RentEasy

Rent It. Use It. Return It.

Sumago Infotech Pvt. Ltd.

SCOPE

Strive With Technology...!

Project Title

RentEasy – P2P Rental Platform

Submitted By

I. Krushna Ramesh Mengal (**Team Leader**)

II. Shreya Laxman Agiwale

III. Bhumika Valmik Ighe

IV. Neha Vitthal Bhoye

Course

Diploma in Computer Technology – Final Year

Institutes

Amrutvahini Polytechnic, Sangamner
Sanjivani K.B.P. Polytechnic, Kopargaon

Project Guide

Ravi Kandekar(React Native Developer)

Sign of Mentor

Sign of Training Master

INDEX

Introduction

RentEasy is a modern digital platform designed to facilitate peer-to-peer sharing of underutilized items. It empowers users to rent their unused belongings to others who need them temporarily, fostering a sharing economy that saves money, reduces waste, and creates new earning opportunities.

> **Use Case Example:**
> A student needing a DSLR camera for a short project can rent it from a person who owns one but rarely uses it, avoiding the cost of purchasing new equipment.

Problem Statement

Many individuals possess items like tools, electronics, or books that sit unused most of the time. Meanwhile, others require these items but face high costs or inconvenience buying them outright. There is a lack of accessible online platforms connecting owners and borrowers efficiently and securely.

Solution Overview

RentEasy addresses this gap by providing an intuitive app that allows:

- Owners to list items with detailed descriptions, images, and prices
- Borrowers to search, filter, and select items based on preference and location
- A secure booking and payment system integrated with reliable payment gateways
- In-app messaging between parties for communication and coordination
- Comprehensive rental histories, user reviews, ratings, and real-time notifications

User Roles

- **Item Owner (Lender):** Manages their item listings, rental approval, pricing, and earnings.
- **Borrower:** Searches for items, requests bookings, rents, and returns items responsibly.

Key Features

No.	Feature	Description
1	User Authentication	Secure login and registration for both owners and borrowers
2	Item Listing	Upload photos, information, rental price, and availability status
3	Search & Filtering	Filter items by location, category, price range, and ratings
4	Booking System	Send rental requests with specified dates
5	Payment Integration	Secure payment processing via Razorpay or Stripe
6	In-App Chat	Real-time communication between users to coordinate rentals
7	Rental History	Track current and past rentals
8	Notifications	Timely alerts on request status, payments, returns

System Architecture / User Flow

```
User (Owner or Borrower)

              |

              ▼

Login/Register → Home / Dashboard

               |

   ------------------------------------------

   |                  |                |

Add/List         Search Items      Request → Payment → Confirmation

Item (Owner)      (Borrower)                 |

                                  Chat / Communication

                                             |

                                  Rent Period → Return Item

                                             |

                                    Rate & Review
```

RentEasy App Screen

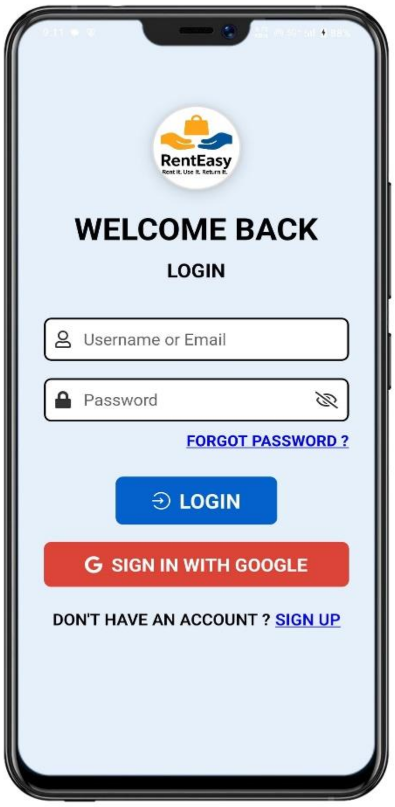
1. Login Screen

Purpose:

Allows existing users to securely log in with their username/email and password, enabling access to the app's main functions. Provides navigation to password recovery and account registration.

Key Features:

- Logo and welcome message ("WELCOME BACK", "LOGIN").
- Username/Email and Password input fields with validation and error display.
- Password visibility toggle icon.
- "Forgot Password" link navigating to password reset.
- Login button handling Firebase authentication.
- Google Sign-In button placeholder for future integration.
- Navigation to SignUp screen for new users.
- Modal popups for success/error feedback.



