

-Project Report On



# **RentKaro**

Submitted in partial fulfillment for the award of  
**Post Graduate Diploma in Advanced Computing**

from

**C-DAC ACTS (Pune)**

**Guided by**

**Mr. Abhilash Bande**

**Presented By**

**Prashant Kumar - 240840120117**

**Saurabh Shirbhate - 240840120158**

**Prathmesh Joshi - 240840120121**

**Siddhesh Darade -240840120189**

**Tejas Koshti - 240840120206**

Centre of Development of Advanced Computing (C-DAC), Pune



# **CERTIFICATE**

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Prashant Kumar - 240840120117

Saurabh Shirbhate - 240840120158

Prathmesh Joshi - 240840120121

Siddhesh Darade - 240840120189

Tejas Koshti - 240840120206

have successfully completed their project titled

## **“RentKaro”**

Under the Guidance of **Mr. Abhilash Bande**

Project Guide

HOD ACTS

# ACKNOWLEDGEMENT

This project “**RentKaro**” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of **Mr. Abhilash Bande** for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

Our most heartfelt thanks goes to Ms **Swati mam** (Course Coordinator, PG-DAC) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

Prashant Kumar - 240840120117

Saurabh Shirbhate - 240840120158

Prathmesh Joshi - 240840120121

Siddhesh Darade- -240840120189

Tejas Koshti- 240840120206

# **TABLE OF CONTENTS**

1. Introduction
2. Software Requirements and Specification
3. Tools and technologies used
4. Project Flow Diagram
5. ER Diagram
6. Advantages
7. Screenshots
8. Future Scope
9. Conclusion
10. Reference

# 1. Introduction

---

## RentKaro - Full Stack Application

Everything on Rent is a full-stack web application designed to facilitate a seamless renting experience for users. The platform allows individuals to rent various items, including cars, furniture, and gadgets, from sellers, negotiate prices, and complete transactions securely. The system consists of three primary roles: Buyer, Seller, and Admin.

This project follows a **Microservices Architecture** and adheres to **SOLID principles** and **design patterns** to ensure scalability and maintainability. With features like real-time price negotiation, a secure booking system, and an admin dashboard for managing users and listings, Everything on Rent provides an intuitive and efficient rental marketplace.

### Features :

- User Authentication & Role-based Access (Buyer, Seller, Admin)
- Listing of Rental Items (Cars, Furniture, Gadgets, etc.)
- Booking System for Renting Items
- Secure Payments (UPI, PayPal, Credit/Debit Card)
- Ratings & Reviews for Sellers and Rental Items
- Admin Dashboard for Managing Users and Listings

## Folder Structure :

```
📁 everything-on-rent
├── 📁 backend
│   ├── 📁 user-service
│   ├── 📁 rental-service
│   ├── 📁 negotiation-service
│   ├── 📁 payment-service
│   └── 📁 review-service
├── 📁 frontend
│   ├── 📁 src
│   ├── 📁 components
│   ├── 📁 pages
│   ├── 📁 redux
│   └── 📁 assets
├── 📄 docker-compose.yml
├── 📄 README.md
├── 📄 .gitignore
└── 📄 package.json
```

## 2. Software/Hardware Requirement

---

### Server Requirements:

- **Processor:** Intel Core i5 or equivalent AMD processor
- **RAM:** Minimum 8GB
- **Storage:** SSD storage for better performance
- **Network:** Ethernet or Wi-Fi connectivity
- **Operating System:** Linux (Ubuntu, CentOS) preferred

### Client Requirements:

- **Processor:** Dual-core processor or higher
- **RAM:** Minimum 4GB
- **Storage:** Sufficient for caching and local data
- **Network:** Stable internet connection
- **Browser:** Latest versions of Google Chrome, Mozilla Firefox, or Safari

### 3. Tools and Technologies Used

---

#### Frontend:

- React.js (Dynamic UI)
- TailwindCSS / Material-UI (UI Components)

#### Backend:

- Spring Boot (Microservices Architecture)
- Spring Security + JWT (Authentication & Authorization)
- MySQL (Relational Database for structured data)
- RESTful APIs (Communication between services)

#### DevOps & Deployment:

- Docker (Containerization)
  - Apache Kafka (Cloud Deployment)
  - CI/CD (GitHub Actions)
-



## **Backend Technologies**

**Spring Boot:** Utilized for building the backend services using a microservices architecture, ensuring scalability, maintainability, and efficient API management. It provides built-in support for dependency injection, security, and database integration.

