColor: Colors.red,
          ),
        );
      }
    } else {
      print('Failed to retrieve data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Failed to retrieve data. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('An error occurred. Please try again later.'),
        backgroundColor: Colors.red,
      ),
    );
  }
}

Future<void> getcurrentData() async {
  final url = Uri.parse('http://'+api+':3000/getcurrentdata');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final List<dynamic> userData = jsonDecode(response.body);


      if (userData != null) {
        setState(() {
          email = userData.last['email'];
          pass = userData.last['pass'];
        });
      } else {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text('Invalid email or password. Please try again.'),
            backgroundColor: Colors.red,
          ),
        );
      }
    } else {
      print('Failed to retrieve data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Failed to retrieve data. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');
```

```
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('An error occurred. Please try again later.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }

}
```

Code for Landlord dashboard:

```
class Body extends StatefulWidget {
  final String? email;
  final String? pass;

  const Body({super.key, required this.email, required this.pass});
  @override
  State<Body> createState() => _BodyState();
}

class _BodyState extends State<Body> {
  String name="";
  List<dynamic> property=[];
  List<dynamic> rental=[];
  List<dynamic> lease_h=[];
  List<dynamic> booking=[];
  List<dynamic> property_p=[];



  @override
  void initState() {
    getownerData();
    gethouseData();
    getrentalData();
    getlease_hData();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: Stack(
        children: [
          Container(
            height: 150,
            decoration: BoxDecoration(color: kPrimaryColor,),
            padding: EdgeInsets.only(bottom: 40),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
        Padding(
          padding: const EdgeInsets.only(left: 14.0),
          child: Text(
            'Landlord Dashboard',
            textAlign: TextAlign.center,
            style: TextStyle(
              color: Colors.white,
              fontSize: getProportionateScreenHeight(24),
            ),),
        ),

      ],
    ),
  ),
  SizedBox(height: getProportionateScreenHeight(20),),
  Padding(
    padding: const EdgeInsets.only(top: 116.372),
    child: Stack(
      children: [
        Container(
          height: SizeConfig.screenHeight*0.798,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.only(topLeft: Radius.circular(25),topRight: Radius.circular(25)),
            color: Color(0xFFF5F6F9),
          ),
          child: Column(
            children: [
              SizedBox(height: getProportionateScreenHeight(150),),
              Container(
                height: SizeConfig.screenHeight*0.545,
                child: SingleChildScrollView(
                  scrollDirection: Axis.vertical,
                  child: Column(
                    children: [
                      near_you(email: widget.email, password: widget.pass),
                      InkWell(child: post_ad(),onTap:()=>Navigator.push(context, MaterialPageRoute(builder:
(context)=>addPropertyScreen(email: widget.email, password: widget.pass)))),
                      SizedBox(height: getProportionateScreenHeight(20),)
                    ],
                  ),
                ),
              ),
            ],
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(left: 22.0,top: 20,bottom: 20),
          child: SingleChildScrollView(
            scrollDirection: Axis.horizontal,
            child:
            Row(
              children: [
                stats(
                  title: 'Properties',
                  number: property.length,
                    color: Colors.green,
                    icon: 'house',
                  ),
                  SizedBox(width: getProportionateScreenWidth(13),),
                  stats(
                    title: 'Rentals',
                    number: rental.length,
                    color: Colors.orange,
                    icon: 'house',
                  ),
                  SizedBox(width: getProportionateScreenWidth(13),),
                  stats(
                    title: 'Lease',
                    number: lease_h.length,
                    color: Colors.blue,
                    icon: 'house',
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    ),
  ],
 ),
);
}

Future<void> getownerData() async {
  final url = Uri.parse('http://'+api+':3000/getownerdata');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final List<dynamic> userData = jsonDecode(response.body);

      final user = userData.firstWhere(
            (user) => user['email'] == widget.email && user['pass'] == widget.pass,
        orElse: () => null,
      );

      if (user != null) {
        setState(() {
          name = user['fname'] + ' ' + user['lname'];
        });
      } else {
        print('wrong credentials');
      }
    } else {
      print('Failed to retrieve data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Failed to retrieve data. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');

  }
}

Future<void> gethouseData() async {
  final url = Uri.parse('http://'+api+':3000/gethousedata');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final List<dynamic> houseData = jsonDecode(response.body);

      final List<dynamic> userHouses = houseData.where(
            (house) => house['owner'] == name,
      ).toList();

      if (userHouses.isNotEmpty) {
        setState(() {
          property = userHouses;
        });
      } else {
        property = [];
      }
    } else {
      print('Failed to retrieve data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Failed to retrieve data. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');

  }
}

Future<void> getrentalData() async {
  final url = Uri.parse('http://'+api+':3000/gethousedata');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final List<dynamic> houseData = jsonDecode(response.body);

      final List<dynamic> userHouses = houseData.where(
        (house) => house['owner'] == name && (house['type']=='Rent'||house['type']=='Both'),
    ).toList();

    if (userHouses.isNotEmpty) {
      setState(() {
        rental = userHouses;
      });
    } else {
      rental = [];
    }
  } else {
    print('Failed to retrieve data');
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text('Failed to retrieve data. Please try again.'),
        backgroundColor: Colors.red,
      ),
    );
  }
} catch (e) {
  print('Error: $e');

}
}

Future<void> getlease_hData() async {
  final url = Uri.parse('http://'+api+':3000/gethousedata');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      final List<dynamic> houseData = jsonDecode(response.body);

      final List<dynamic> userHouses = houseData.where(
          (house) => house['owner'] == name && (house['type']=='Lease'||house['type']=='Both'),
      ).toList();

      if (userHouses.isNotEmpty) {
        setState(() {
          lease_h = userHouses;
        });
      } else {
        lease_h = [];
      }
    } else {
      print('Failed to retrieve data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Failed to retrieve data. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');

   }
  }
}
```

Code For Favourites Page:

```
class Body extends StatefulWidget {
  const Body({
    Key? key,
  }) : super(key: key);

  @override
  State<Body> createState() => _BodyState();
}

class _BodyState extends State<Body> {
  List<dynamic> property=[];
  String name='';
  String address='';

  @override
  void initState() {
    super.initState();
    getfavData();
    gethouseData(name, address);
  }

  bool isListView = false; // Track the selected state of the toggle button

  @override
  Widget build(BuildContext context) {
    if(property.isEmpty){
      return Stack(
        children: [
          Container(

            height: 150,
            decoration: BoxDecoration(color: kPrimaryColor,),
            child: Padding(
              padding: const EdgeInsets.only(bottom: 40.0,right: 10,left: 10),
              child: Row(
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  Padding(
                    padding: const EdgeInsets.only(left: 14.0),
                    child: Text(
                      'Empty Properties',
                      textAlign: TextAlign.center,
                      style: TextStyle(
                        color: Colors.white,
                        fontSize: getProportionateScreenHeight(24),
                      ),),
          ),
        ],
      ),
    ),
  ),
  Padding(
    padding: const EdgeInsets.only(top: 116.372),
    child: Stack(
      children: [
        Container(
          width: SizeConfig.screenWidth,
          height: SizeConfig.screenHeight*0.798,
          decoration: BoxDecoration(
              borderRadius: BorderRadius.only(topLeft: Radius.circular(25),topRight: Radius.circular(25)),
              color: Color(0xFFF5F6F9)
          ),
          padding: EdgeInsets.only(left: 36,right: 24),
          child: Row(
            children: [
              SvgPicture.asset('assets/icons/addProperty.svg',height: 55,),
              SizedBox(width: getProportionateScreenWidth(14),),
              Text(
                'No favourite property',
                style: TextStyle(
                    fontWeight: FontWeight.bold,
                    fontSize: 24,
                    color: kblackcolor
                ),
              )
            ],
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(left: 320.0,right: 30,top: 625,bottom: 20),
          child: Container(
            height: 80.0,
            width: 80.0,
            child: FittedBox(
              child: FloatingActionButton(
                backgroundColor: kPrimaryColor,
                onPressed: () {},
                child: SvgPicture.asset('assets/icons/add.svg',color: Colors.white,height: 55,),
              ),
            ),
          ),
        )
      ],
    ),
  ),
 ],
);
}
return SafeArea(
  child: Padding(
    padding: const EdgeInsets.symmetric(horizontal: 20.0, vertical: 10),
  child: SingleChildScrollView(
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(height: getProportionateScreenHeight(5),),
        Row(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Padding(
              padding: const EdgeInsets.only(left: 4.0),
              child: Text(
                'Favourites',
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  fontSize: getProportionateScreenHeight(30),
                  color: kblackcolor
                ),
              ),
            ),
            Padding(
              padding: const EdgeInsets.only(right:7.0),
              child: list_gridToggle(onToggle: (bool value) {
                // Update the selected state of the toggle button
                setState(() {
                  isListView = !isListView;
                });
              }),
            ),
          ],
        ),
        SizedBox(height: getProportionateScreenHeight(15)),
        // Conditional rendering of the appropriate view based on selected state
        isListView
            ? SingleChildScrollView(
          child:Column(
            children: List.generate(
              property.length,
                  (index) => house_tab(property: property[index]),
            ),
          ),
        )
            : SingleChildScrollView(
          child: Column(
            children: List.generate(
              property.length,
                  (index) => house_list(property: property[index]),
            ),
          ),
        ),
      ],
    ),
  ),
 ),
);
```

```
}
Future<void> getfavData() async {
  final url = Uri.parse('http://'+api+':3000/getfav');

  try {
    final response = await http.get(url);

    if (response.statusCode == 200) {
      // Parse the JSON response
      final List<dynamic> favData = jsonDecode(response.body);

      // Filter fav data for the specific user
      final List<dynamic> userFavData = favData.where(
            (fav) => fav['user_name'] == 'tahir',
      ).toList();

      if (userFavData.isNotEmpty) {
        // Extract the name and address values from the fav data
        final List<String> names = [];
        final List<String> addresses = [];

        for (final fav in userFavData) {
          names.add(fav['name']);
          addresses.add(fav['address']);
        }

        // Use the names and addresses to retrieve additional data from the house table
        final List<dynamic> houseData = [];

        for (int i = 0; i < names.length; i++) {
          final user = await gethouseData(names[i], addresses[i]);
          if (user != null) {
            houseData.add(user);
          }
        }

        // Set the list of houses for the UI
        setState(() {
          property = houseData;
        });
      } else {
        property = []; // No favorite data found for the user
        // Handle the case where no favorite data is found
      }
    } else {
      print('Failed to retrieve data');
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('Failed to retrieve data. Please try again.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  } catch (e) {
    print('Error: $e');
```

```
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text('An error occurred. Please try again later.'),
          backgroundColor: Colors.red,
        ),
      );
    }
  }

  Future<dynamic> gethouseData(String name, String address) async {
    final url = Uri.parse('http://'+api+':3000/gethousedata');

    try {
      final response = await http.get(url);

      if (response.statusCode == 200) {
        // Parse the JSON response
        final List<dynamic> userData = jsonDecode(response.body);

        // Check if the provided name and address match any user record
        final user = userData.firstWhere(
              (user) => user['name'] == name && user['address'] == address,
          orElse: () => null,
        );

        return user;
      }
    } catch (e) {
      print('Error: $e');
    }

    return null; // Return null if no user data is found
  }



}
```

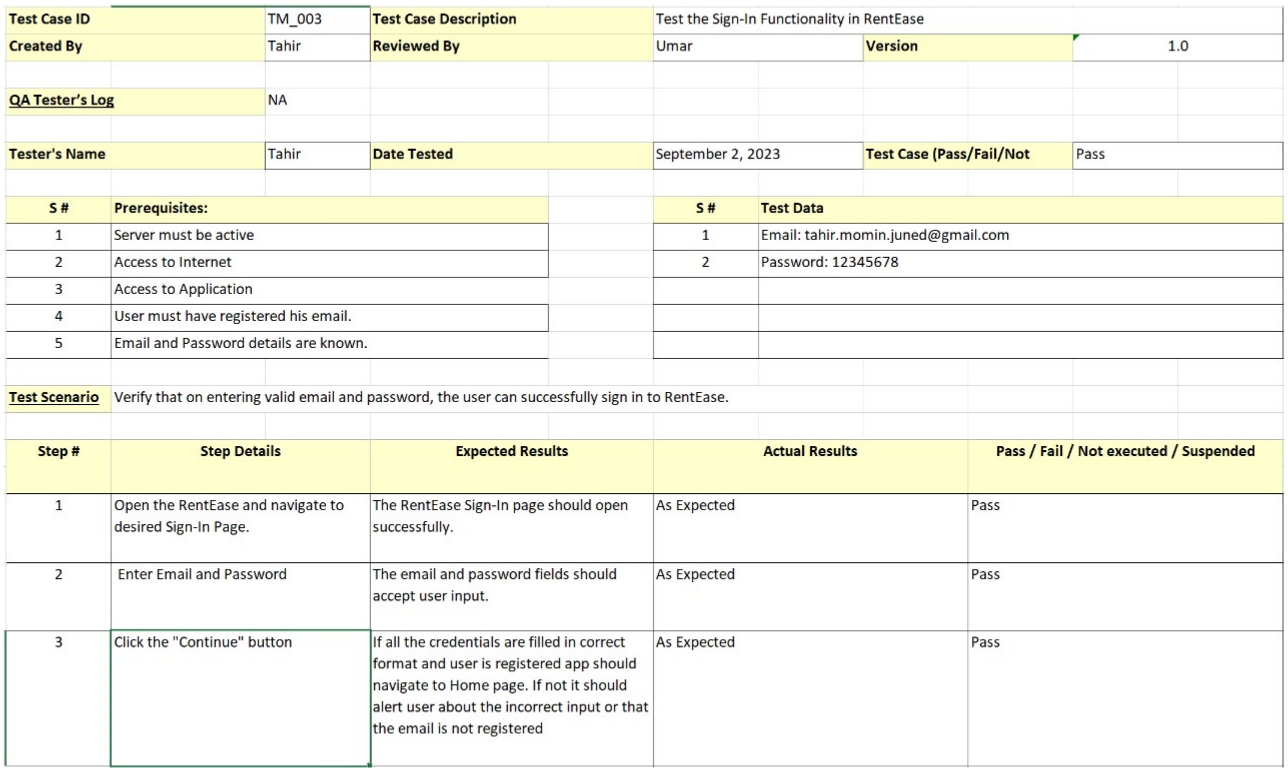
Code for Posting Property:

```
List<String> houseType=['House','Apartment','Villa'];

class FirstPage extends StatefulWidget {
  final String? email;
  final String? password;

  const FirstPage({
    Key? key,
    required this.email,
    required this.password,
  }) : super(key: key);

  @override
  State<FirstPage> createState() => _FirstPageState();
}
```

```
class _FirstPageState extends State<FirstPage> {


  String owner = "";
  String area ="";
  String rent ="";
  String security ="";
  String dropdownValue=houseType.first;
  String? selectedOption='Both';

  List<io.File> pickedImages = [];

