محمد احمد عبدالواحد الشاذلي

لؤي محمد السيد ابوزيد

محمد أحمد محمد محمد الصوت

Functional Requirements:

1. User Management:

- **User Registration**: Users should be able to create an account by providing their name, email, password, and contact details.
- **User Authentication**: Users should log in using their email and password. Implement multi-factor authentication for added security.
- **Profile Management**: Users should be able to view and update their profile information, including contact details and preferences.
- **Admin Management**: Admins should have additional permissions to manage the system, including user roles and access levels.

2. Product Catalog:

- **Product Listing**: Display a catalog of watches available for sale, including images, descriptions, prices, and specifications.
- **Search and Filter**: Users should be able to search for products and apply filters based on brand, price range, features, etc.
- **Product Details**: Provide detailed information about each product, including high-resolution images, reviews, and ratings.

3. Shopping Cart and Checkout:

- **Add to Cart**: Users should be able to add products to a shopping cart for purchase.
- **View Cart**: Display the contents of the shopping cart, including product details, quantities, and total price.
- **Checkout Process**: Guide users through the checkout process, including entering shipping information, selecting payment methods, and reviewing order details.
- **Payment Integration**: Integrate with payment gateways (e.g., PayPal, Stripe) for secure online transactions, supporting various payment methods including credit/debit cards and digital wallets.
- **Order Confirmation**: Send order confirmation emails to users with order details and estimated delivery times.

4. Order Management:

- **Order Tracking**: Allow users to track the status of their orders from placement to delivery.
- **Order History**: Provide users with a history of their past orders, including details and status.
- **Return and Refund**: Facilitate the process for users to request returns and refunds, including reasons for the return.

5. Maintenance Services:

- **Service Requests**: Allow users to request maintenance services for their watches, including details about the issue and preferred service date.
- **Service Booking**: Redirect users to a booking page where they can schedule an appointment for their maintenance service, selecting the date and time that suits them.
- **Service Tracking**: Enable users to track the status of their service requests and receive notifications about updates.
- **Service History**: Maintain a record of all service requests and completed services for users to review.

6. Inventory Management:

- **Stock Levels**: Track inventory levels for all products, including watches and spare parts.
- **Restocking Alerts**: Notify admins when stock levels are low and require restocking.
- **Supplier Management**: Maintain information about suppliers and manage orders for restocking products.

7. Notifications:

- **Email Notifications**: Send email notifications for account activities, order updates, service requests, and promotional offers.

8. Reporting and Analytics:

- **Sales Reports**: Generate reports on sales performance, including revenue, top-selling products, and customer demographics.
- **Service Reports**: Provide insights into maintenance services, including service types, turnaround times, and customer satisfaction.
- **Inventory Reports**: Monitor inventory levels and identify trends in stock movement and product demand.

Non Functional Requirements:

1. Performance:

- **Response Time**: The system should handle up to 1,000 simultaneous users without significant degradation in performance, with an average page load time not exceeding 2 seconds.
- **Throughput**: The system should process at least 10 transactions per second during peak hours.
- **Load Testing**: Regular load testing should be conducted to ensure the system can handle peak loads without crashing.

2. Security:

- **Data Encryption**: User data, including payment information, must be encrypted using AES-256 both in transit (TLS 1.2 or higher) and at rest.
- **Authentication**: Implement multi-factor authentication (MFA) for user accounts, including both OTP (One-Time Password) and biometric options.
- **Access Control**: Role-based access control (RBAC) to ensure that users only have access to the features and data they are authorized to view or modify.
- **Vulnerability Management**: Conduct regular security audits and vulnerability assessments to identify and mitigate potential security risks.

3. Reliability:

- **Uptime**: The system should have an uptime of 99.9%, ensuring high availability for users.
- **Redundancy**: Implement redundant servers and failover mechanisms to minimize downtime and ensure continuous operation in case of server failure.
- **Backup**: Daily backups should be conducted to prevent data loss, with a retention policy of at least 30 days.

4. Usability:

- **User Interface**: The interface should be intuitive and easy to navigate, requiring no more than three clicks to access any major feature. Use clear and consistent design patterns.
- **Help and Support**: Provide comprehensive documentation, help guides, and a customer support system with chat and email support options.
- **Accessibility**: Ensure the system complies with WCAG 2.1 AA standards to make it accessible to users with disabilities, including screen reader support and keyboard navigation.