User Flow:

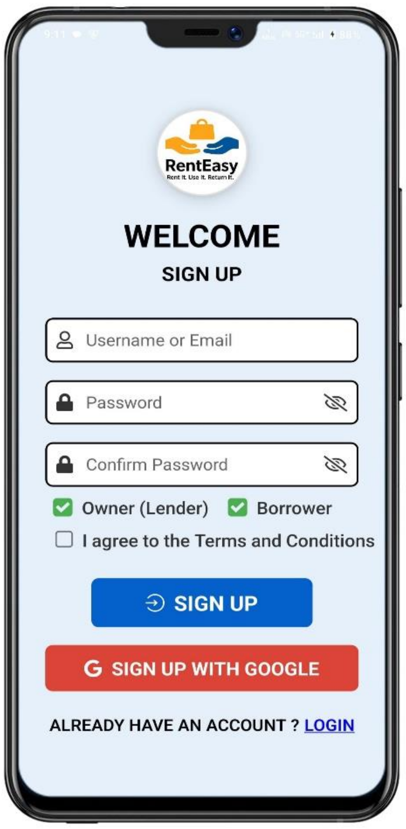
User inputs credentials → inputs are validated → Firebase Auth login → success leads to Home screen → error shows modal.

2. SignUp Screen

Purpose:
Allows new users to create an account by providing username/email, password, confirming password, selecting roles (Owner and/or Borrower), and agreeing to terms.

Key Features:

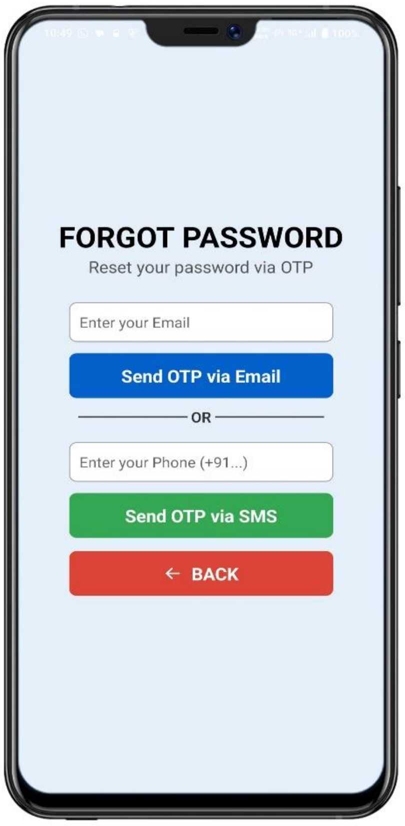
- Logo and sign-up titles (“WELCOME”, “SIGN UP”).
- Inputs for username/email, password, confirm password with chained validation.
- Role selection checkboxes for Owner/Borrower.
- Mandatory Terms and Conditions agreement toggle.
- Account creation with Firebase Authentication and data storage in Realtime Database.
- Navigation back to Login screen.
- Modal dialogs for feedback.



User Flow:
User fills form → validation checks → creates Firebase user → stores user profile → navigates to Home.

3. Forgot Password Screen

Purpose:
Facilitates password reset when users forget their credentials, via OTP sent through email or SMS.



Key Features:

- Inputs for email and phone.
- Buttons to send OTP (One-Time Password) via EmailJS (email) or Firebase SMS OTP (phone).
- Input validation for email format and presence of phone number.
- Modal feedback for success or errors.
- Masked email display for privacy when OTP is sent.

User Flow:
User chooses email or phone → inputs info → OTP sent → navigates to OTP verification.

4. OTP Verification Screen

Purpose:
Verifies the OTP sent via Email or SMS to authenticate password reset requests.

Key Features:

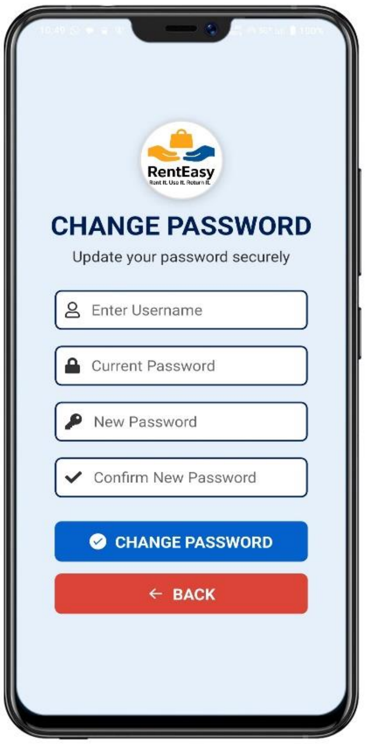
- Displays instruction based on verification method.
- Numerical OTP input with formatted styling.
- Verifies OTP by matching generated code (email) or confirming via Firebase (SMS).
- Navigates to ResetPassword screen upon successful verification.
- Modal dialogs for error or success messages.