  @override
  void initState() {
    super.initState();
    getownerData();
  }


  TextEditingController name_c = new TextEditingController();
  TextEditingController address_c = new TextEditingController();
  TextEditingController city_c = new TextEditingController();
  TextEditingController bedroom_c = new TextEditingController();
  TextEditingController bathroom_c = new TextEditingController();
  TextEditingController area_c = new TextEditingController();
  TextEditingController rent_c = new TextEditingController();
  TextEditingController description_c = new TextEditingController();
  TextEditingController security_c = new TextEditingController();
  TextEditingController guest_c = new TextEditingController();
  TextEditingController state_c = new TextEditingController();
  TextEditingController year = new TextEditingController();


  Future<void> pickImages() async {
    if (pickedImages.isNotEmpty) {
      // If 10 images have already been picked, show a message or handle accordingly
      removeError(error: kimageNullError);
      if (pickedImages.length > 10) {
        pickedImages = pickedImages.sublist(0, 10);
        removeError(error: kimagelongError);

      }
      else{
        addError(error: kimagelongError);
      }
    }
    else{
      addError(error: kimageNullError);
    }

    List<XFile>? picked = await ImagePicker().pickMultiImage(
      maxWidth: 800, // Adjust the image quality and size as needed
      imageQuality: 80,
    );
    if (picked != null) {
      setState(() {
        pickedImages.addAll(picked.map((XFile xFile) => io.File(xFile.path)));
        // Limit the number of images to 10
      });
    }
  }

  final List<String?> errors = [];
  final _formKey = GlobalKey<FormState>();


  void addError({String? error}) {
    if (!errors.contains(error))
      setState(() {
        errors.add(error);
      });
  }

  void removeError({String? error}) {
    if (errors.contains(error))
      setState(() {
        errors.remove(error);
      });
  }

  @override
  Widget build(BuildContext context) {
    return SingleChildScrollView(
      child: Form(
        key: _formKey,
        child: Column(
          children: [
            Container(
              height: SizeConfig.screenHeight*0.535,
              child: Padding(
                padding: const EdgeInsets.symmetric(horizontal: 15.0),
                child: SingleChildScrollView(
                  scrollDirection: Axis.vertical,
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      SizedBox(height: getProportionateScreenHeight(5),),
                      TextFormField(
                        textCapitalization: TextCapitalization.characters,
                        onChanged: (value) {
                          if (value.isNotEmpty) {
                            removeError(error: khNamelNullError);
                          }
                          return null;
                        },
                        validator: (value) {
                          if (value!.isEmpty) {
                            addError(error: khNamelNullError);
        return "";
      }
      return null;
    },
    controller: name_c,
    decoration: InputDecoration(
        contentPadding: EdgeInsets.symmetric(horizontal: 15),
        label: Text(
          'Property Name',
          style: TextStyle(
              fontSize: getProportionateScreenHeight(18),
              color: kblackcolor
          ),
        ),
        floatingLabelBehavior: FloatingLabelBehavior.always,
        hintText: 'Enter Property Name',
        hintStyle: TextStyle(
            color: kTextColor,
            fontSize: getProportionateScreenHeight(16)
        )
    ),

    style: TextStyle(
        color: kblackcolor,
        fontSize: getProportionateScreenHeight(16)
    ),

  ),
  SizedBox(height: getProportionateScreenHeight(25),),
  TextFormField(
    textCapitalization: TextCapitalization.characters,
    onChanged: (value) {
      if (value.isNotEmpty) {
        removeError(error: khaddresslNullError);
      }
      return null;
    },
    validator: (value) {
      if (value!.isEmpty) {
        addError(error: khaddresslNullError);
        return "";
      }
      return null;
    },
    controller: address_c,
    style: TextStyle(
        color: kblackcolor,
        fontSize: getProportionateScreenHeight(16)
    ),
    decoration: InputDecoration(
        contentPadding: EdgeInsets.symmetric(horizontal: 15),
        label: Text(
          'Address',
          style: TextStyle(
              fontSize: getProportionateScreenHeight(18),
              color: kblackcolor
            ),
          ),
          floatingLabelBehavior: FloatingLabelBehavior.always,
          hintText: 'Enter Your Address',
          hintStyle: TextStyle(
            color: kTextColor,
            fontSize: getProportionateScreenHeight(16)
          )
      ),
    ),
    SizedBox(height: getProportionateScreenHeight(25),),
    Row(
      children: [
        Container(
          width: getProportionateScreenWidth(167),
          child: TextFormField(
            textCapitalization: TextCapitalization.characters,
            onChanged: (value) {
              if (value.isNotEmpty) {
                removeError(error: kdecityNullError);
              }
              return null;
            },
            validator: (value) {
              if (value!.isEmpty) {
                addError(error: kdecityNullError);
                return "";
              }
              return null;
            },
            controller: city_c,
            style: TextStyle(
                color: kblackcolor,
                fontSize: getProportionateScreenHeight(16)
            ),
            decoration: InputDecoration(
                contentPadding: EdgeInsets.symmetric(horizontal: 15),
                label: Text(
                  'City',
                  style: TextStyle(
                      fontSize: getProportionateScreenHeight(18),
                      color: kblackcolor
                  ),
                ),
                floatingLabelBehavior: FloatingLabelBehavior.always,
                hintText: 'Enter Your City',
                hintStyle: TextStyle(
                    color: kTextColor,
                    fontSize: getProportionateScreenHeight(16)
                )
            ),
          ),
        ),
        SizedBox(width: getProportionateScreenWidth(10),),
        Container(
          width: getProportionateScreenWidth(167),
          child: TextFormField(
            textCapitalization: TextCapitalization.characters,
            onChanged: (value) {
              if (value.isNotEmpty) {
                removeError(error: kstateNullError);
              }
              return null;
            },
            validator: (value) {
              if (value!.isEmpty) {
                addError(error: kstateNullError);
                return "";
              }
              return null;
            },
            controller: state_c,
            style: TextStyle(
                color: kblackcolor,
                fontSize: getProportionateScreenHeight(16)
            ),
            decoration: InputDecoration(
                contentPadding: EdgeInsets.symmetric(horizontal: 15),
                label: Text(
                  'State',
                  style: TextStyle(
                      fontSize: getProportionateScreenHeight(18),
                      color: kblackcolor
                  ),
                ),
                floatingLabelBehavior: FloatingLabelBehavior.always,
                hintText: 'Enter Your State',
                hintStyle: TextStyle(
                    color: kTextColor,
                    fontSize: getProportionateScreenHeight(16)
                )
            ),
          ),
        ),
      ],
     ),
     SizedBox(height: getProportionateScreenHeight(10),),
     Padding(
      padding: const EdgeInsets.only(left: 10.0,),
      child: Text(
        'Select Rent Type',
        style: TextStyle(
          color: kblackcolor,
          fontSize: getProportionateScreenHeight(18),
        ),
      ),
     ),
     Padding(
      padding: const EdgeInsets.only(left: 5.0,bottom: 5),
      child: Row(
        children: [
          Radio<String>(
            value: 'Rent',
            groupValue: selectedOption,
            onChanged: (value) {
              setState(() {
                selectedOption = value!;
              });
            },
          ),
          Text(
            'Rent',
            style: TextStyle(
              fontSize: getProportionateScreenHeight(16),
              color: kblackcolor,
            ),
          ),
          SizedBox(width: getProportionateScreenWidth(10),),
          Radio<String>(
            value: 'Lease',
            groupValue: selectedOption,
            onChanged: (value) {
              setState(() {
                selectedOption = value!;
              });
            },
          ),
          Text(
            'Lease',
            style: TextStyle(
              fontSize: getProportionateScreenHeight(16),
              color: kblackcolor,
            ),
          ),
          Radio<String>(
            value: 'Both',
            groupValue: selectedOption,
            onChanged: (value) {
              setState(() {
                selectedOption = value!;
              });
            },
          ),
          Text(
            'Both',
            style: TextStyle(
              fontSize: getProportionateScreenHeight(16),
              color: kblackcolor,
            ),
          ),

        ],
      ),
     ),
     SizedBox(height: getProportionateScreenHeight(25),),
     Row(
      children: [
        Container(
          width: getProportionateScreenWidth(167),
          child: TextFormField(
            textCapitalization: TextCapitalization.characters,
            onChanged: (value) {
              if (value.isNotEmpty) {
                removeError(error: kbuilt_yearNullError);
              }
              return null;
            },
            validator: (value) {
              if (value!.isEmpty) {
                addError(error: kbuilt_yearNullError);
                return "";
              }
              return null;
            },
            controller: year,
            style: TextStyle(
                color: kblackcolor,
                fontSize: getProportionateScreenHeight(16)
            ),
            keyboardType: TextInputType.number,
            decoration: InputDecoration(
                contentPadding: EdgeInsets.symmetric(horizontal: 15),
                label: Text(
                  'Built Year',
                  style: TextStyle(
                      fontSize: getProportionateScreenHeight(18),
                      color: kblackcolor
                  ),
                ),
                floatingLabelBehavior: FloatingLabelBehavior.always,
                hintText: 'Enter Built Year',
                hintStyle: TextStyle(
                    color: kTextColor,
                    fontSize: getProportionateScreenHeight(16)
                )
            ),
          ),
        ),
        SizedBox(width: getProportionateScreenWidth(10),),
        Container(
          width: getProportionateScreenWidth(167),
          child: TextFormField(
            textCapitalization: TextCapitalization.characters,
            onChanged: (value) {
              if (value.isNotEmpty) {
                removeError(error: kguestNullError);
              }
              return null;
            },
            validator: (value) {
              if (value!.isEmpty) {
                addError(error: kguestNullError);
                return "";
              }
              return null;
            },
            controller: guest_c,
            style: TextStyle(
                color: kblackcolor,
                fontSize: getProportionateScreenHeight(16)
            ),
            keyboardType: TextInputType.number,
            decoration: InputDecoration(
                contentPadding: EdgeInsets.symmetric(horizontal: 15),
                label: Text(
                  'No. of Guests',
                  style: TextStyle(
                      fontSize: getProportionateScreenHeight(18),
                      color: kblackcolor
                  ),
                ),
                floatingLabelBehavior: FloatingLabelBehavior.always,
                hintText: 'Allowed Guests',
                hintStyle: TextStyle(
                    color: kTextColor,
                    fontSize: getProportionateScreenHeight(16)
                )
            ),
          ),
        ),
      ],
    ),
    SizedBox(height: getProportionateScreenHeight(25),),
    DropdownButtonFormField<String>(
      decoration: InputDecoration(
        contentPadding: EdgeInsets.symmetric(horizontal: 15),
        label: Text(
          'Property Type',
          style: TextStyle(
              fontSize: getProportionateScreenHeight(18),
              color: kblackcolor
          ),
        ),
        floatingLabelBehavior: FloatingLabelBehavior.always,
      ),
      value: dropdownValue,
      icon: const Icon(Icons.arrow_drop_down_rounded,),
      elevation: 16,
      style: TextStyle(
          color: kblackcolor,
          fontSize: getProportionateScreenHeight(16)
      ),
      onChanged: (String? value) {
```

```
    setState(() {
      dropdownValue = value!;
    });
  },
  items: houseType.map<DropdownMenuItem<String>>((String value) {
    return DropdownMenuItem<String>(
      value: value,
      child: Text(value),
    );
  }).toList(),
),
SizedBox(height: getProportionateScreenHeight(25),),
Row(
  children: [
    Container(
      width: getProportionateScreenWidth(167),
      child: TextFormField(
        textCapitalization: TextCapitalization.characters,
        onChanged: (value) {
          if (value.isNotEmpty) {
            removeError(error: kbedroomlNullError);
          }
          return null;
        },
        validator: (value) {
          if (value!.isEmpty) {
            addError(error: kbedroomlNullError);
            return "";
          }
          return null;
        },
        controller: bedroom_c,
        keyboardType: TextInputType.number,
        style: TextStyle(
            color: kblackcolor,
            fontSize: getProportionateScreenHeight(16)
        ),
        decoration: InputDecoration(
            contentPadding: EdgeInsets.symmetric(horizontal: 15),
            label: Text(
              'Bedrooms',
              style: TextStyle(
                  fontSize: getProportionateScreenHeight(18),
                  color: kblackcolor
              ),
            ),
            floatingLabelBehavior: FloatingLabelBehavior.always,
            hintText: 'No. of Bedrooms',
            hintStyle: TextStyle(
                color: kTextColor,
                fontSize: getProportionateScreenHeight(16)
            )
        ),
      ),
    ),
    SizedBox(width: getProportionateScreenWidth(10),),
    Container(
      width: getProportionateScreenWidth(167),
      child: TextFormField(
        textCapitalization: TextCapitalization.characters,
        onChanged: (value) {
          if (value.isNotEmpty) {
            removeError(error: kbathroomNullError);
          }
          return null;
        },
        validator: (value) {
          if (value!.isEmpty) {
            addError(error: kbathroomNullError);
            return "";
          }
          return null;
        },
        controller: bathroom_c,
        style: TextStyle(
            color: kblackcolor,
            fontSize: getProportionateScreenHeight(16)
        ),
        keyboardType: TextInputType.number,
        decoration: InputDecoration(
            contentPadding: EdgeInsets.symmetric(horizontal: 15),
            label: Text(
              'Bathrooms',
              style: TextStyle(
                  fontSize: getProportionateScreenHeight(18),
                  color: kblackcolor
              ),
            ),
            floatingLabelBehavior: FloatingLabelBehavior.always,
            hintText: 'No. of Bathrooms',
            hintStyle: TextStyle(
                color: kTextColor,
                fontSize: getProportionateScreenHeight(16)
            )
        ),
      ),
    ),
  ],
 ),
 SizedBox(height: getProportionateScreenHeight(25),),
 Row(
  children: [
    Container(
      width: getProportionateScreenWidth(167),
      child: TextFormField(
        textCapitalization: TextCapitalization.characters,
        onChanged: (value) {
          if (value.isNotEmpty) {
            setState(() {
              area = value;
            });
            removeError(error: kareaNullError);
          }
          return null;
        },
        validator: (value) {
          if (value!.isEmpty) {
            addError(error: kareaNullError);
            return "";
          }
          return null;
        },
        controller: area_c,
        keyboardType: TextInputType.number,
        style: TextStyle(
            color: kblackcolor,
            fontSize: getProportionateScreenHeight(16)
        ),
        decoration: InputDecoration(
            suffixText: area.isNotEmpty?'sq/ft':'',
            suffixStyle: TextStyle(
              color: kblackcolor,
              fontSize: getProportionateScreenHeight(16),
            ),
            contentPadding: EdgeInsets.symmetric(horizontal: 15),
            label: Text(
              'Area/Size',
              style: TextStyle(
                  fontSize: getProportionateScreenHeight(18),
                  color: kblackcolor
              ),
            ),
            floatingLabelBehavior: FloatingLabelBehavior.always,
            hintText: 'Enter Area/Size',
            hintStyle: TextStyle(
                color: kTextColor,
                fontSize: getProportionateScreenHeight(16)
            )
        ),
      ),
    ),
    SizedBox(width: getProportionateScreenWidth(10),),
    Container(
      width: getProportionateScreenWidth(167),
      child: TextFormField(
        textCapitalization: TextCapitalization.characters,
        onChanged: (value) {
          if (value.isNotEmpty) {
            setState(() {
              rent = value; // Update the stored value when the user enters input
            });
            removeError(error: krentNullError);
          }
          return null;
        },
        validator: (value) {
          if (value!.isEmpty) {
            addError(error: krentNullError);
            return "";
          }
          return null;
        },
        controller: rent_c,
        style: TextStyle(
            color: kblackcolor,
            fontSize: getProportionateScreenHeight(16)
        ),
        keyboardType: TextInputType.number,
        decoration: InputDecoration(
            contentPadding: EdgeInsets.symmetric(horizontal: 15,),
            label: Text(
              'Rent Amount',
              style: TextStyle(
                  fontSize: getProportionateScreenHeight(18),
                  color: kblackcolor
              ),
            ),
            prefixText: rent.isNotEmpty ? '₹':'',
            prefixIconColor: kblackcolor,
            floatingLabelBehavior: FloatingLabelBehavior.always,
            hintText: 'Enter Rent',