5. Scalability:

- **Horizontal Scalability**: The system should be able to scale horizontally to accommodate increased load, allowing for additional servers or services to be added without significant refactoring.
- **Elasticity**: Implement auto-scaling to dynamically adjust resources based on current demand, ensuring optimal performance during peak and off-peak times.

6. Maintainability:

- **Code Quality**: Follow established coding standards and best practices, including meaningful comments and documentation to facilitate maintenance and future development.
- **Modular Design**: Use a modular architecture to allow for easier updates, maintenance, and feature additions.
- **Version Control**: Implement version control using systems like Git to track changes and manage codebase effectively.

7. Compatibility:

- **Browser Compatibility**: The system should be compatible with all major web browsers, including Chrome, Firefox, Safari, and Edge.
- **Responsive Design**: Ensure responsive design for optimal performance and user experience on mobile devices, tablets, and desktops.
- **Integration**: Provide APIs for seamless integration with third-party services like payment gateways, inventory management systems, and customer relationship management (CRM) tools.

8. Availability:

- **24/7 Availability**: The system should be available 24/7, with scheduled maintenance windows communicated to users at least 48 hours in advance.
- **Failover Systems**: Implement automatic failover systems to ensure high availability and minimize downtime in case of failures.

9. Disaster Recovery:

- **Disaster Recovery Plan**: Develop and implement a comprehensive disaster recovery plan that includes regular data backups, off-site storage, and recovery procedures.

- **Recovery Time Objective (RTO)**: Define a maximum acceptable downtime (RTO) of 2 hours in the event of a disaster.
- **Recovery Point Objective (RPO)**: Define a maximum acceptable data loss (RPO) of 1 hour, ensuring data is frequently backed up and recoverable.

10. Legal and Regulatory Compliance:

- **GDPR Compliance**: Ensure the system complies with GDPR regulations for users in the European Union, including user consent management and data protection measures.
- **PCI-DSS Compliance**: For payment processing, ensure the system complies with PCI-DSS standards to protect cardholder data.
- **Audit Logs**: Maintain detailed audit logs of user activities and system events for compliance and forensic purposes.

SYSTEM REQUIRMENTS FOR A WEBSITE FOR SELLING AND FIXING WATCHES

1. Front-End Development - JavaScript Framework : - Example : React, Angular, Vue.js - Use: For building interactive and responsive user interfaces, enabling dynamic

content updates without refreshing the page. ----------

2. Back-End Development - Server-Side Language : - Example : Node.js, Python (Django/Flask), Ruby (Ruby on Rails) - Use : For handling application logic, processing requests, managing databases,

and serving data to the front end. ----------

3. Database Management - Relational or NoSQL Database : - Example : PostgreSQL, MySQL, MongoDB - Use : For storing and managing user data, course content, progress tracking,

and other structured or unstructured information. ----------

4. Hosting and Deployment - Cloud Hosting Service : - Example : AWS (Amazon Web Services), Google Cloud Platform, Heroku - Use : For deploying the application to the internet, ensuring scalability,

reliability, and performance under varying traffic loads. ----------

5. Payment Processing - Payment Gateway : - Example : Stripe, PayPal, Square

- Use : For securely processing online payments, handling transactions, and

managing subscription billing. ----------

6. Security Technologies - Authentication Protocol : - Example : OAuth, JWT (JSON Web Tokens) - Use : For securing user accounts, enabling secure login and authorization, and

protecting sensitive data. ----------

7. Analytics and Monitoring - Analytics Tool : - Example : Google Analytics, Mixpanel, Hotjar - Use : For tracking user behavior, engagement metrics, course completion rates,

and overall system performance. ----------

8. Development Tools - Version Control System : - Example : Git, GitHub, GitLab - Use : For managing code changes, collaborating with team members, and

maintaining a history of project development. ----------

9. Collaboration and Communication Tools - Team Communication Platform : - Example : Slack, Microsoft Teams, Trello - Use : For facilitating team collaboration, project management, and real-time