5. Reset Password Screen

Purpose:
Allows users to securely set a new password after successful OTP verification.

Key Features:

- Inputs for new password and confirmation with matching validation.
- Updates user password in the backend (Firebase Realtime Database).
- Feedback via modal and navigation back to Login screen.

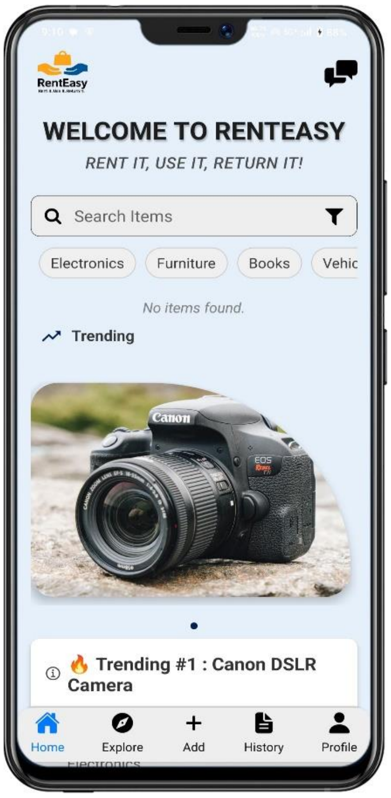


6. Home Screen

Purpose:
Main landing page post-login; welcomes users, displays trending rental items, search/filter options, and app navigation shortcuts.

Key Features:

- Header with logo (home navigation) and chat icon.
- Welcome title and tagline.
- SearchWithFilter component for live item searching.
- Trending section showing top 3 rental items by popularity with ProductCard components.
- Prominent button navigating to BrowseItems screen.
- Fixed bottom navigation bar (Home, Explore, Add, History, Profile).
- Loader and modal for async feedback.

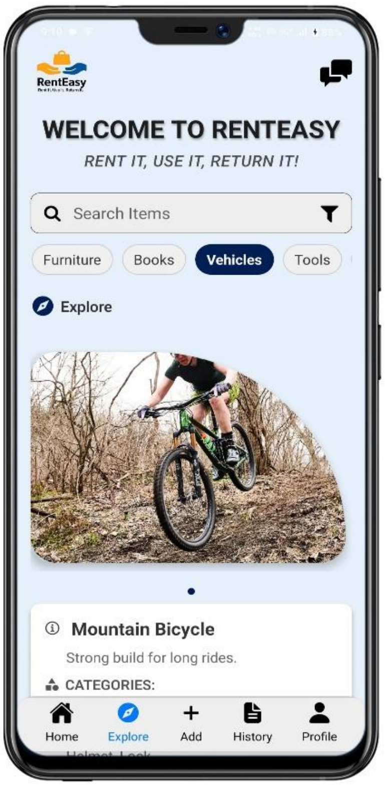


7. BrowseItems Screen

Purpose:
Provide a comprehensive listing of all rental items, with search and filter functionality.

Key Features:

- Back navigation header with chat access.
- Page titles and SearchWithFilter component managing filtered results.
- Dynamic rendering of filtered items via ProductCard component, including image, description, pricing, owner, and availability.
- Bottom navigation consistent with app.

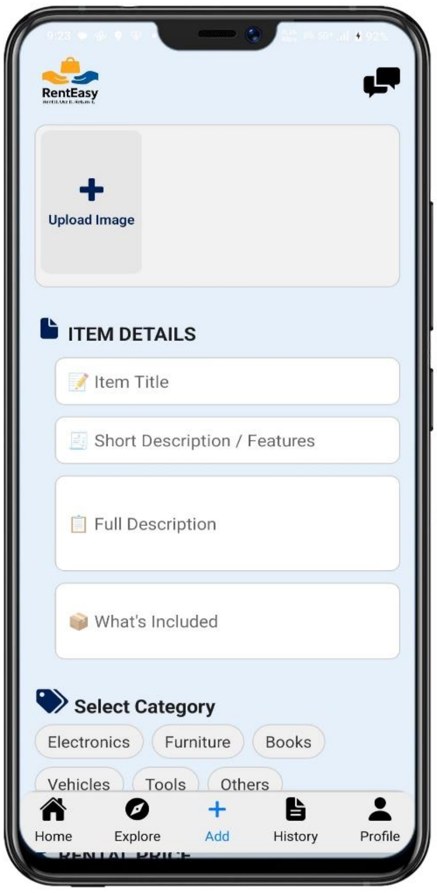


8. AddItem Screen

Purpose:
Allows users to add new rental items with detailed info, images, pricing, availability, and terms.