            hintStyle: TextStyle(
                color: kTextColor,
                fontSize: getProportionateScreenHeight(16)
            )
        ),
      ),
    ),
    ],
    ),
    SizedBox(height: getProportionateScreenHeight(25),),
    Container(
      width: getProportionateScreenWidth(167),
      child: TextFormField(
    textCapitalization: TextCapitalization.characters,
    onChanged: (value) {
      if (value.isNotEmpty) {
        removeError(error: ksecuritylNullError);
        setState(() {
          security = value; // Update the stored value when the user enters input
        });
      }
      return null;
    },
    validator: (value) {
      if (value!.isEmpty) {
        addError(error: ksecuritylNullError);
        return "";
      }
      return null;
    },
    controller: security_c,
    style: TextStyle(
      color: kblackcolor,
      fontSize: getProportionateScreenHeight(16)
    ),
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
      contentPadding: EdgeInsets.symmetric(horizontal: 15,),
      label: Text(
        'Security Amount',
        style: TextStyle(
          fontSize: getProportionateScreenHeight(18),
          color: kblackcolor
        ),
      ),
      prefixText: rent.isNotEmpty ? '₹':'',
      prefixIconColor: kblackcolor,
      floatingLabelBehavior: FloatingLabelBehavior.always,
      hintText: 'Enter Security',

      hintStyle: TextStyle(
        color: kTextColor,
        fontSize: getProportionateScreenHeight(16)
      )
    ),
  ),
),
SizedBox(height: getProportionateScreenHeight(25),),
TextFormField(
  textCapitalization: TextCapitalization.characters,
  controller: description_c,
  decoration: InputDecoration(
    contentPadding: EdgeInsets.symmetric(horizontal: 15,vertical: 10),
    label: Text(
      'Description (Optional)',
      style: TextStyle(
        fontSize: getProportionateScreenHeight(18),
        color: kblackcolor
      ),
    ),
    floatingLabelBehavior: FloatingLabelBehavior.always,
    hintText: 'Enter Description',
    hintStyle: TextStyle(
      color: kTextColor,
      fontSize: getProportionateScreenHeight(16)
    ),
    alignLabelWithHint: true
  ),
  style: TextStyle(
    color: kblackcolor,
    fontSize: getProportionateScreenHeight(16)
  ),
  maxLines: 5,
),
SizedBox(height: getProportionateScreenHeight(25),),
InkWell(
  onTap: () => pickImages(),
  child: Column(
    children: [
      Padding(
        padding: const EdgeInsets.only(left: 8.0),
        child: Text(
          'Add Image',
          style: TextStyle(
            fontSize: 16,
          ),
        ),
      ),
      SizedBox(height: 8),
      Visibility(
        visible: pickedImages.isEmpty,
        child: Container(
          height: 120,
          width: 120,
          decoration: BoxDecoration(
            border: Border.all(),
            borderRadius: BorderRadius.circular(5),
          ),
          padding: EdgeInsets.all(15),
          child: Icon(Icons.camera_alt_outlined, size: 50),
        ),
      ),
      Visibility(
        visible: pickedImages.isNotEmpty,
        child: SingleChildScrollView(
          scrollDirection: Axis.horizontal,
          child: Row(
            children: pickedImages.map((io.File image) {
              return Padding(
                padding: const EdgeInsets.only(left: 8.0),
                child: Container(
                  height: 120,
                  width: 120,
                  decoration: BoxDecoration(
                    border: Border.all(),
                    borderRadius: BorderRadius.circular(5),
                  ),
                  child: Image.file(
                    image,
                    fit: BoxFit.cover,
                  ),
                ),
              );
            }).toList(),
          ),
        ),
      ),
    ],
                    ),
                  ),
                  SizedBox(height: getProportionateScreenHeight(25),),
                ],
              ),
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.only(left:20, right: 20,bottom: 45),
          child: DefaultButton(
            text: 'Save',
            icon: 'signIn_forward',
            press: () {
              if (_formKey.currentState!.validate()) {
                _formKey.currentState!.save();
                setState(() {
                  insertData();
                  sendImagesToServer(
                      pickedImages, name_c.text, address_c.text);
                  Navigator.push(context, MaterialPageRoute(builder: (
                      context) =>
                      HomeScreen_l(
                          email: widget.email, password: widget.password)));
                });
              }
            },
          ),
        ),
      ],
    ),
  ),
 );
}
Future<void> insertData() async {
 print("hi");

 final url = Uri.parse('http://' + api + ':3000/inserthouse');

 try {
   final response = await http.post(
     url,
     headers: {
       'Content-Type': 'application/json',
     },
     body: jsonEncode({
       'owner': owner,
       'name': name_c.text,
       'address': address_c.text,
       'city': city_c.text,
       'built_type': dropdownValue,
       'bedroom': bedroom_c.text,
       'bathroom': bathroom_c.text,
       'area': area_c.text,
       'rent': rent_c.text,
       'state': state_c.text,
       'type':selectedOption,
       'description': description_c.text,
       'security': security_c.text,
       'built_year': year.text,
       'guest_limit': guest_c.text,
     }),
   );

   if (response.statusCode == 200) {
     print('Data inserted successfully');
     ScaffoldMessenger.of(context).showSnackBar(
       SnackBar(
        content: Text('Congratulations your property is listed'),
        backgroundColor: Colors.green,
       ),
     );
   } else {
     print('Failed to insert data');
     ScaffoldMessenger.of(context).showSnackBar(
       SnackBar(
        content: Text('Listing failed. Please try again.'),
        backgroundColor: Colors.red,
       ),
     );
   }
 } catch (e) {
   print('Error: $e');
   ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
       content: Text('An error occurred. Please try again later.'),
       backgroundColor: Colors.red,
     ),
   );
 }
}

Future<void> getownerData() async {
 final url = Uri.parse('http://'+api+':3000/getownerdata');

 try {
   final response = await http.get(url);

   if (response.statusCode == 200) {
     // Parse the JSON response
     final List<dynamic> userData = jsonDecode(response.body);

     // Check if the provided email and password match any user record
     final user = userData.firstWhere(
          (user) => user['email'] == widget.email && user['pass'] == widget.password,
       orElse: () => null,
     );

     if (user != null) {
       owner= user['fname']+" "+user['lname'];
       print(owner);
       // Credentials are correct, navigate to the next page (HomeScreen)
     } else {
       print('credentials are wrong');
       // Credentials are incorrect, show an error message
     }
   } else {
     print('Failed to retrieve data');

   }
 } catch (e) {
   print('Error: $e');
   ScaffoldMessenger.of(context).showSnackBar(
     SnackBar(
       content: Text('An error occurred. Please try again later.'),
       backgroundColor: Colors.red,
     ),
   );
 }
}

Future<void> sendImagesToServer(List<io.File?> images, String name, String address) async {
 try {
   final url = Uri.parse('http://' + api + ':3000/insertimage'); // Replace with your server URL
   final request = http.MultipartRequest('POST', url);
   request.fields['name'] = name;
   request.fields['address'] = address;

   for (int i = 0; i < images.length; i++) {
     io.File? image = images[i];

     if (image != null) {
       final imageStream = http.ByteStream(image.openRead());
       final length = await image.length();
       final multipartFile = http.MultipartFile(
        'image$i+1', // Change 'image$i' to your desired field name
        imageStream,
        length,
        filename: 'image$i.png', // Change to an appropriate file extension
       );
       request.files.add(multipartFile);
     }
   }

   final response = await request.send();