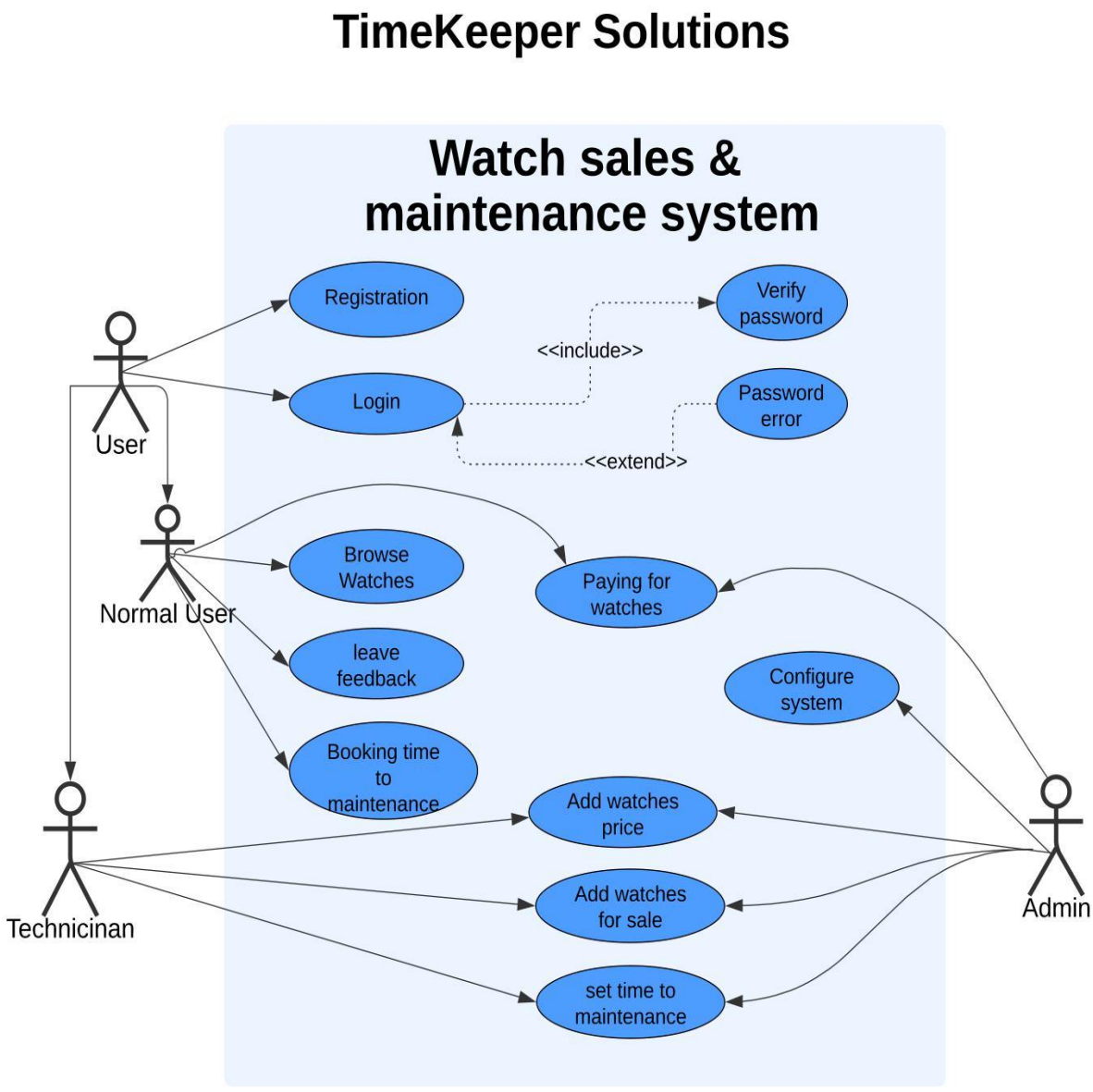
communication among team members. ----------

