Infosys springboard

Waste Management System

Team :4
19 NOV,2024

Table of Contents

1. Project Overview

Goal

The Waste Management System aims to create a platform that facilitates efficient waste collection, management, and recycling processes. The platform will allow users to schedule waste pickups, track collection status, and provide information on waste disposal best practices. The project focuses on improving waste management by utilising digital solutions but does not cover on-ground logistics, such as transport or actual disposal services.

Importance

The key goals of this project are:

- Environmental Impact: Reduce waste accumulation by encouraging recycling and responsible disposal.
- Efficiency: Provide a streamlined waste management process for communities and organisations.
- User Awareness: Educate users on waste disposal and recycling practices.

Key People

- Project Leader: [K.Rohithreddy]
- Development Team: [T4 and their roles, e.g., Frontend Developer, Backend Developer, etc.]
- End Users: Community members, waste collection agencies, and recycling centers.
- Stakeholders: Company management, environmental agencies, and local government bodies.

2.Requirements Documentation

Functional Requirements

- **User Registration and Authentication**: Users can register, log in, and manage profiles.
- **Schedule Waste Pickup**: Users can request and schedule waste collection.
- **Tracking Collection Status**: Allows users to check the status of their scheduled pickups.
- **Notifications**: Email or SMS notifications for pickup schedules, delays, and successful pickups.
- **Waste Disposal Information**: Educational resources on waste segregation and recycling.

Quality Standards

- **Performance**: The system should handle multiple user requests efficiently without significant delays.
- **Security**: User data should be protected with encryption and secure authentication protocols.
- **Usability**: User-friendly interfaces and intuitive navigation are essential for adoption and usage.
- **Reliability**: The system should operate with high availability and minimal downtime.

Business Objectives

- **Digital Transformation**: Aligns with the company's goal of digitising traditional processes.
- **Community Engagement**: Supports the company's sustainability and corporate social responsibility (CSR) goals by engaging users in waste reduction initiatives.

User Interaction

- **User Stories**:
    - **As a registered user**, I want to schedule a waste pickup to ensure timely waste disposal.
    - **As a waste collection agency**, I need access to a list of scheduled pickups to optimise collection routes.
    - **As an admin**, I want to monitor user activity and manage scheduled pickups to ensure efficient operation.

3.Project Plan

Timeline and Phases

- **Planning**: - Define requirements and gather resources.
- **Development**: - Begin coding features and setting up the database.
- **Testing**: - Conduct unit and integration testing for functionality and security.
- **Deployment**: - Roll out the system and ensure initial user training.

Resource Requirements

- **Personnel**: Developers, UI/UX designer, QA testers.
- **Equipment**: Development laptops, cloud server (for deployment and storage), monitoring tools.
- **Software**: Django, PostgreSQL/MySQL, AWS/Azure for cloud hosting, Twilio (for SMS notifications), Mailgun (for email notifications).

4.Architecture and Design Documentation

System Architecture

The system follows a client-server architecture with Django as the backend framework. The core components include:

- **Frontend**: HTML/CSS for page layouts, JavaScript for interactive features.
- **Backend**: Django-based application managing requests, database interactions, and business logic.
- **Database**: PostgreSQL/MySQL for storing user information, schedules, and notifications.
- **APIs**: Twilio and Mailgun APIs for sending SMS and email notifications.

System Design

- **User Interface**:
  - **Home Page**: Overview of services and options for waste disposal.
  - **User Dashboard**: For scheduling pickups and tracking requests.
  - **Admin Dashboard**: For managing schedules and monitoring system activity.
- **Diagrams**:
  - **Component Diagram**: Illustrating interaction between frontend, backend, and external APIs.
  - **UML Diagrams**: For user actions like scheduling pickups and receiving notifications.
  - **ER Diagram**: Displaying the database structure and relationships (User, Schedule, Pickup Status, Notification).

Data Storage

- **Database Structure**:
  - **User Table**: Stores user details such as user_id, name, email, and contact_info.
  - **Schedule Table**: Records pickup schedules with fields like schedule_id, user_id, pickup_date, and status.

5.Testing and Quality Assurance

Our testing approach is designed to verify functionality, ensure system reliability, and confirm stakeholder satisfaction. The key testing types and tools include:

1. **Unit Testing:**
    - Testing individual components like user registration, login, and booking modules.
    - Tools: Pytest, unittest.
    - Example: Verify user signup form validations and correct storage in the database.
2. **Integration Testing:**
    - Testing interactions between modules, such as communication between the booking system and the email notification service.
    - Tools: Selenium, Postman.
    - Example: Confirm email notifications are sent after a booking request.
3. **System Testing:**
    - Testing the entire system end-to-end in an environment similar to production.
    - Tools: Selenium for UI tests, load testing tools like Apache JMeter.
    - Example: Simulate multiple users booking requests to test system scalability.
4. **Regression Testing:**
    - Re-testing previously tested functionalities after updates.
    - Example: Verify existing features remain functional after deploying new features.
5. **User Acceptance Testing (UAT):**
    - Involve stakeholders in testing the system against real-world use cases to ensure it meets requirements.
    - Tools: Test scripts and feedback forms.

What makes it complete?

The project will be considered complete when the following conditions are met:

- All defined functionalities are implemented and thoroughly tested.
- UAT feedback is addressed and resolved.
- Performance standards are met under expected loads.
- Deployment is successful in the designated environment.
- Documentation, including user manuals and technical guides, is delivered.

When will we test?