   if (response.statusCode == 200) {
     // Images uploaded successfully
     print('Images uploaded successfully');
   } else {
     // Handle the error
     print('Failed to upload images');
   }
 } catch (e) {
   print('Error: $e');
```

6. Test Cases

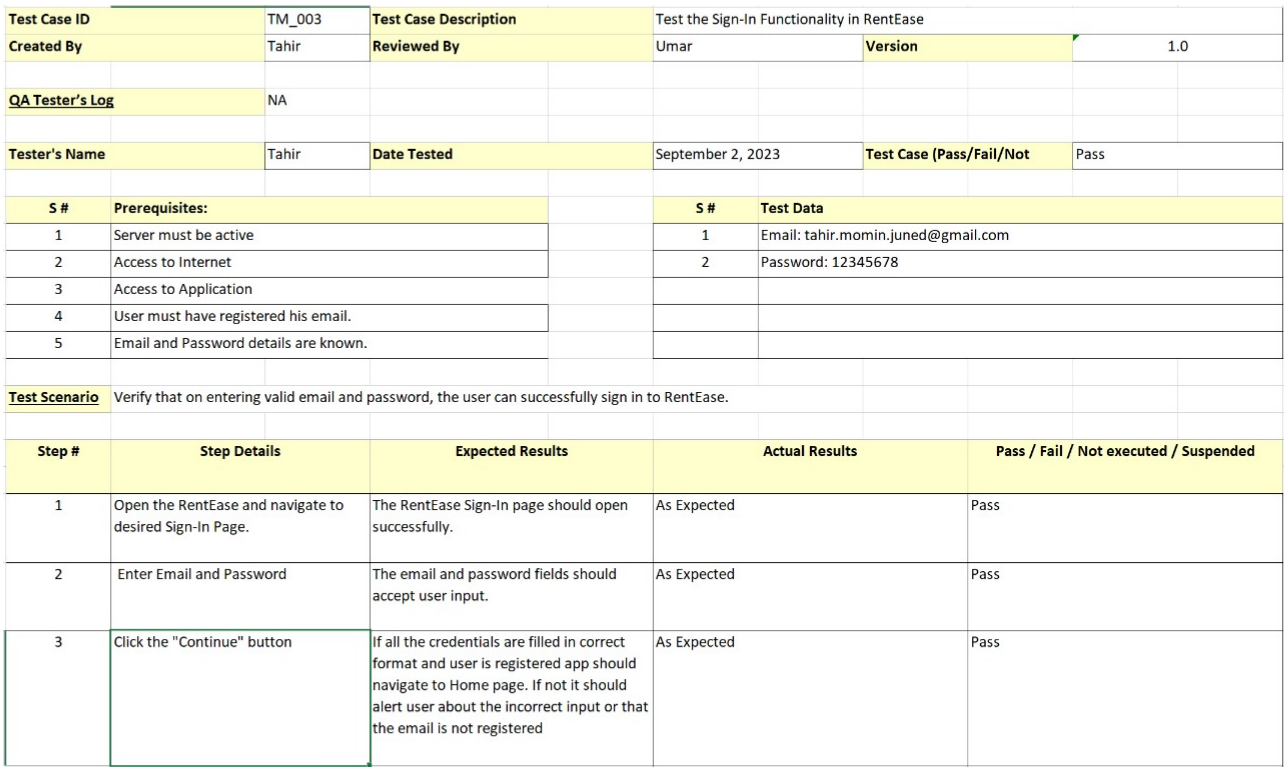


Test Case ID	TM_003	**Test Case Description**	Test the Sign-In Functionality in RentEase		
Created By	Tahir	**Reviewed By**	Umar	**Version**	1.0
QA Tester's Log	NA				
Tester's Name	Tahir	**Date Tested**	September 2, 2023	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Server must be active	1	Email: tahir.momin.juned@gmail.com
2	Access to Internet	2	Password: 12345678
3	Access to Application		
4	User must have registered his email.		
5	Email and Password details are known.		

Test Scenario Verify that on entering valid email and password, the user can successfully sign in to RentEase.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Open the RentEase and navigate to desired Sign-In Page.	The RentEase Sign-In page should open successfully.	As Expected	Pass
2	Enter Email and Password	The email and password fields should accept user input.	As Expected	Pass
3	Click the "Continue" button	If all the credentials are filled in correct format and user is registered app should navigate to Home page. If not it should alert user about the incorrect input or that the email is not registered	As Expected	Pass

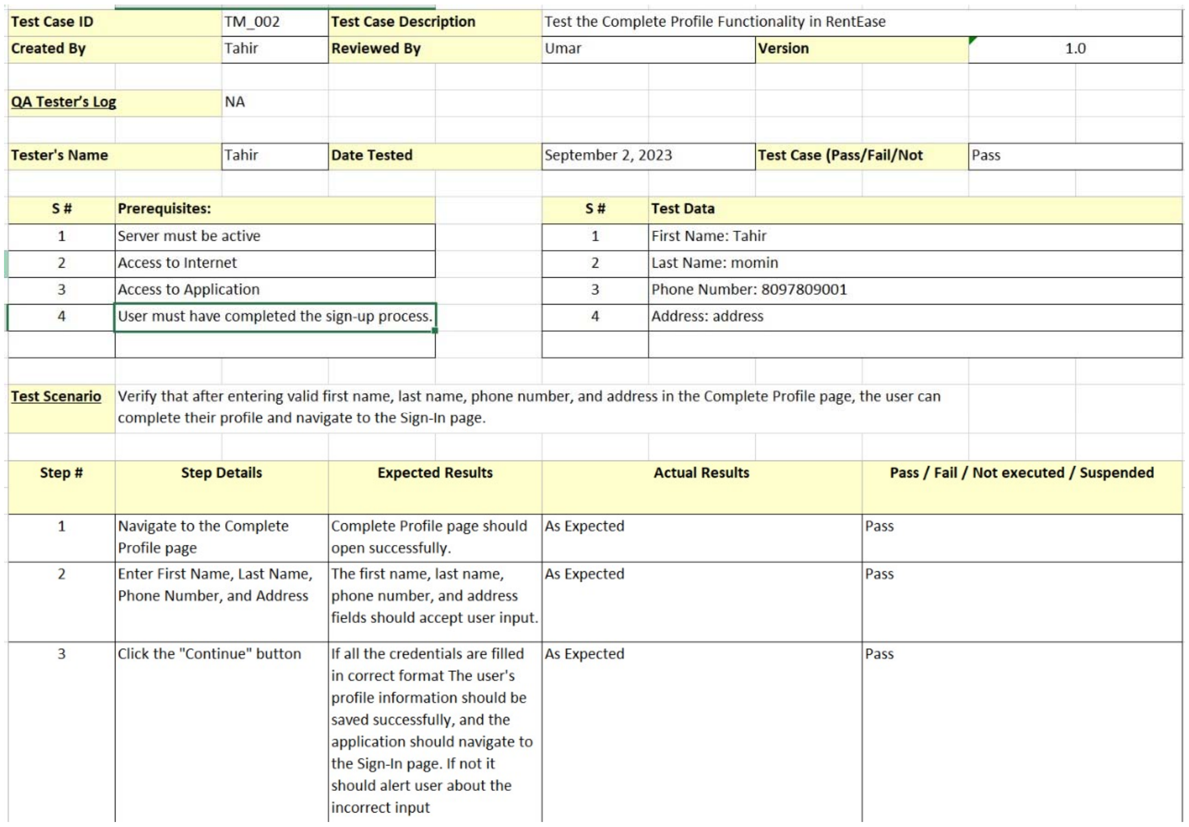


Test Case ID	TM_003	**Test Case Description**	Test the Sign-In Functionality in RentEase		
Created By	Tahir	**Reviewed By**	Umar	**Version**	1.0
QA Tester's Log	NA				
Tester's Name	Tahir	**Date Tested**	September 2, 2023	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Server must be active	1	Email: tahir.momin.juned@gmail.com
2	Access to Internet	2	Password: 12345678
3	Access to Application		
4	User must have registered his email.		
5	Email and Password details are known.		

Test Scenario Verify that on entering valid email and password, the user can successfully sign in to RentEase.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Open the RentEase and navigate to desired Sign-In Page.	The RentEase Sign-In page should open successfully.	As Expected	Pass
2	Enter Email and Password	The email and password fields should accept user input.	As Expected	Pass
3	Click the "Continue" button	If all the credentials are filled in correct format and user is registered app should navigate to Home page. If not it should alert user about the incorrect input or that the email is not registered	As Expected	Pass

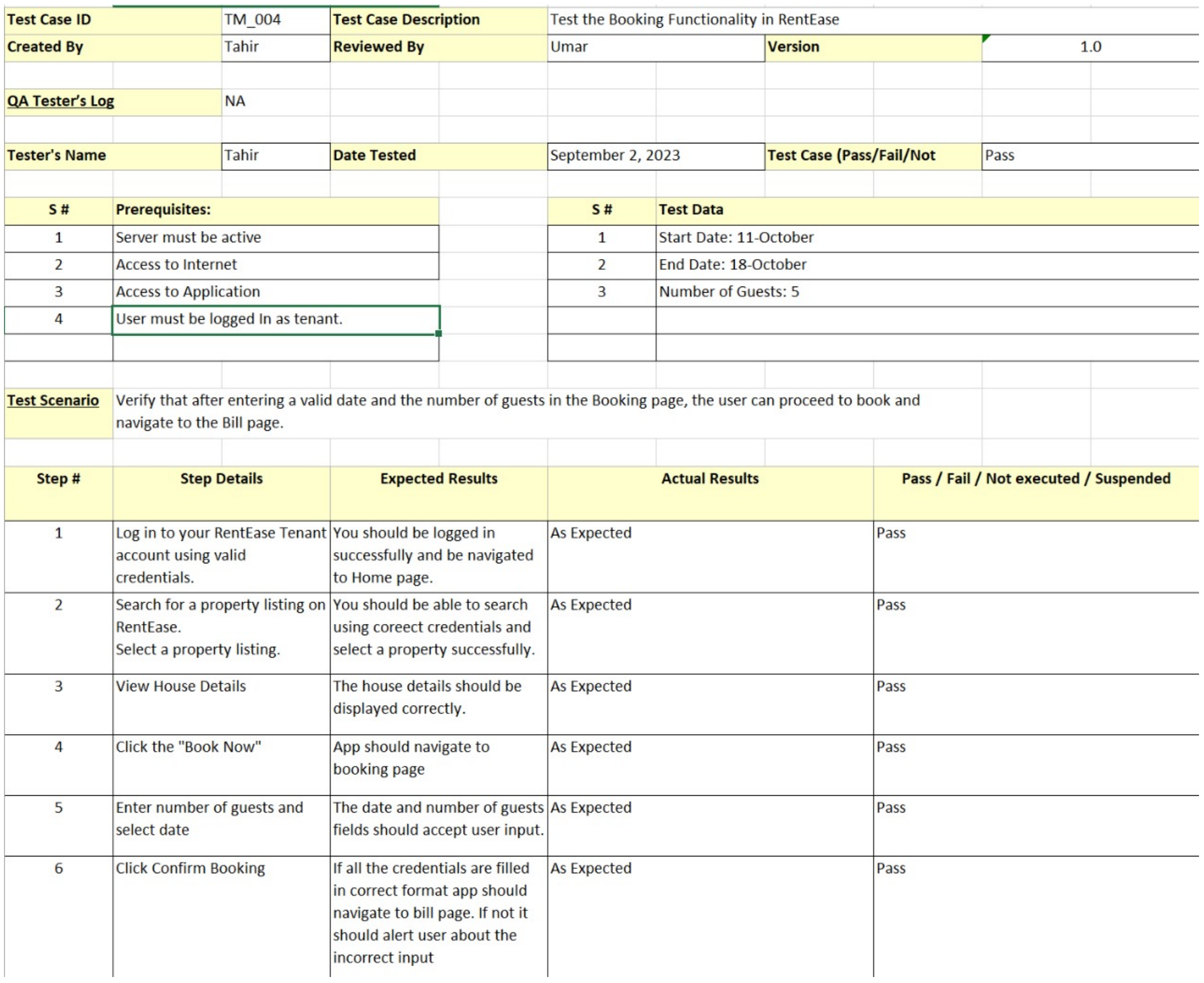


Test Case ID	TM_002	**Test Case Description**	Test the Complete Profile Functionality in RentEase		
Created By	Tahir	**Reviewed By**	Umar	**Version**	1.0
QA Tester's Log	NA				
Tester's Name	Tahir	**Date Tested**	September 2, 2023	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Server must be active	1	First Name: Tahir
2	Access to Internet	2	Last Name: momin
3	Access to Application	3	Phone Number: 8097809001
4	User must have completed the sign-up process.	4	Address: address

Test Scenario Verify that after entering valid first name, last name, phone number, and address in the Complete Profile page, the user can complete their profile and navigate to the Sign-In page.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Navigate to the Complete Profile page	Complete Profile page should open successfully.	As Expected	Pass
2	Enter First Name, Last Name, Phone Number, and Address	The first name, last name, phone number, and address fields should accept user input.	As Expected	Pass
3	Click the "Continue" button	If all the credentials are filled in correct format The user's profile information should be saved successfully, and the application should navigate to the Sign-In page. If not it should alert user about the incorrect input	As Expected	Pass

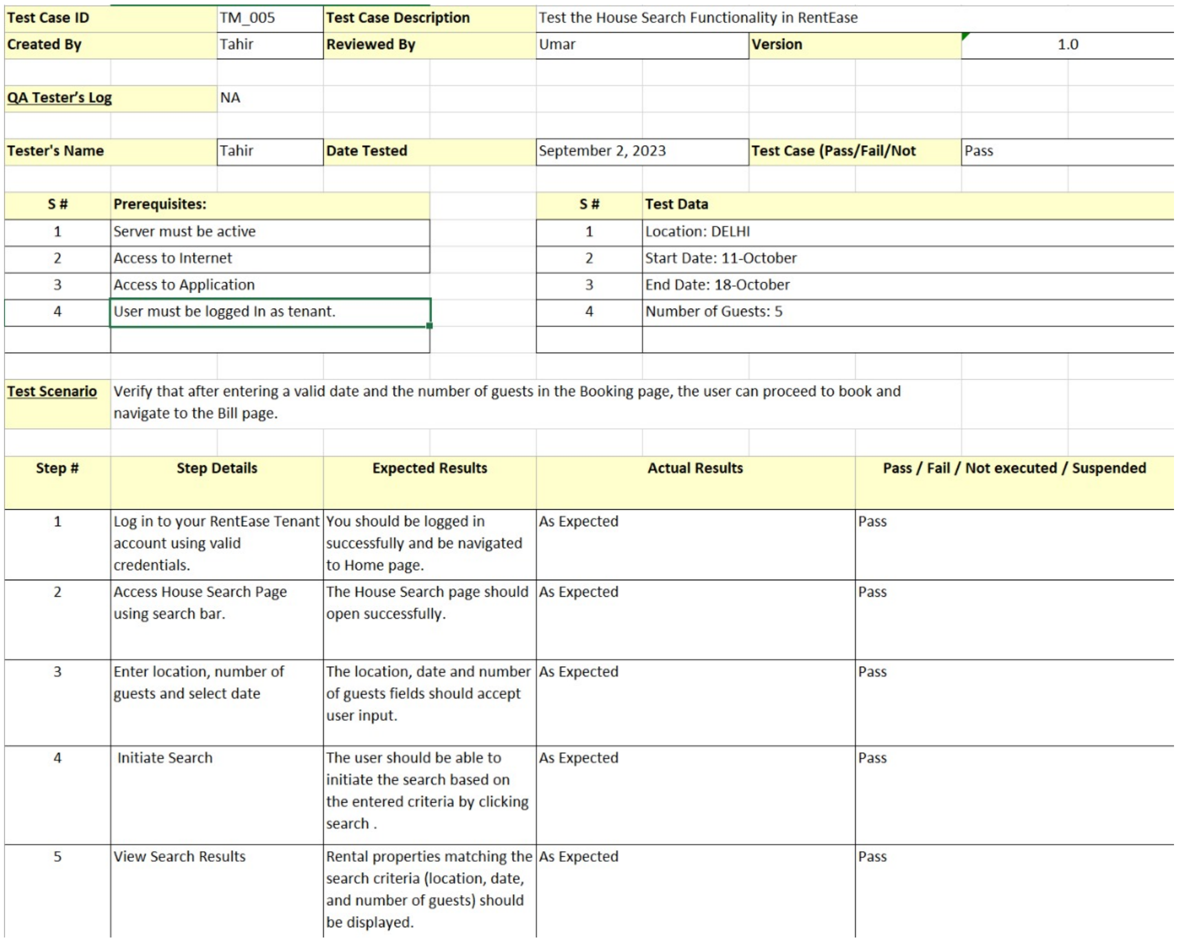


Test Case ID	TM_004	**Test Case Description**	Test the Booking Functionality in RentEase		
Created By	Tahir	**Reviewed By**	Umar	**Version**	1.0
QA Tester's Log	NA				
Tester's Name	Tahir	**Date Tested**	September 2, 2023	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Server must be active	1	Start Date: 11-October
2	Access to Internet	2	End Date: 18-October
3	Access to Application	3	Number of Guests: 5
4	User must be logged In as tenant.		

Test Scenario Verify that after entering a valid date and the number of guests in the Booking page, the user can proceed to book and navigate to the Bill page.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Log in to your RentEase Tenant account using valid credentials.	You should be logged in successfully and be navigated to Home page.	As Expected	Pass
2	Search for a property listing on RentEase. Select a property listing.	You should be able to search using coreect credentials and select a property successfully.	As Expected	Pass
3	View House Details	The house details should be displayed correctly.	As Expected	Pass
4	Click the "Book Now"	App should navigate to booking page	As Expected	Pass