Key Features:

- Header with navigation icons.
- Image picker allowing up to 5 images with preview.
- Inputs for title, descriptions, categories, pricing, security deposit, location.
- Availability status toggles.
- Owner details inputs.
- Terms and conditions toggles with custom terms input.
- Preview modal for checking item before posting.
- Save/post and reset buttons.
- Bottom navigation, loader, modals for status feedback.



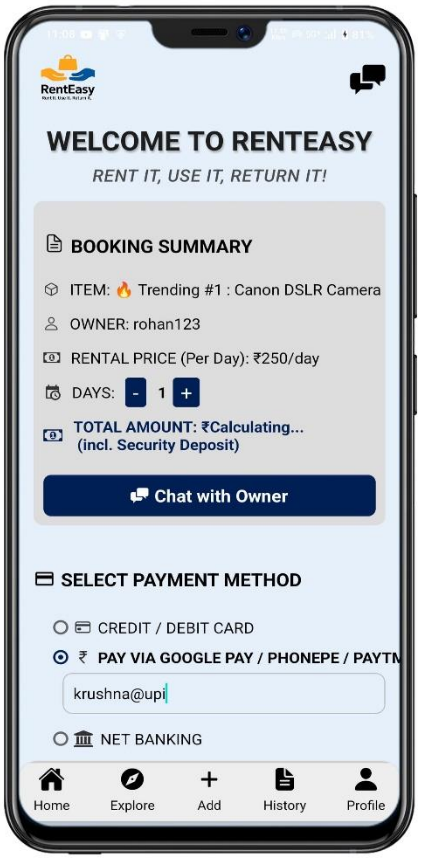
9. Payment Screen

Purpose:

Enables users to complete rental payments for selected items, choosing payment method, and confirming rental duration.

Key Features:

- Booking summary with item details, rental days selection (+/-), and total cost dynamically calculated.
- Payment method options: Card, UPI, Net Banking — each with contextual inputs.
- Agreement checkbox enforcing rental policy consent.
- “Pay Now” button validating inputs, saving transaction, and updating item purchase counts.
- Chat button to communicate with owner.
- Modal dialogs for confirmation and errors.



RentEasy

WELCOME TO RENTEASY

RENT IT, USE IT, RETURN IT!

BOOKING SUMMARY

ITEM: 🔥 Trending #1 : Canon DSLR Camera

OWNER: rohan123

RENTAL PRICE (Per Day): ₹250/day

DAYS: - 1 +

TOTAL AMOUNT: ₹Calculating...
(incl. Security Deposit)

Chat with Owner

SELECT PAYMENT METHOD

CREDIT / DEBIT CARD

[✓ **PAY VIA GOOGLE PAY / PHONEPE / PAYTM**]

krushna@upi

NET BANKING

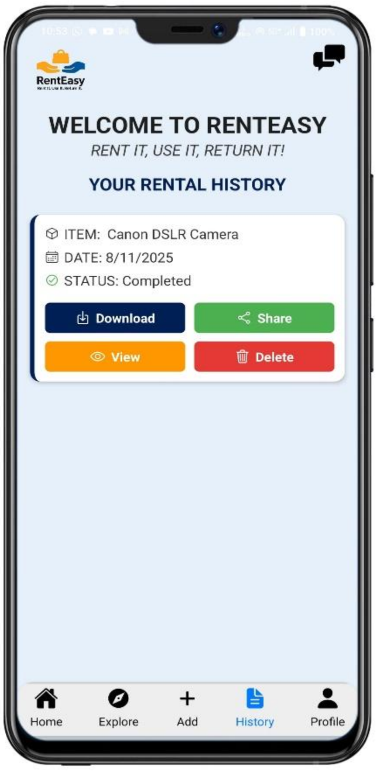
Home Explore Add History Profile

10. History Screen

Purpose:
Displays the user's rental history with options to view, download, share receipts, or delete entries.

Key Features:

- Rental entries display with title, owner, price, date, status icons.
- Action buttons: Download (PDF), Share (PDF), View (PDF preview), Delete (with confirmation).
- Loader and modal for process feedback.
- Fixed bottom navigation bar.



