WASTE MANAGEMENT SYSTEM

Project Documentation

Team - 2

1. Project Overview:

What's the Goal?

The primary objective of this project is to develop an automated, digital *Waste Management System* that simplifies and streamlines waste collection processes. By enabling users to easily request, schedule, and track waste pickups, this system aims to enhance efficiency, transparency, and user satisfaction in waste management. Key goals include improving service speed, maintaining clear lines of communication, and providing a user-friendly experience for all stakeholders involved.

Features Included:

- **User Registration and Authentication:** Users can create accounts and securely log in to the platform.
- **Request Scheduling:** Users can submit waste collection requests and specify convenient pickup dates.
- **Real-time Tracking**: Users have the ability to monitor the status of their requests, fostering transparency and peace of mind.
- **Automated Notifications:** The system sends timely updates to users regarding the progress of their requests.
- **Admin Dashboard:** A centralized dashboard for administrators allows efficient management of requests, scheduling, and communication.

Excluded Features:

- **Advanced Analytics:** While useful for analyzing waste generation trends, this feature is beyond the scope of the project.
- **Third-Party Integrations:** Integration with external waste processing facilities is not included in this project.

Why is it Important?

To address existing inefficiencies in waste management, this project focuses on three main objectives:

- **Timely Service:** Automation reduces delays in waste collection by facilitating quick request submission, scheduling, and real-time notifications, ensuring punctual and reliable service.
- **Enhanced Communication:** Real-time tracking and automated notifications keep users informed and build trust, eliminating the need for manual updates from administrators.
- **Efficient Resource Management:** The admin dashboard allows streamlined request handling, effective scheduling, and optimized resource allocation, enhancing the service's capacity to meet demand efficiently.

Who's Involved?

- **Narayanan Sreeraman (Project Leader):** Responsible for overseeing the project's milestones, timelines, and ensuring alignment with the company's goals. Narayan coordinates tasks among team members and ensures project progression.
- **Chintakunta Pranay Teja (Frontend Developer):** Specializes in creating an intuitive, user-friendly interface, designing and implementing features like user registration, request submission, and real-time tracking.
- **Dharsika Sengamalam (Backend Developer and Presenter):** Manages backend functionality, ensuring seamless system performance and efficient data management. Dharsika also develops the admin dashboard, allowing administrators to manage requests effectively, and serves as the presenter to communicate the project's objectives and progress to stakeholders.

This team's combined expertise ensures the Waste Management System will meet its goals, providing a streamlined experience for users and effective management tools for administrators.

---

2. Requirements Documentation

What Does it Need to Do?

The system is designed to fulfill the following core functions, tailored to meet the needs of both users and administrators:

- **User Features:**
    - **Registration & Login:** Users can create an account and securely access the platform.
    - **Request Submission:** Users can submit a waste collection request with relevant details, such as address and preferred pickup time.
    - **Real-time Tracking:** Users can monitor the progress of their requests in real time, from submission to completion.
    - **Cancellation Option:** Users have the flexibility to cancel requests if needed, providing control over their interactions with the service.

- **Admin Features:**
    - **Dashboard Access:** Administrators have access to a secure dashboard where they can view all requests in a consolidated format.
    - **Request Management:** Admins can approve or reject requests based on availability, resources, and scheduling feasibility.
    - **Status Monitoring:** The system provides administrators with tools to update and monitor the status of collection requests in real time.

How Well Does it Need to Work?

To ensure an effective and reliable platform, the following quality standards are prioritized:

- **Performance:** The system must handle simultaneous access by multiple users without performance lags, ensuring smooth operation even under heavy load.
- **Security:** User data must be protected with secure login processes, encrypted storage, and restricted access to sensitive admin functions.
- **Usability:** The interface should be intuitive, allowing users to navigate, submit requests, and track their status with ease.

Why is it Needed?

The Waste Management System aligns with several key business objectives:

- **Improved Service Efficiency:** By automating waste collection processes, the system supports the company's mission to provide prompt, organized, and reliable waste management services.
- **Enhanced User Engagement:** By offering transparent access to waste management services, the platform increases user satisfaction and encourages active engagement with the service.

How Will Users Use It?

User Stories:

- **As a User:** I want to quickly register, log in, submit my waste collection requests, track their status, and cancel requests if needed.
- **As an Admin:** I want to view all incoming requests, approve or reject them based on availability, and monitor each request's status to ensure timely and reliable waste collection.

These user stories illustrate the practical applications of the system for both users and administrators, showcasing its functionality and benefits.

—----------

5. Testing and Quality Assurance

How Will We Ensure It Works?

To ensure the Waste Management System functions as intended, we will implement a comprehensive testing strategy:

- Unit Testing: Each function and module will undergo unit tests to verify individual components. For example, registration, login, and request submission will be tested independently.
- Integration Testing: We'll test interactions between frontend and backend components to ensure smooth data flow and functionality. For instance, testing the interaction between the real-time tracking system and the user interface.
- System Testing: We will perform end-to-end testing on the complete system to validate its performance under various conditions.
- User Acceptance Testing (UAT): Users will test the system to confirm it meets their requirements and is user-friendly.
- Tools: Testing will be managed through GitHub Actions to automate CI/CD pipelines. For bug tracking and issue management, GitHub Issues will be used to log, assign, and resolve issues.

What Makes It Complete?

