Final Report - Rental Website

Contents

Title Page

Title: Tool Rental Website Architecture

Student Name: [Name]

Student ID: [ID]

Date: [May 30. 2024]

Abstract

This report presents a detailed architecture for a tool rental website that allows users to rent tools from various categories such as electronics, photography, plumbing etc. The system accommodates three types of users: Managers, Employees, and Customers. Each user type has specific functionalities, including tool management, rental processing, and user administration. The website uses React for the frontend, Node.js for the backend, and MySQL for the database.

Introduction

Background

The increasing demand for online rental services necessitates a robust and scalable architecture to manage tool rentals efficiently. The proposed system aims to streamline the rental process, providing a user-friendly interface for different types of users while ensuring secure transactions and data management.

Motivation

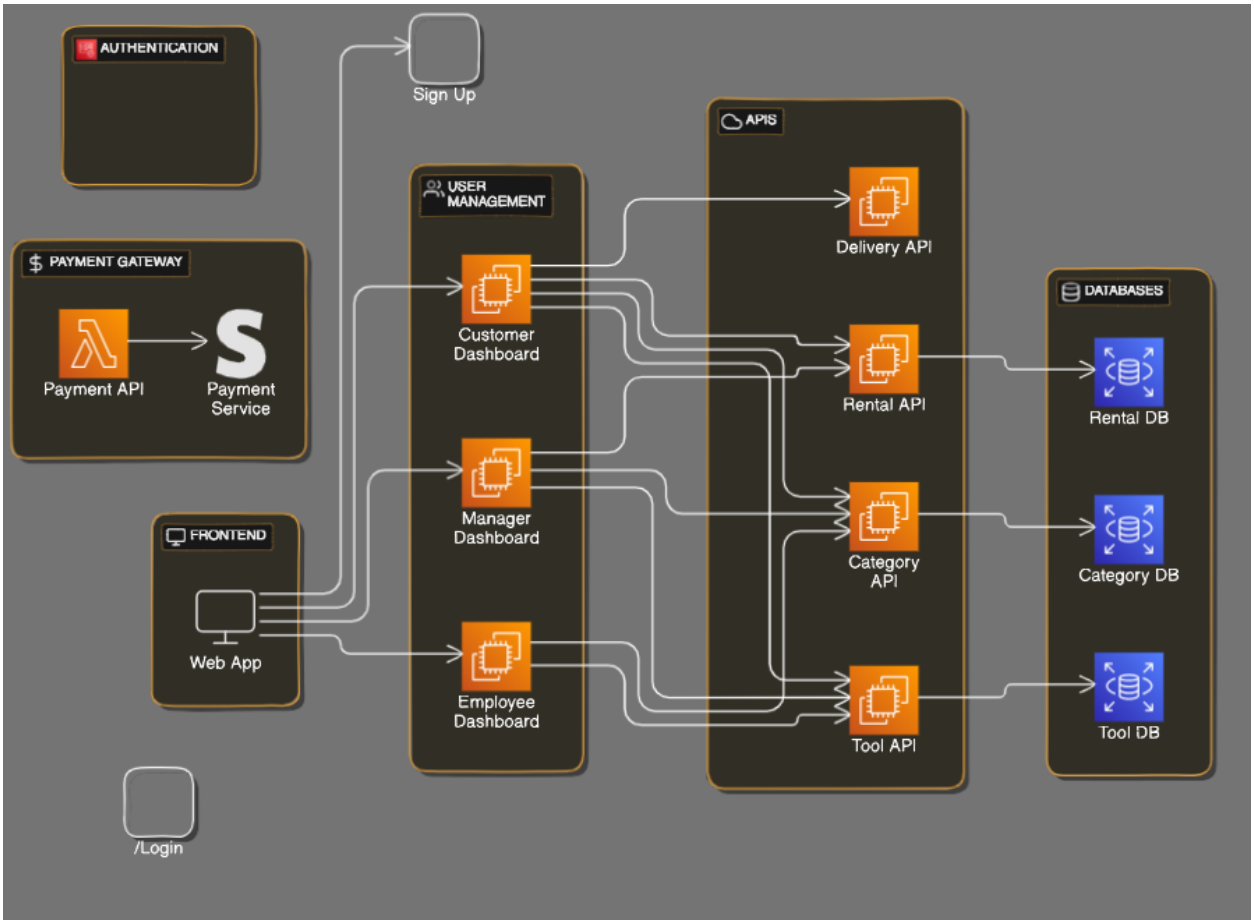
The primary motivation is to create a comprehensive tool rental platform that caters to different user roles with distinct functionalities. This platform will facilitate efficient tool management, rental processing, and secure payment transactions, improving the overall user experience.

Solution

The solution involves developing a tool rental website with a well-defined architecture that includes authentication, user management, various APIs, and a secure payment gateway. The system will ensure scalability, reliability, and ease of maintenance.

System Architecture

Diagrams



The architecture diagram outlines the following components:

1. Authentication
2. Payment Gateway
3. Frontend
4. User Management
5. APIs
6. Databases

Each component and its interactions are described in detail below.

1. Authentication
The authentication module handles user sign-up and login. It ensures secure access to the system by verifying user credentials.

2. Payment Gateway
The payment gateway processes payments through a payment service like Stripe. It integrates with a Payment API for transaction processing.

- **Payment API**: Interfaces with the payment service to handle financial transactions.