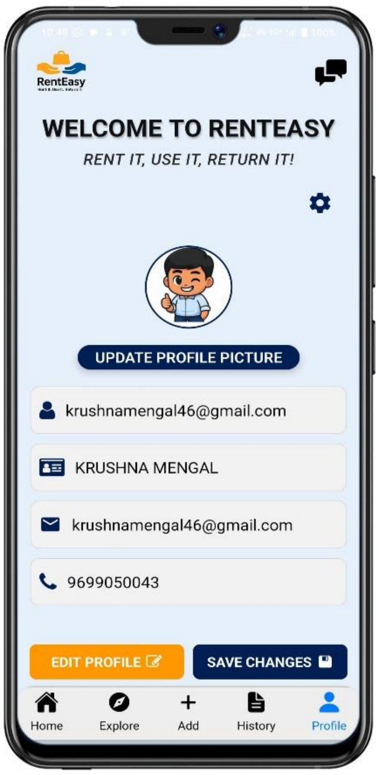
ITEM: Canon DSLR Camera
DATE: 8/11/2025
STATUS: Completed

11. Profile Screen

Purpose:
 Showcases personal user information, owned listings, rental history, and allows profile editing including image updates.

Key Features:

- Profile picture with update feature.
- Editable fields for username, full name, email, phone (edit mode toggled).
- Dynamic sections:
    - Listings (for Owner role)
    - Rentals (for Borrower role)
- Settings navigation button.
- Edit and Save buttons to update backend and local storage.
- Loader and modal for feedback.
- Dark and light theme support.
- Bottom navigation.



WELCOME TO RENTEASY

RENT IT, USE IT, RETURN IT!

krushnamengal46@gmail.com

KRUSHNA MENGAL

krushnamengal46@gmail.com

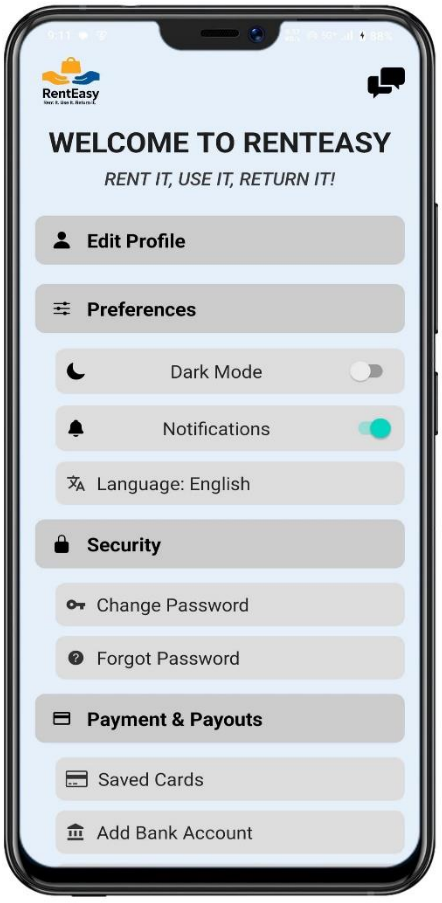
9699050043

12. Settings Screen

Purpose:
Provides user with app preferences, security options, payment & payout management, help & support, and account management features.

Key Features:

- Toggles for dark mode and notifications.
- Language selection cycling among English, Marathi, Hindi.
- Navigation to change/forgot password screens.
- Payment and banking info management shortcuts.
- Help & Support modal with FAQs, contact options, feedback submission, and about section.
- Account options:
  - Logout with confirmation
  - Clear app cache
  - Account deactivation with deletion confirmation
- Modal dialogs for confirmations and errors.

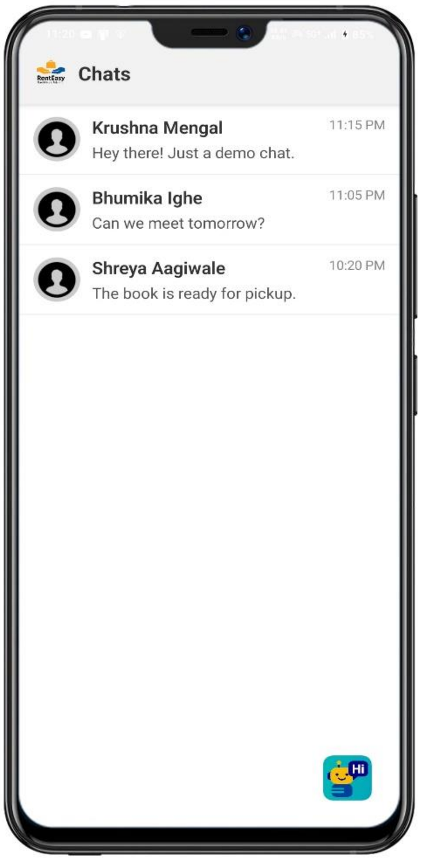


13. Chat Screen

Purpose:
Allows real-time messaging between the current user and another user (typically an item owner).

Key Features:

- Header with app logo and chat icon.
- Chat title indicating chat partner.
- ScrollView displaying messages with left/right alignment based on sender.
- Input bar with text input and send button.
- ChatBot shortcut button.
- Live synchronization with Firebase Realtime Database messages node.
- Modal dialogs for errors and confirmations.
- Auto-scroll on new messages.