5	Enter number of guests and select date	The date and number of guests fields should accept user input.	As Expected	Pass
6	Click Confirm Booking	If all the credentials are filled in correct format app should navigate to bill page. If not it should alert user about the incorrect input	As Expected	Pass

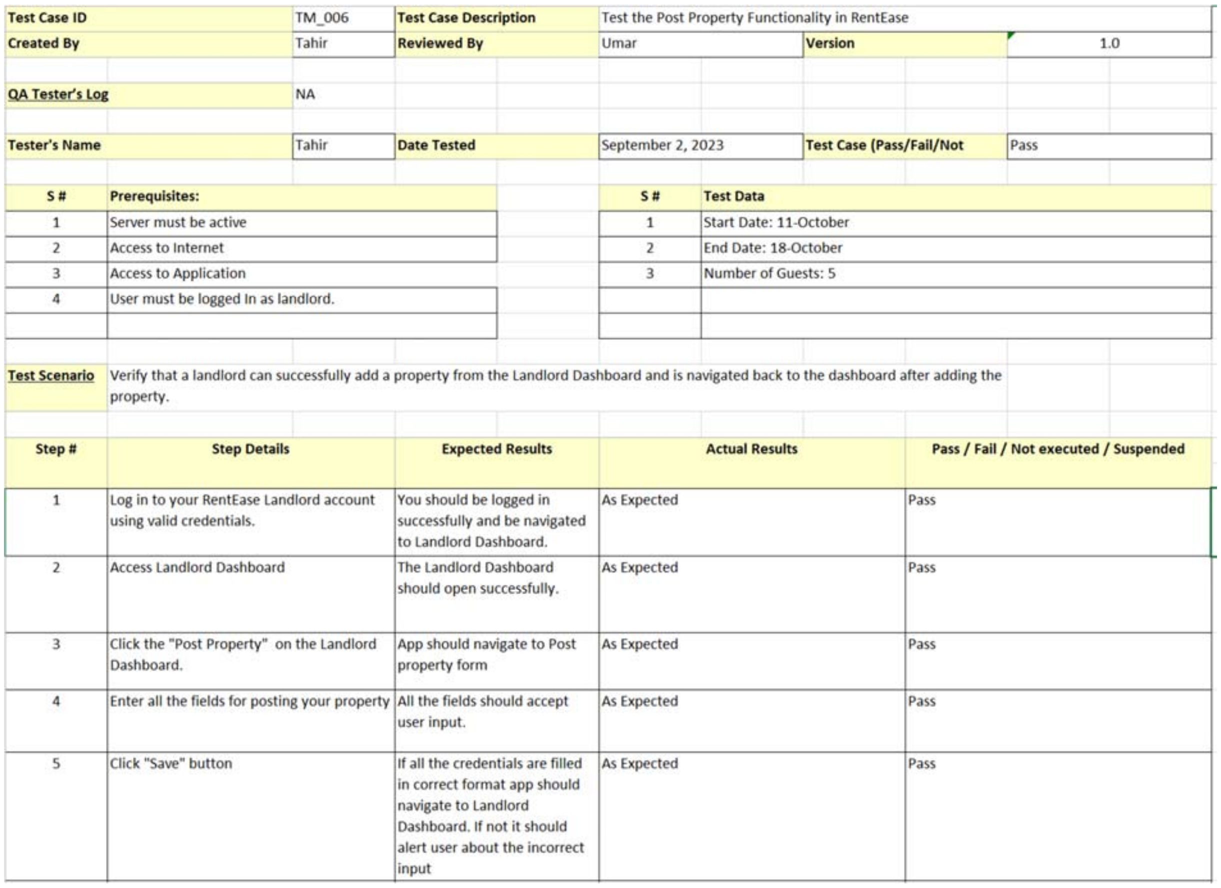


Test Case ID	TM_005	**Test Case Description**	Test the House Search Functionality in RentEase		
Created By	Tahir	**Reviewed By**	Umar	**Version**	1.0
QA Tester's Log	NA				
Tester's Name	Tahir	**Date Tested**	September 2, 2023	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Server must be active	1	Location: DELHI
2	Access to Internet	2	Start Date: 11-October
3	Access to Application	3	End Date: 18-October
4	User must be logged In as tenant.	4	Number of Guests: 5

Test Scenario Verify that after entering a valid date and the number of guests in the Booking page, the user can proceed to book and navigate to the Bill page.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Log in to your RentEase Tenant account using valid credentials.	You should be logged in successfully and be navigated to Home page.	As Expected	Pass
2	Access House Search Page using search bar.	The House Search page should open successfully.	As Expected	Pass
3	Enter location, number of guests and select date	The location, date and number of guests fields should accept user input.	As Expected	Pass
4	Initiate Search	The user should be able to initiate the search based on the entered criteria by clicking search .	As Expected	Pass
5	View Search Results	Rental properties matching the search criteria (location, date, and number of guests) should be displayed.	As Expected	Pass



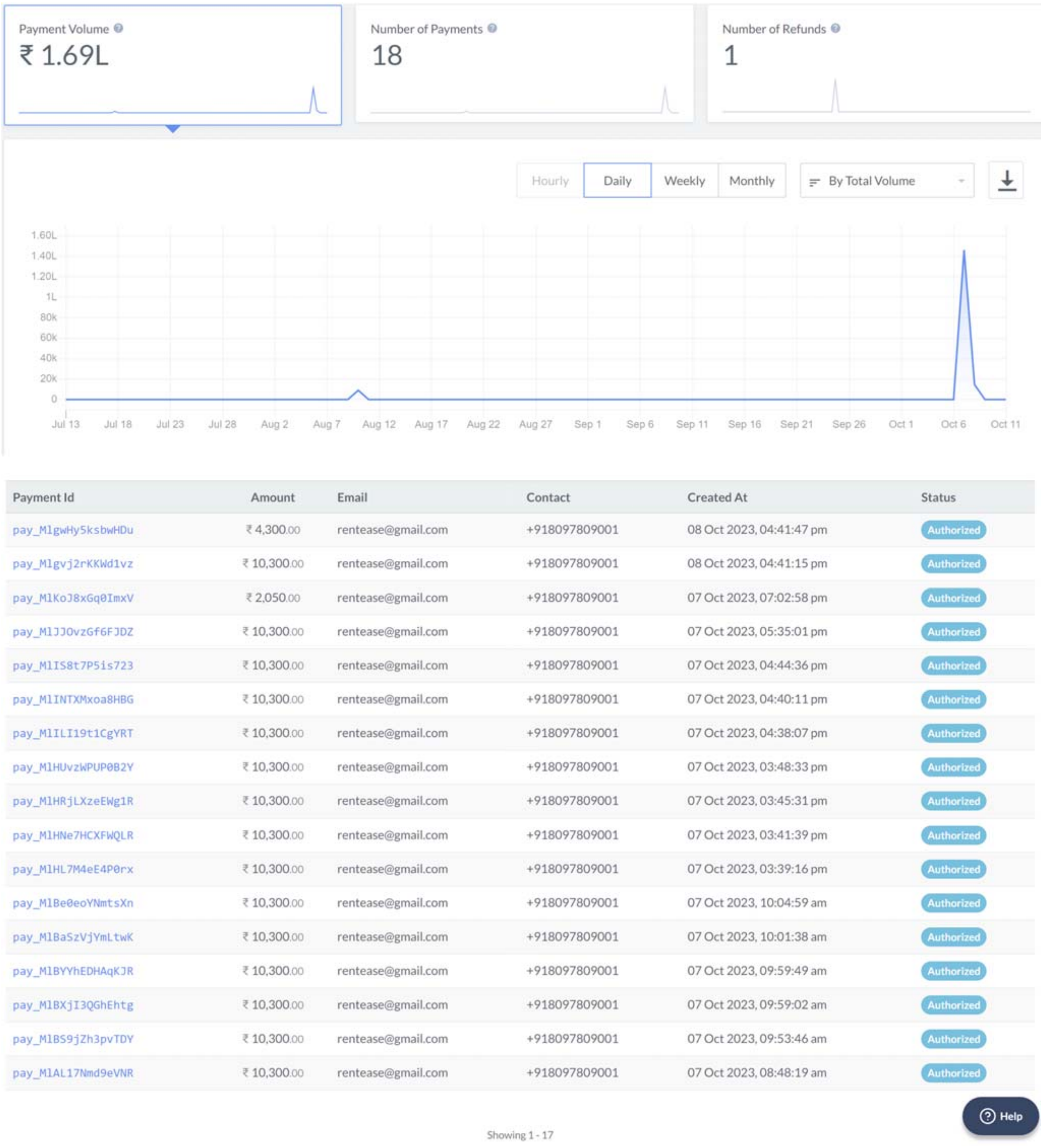
Test Case ID	TM_006	**Test Case Description**	Test the Post Property Functionality in RentEase		
Created By	Tahir	**Reviewed By**	Umar	**Version**	1.0
QA Tester's Log	NA				
Tester's Name	Tahir	**Date Tested**	September 2, 2023	**Test Case (Pass/Fail/Not**	Pass

S #	**Prerequisites:**	**S #**	**Test Data**
1	Server must be active	1	Start Date: 11-October
2	Access to Internet	2	End Date: 18-October
3	Access to Application	3	Number of Guests: 5
4	User must be logged In as landlord.		

Test Scenario Verify that a landlord can successfully add a property from the Landlord Dashboard and is navigated back to the dashboard after adding the property.

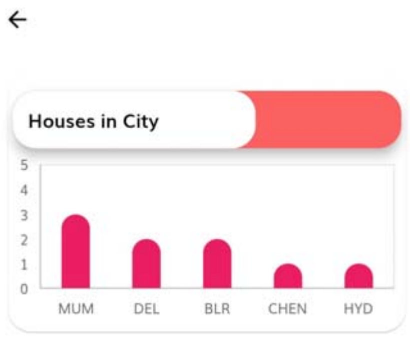
Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Log in to your RentEase Landlord account using valid credentials.	You should be logged in successfully and be navigated to Landlord Dashboard.	As Expected	Pass
2	Access Landlord Dashboard	The Landlord Dashboard should open successfully.	As Expected	Pass
3	Click the "Post Property" on the Landlord Dashboard.	App should navigate to Post property form	As Expected	Pass
4	Enter all the fields for posting your property	All the fields should accept user input.	As Expected	Pass
5	Click "Save" button	If all the credentials are filled in correct format app should navigate to Landlord Dashboard. If not it should alert user about the incorrect input	As Expected	Pass

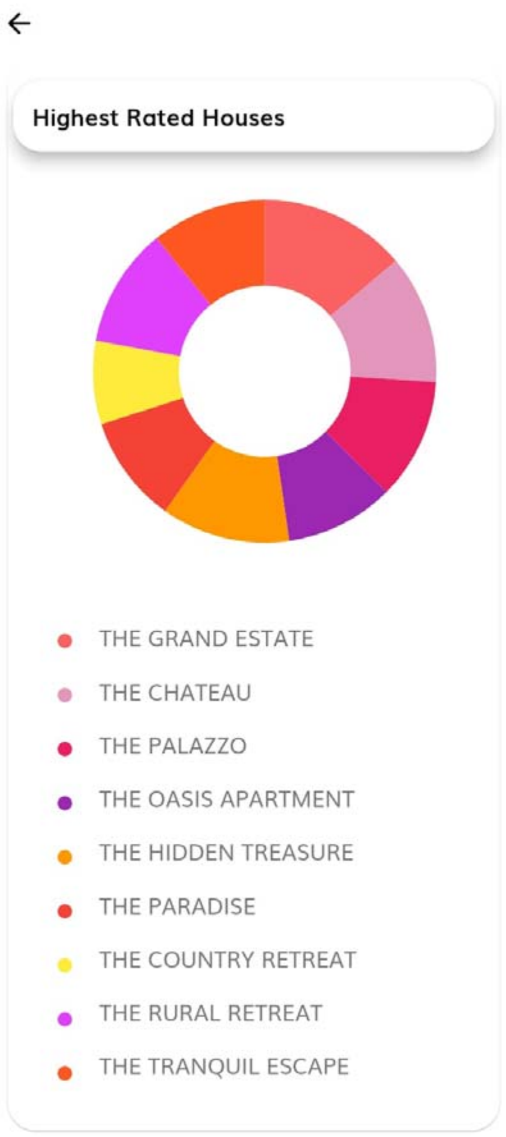
7. Reports



Payment Id	Amount	Email	Contact	Created At	Status
pay_MlgwHy5ksbwHDu	₹ 4,300.00	rentease@gmail.com	+918097809001	08 Oct 2023, 04:41:47 pm	Authorized
pay_Mlgvj2rKKWd1vz	₹ 10,300.00	rentease@gmail.com	+918097809001	08 Oct 2023, 04:41:15 pm	Authorized
pay_MlKoJ8xGq0ImxV	₹ 2,050.00	rentease@gmail.com	+918097809001	07 Oct 2023, 07:02:58 pm	Authorized
pay_MlJJOvzGf6FJDZ	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 05:35:01 pm	Authorized
pay_MlIS8t7P5is723	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 04:44:36 pm	Authorized
pay_MlINTXMxoa8HBG	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 04:40:11 pm	Authorized
pay_MlILI19t1CgYRT	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 04:38:07 pm	Authorized
pay_MlHUvzWPUP0B2Y	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 03:48:33 pm	Authorized
pay_MlHRjLXzeEWg1R	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 03:45:31 pm	Authorized
pay_MlHNe7HCXFWQLR	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 03:41:39 pm	Authorized
pay_MlHL7M4eE4P0rx	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 03:39:16 pm	Authorized
pay_MlBe0eoYNmtsXn	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 10:04:59 am	Authorized
pay_MlBaSzVjYmLtwK	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 10:01:38 am	Authorized
pay_MlBYYhEDHAqKJR	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 09:59:49 am	Authorized
pay_MlBXjI3QGhEhtg	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 09:59:02 am	Authorized
pay_MlBS9jZh3pvTDY	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 09:53:46 am	Authorized
pay_MlAL17Nmd9eVNR	₹ 10,300.00	rentease@gmail.com	+918097809001	07 Oct 2023, 08:48:19 am	Authorized

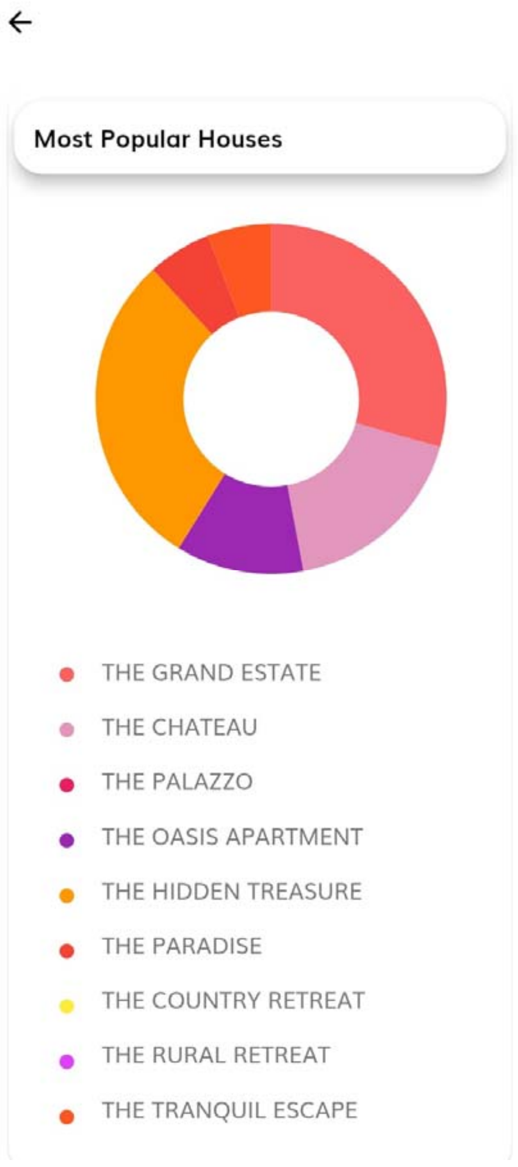
Showing 1 - 17





Highest Rated Houses

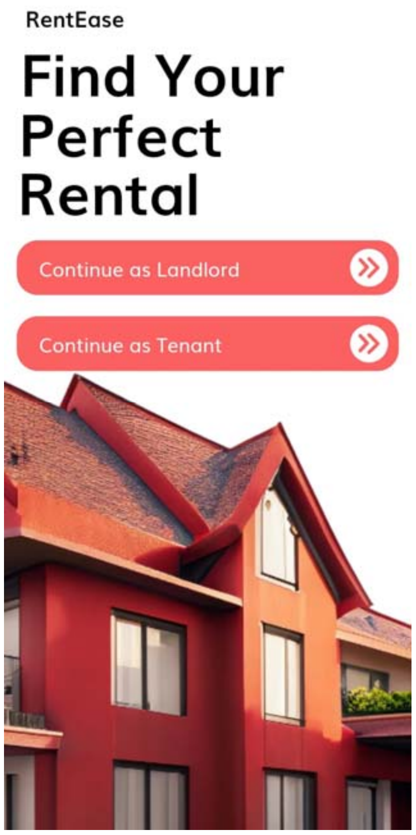
● THE GRAND ESTATE
● THE CHATEAU
● THE PALAZZO