10. Custom Appointment

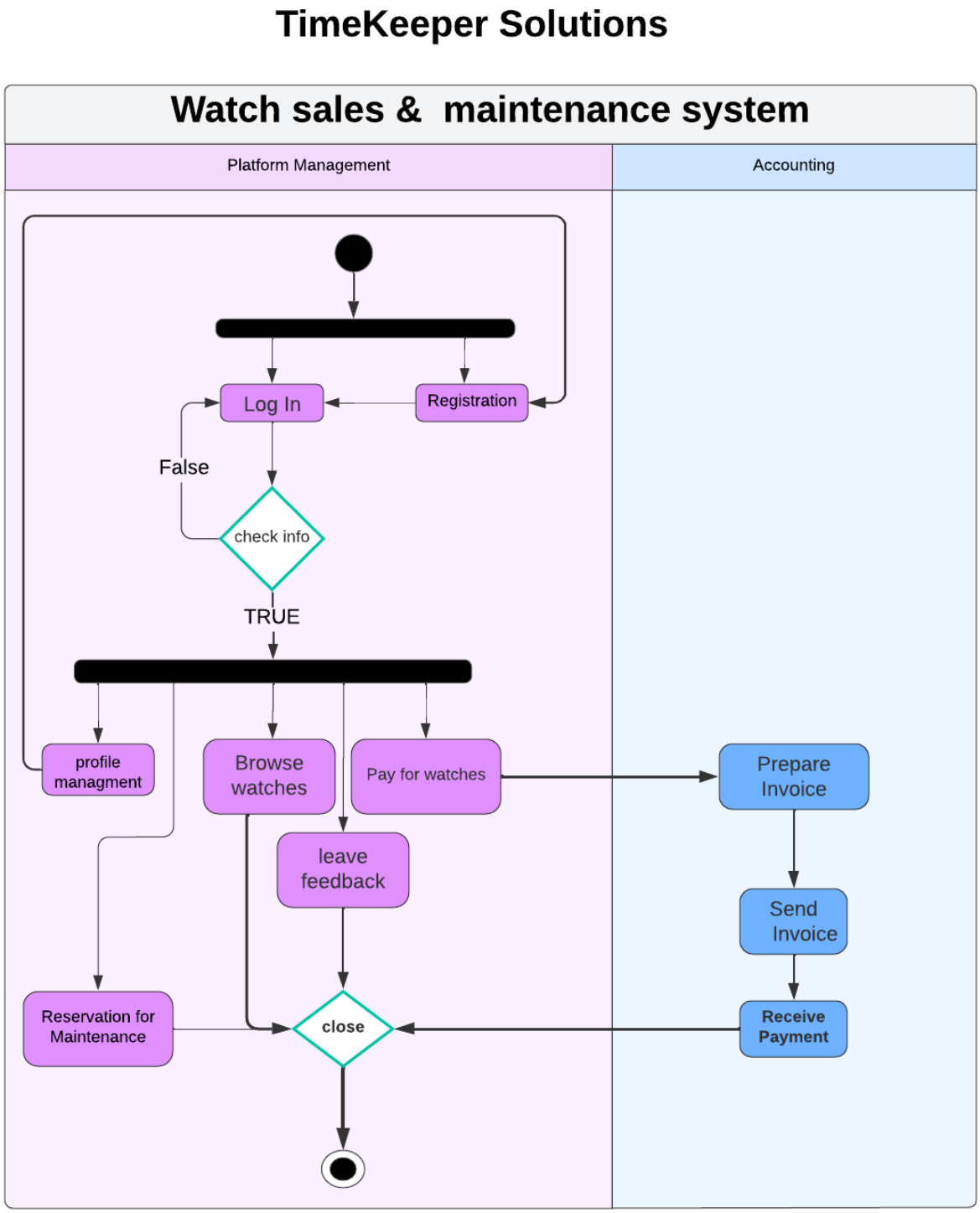
- Example: Custom-built calendar integration (using APIs like Google Calendar API) - Use: You can integrate a calendar directly into your website, allowing users to pick

available slots, book appointments, and sync with your or their personal calendars.