Krushna Mengal 11:15 PM
Hey there! Just a demo chat.

Bhumika Ighe 11:05 PM
Can we meet tomorrow?

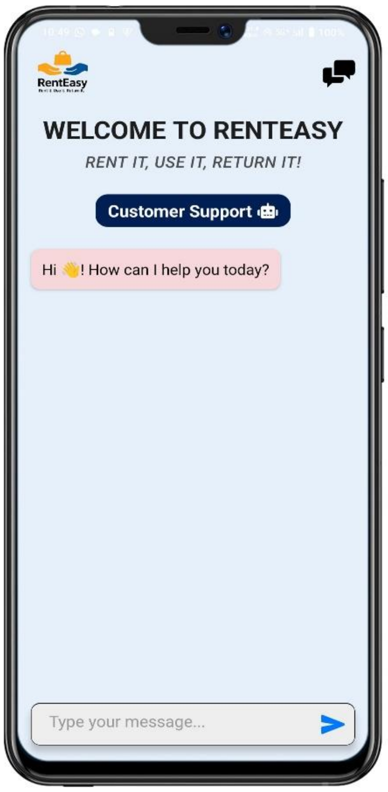
Shreya Aagiwale 10:20 PM
The book is ready for pickup.

14. ChatBot Screen

Purpose:
Interactive AI-powered chatbot for customer support, using Google Gemini API for dynamic, context-aware replies.

Key Features:

- Header with logo and navigation to Chat screen.
- Welcome messages and customer support button with robot icon.
- Scrollable message list with distinct user/bot message styles.
- Message input with send button and loading state.
- API integration with Google Gemini for generating bot responses.
- Modal dialog for errors.
- Bottom navigation consistent with the app.



Hi 👋! How can I help you today?

RentEasy Components

1. Loader Component

Purpose:
The Loader component provides a reusable, full-screen modal overlay displaying an activity spinner and the RentEasy logo to indicate ongoing background processing such as data fetching or network operations. It helps enhance user experience by preventing UI interactions while the app is busy.

Key Features:

- Controlled visibility via the visible prop.
- Modal with semi-transparent backdrop dimming the underlying screen.
- Animated logo display with shadow and glow effects (especially on iOS).
- Native activity spinner colored in App's accent orange (#FFA500).
- Platform-specific styling to handle differences (notably hides the glow effect on Android as per current styles).

Usage Flow:
Parent components toggle visible to show/hide the loader during async operations like login, data fetch, or uploads.

2. ProductCard Component

Purpose:
Displays detailed and interactive previews of rental items, including image carousels, ratings, descriptions, pricing, and owner info. It acts as a browse-and-book card throughout the app.

Key Features:

- Supports horizontal scrolling through multiple images with visible dots indicating the selected image.
- Dynamically renders star ratings with full and half stars.
- Presents item info sections such as features, categories, included items, pricing, security deposit, owner, location, and availability status.
- Includes a "Book Now" button which navigates to the Payment screen, passing along item identifiers and details.

Interaction Flow:
Users swipe images, read info, and tap "Book Now" to proceed with rental bookings.

3. RentEasy Modal Component

Purpose:
Reusable modal dialog to display alerts, confirmations, and informational messages across the app consistently.

Key Features:

- Displays a centered popup with the RentEasy logo, a customizable title, message, and an "OK" button.
- Accepts optional confirmation callback invoked on user acknowledgement.
- Semi-transparent black backdrop dims and blocks interaction with the rest of the UI.
- Smooth fade-in/out animation provides a polished feel.

Usage Flow:
Invoked whenever feedback to the user is required, such as errors, success messages, confirmations, etc.

4. RentEasy Welcome Component

Purpose:
Animated splash/welcome screen presenting the RentEasy logo and branding messages with smooth scale and fade-in animations to create a positive first impression.

Key Features:

- Logo scales up with spring animation on mount.
- Welcome text fades in and slides upward in coordination with logo animation.
- Styled with app-themed colors (soft blue background, navy text, warm orange highlights).
- Layout centered both vertically and horizontally on screen.

Usage Flow:
Shown at app launch before routing to auth or main app screens.

5. SearchWithFilter Component

Purpose:
Provides a combined search bar and category filter UI enabling users to efficiently browse rental items by text queries and/or category selections.

Key Features:

- Text input with debounced search triggering live filtering after 300 ms of inactivity.
- Horizontal inline scrollable chips displaying category filters, with clear visual indication of selection.
- Modal popup for expanded category filter selection with "Clear Filter" option.