3. Frontend
The frontend is a web application built with React. It provides different dashboards based on user roles:

- **Customer Dashboard**: Allows customers to view and rent tools.
- **Manager Dashboard**: Enables managers to manage employees and view rental summaries.
- **Employee Dashboard**: Lets employees register tools and manage categories.

4. User Management
The user management module contains the dashboards mentioned above and is responsible for routing users to their respective interfaces.

5. APIs
The system exposes several APIs for interacting with the databases and handling various operations:

- **Delivery API**: Manages tool delivery and pickup logistics.
- **Rental API**: Handles tool rental transactions and availability.
- **Category API**: Manages tool categories.
- **Tool API**: Handles tool-specific operations such as registration and updates.

6. Databases
The backend uses MySQL databases to store and manage data. There are three main databases:

- **Rental DB**: Stores rental transactions and related data.
- **Category DB**: Contains information about tool categories.
- **Tool DB**: Manages tool inventory data.

Functional and Non-Functional Requirements

Functional Requirements

For the REST API

User Authentication and Authorization

- API endpoints for user sign-up, login, and logout.
- Role-based access control (Manager, Employee, Customer).

Tool Management

- CRUD operations for tools.
- API to fetch tool details by category.
- API to update tool availability status.

Category Management

- CRUD operations for tool categories.
- API to fetch all categories.

Rental Management

- API for initiating and processing tool rentals.
- API for managing rental returns.
- API to fetch rental history for users.

Delivery Management

- API to schedule and manage tool deliveries and pickups.
- API to update delivery status.

Payment Processing

- API to handle payment transactions.
- Integration with the payment service provider.

For the Client Web Application

User Interface

- Responsive design compatible with various devices.
- Separate dashboards for Managers, Employees, and Customers.

Tool Browsing and Renting

- Interface for customers to browse and filter tools by category.
- Tool rental workflow including selection, rental period, and payment.

Employee and Manager Functions

- Interface for employees to add, edit, and delete tools and categories.
- Interface for managers to manage employees and view rental summaries.

Account Management

- User profile management.
- Password reset and update features.

Notifications and Alerts

- Email/SMS notifications for rental confirmations, reminders, and delivery updates.

Non-Functional Requirements

For the REST API

Performance

- The API should handle a high number of concurrent requests efficiently.
- Response times should be within acceptable limits (<200ms for most operations).

Scalability

- The system should scale horizontally to handle increased load.
- Support for auto-scaling based on traffic.

Security

- Data encryption for sensitive information.
- Secure communication using HTTPS.
- Regular security audits and vulnerability assessments.

Reliability

- High availability with minimal downtime.
- Robust error handling and logging mechanisms.

Maintainability

- Well-documented codebase and APIs.
- Modular architecture to facilitate updates and maintenance.

For the Client Web Application

Usability

- Intuitive and user-friendly interface.
- Accessibility features to support users with disabilities.

Performance

- Fast loading times and smooth navigation.
- Efficient data fetching to minimize latency.

Scalability

- Ability to handle a large number of concurrent users.
- Efficient caching mechanisms to reduce server load.

Security

- Protection against common web vulnerabilities (e.g., XSS, CSRF).
- Regular updates and patches to address security issues.

Compatibility

- Cross-browser compatibility.
- Support for various screen sizes and resolutions.

Detailed Design

Frontend:

- **Technology**: React
- **Components**: Separate components for Customer, Manager, and Employee dashboards to ensure modularity and ease of maintenance.
- **Libraries**: MaterialUI for UI components, Axios for HTTP requests, React Router for navigation.

Backend:

- **Technology**: Node.js
- **APIs**: RESTful APIs to handle requests from the frontend and perform CRUD operations on the databases.
- **Libraries**: Express.js for building the APIs, JWT for authentication, Sequelize for ORM.

Databases:

- **MySQL**: Chosen for its reliability and support for complex queries.
- **Schema Design**: Separate tables for Users, Tools, Categories, Rentals, and Deliveries.

Implementation Plan

1. Phase 1: Research and Setup (Weeks 1-3)
    - Research on required technologies.
    - Setup development environments for frontend, backend, and databases.
2. Phase 2: Backend Development (Weeks 4-7)
    - Develop and test the APIs.
    - Integrate MySQL databases.
3. Phase 3: Frontend Development (Weeks 8-11)
    - Develop user dashboards.
    - Ensure responsiveness and user-friendly interface.
4. Phase 4: Integration and Testing (Weeks 12-14)
    - Integrate frontend with backend.
    - Perform thorough testing including unit tests, integration tests, and user acceptance testing.
5. Phase 5: Deployment (Weeks 15-16)
    - Deploy the application on a cloud platform.
    - Monitor and optimize performance.

Challenges and Solutions

1. **Scalability**: Ensuring the system can handle a large number of users and transactions.
    - **Solution**: Use load balancing and optimize database queries.
2. **Security**: Protecting sensitive user information and transaction data.
    - **Solution**: Implement strong encryption and regular security audits.
3. **User Experience**: Providing a seamless and intuitive interface for all user types.
    - **Solution**: Conduct user testing and gather feedback for continuous improvement.

Conclusion

The tool rental website architecture outlined in this report aims to provide a robust and scalable solution for renting tools across various categories. By leveraging modern technologies such as React, Node.js, and MySQL, the system ensures a smooth and efficient user experience for managers, employees, and customers