**Spring Data JPA:** Implemented for seamless interaction with MySQL, allowing efficient data storage, retrieval, and management of rental items, user details, bookings, and transactions.

**Spring Security + JWT:** Used to implement secure authentication and authorization, ensuring role-based access for Buyers, Sellers, and Admins. JWT (JSON Web Tokens) is utilized for maintaining session security.

**RESTful Web Services:** Designed to facilitate communication between frontend and backend components using REST APIs, following RESTful principles for efficient and scalable data exchange.

**WebSockets:** Integrated for real-time price negotiations, enabling instant communication between users within the application. This ensures a smooth and interactive experience for buyers and sellers.

**MySQL:** Chosen as the primary relational database to store structured data such as user information, rental listings, bookings, payments, and reviews.

---

## **Frontend Technologies**

**React.js :** Used to develop the frontend of the application, providing a dynamic, responsive, and component-based architecture for smooth user interactions.

**Material-UI / TailwindCSS:** Leveraged to design a visually appealing user interface with pre-built and customizable UI components, enhancing the user experience.

**Axios:** Used for making asynchronous HTTP requests from the frontend to backend services, facilitating seamless API communication and data fetching.

**CSS:** Applied for styling frontend components, enabling rapid customization and improving the overall aesthetics of the application.

---

## **Payment Integration**

**Razorpay / PayPal / UPI:** Integrated as secure payment gateways, allowing users to complete transactions for rental bookings. Ensures smooth, secure, and reliable payment processing.

---

## **Deployment & DevOps**

**Docker:** Utilized for containerizing microservices, ensuring consistency in deployment across different environments.

**Apache Kafka:** Used for cloud deployment, ensuring high availability, scalability, and security of the application.

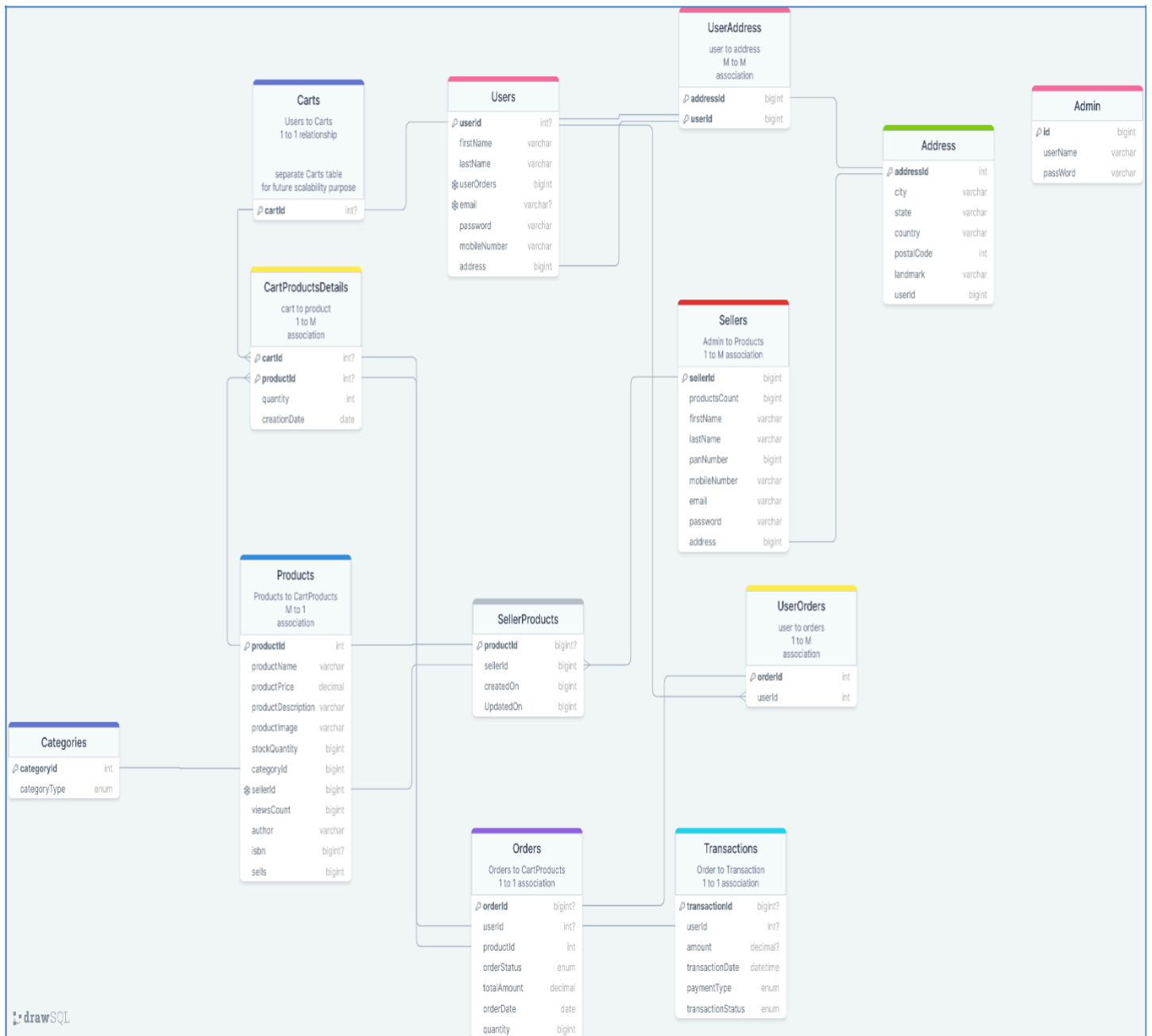
**CI/CD (GitHub Actions ):** Implemented for continuous integration and deployment, automating testing, builds, and deployment workflows.