Testing Schedule:

- Unit Testing: During development of each module.
- Integration Testing: After completing development of related modules.
- System Testing: Before deployment in a staging environment.
- UAT: Final phase, before production deployment.
- CI/CD Integration: Automated tests are triggered during every build and deployment.

6.Deployment and Implementation Plan

Where will it live?

- Environment: Cloud-based deployment using AWS for scalability.
- Future growth considerations include database scaling with Amazon RDS and CDN for static assets.

How will we get it there?

- Strategy:
    - Initial deployment on a staging server for testing.
    - Phased rollout for production, starting with a limited region or group of users.

What if something goes wrong?

- Backup Plans:
    - Daily backups of databases and user data.
    - Versioned deployments with rollback capabilities using Docker and Kubernetes.

How will users learn to use it?

- Training and Onboarding:
    - Provide a comprehensive User Manual.
    - Conduct live training sessions for stakeholders.

7.Maintenance and Support

Who's in charge of keeping it running?

- Support Team: A dedicated team will handle ongoing support, including developers, a DevOps engineer, and a customer support representative.

What needs to be done to keep it running?

- Regular software updates.
- Monitor system performance using AWS CloudWatch.
- Address bugs and enhance features based on user feedback.

How will we know what users think?

- Feedback Collection:
    - Use surveys and in-app feedback forms.
    - Analyse user feedback periodically to identify improvement areas.

What are our service commitments?

- SLAs:
    - Critical issues: Resolved within 2 hours.
    - High priority issues: Resolved within 4 hours.
    - General queries: Resolved within 1 business day.

8.Risk Management

What could go wrong?

- Technical Risks:
    - System downtime during deployment.
    - Security vulnerabilities in data handling.
- Operational Risks:
    - Miscommunication with stakeholders leading to incorrect features.
    - Lack of user adoption.
- Financial Risks:
    - Budget overruns during implementation.
    - Higher than expected costs for scaling.

How can we prevent problems?

- Mitigation Strategies:
    - Implement CI/CD pipelines for reliable deployments.
    - Conduct security audits and regular penetration tests.
    - Ensure clear and regular communication with stakeholders.

What's our backup plan?

- Contingency Plan:
    - Maintain a fallback server in case the primary one fails.
    - Ensure daily backups of databases and configurations.

Who's watching for problems?

- Monitoring:
    - Use tools like New Relic and AWS CloudWatch for real-time monitoring.

- Assign a team lead to oversee risk management and ensure timely responses.

9.Security and Privacy

Data Protection

- **Encryption**: Sensitive data (e.g., user details) will be encrypted using SSL for data transmission and AES for storage.
- **Access Controls**: Role-based access control (RBAC) will restrict data access to authorised personnel only.
- **Data Privacy Regulations**: The system adheres to GDPR for user data protection and HIPAA for secure handling of sensitive information where applicable.

Security Measures

- Implementation of **firewalls** and **intrusion detection systems (IDS)**.
- Regular **security audits** to identify and mitigate vulnerabilities.

Incident Response Plan

- Immediate containment measures for detected breaches.
- Notification to stakeholders within 24 hours of incident detection.
- Detailed post-incident analysis and reporting.

10.Legal and Compliance

Licensing

- Ensuring proper licensing for Django, third-party libraries, and database systems used.
- Compliance with software usage licence and agreements.

Regulatory Compliance

- **Local Environmental Regulations**: Adherence to waste disposal standards mandated by regulatory authorities.
- **ISO Standards**: Ensuring compliance with ISO 14001 for environmental management.

Intellectual Property

- Securing copyrights for proprietary code and design assets.
- Monitoring and protection of the system against unauthorised use or duplication.

11.Environmental Impact Assessment

Sustainability

- Promoting waste segregation and recycling to reduce landfill contributions.
- Minimising carbon footprint by optimising vehicle routes and scheduling.

Green IT Practices

- Using energy-efficient servers and virtualized infrastructure to conserve energy.
- Encouraging digital processes to reduce paper usage.

12.User Documentation

User Manuals

- Comprehensive step-by-step guides for system usage.
- Available in multiple formats (PDF, online, and physical copies).

Online Help and Tutorials

- FAQs addressing common issues.
- Video tutorials demonstrating platform functionality.

User Interface Design

- An intuitive dashboard for managing waste requests and tracking services.
- Mobile-first design ensures accessibility across devices.

13.Project Management and Monitoring

Project Planning

- Detailed timelines and task allocation using project management tools like Trello or Jira.
- Clear milestones for development, testing, and deployment.

Risk Management

- Identification of risks (e.g., data breaches, operational delays).
- Developing fallback plans to mitigate these risks effectively.

14.Testing and Quality Assurance

Unit Testing

- Testing individual components, such as forms, APIs, and database operations.

Integration Testing

- Ensuring seamless communication between the system's modules, including user interfaces, databases, and third-party integrations.

Conclusion

The Waste Management System is designed to address the growing need for efficient and environmentally responsible waste disposal through a digital platform. By enabling users to schedule waste pickups, track collection statuses, and receive notifications, the system promotes a streamlined approach to waste management that is accessible and user-friendly. Additionally, the platform contributes to raising awareness about sustainable waste practices by providing educational resources on waste segregation and recycling.

Aligned with the company's commitment to digital transformation and environmental responsibility, this system supports key business objectives while also benefiting community stakeholders. By leveraging Django for a robust backend, integrating secure data storage, and using API services for notifications, the project achieves both technical soundness and user-centered functionality. Future enhancements may include analytics for tracking waste collection efficiency and expanding the platform to support more complex waste management workflows.

Overall, this Waste Management System stands as a scalable, secure, and impactful solution that meets both the immediate needs of users and long-term sustainability goals.