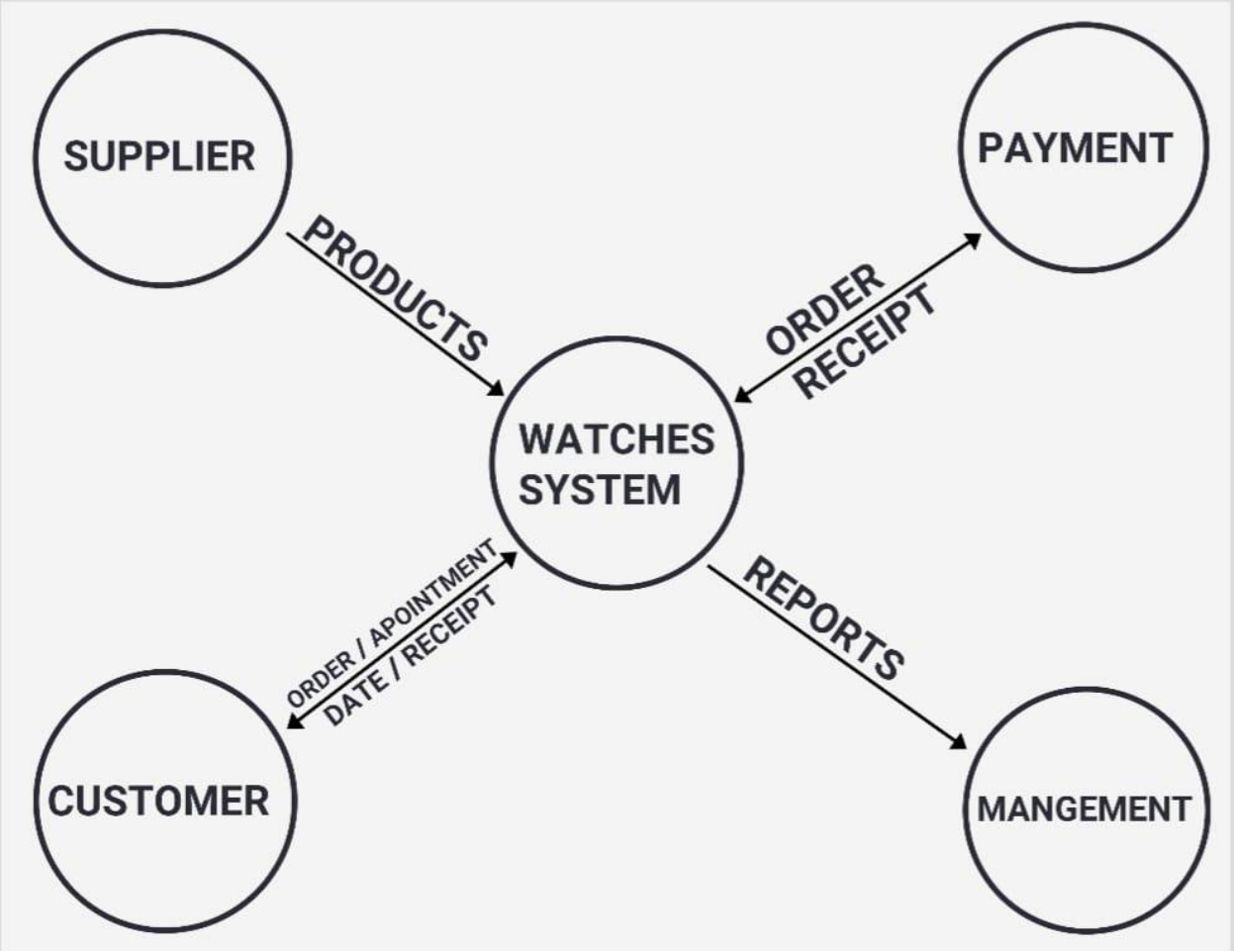
Use case :


TimeKeeper Solutions
Watch sales & maintenance system
Registration
Verify password
<<include>>
Login
Password error
<<extend>>
User
Normal User
Browse Watches
Paying for watches
leave feedback
Configure system
Booking time to maintenance
Add watches price
Add watches for sale
set time to maintenance
Technicinan
Admin

Activity diagram :