---

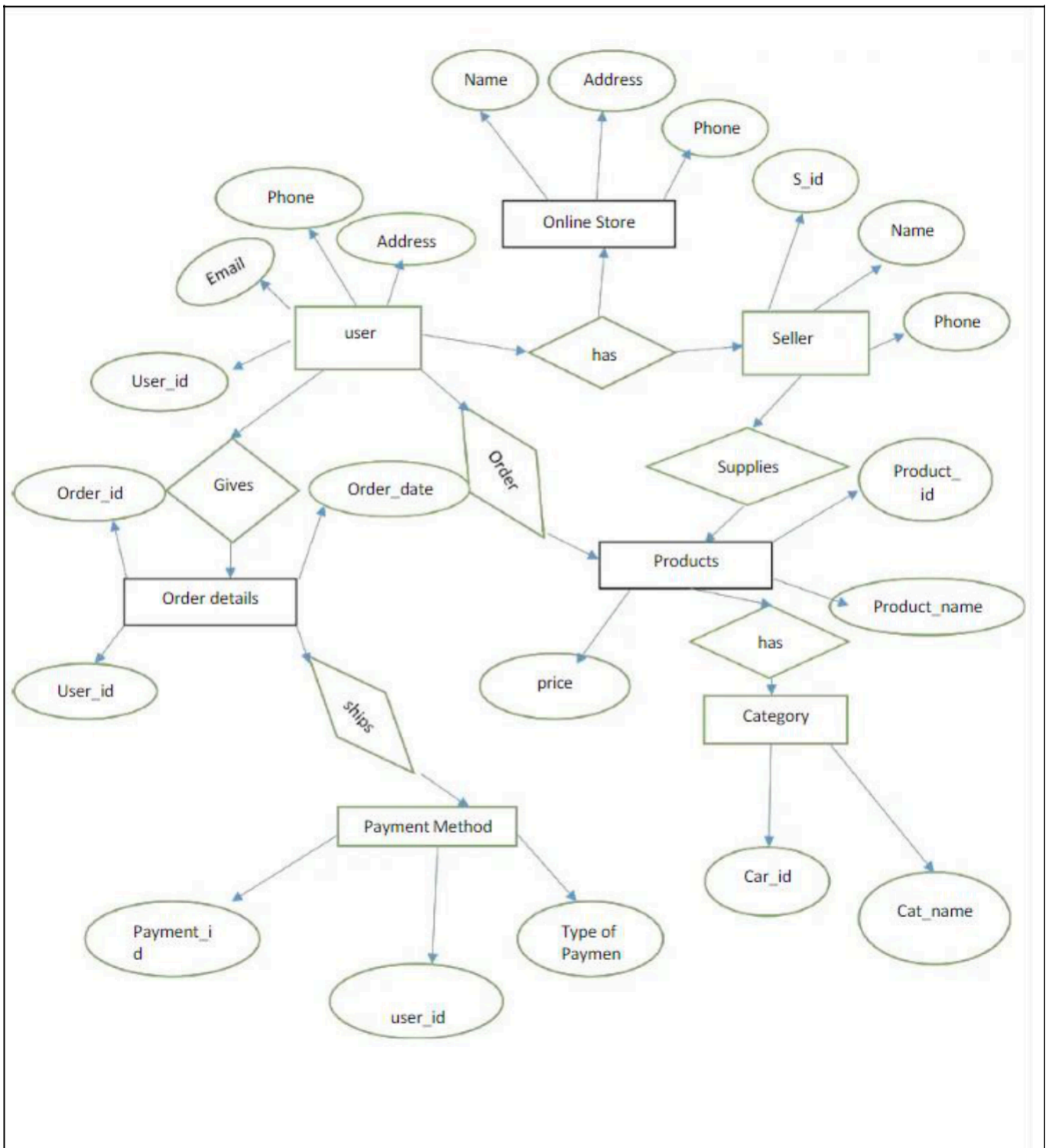
## **Version Control & Collaboration**

**Git:** Used for version control, tracking code changes, and facilitating team collaboration through GitHub repositories.

## 4. Project Database Diagram



## 5.Project E-R(Entity relationship) Diagram



## 6. Advantages

---

### Use of MySQL Cloud Database (Aiven)

The **Everything on Rent** platform leverages the **Aiven cloud-based MySQL database** to provide efficient, secure, and scalable data management.

1. **Reliability** – Aiven ensures high availability, reducing downtime and offering uninterrupted service to users.
2. **Scalability** – The cloud-based model allows seamless scalability as the platform grows.
3. **Managed Services** – Automated backups, maintenance, and monitoring reduce operational complexity.
4. **Security** – Encryption and access control mechanisms safeguard user and rental data.
5. **Automatic Backups** – Ensures data integrity and quick recovery in case of failures.
6. **API Compatibility** – Supports standard MySQL APIs, enabling smooth backend integration.
7. **Developer-Friendly** – Simplified database management makes it easy to configure and optimize performance.
8. **Cost-Effective** – Pay-as-you-go pricing ensures efficient resource utilization without high infrastructure costs.
9. **Data Durability** – Redundant storage enhances data safety against potential losses.

## **Use of JWT for Authorization**

JSON Web Tokens (JWT) provide a **secure and scalable authentication mechanism** for the **Everything on Rent** platform.

**A      Stateless Authentication** – No need for session storage, reducing server load and improving response times.

**B      Enhanced Security** – Digital signatures ensure data integrity, preventing tampering or unauthorized access.

**C      Cross-Domain Compatibility** – Easily transmittable over HTTP headers or URLs, making integration with various frontend and backend technologies seamless.