● THE OASIS APARTMENT
● THE HIDDEN TREASURE
● THE PARADISE
● THE COUNTRY RETREAT
● THE RURAL RETREAT
● THE TRANQUIL ESCAPE



Most Popular Houses

● THE GRAND ESTATE
● THE CHATEAU
● THE PALAZZO
● THE OASIS APARTMENT
● THE HIDDEN TREASURE
● THE PARADISE
● THE COUNTRY RETREAT
● THE RURAL RETREAT
● THE TRANQUIL ESCAPE

8. UI Screen Shots

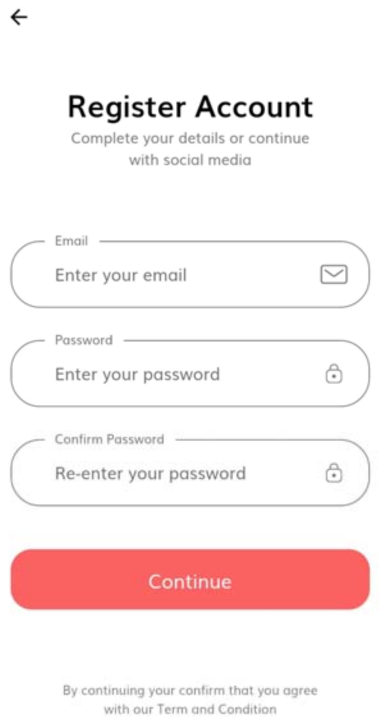


RentEase

Find Your Perfect Rental

Continue as Landlord

Continue as Tenant



Register Account

Complete your details or continue with social media

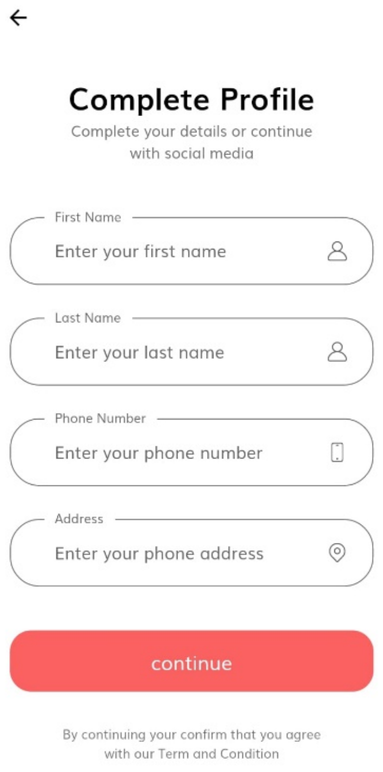
Email: Enter your email

Password: Enter your password

Confirm Password: Re-enter your password

Continue

By continuing your confirm that you agree with our Term and Condition



Complete Profile

Complete your details or continue with social media

First Name: Enter your first name

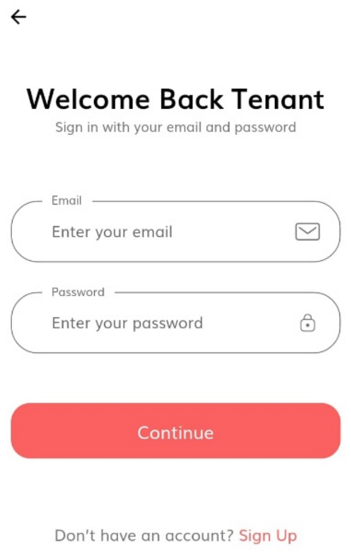
Last Name: Enter your last name

Phone Number: Enter your phone number

Address: Enter your phone address

continue

By continuing your confirm that you agree with our Term and Condition



Welcome Back Tenant

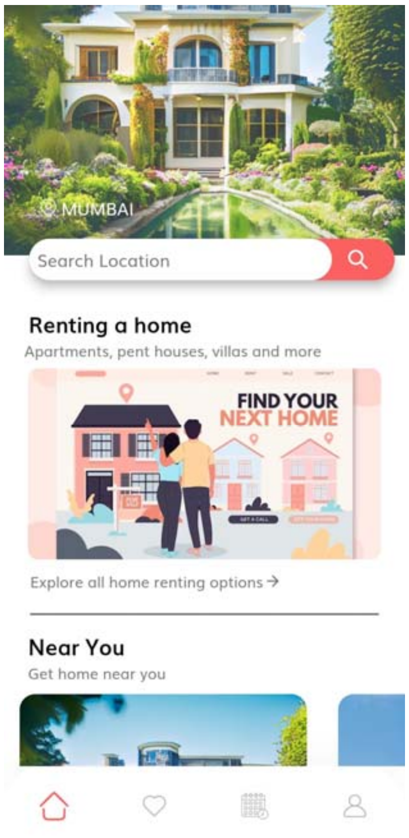
Sign in with your email and password

Email: Enter your email

Password: Enter your password

Continue

Don't have an account? Sign Up


MUMBAI

Search Location

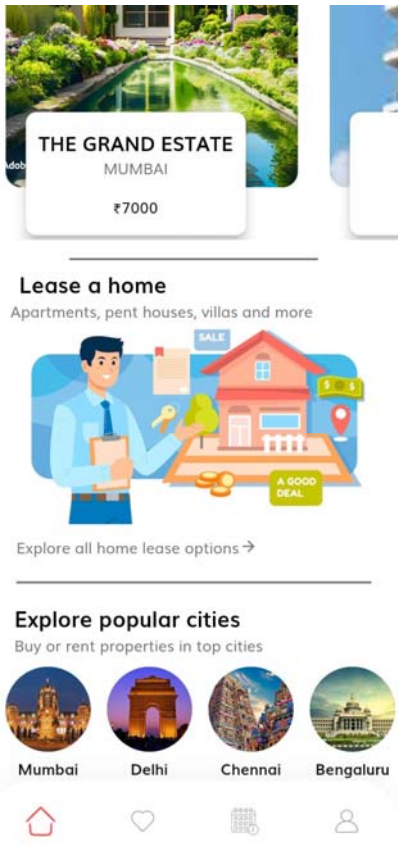
Renting a home

Apartments, pent houses, villas and more

Explore all home renting options →

Near You

Get home near you


THE GRAND ESTATE

MUMBAI

₹7000

Lease a home

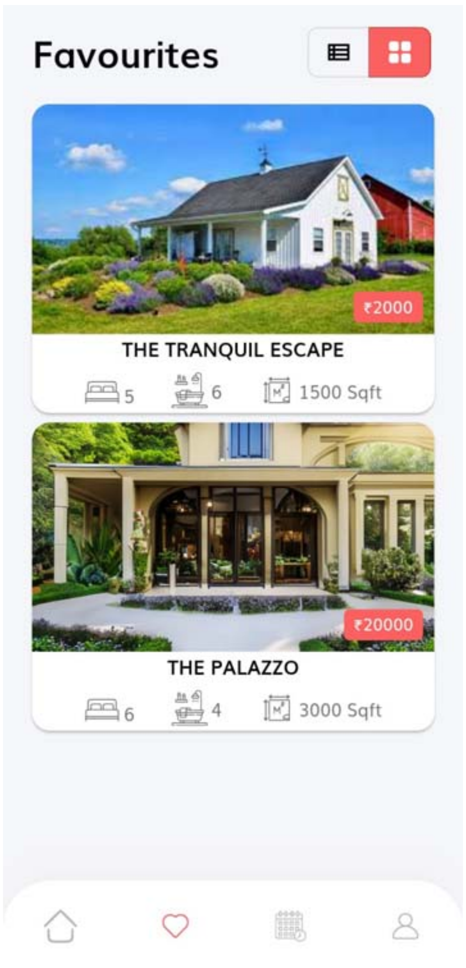
Apartments, pent houses, villas and more

Explore all home lease options →

Explore popular cities

Buy or rent properties in top cities

Mumbai Delhi Chennai Bengaluru


Favourites

₹2000

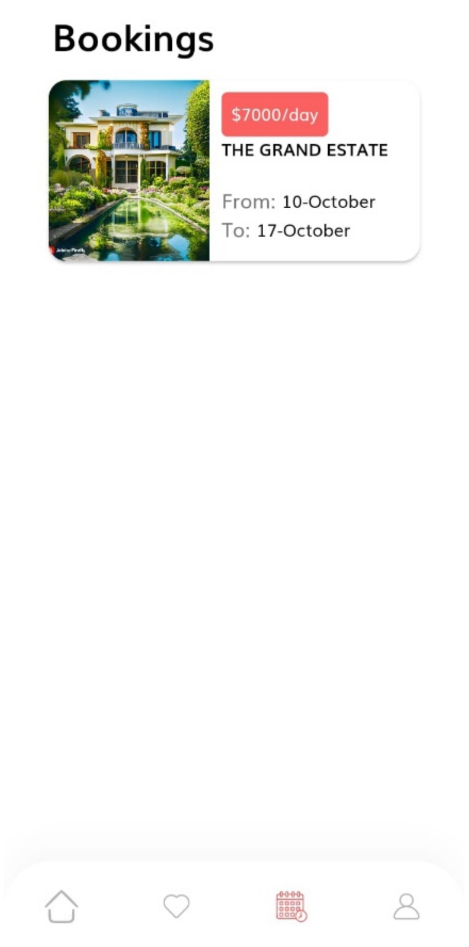
THE TRANQUIL ESCAPE

5 6 1500 Sqft

₹20000

THE PALAZZO

6 4 3000 Sqft

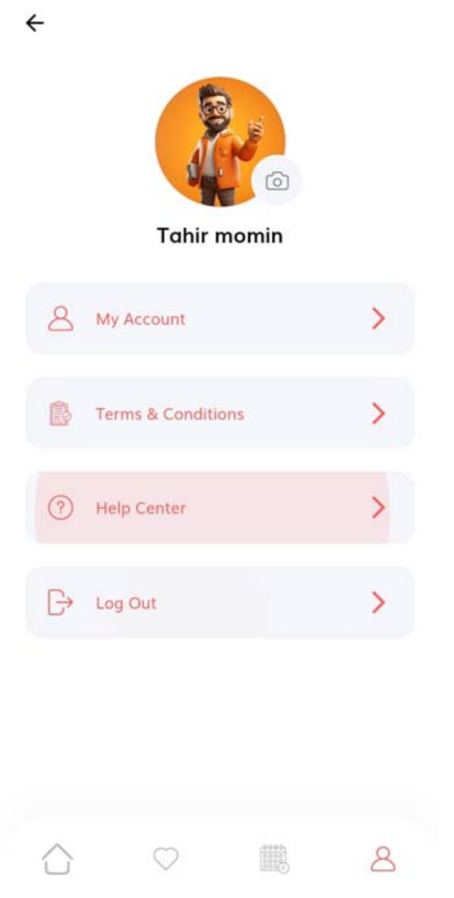

Bookings

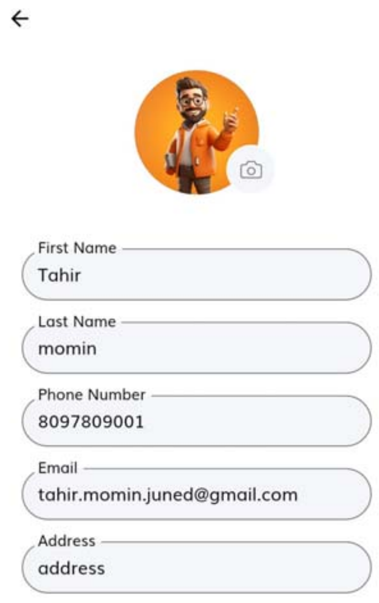
$7000/day

THE GRAND ESTATE

From: 10-October

To: 17-October



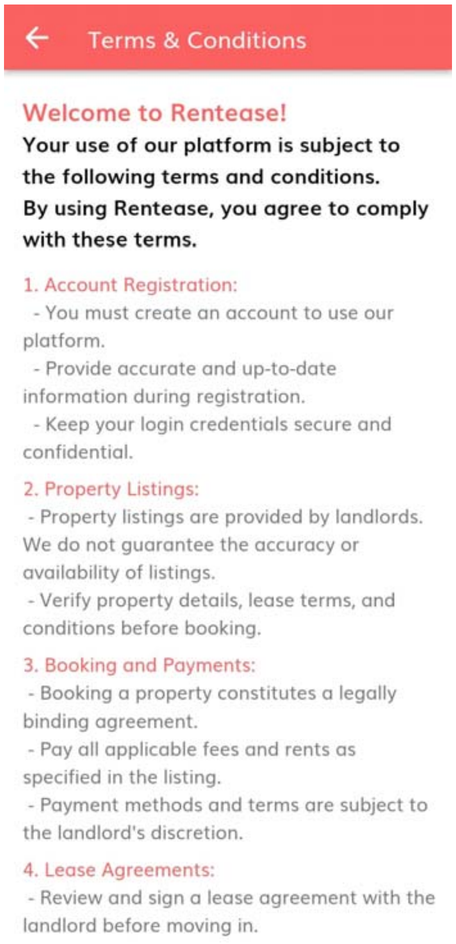

First Name: Tahir

Last Name: momin

Phone Number: 8097809001

Email: tahir.momin.juned@gmail.com

Address: address


Terms & Conditions

Welcome to Rentease!

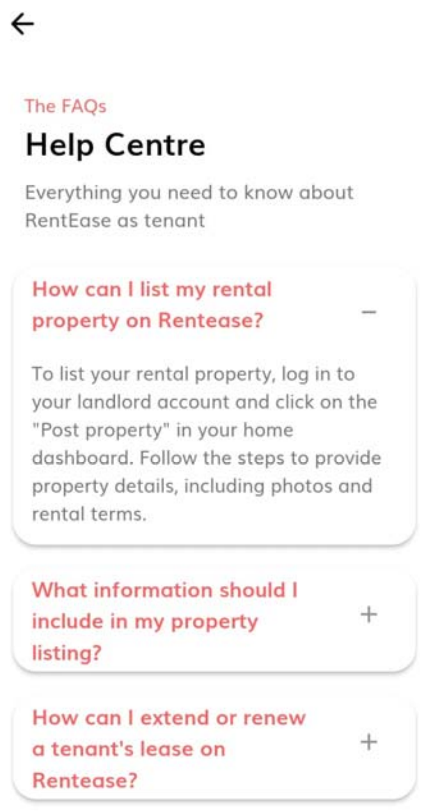
Your use of our platform is subject to the following terms and conditions. By using Rentease, you agree to comply with these terms.

1. Account Registration:
 - You must create an account to use our platform.
 - Provide accurate and up-to-date information during registration.
 - Keep your login credentials secure and confidential.

2. Property Listings:
 - Property listings are provided by landlords. We do not guarantee the accuracy or availability of listings.
 - Verify property details, lease terms, and conditions before booking.

3. Booking and Payments:
 - Booking a property constitutes a legally binding agreement.
 - Pay all applicable fees and rents as specified in the listing.
 - Payment methods and terms are subject to the landlord's discretion.

4. Lease Agreements:
 - Review and sign a lease agreement with the landlord before moving in.


The FAQs

Help Centre

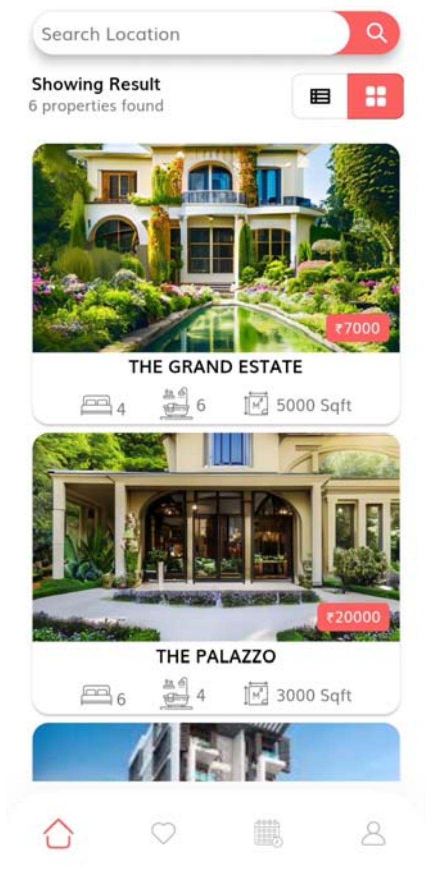
Everything you need to know about RentEase as tenant

How can I list my rental property on Rentease? −

To list your rental property, log in to your landlord account and click on the "Post property" in your home dashboard. Follow the steps to provide property details, including photos and rental terms.

What information should I include in my property listing? +

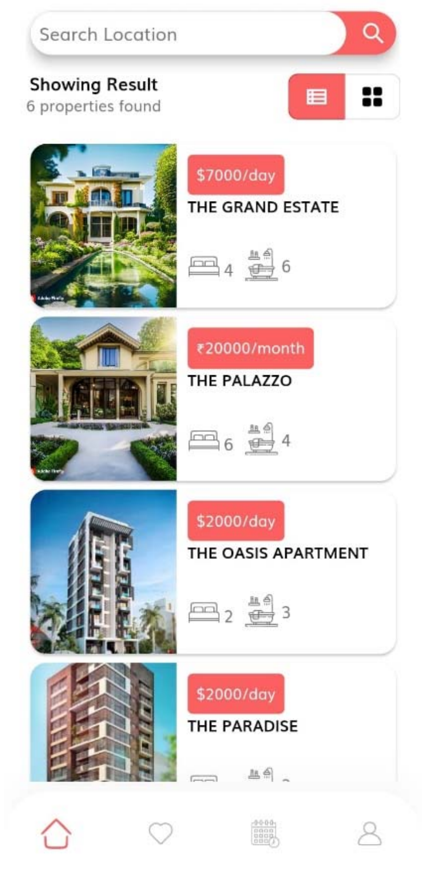
How can I extend or renew a tenant's lease on Rentease? +


Search Location

Showing Result
6 properties found

THE GRAND ESTATE
₹7000
4 6 5000 Sqft

THE PALAZZO
₹20000
6 4 3000 Sqft


Search Location

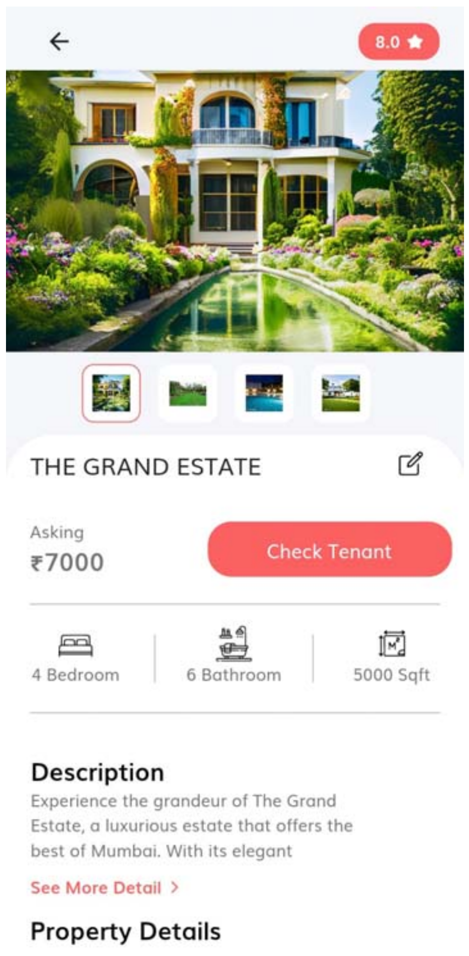
Showing Result
6 properties found

$7000/day
THE GRAND ESTATE
4 6

₹20000/month
THE PALAZZO
6 4

$2000/day
THE OASIS APARTMENT
2 3

$2000/day
THE PARADISE


8.0 ★

THE GRAND ESTATE

Asking
₹7000

Check Tenant

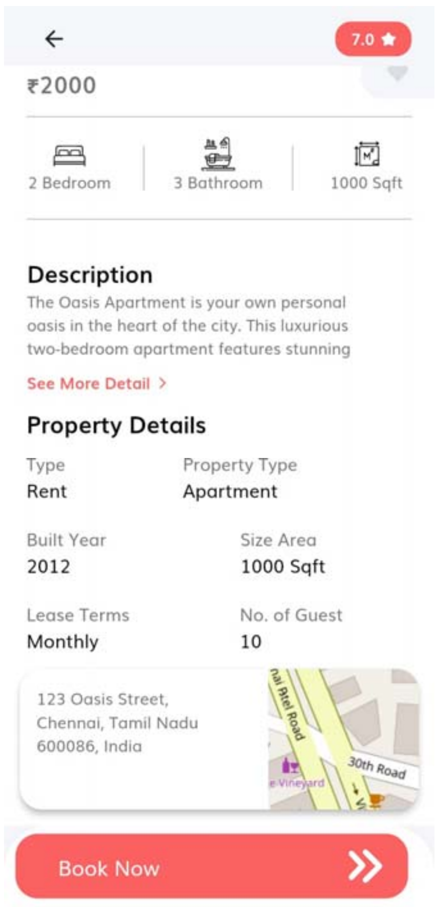
4 Bedroom | 6 Bathroom | 5000 Sqft

Description

Experience the grandeur of The Grand Estate, a luxurious estate that offers the best of Mumbai. With its elegant

See More Detail >

Property Details


7.0 ★

₹2000

2 Bedroom | 3 Bathroom | 1000 Sqft

Description

The Oasis Apartment is your own personal oasis in the heart of the city. This luxurious two-bedroom apartment features stunning

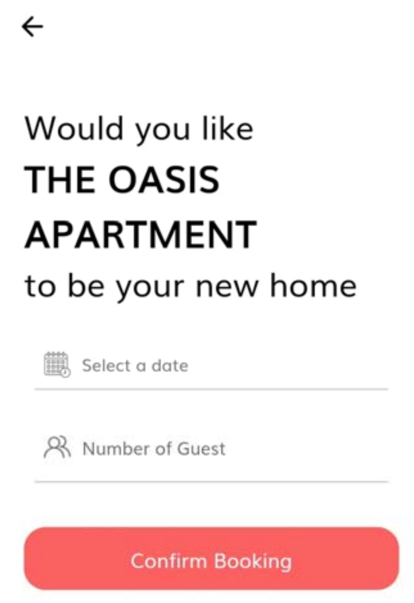
See More Detail >

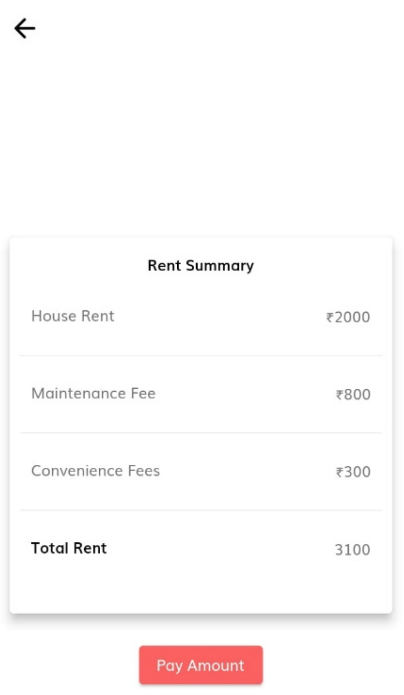
Property Details

Type	Property Type
Rent	Apartment
Built Year	Size Area
2012	1000 Sqft
Lease Terms	No. of Guest
Monthly	10

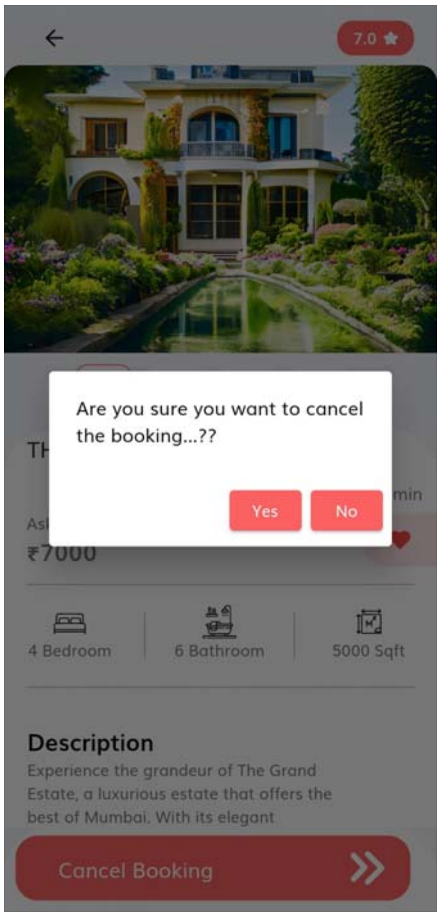
123 Oasis Street,
Chennai, Tamil Nadu
600086, India

Book Now

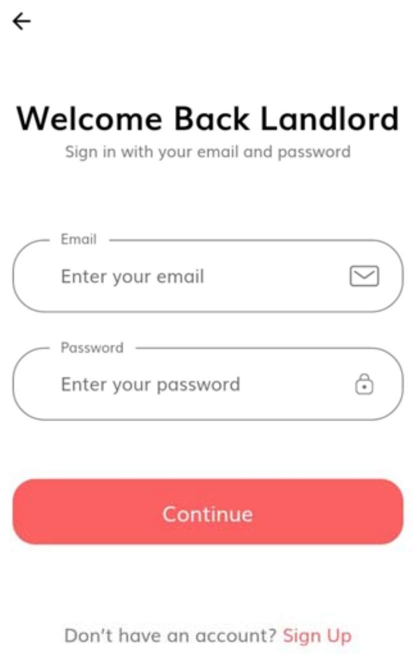

Would you like
THE OASIS APARTMENT
to be your new home


Rent Summary

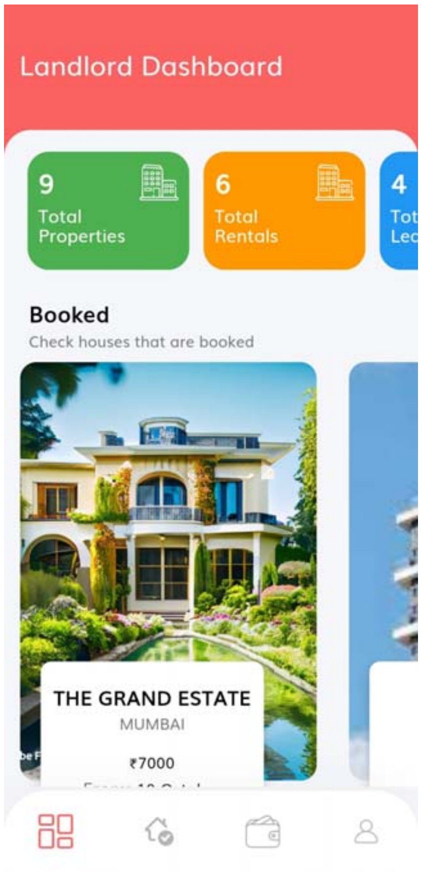
House Rent	₹2000
Maintenance Fee	₹800
Convenience Fees	₹300
Total Rent	3100


Are you sure you want to cancel the booking...??