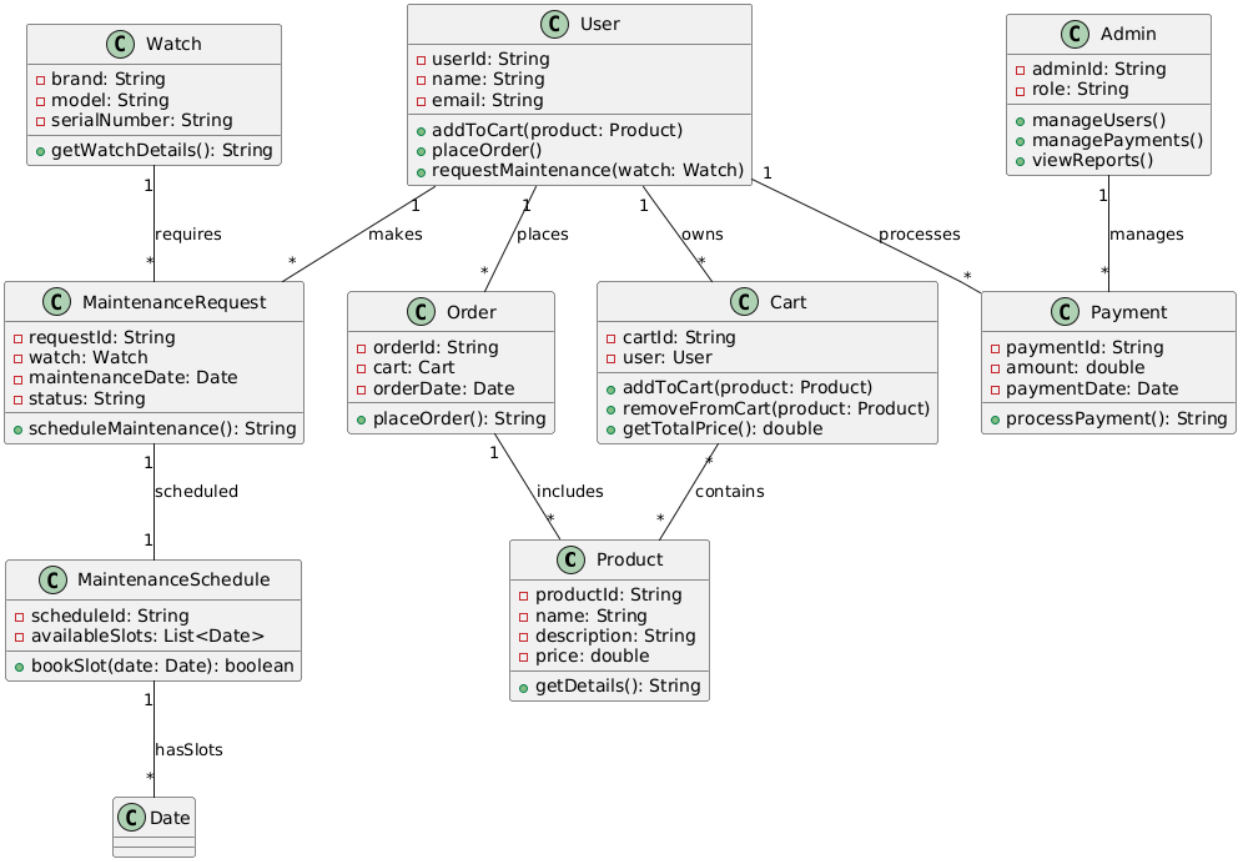


Context diagram

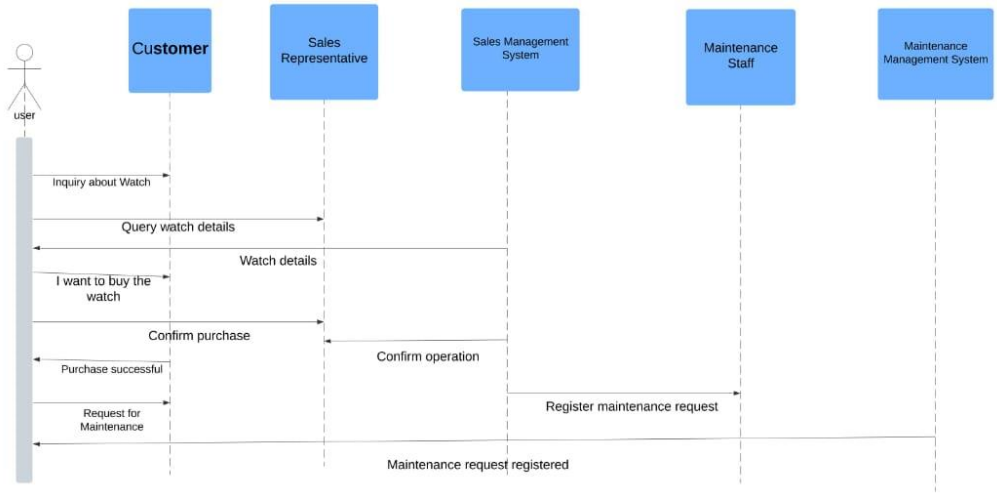


Context diagram

Class diagram :



Sequence diagram :



Sequence diagram :