## 7. Screenshots

### A) User Related Functionalities

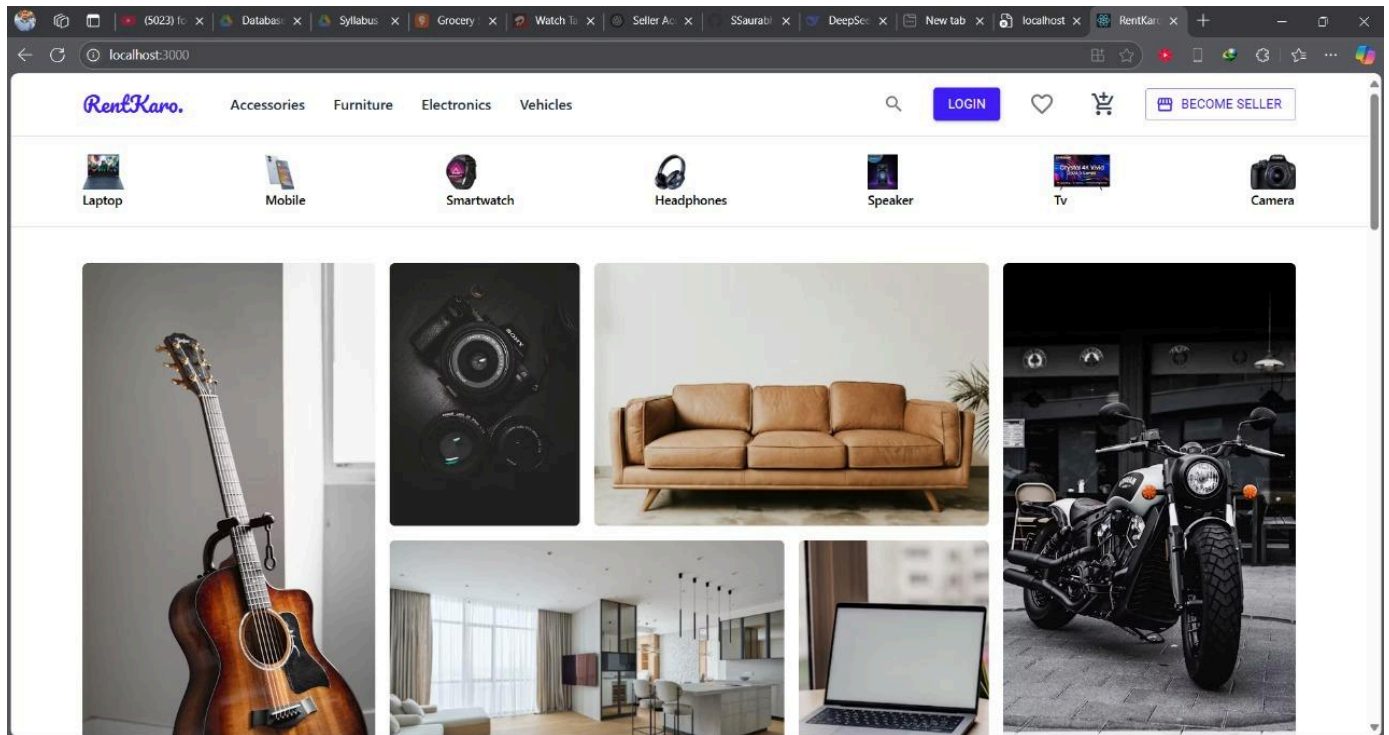
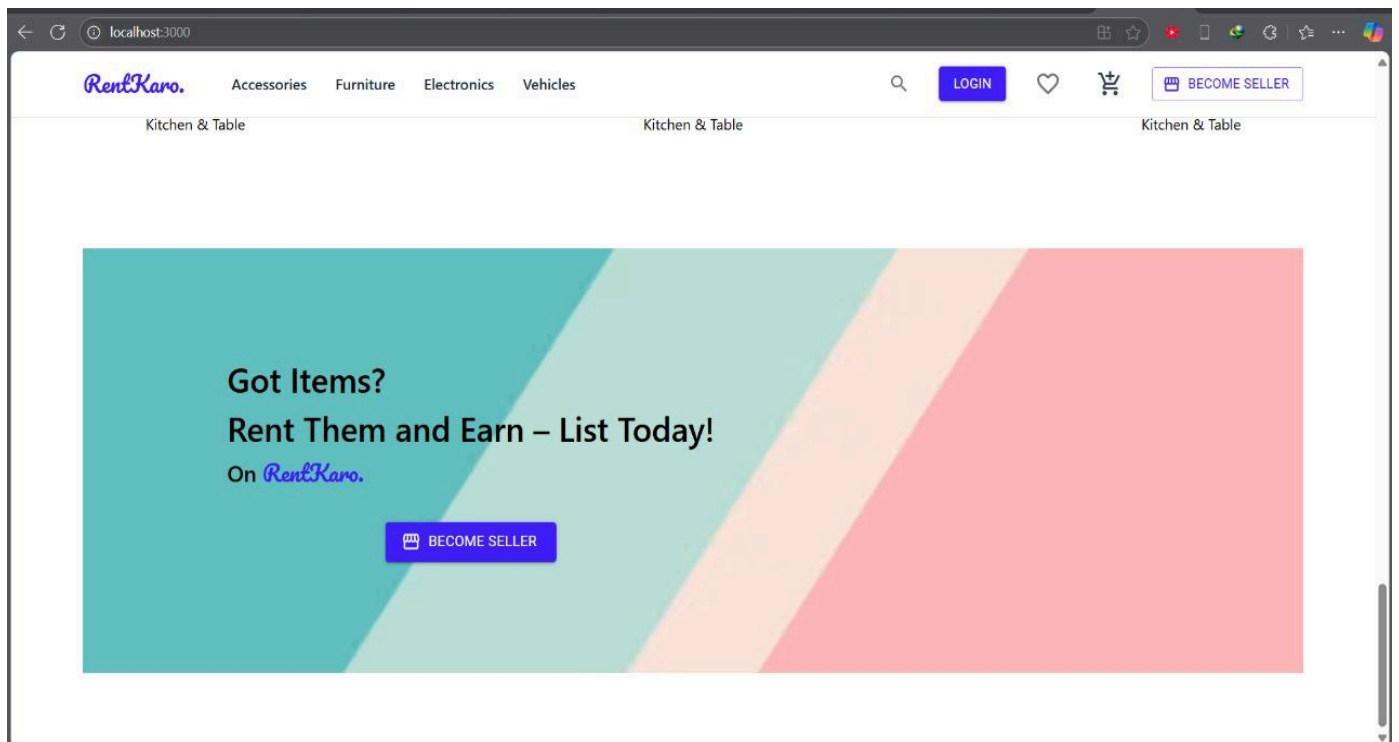