Description
Experience the grandeur of The Grand Estate, a luxurious estate that offers the best of Mumbai. With its elegant


Welcome Back Landlord
Sign in with your email and password

Don't have an account? Sign Up



Landlord Dashboard

9 Total Properties

6 Total Rentals

4 Tot Lec

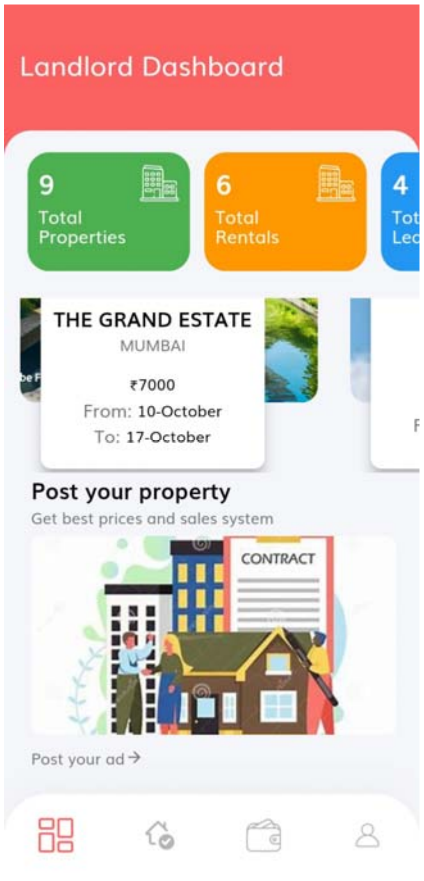
Booked

Check houses that are booked

THE GRAND ESTATE

MUMBAI

₹7000



Landlord Dashboard

9 Total Properties

6 Total Rentals

4 Tot Lec

THE GRAND ESTATE

MUMBAI

₹7000

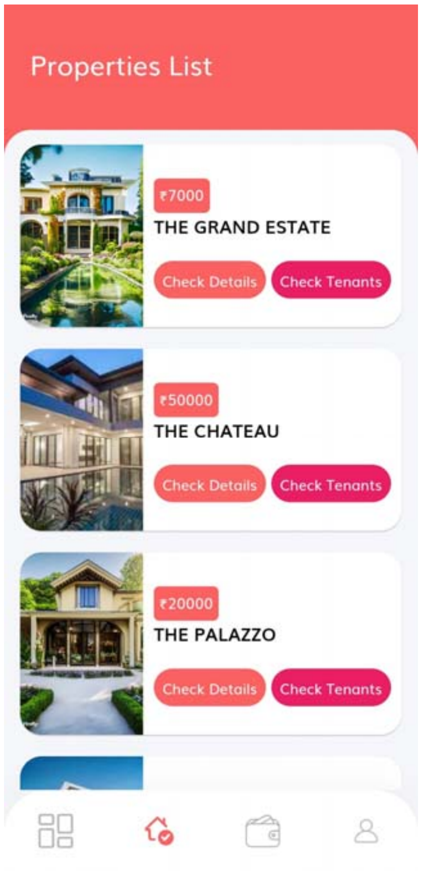
From: 10-October

To: 17-October

Post your property

Get best prices and sales system

Post your ad →



Properties List

₹7000

THE GRAND ESTATE

Check Details Check Tenants

₹50000

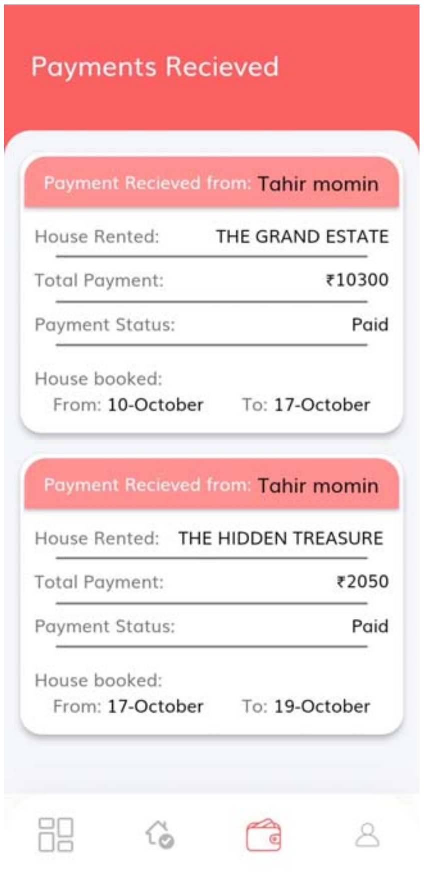
THE CHATEAU

Check Details Check Tenants

₹20000

THE PALAZZO

Check Details Check Tenants



Payments Recieved

Payment Recieved from: Tahir momin

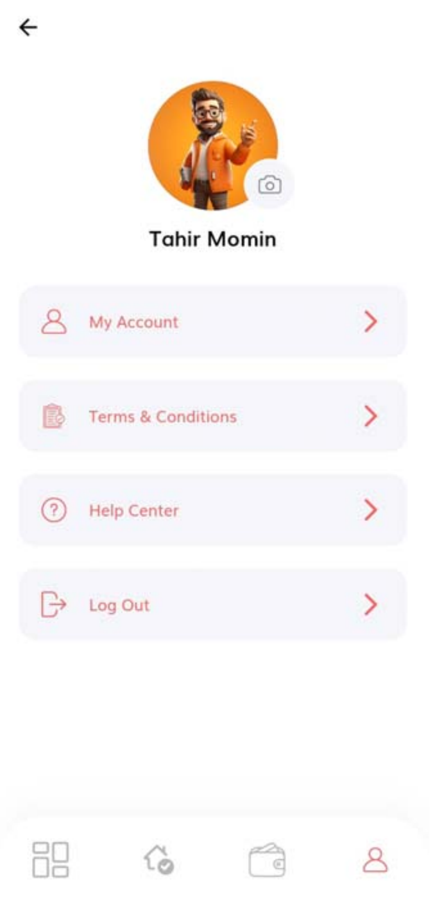
House Rented:	THE GRAND ESTATE
Total Payment:	₹10300
Payment Status:	Paid

House booked:
From: 10-October To: 17-October

Payment Recieved from: Tahir momin

House Rented:	THE HIDDEN TREASURE
Total Payment:	₹2050
Payment Status:	Paid

House booked:
From: 17-October To: 19-October

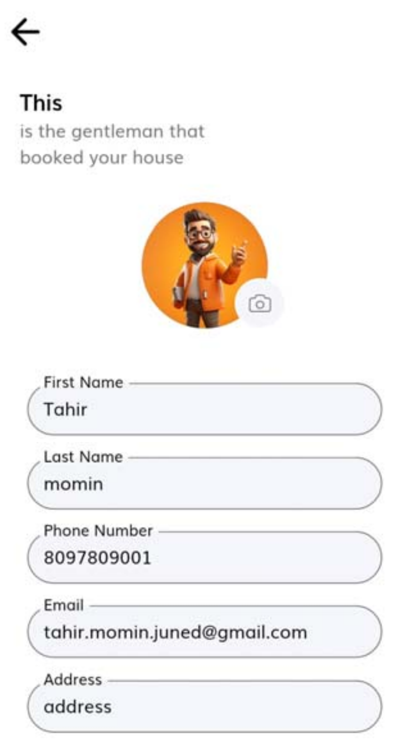

Tahir Momin

My Account

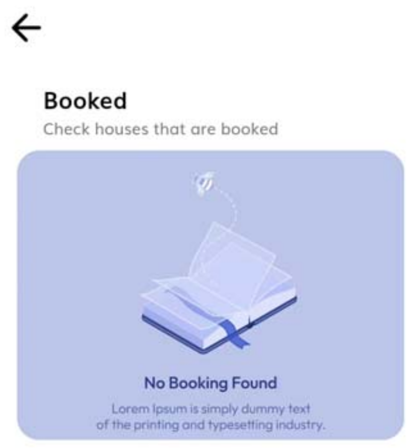
Terms & Conditions

Help Center

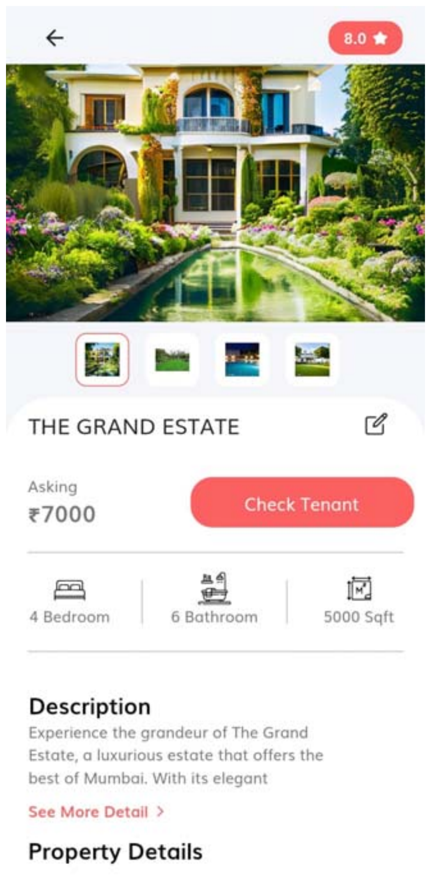
Log Out


This
is the gentleman that
booked your house

First Name	Tahir
Last Name	momin
Phone Number	8097809001
Email	tahir.momin.juned@gmail.com
Address	address


Booked
Check houses that are booked

No Booking Found
Lorem Ipsum is simply dummy text
of the printing and typesetting industry.



THE GRAND ESTATE

Asking
₹7000

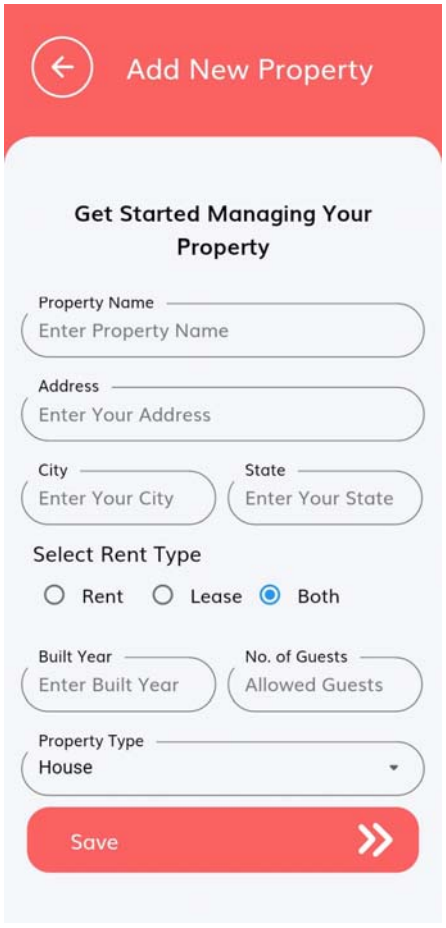
Check Tenant

4 Bedroom | 6 Bathroom | 5000 Sqft

Description
Experience the grandeur of The Grand Estate, a luxurious estate that offers the best of Mumbai. With its elegant

See More Detail >

Property Details



Add New Property

Get Started Managing Your Property

Property Name: Enter Property Name

Address: Enter Your Address

City: Enter Your City

State: Enter Your State

Select Rent Type

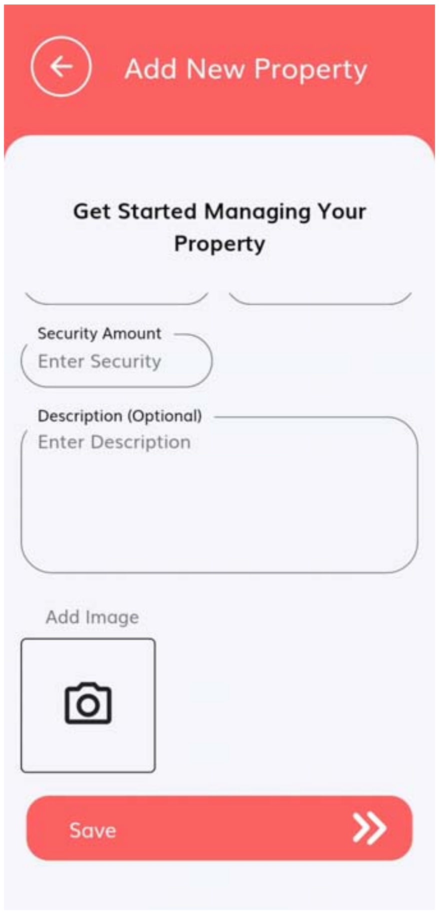
Rent [✓ Lease] Both — chosen: Both

Built Year: Enter Built Year

No. of Guests: Allowed Guests

Property Type: House

Save



Add New Property

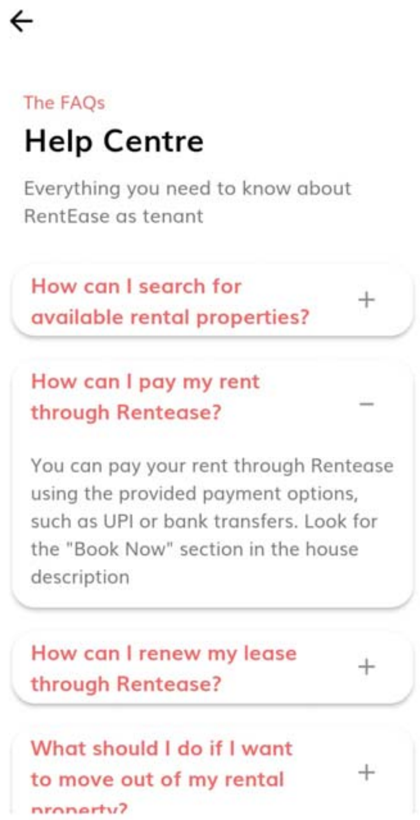
Get Started Managing Your Property

Security Amount: Enter Security

Description (Optional): Enter Description

Add Image

Save



The FAQs

Help Centre

Everything you need to know about RentEase as tenant

How can I search for available rental properties? +

How can I pay my rent through Rentease? –

You can pay your rent through Rentease using the provided payment options, such as UPI or bank transfers. Look for the "Book Now" section in the house description

How can I renew my lease through Rentease? +

What should I do if I want to move out of my rental property? +

9. Future Scope

The future scope for RentEase is promising, with potential opportunities for expansion and improvement in various aspects of the platform. Here are some key areas of future scope for RentEase:

1. Geographical Expansion:
   - RentEase can consider expanding its services to cover a wider geographical area, both nationally and internationally. This expansion would open up new markets and increase the user base.

2. Property Types and Categories:
   - Expanding the types of properties available on the platform, such as commercial properties, vacation rentals, or short-term accommodations, can attract a diverse set of users and property owners.

3. Advanced Search and Recommendation Algorithms:
   - Implementing advanced search and recommendation algorithms can enhance the user experience. Machine learning and AI-driven algorithms can provide personalized property recommendations based on user preferences and behavior.

4. Virtual Reality (VR) Property Tours:
   - Integrating VR technology to offer virtual property tours can revolutionize the way users explore rental properties. Users can virtually walk through properties from the comfort of their homes, saving time and effort.

5. Property Management Tools:
   - Developing property management tools for landlords to efficiently manage their rental properties, including maintenance requests, rent collection, and financial tracking, can add significant value to the platform.

6. Payment Features and Financial Services:
   - Expanding financial services within the platform, such as rent payment processing, financial planning tools, and insurance options, can make RentEase a comprehensive solution for both tenants and landlords.

7. Blockchain and Smart Contracts:
   - Integrating blockchain technology for secure and transparent lease agreements through smart contracts can reduce disputes and enhance trust between parties.

8. Sustainability and Green Housing:
   - Promoting and listing environmentally sustainable and energy-efficient properties can align with the growing trend of eco-conscious living.