- Displays "No items found" message dynamically when filters yield no results.
- Prop-driven, updating the filtered items list in parent components.

Interaction Flow:
Users type to search, tap chips to filter, and clear inputs or filters as needed.

6. Upload Images to Firebase

Purpose:
Utility function for uploading multiple images from local device URIs to Firebase Storage and returning publicly accessible download URLs for each.

Key Features:

- Uses Firebase Storage API to convert local URI images to blobs and upload them one by one.
- Generates unique file names for storage using UUID to avoid collisions.
- Returns an array of downloadable URLs upon completion.
- Designed for integration with screens managing item photo uploads (such as Add Item).

Usage Flow:
Call the function with an array of image URIs; use returned URLs to save in database and display in the app.

<u>RentEasy Constant</u>

Categories Constant

Definition:

Static array of category objects for classifying rental items.

Details:

- Each category has an id (internal key) and label (display name).
- Categories include Electronics, Furniture, Books, Vehicles, Tools, Others.

Usage:

- Used in item tagging during creation.
- Supports filtering and display in UI components like SearchWithFilter and ProductCard.

RentEasy Navigation Setup - AppNavigator

Purpose:
The AppNavigator is the root navigation component of the RentEasy React Native app. It manages the routing and screen transitions throughout the app, directing users to the appropriate screen based on their authentication status. It ensures a smooth and structured flow from launching the app to accessing all core features.

Key Features and Behavior:

1. **Authentication Check & Initial Route:**
   Upon startup, the app checks local storage (AsyncStorage) for existing logged-in user data ("loggedInUser").

   - If user data is found, the navigation starts at the **Home** screen, enabling a direct entry to the main app experience.

   - If no user data exists (new or logged-out user), it starts at the **Login** screen for authentication.

   - During the check, a **Welcome Splash Screen** (RentEasyWelcome) is displayed to engage the user while background tasks complete.

2. **Stack Navigator:**
   The app employs React Navigation's **native stack navigator** to map each screen appropriately. Each screen is registered with a unique route name, enabling easy navigation calls throughout the app.

3. **Screen List:**
   The navigator includes all major screens of the app:

   - Authentication screens: **Login**, **SignUp**, **ForgotPassword**, **OTPVerification**, **ResetPassword**, **ChangePassword**

   - Main app flows: **Home**, **BrowseItems**, **AddItem**, **History**, **Profile** **Payment**

    - User profile and settings: **Profile**, **Settings**, **About**
    - Communication: **Chat**, **ChatBot**

4. **Navigation Options:**
   All screens have the navigation header disabled (headerShown: false), allowing each screen to implement custom headers or use full-screen layouts for a consistent UI style.

5. **Navigation State Management:**
   The component maintains an internal state variable initialRoute that captures the target starting screen after checking login status. This state ensures the navigator doesn't render until the user's authentication state is known.

6. **Extensibility:**
   New screens can be added simply by importing them and adding an entry to the Stack.Navigator. Navigation between screens is done using navigation.navigate("ScreenName"), along with passing any necessary parameters.

Typical User Flow via Navigation Component:

- App launch → splash animation on RentEasyWelcome → check authentication → navigate to
    - **Login** (if unauthenticated) → ability to move to SignUp or password recovery; or
    - **Home** (if authenticated) → access to app core functionality including browsing, chatting, profile, and settings.
- From any screen, users can navigate to various parts of the app like BrowseItems, AddItem, Chat, Payment, etc., facilitated by the stack navigator.

RentEasy App.jsx

App Component

Purpose:
This is the root component of your React Native app. It serves as the main entry point and renders the navigation container defined in your AppNavigator. This setup is essential to bootstrap the app's navigation flow.

Key Details:

- Imports the AppNavigator component, which handles all navigation and screen routing in the app.
- The App functional component simply returns the AppNavigator component.
- Exports the App component as default, which will be used by React Native's app registry to run the app.

How it fits in the app:

- AppNavigator decides which screen to show first based on whether the user is logged in.
- By rendering AppNavigator here, App delegates all screen and flow management to it.
- This separation keeps the main app bootstrap clear and modular.

Usage:

- This App component is typically referenced in your index.js (or index.tsx) as the root to render the entire app.
- Allows easy replacement or modification of navigation logic without touching the app's root

RentEasy – Technology Stack and Dependencies

The RentEasy app is built using a modern, efficient, and widely adopted technology stack centered around **React Native** with a combination of powerful libraries and services that enable fast development, cross-platform compatibility, rich user experience, and backend integration.