Fig-1: Home Page



localhost:3000/become-seller

RentKaro. Accessories Furniture Electronics Vehicles

LOGIN

BECOME SELLER

### Register as Seller

Email

Password

Username

First Name Last Name

Address

Pincode Locality

City State

REGISTER

### Join the Marketplace Revolution

Boost Your Sales Today




Fig2 – User Registration page



## B) Seller Related Functionalities

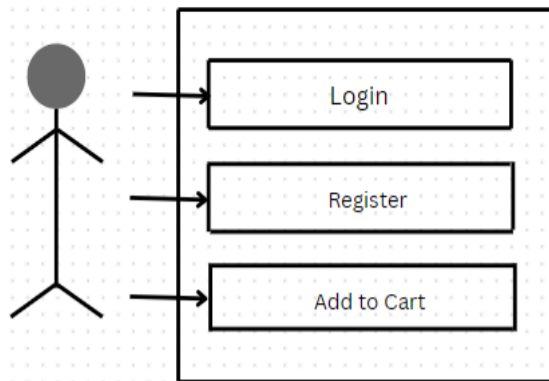
The screenshot shows the 'become-seller' page in a web browser. The browser's address bar displays 'localhost:3000/become-seller'. The website's header includes the 'RentKaro.' logo, navigation links for 'Accessories', 'Furniture', 'Electronics', and 'Vehicles', a search icon, a 'LOGIN' button, a heart icon, a shopping cart icon, and a 'BECOME SELLER' button. The main content area is split into two columns. The left column, titled 'Register as Seller', contains a registration form with the following fields: 'Email', 'Password', 'Username', 'First Name', 'Last Name', 'Address', 'Pincode', 'Locality', 'City', and 'State'. A blue 'REGISTER' button is positioned at the bottom of this form. The right column features a promotional banner with the text 'Join the Marketplace Revolution' and 'Boost Your Sales Today'. Below the text is an illustration of a man in a suit standing on a red location pin, surrounded by various product icons like a guitar, a chair, a car, and a dress, all floating around a large smartphone.