9. Community Features:
   - Creating a community or social networking aspect within the platform where users can share experiences, tips, and advice related to renting can foster a sense of belonging and engagement.

10. Data Analytics and Insights:

- Enhancing data analytics capabilities can provide users with valuable insights into rental trends, pricing dynamics, and neighborhood information, aiding in informed decision-making.

11. Enhanced Security and Privacy:

- Continuously improving security measures to protect user data and privacy is crucial in maintaining user trust. This includes robust identity verification and data encryption.

12. Partnerships and Integrations:

- Collaborating with real estate agencies, property management companies, financial institutions, and other service providers can expand the range of services available on the platform.

14. Regulatory Compliance:

- Keeping abreast of evolving rental and real estate regulations and ensuring RentEase complies with them is essential for long-term sustainability.

15. User Feedback and Continuous Improvement:

- Actively seeking user feedback and making continuous improvements based on user suggestions and needs is fundamental to staying competitive and user-centric.

16. AI-Powered Predictive Analytics:

- Utilizing predictive analytics and AI to forecast rental market trends, helping users make informed decisions about when and where to rent.

17. Localized Content and Language Support:

- Offering localized content and language support to cater to international markets and non-English-speaking users.

RentEase's future success lies in its ability to adapt, innovate, and meet the evolving needs of both renters and property owners. By embracing these future scopes, RentEase can solidify its position as a leading rental platform in the real estate industry.

10. Conclusion

In conclusion, RentEase stands as a transformative force in the world of house rentals, offering a user-centric platform that streamlines the entire rental journey for both tenants and landlords. With a commitment to user convenience, security, and transparency, RentEase has redefined the rental experience, setting a new standard in the industry.

One of RentEase's key strengths is its comprehensive approach to rental management. Through its well-designed modules, including User, Admin, and Landlord functionalities, RentEase ensures a seamless and efficient process from the initial property search to lease agreements and rental payments. Tenants can easily find their ideal rental properties, connect with landlords, and navigate the rental process with confidence, while landlords benefit from simplified property management tools.

What sets RentEase apart is its dedication to innovation. The incorporation of cutting-edge technologies like Flutter, Dart, Node.js, Express, and SQLite3 provides a robust and scalable foundation for the platform. Moreover, RentEase embraces artificial intelligence to offer personalized property recommendations, virtual property tours, and even blockchain-based smart contracts, opening up exciting possibilities for the platform's continued growth and improvement.

As RentEase looks toward the future, its scope remains promising. The potential for geographical expansion presents opportunities to serve a broader audience. Advanced search algorithms will enhance user experiences by ensuring more accurate property matches. Sustainability initiatives will align RentEase with eco-friendly practices, contributing to a greener planet. Community-building features will foster connections among users, creating a sense of belonging within the RentEase community.

Above all, RentEase places a premium on user satisfaction. The platform continuously seeks feedback and actively implements improvements to meet the ever-changing needs of its diverse user base. The unwavering commitment to security, convenience, and innovation sets RentEase apart, making it an indispensable tool for anyone embarking on the journey of finding or renting a home.

In the competitive landscape of rental platforms, RentEase shines as a beacon of user-centricity, innovation, and efficiency. It has not only simplified house rentals but also set a high standard for the entire industry. With RentEase, the future of house rentals is brighter than ever, offering a promising and transformative rental experience for all.

11. Bibliography

• Tutorials and Documentation:

- GeeksforGeeks - geeksforgeeks.org/flutter-tutorial
- TutorialsPoint - tutorialspoint.com/dart
- W3Schools - w3schools.com/Django

• Official Documentation:

- Node JS - https://nodejs.org/docs/
- Express JS - https://expressjs.com/
- Sqlite3 - https://www.sqlite.org/docs.html

• Flutter Documentation

- flutter.dev/docs

• Dart Documentation

- dart.dev/guides

• Forums and Communities:

- Stack Overflow - stackoverflow.com/questions/tagged/flutter
- Reddit - reddit.com/r/dartlang
- GitHub - github.com/djangorsestframework/Django

• Blogs:

- Dev.to - dev.to/t/flutter