1. Core Framework

- **React Native**
  React Native is the core framework for building native mobile applications for both Android and iOS using a single JavaScript/JSX codebase. It enables building performant apps with native look and feel, leveraging React's component-based architecture and a rich ecosystem. React Native uses a bridge to communicate between JavaScript and native modules, providing high flexibility and performance.reactnative+1

2. Navigation

- **@react-navigation/native** & **@react-navigation/native-stack**
  React Navigation is the community standard for routing and navigation in React Native apps. It supports stack, tab, and drawer navigators with smooth transitions and customizable headers. Native-stack navigator leverages native platform capabilities for performance and seamless experience.reactnative

- **Required Peer Dependencies**
  - react-native-screens
  - react-native-safe-area-context
    These optimize stack navigation and respect device safe areas (notches, status bars), enhancing UI correctness and performance.reactnative

3. Backend and Database

- **Firebase (Realtime Database, Authentication, Storage)**
  Firebase is used for realtime data storage, authentication, and media

storage in this app. It supports user login, storing rental items, chats, history, and image files with realtime syncing capabilities and scalability.geeksforgeeks+1

- **Firebase Storage**
  Used alongside Realtime Database for uploading and serving item images securely.

4. State Management & Storage

- **AsyncStorage** (@react-native-async-storage/async-storage)
  A persistent, asynchronous, key-value storage system for React Native. Used for caching user session and profile info locally on the device.

5. HTTP Client

- **Axios**
  A promise-based HTTP client used to handle API requests to Firebase and other REST endpoints. It simplifies asynchronous communication and error handling.geeksforgeeks+1

6. UI Components and Utilities

- **react-native-image-picker**
  Allows users to select images from their device's media library or camera. Supports multiple image selection, media type filtering, and customization options for image quality.npmjs

- **react-native-vector-icons**
  Provides a vast collection of customizable icons from popular icon sets like FontAwesome, Ionicons, MaterialIcons, and Entypo. Enhances UI readability and aesthetics with scalable vector icons.dev+1

- **React Native Modal & Animations**
  Custom modal components like RentEasyModal are implemented to

show alerts and confirmations. React Native's built-in Modal and Animated APIs are used for modals and animated welcome screens.

7. Environment Variables

- **@env** or similar environment variable management
  Used to securely store API keys and secrets (e.g., Gemini AI API key) without hardcoding them in source code, ensuring safe and configurable environment-specific settings.

8. Additional Libraries/Tools

- **UUID (uuid package)**
  For generating unique identifiers when uploading images or creating item keys, ensuring conflict-free storage and database records.

- **React Native Gesture Handler & Reanimated (implied in navigation dependencies)**
  Often used to enhance gesture-based navigation and animations in React Native apps.

- **JavaScript (ES6+) and JSX**
  For application logic and UI components.

- **Node.js and npm/yarn**
  For package management and running development scripts.

- **Android Studio / Xcode**
  Native development environments used for building and debugging Android and iOS apps.

Dependency	**Purpose**
react-native	Core framework for cross-platform mobile app development
@react-navigation/native	Navigation container and utilities
@react-navigation/native-stack	Stack navigation optimized for native transitions
react-native-screens	Enhances native navigation screen performance
react-native-safe-area-context	Handles device safe area insets
@react-native-async-storage/async-storage	Persistent local storage for user session and profile
firebase (auth, database, storage)	Backend realtime data, auth, and media storage
axios	HTTP client for API requests
react-native-image-picker	Image picking from device camera/gallery
react-native-vector-icons	Cross-platform vector icons for UI
uuid	Unique ID generation
@env (or similar)	Environment variable management

Summary Table of Main DependenciesDevelopment Tools & Languages

Future Scope

- Location-based rental suggestions integrated with Google Maps API.
- Incorporation of security deposits and identity verification for enhanced trust.
- Launch of dedicated Android and iOS apps.
- Optional insurance or damage protection integrations to safeguard rentals.

User Manual Summary

RentEasy simplifies sharing by enabling users to rent items rather than buy, saving cost and resources. Item owners can monetize things otherwise unused, and borrowers can access goods affordably and conveniently. User-friendly search, booking, payment, chat, and review features finally connect communities under a sharing economy ethos.

Conclusion

RentEasy is more than an app; it’s a movement towards responsible consumption through sharing. Encouraging connection, trust, and sustainability, the platform offers modern tech solutions to everyday challenges in peer-to-peer access. Its continuous evolution aims to foster a robust, user-friendly rental community supporting environmental and economic benefits.

This detailed explanation should comprehensively capture your project's vision, features, and technical depth for academic or professional reporting. If you need formatting assistance or section-wise breakdowns, please let me know!

Logo

RentEasy

Rent It. Use It. Return It.

Project Team

Code4Change

Building digital solutions for real-world impact