Fig 7 – Seller Registration Page

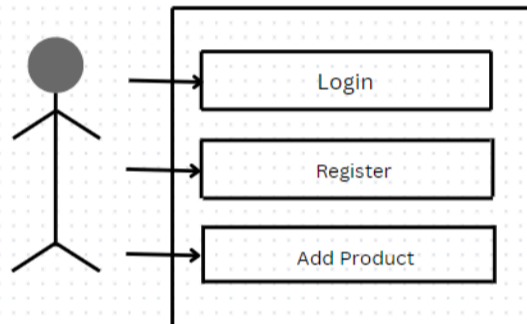
The screenshot shows the 'become-seller' page in a web browser, but with the login form visible. The browser's address bar displays 'localhost:3000/become-seller'. The website's header is identical to the previous screenshot. The main content area is split into two columns. The left column, titled 'Login as Seller', contains a login form with 'Email' and 'Password' fields, a blue 'LOGIN' button, and a link that says 'Don't have an account? REGISTER'. The right column features the same promotional banner as in the registration page, with the text 'Join the Marketplace Revolution' and 'Boost Your Sales Today', and the illustration of a man in a suit standing on a red location pin surrounded by product icons.

Fig 8 – Seller Login Page

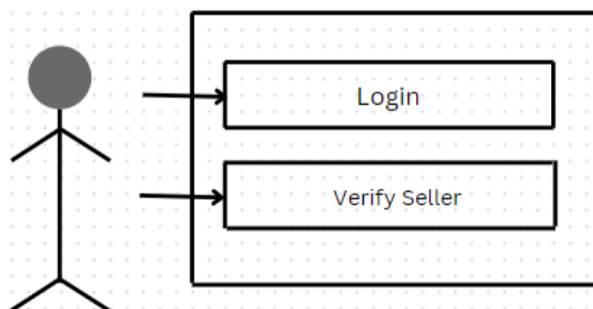
## USER



## SELLER



## ADMIN



## 8. Future Scope :

---

### 1. Subscription-based Model:

- Introduce a premium subscription plan offering exclusive benefits such as discounted rental rates, priority booking, and early access to newly listed rental items.
- Implement tiered subscription levels to cater to different user needs, such as short-term renters, frequent users, or business accounts.

### 2. Mobile App Development:

- Expand the reach of **Everything on Rent** by developing a dedicated mobile application for both **iOS and Android** using **React Native** or **Flutter**.
- Optimize the user interface for mobile devices, ensuring a seamless experience for users to browse, negotiate, and book rentals on the go.

### 3. AI-based Price Recommendations:

- Implement **AI-driven rental price suggestions** based on market trends, demand, and user preferences to help sellers set competitive rates.
- Utilize machine learning algorithms to suggest optimal rental durations and pricing strategies based on historical data.

### 4. Enhanced Personalization & User Experience:

- Introduce **personalized item recommendations** based on users' rental history, location, and preferences.
- Implement **profile-based customization**, where users can set preferences for notifications, favorite rental categories, and frequently rented items.

### 5. Expansion to B2B Rentals:

- Extend the platform beyond individual rentals by introducing a **business-to-business (B2B) rental marketplace** for corporate needs such as office furniture, IT equipment, and heavy machinery.
- Provide **bulk rental options** for businesses requiring long-term rentals with flexible pricing.

## 9. Conclusion

---

In conclusion, the **"Everything on Rent"** project successfully integrates a variety of modern technologies to deliver a seamless and efficient rental marketplace. By leveraging **React.js** for the frontend, **JEE** for the backend for data storage, the platform ensures a robust and scalable solution for users to rent and list items with ease.

The system incorporates **JWT-based authentication** for secure user access, **voice-to-text transcription** for improved accessibility, and an efficient **search and filtering system** to enhance the user experience. With features like **real-time chat, secure payment gateways, and analytics-driven recommendations**, the platform aims to provide a **user-centric, reliable, and scalable** solution for rental services.

Future enhancements such as **mobile app development, AI-driven price recommendations, blockchain-based smart contracts, and personalized rental suggestions** will further elevate the platform's capabilities. With a strong foundation and a vision for continuous innovation, **Everything on Rent** is well-positioned to transform the rental marketplace and offer a more convenient and intelligent way to access rental services. 🚀

## 10. References

---

1. [Spring Boot](#) – Official documentation for Spring Boot framework.
2. [Spring Data JPA](#) – Guide on implementing JPA-based repositories.
3. [RESTful APIs](#) – Reference for building RESTful web services.
4. [MySQL](#) – Official MySQL database documentation.
5. [Spring Web](#) – Spring Web module for handling web requests and responses.
6. [React.js](#) – Official documentation for building UI components.
7. [Node.js](#) – Node.js official website for backend development and APIs.