The project will be deemed complete and ready for stakeholder approval once:

- All core features are implemented and pass unit, integration, and system tests.
- No critical or high-priority bugs remain unresolved.
- The system meets performance, security, and usability standards.
- The application undergoes a successful UAT and stakeholders approve the final deliverable.

When Will We Test?

The testing schedule will be as follows:

- Phase 1: Initial unit tests during the development stage for each feature.
- Phase 2: Integration tests upon completion of the frontend and backend components.
- Phase 3: System testing after integrating all modules.
- Phase 4: User Acceptance Testing before final deployment.

CI/CD will be used to automate testing during each development cycle, ensuring continuous quality control.

What if Something Goes Wrong?

If issues are detected, they will be reported and tracked using GitHub Issues. Each issue will be assigned a priority level and designated to team members (Pranay for frontend issues, Dharshika for backend issues). Narayanan will oversee the resolution process. For analysis, we'll employ A/B testing to test code alternatives, while version history in GitHub will facilitate rollback if necessary.

—----------

6. Deployment and Implementation Plan

Where Will It Live?

The system will be deployed on a cloud platform (such as AWS or PythonAnywhere) to ensure scalability, reliability, and easy access for users. This setup will also facilitate future expansion.

How Will We Get It There?

Deployment will follow a phased rollout strategy:

- Testing Environment: Initial deployment to a testing environment for internal testing and UAT.
- Staging Environment: A staging environment for final testing and quality checks.
- Production Environment: Once all testing is complete, the final deployment to the production environment will allow users to access the platform.

What if Something Goes Wrong?

In the event of deployment issues, a rollback strategy will be implemented. GitHub's version control will allow us to revert to the last stable version if necessary, ensuring minimal downtime. Additionally, backup snapshots will be taken before deployment to restore data if required.

How Will Users Learn to Use It?

A user onboarding process will be provided, including:

- An introductory video tutorial that guides users through the registration, request scheduling, and tracking features.
- Help documentation with step-by-step instructions for using the platform.
- An FAQ section for common questions and troubleshooting tips.

—----------

7. Maintenance and Support

Who's in Charge of Keeping It Running?

Narayanan, as the project leader, will oversee continuous operations, while Pranay (frontend) and Dharsika (backend) will be responsible for ongoing support and maintenance.

What Needs to Be Done to Keep It Running?

Regular maintenance tasks will include:

- Software Updates: Frequent updates to address security vulnerabilities, improve performance, and introduce enhancements.
- Bug Fixes: Ongoing identification and resolution of bugs or issues reported by users.
- Performance Monitoring: Routine performance checks to identify any potential slowdowns or issues and address them proactively.

How Will We Know What Users Think?

User feedback will be gathered through:

- An in-app feedback form for users to submit comments or report issues.
- Regular review sessions to analyze feedback and identify common concerns or improvement areas
- Feedback sessions with key stakeholders and administrators.

What Are Our Service Commitments?

The project will follow defined Service Level Agreements (SLAs):

- Response Time: Critical issues will be acknowledged within 1 hour and non-critical issues within 24 hours.
- Resolution Time: Major issues should be resolved within 48 hours, while minor issues are to be addressed within one week.

—---

8. Risk Management

What Could Go Wrong?

Potential risks include:

- Technical Risks: Possible bugs, crashes, or integration failures due to complex system interactions.
- Operational Risks: Delays in deployment or unavailability of resources.
- Financial Risks: Unexpected costs associated with cloud hosting or maintenance, which may strain the budget.

How Can We Prevent Problems?

To mitigate risks:

- Continuous Testing: Ongoing testing to catch issues early and ensure stability.
- Backup Systems: Regular data backups to minimise data loss in case of failures.
- Budget Monitoring: Periodic budget reviews to ensure costs stay within projections.

What's Our Backup Plan?

In the event of a significant issue, the contingency plan includes:

- Rollback Mechanism: Ability to revert to previous stable versions using GitHub's version history.
- Alternative Workflows: Development of alternative workflows for essential features if issues arise.

Who's Watching for Problems?

- All team members will monitor for issues, with Narayanan taking lead on risk management. Regular status checks and team meetings will ensure prompt identification and resolution of any emerging risks.

—----------

9. Security and Privacy

Data Protection:

To ensure the protection of sensitive user data, the following measures will be implemented:

- Encryption: All user data will be encrypted both in transit and at rest using industry-standard protocols (e.g., HTTPS for transit and AES-256 for storage). Passwords will be hashed using secure algorithms like 'bcrypt' before storage.

- Access Controls: Role-based access control (RBAC) will restrict access to sensitive data. Only authorised personnel (e.g., administrators) can access specific system areas. Multi-factor authentication (MFA) will be implemented for admin accounts to enhance security.

- Compliance with Data Privacy Regulations: The system will align with GDPR requirements, ensuring users have control over their data (e.g., data deletion upon request). Data collection will be limited to only what is necessary, and users will be informed about data usage via a clear privacy policy.

Security Measures:

- Firewalls and Intrusion Detection: Firewalls will be set up on PythonAnywhere to protect the system from unauthorised access. An intrusion detection system (IDS) will monitor network traffic for suspicious activity.

- Regular Security Audits: Periodic audits will be conducted to identify and address vulnerabilities. Tools like OWASP ZAP will be used to scan for common vulnerabilities in the web application.

- Software Updates: Regular patching of third-party libraries and frameworks will prevent exploitation of known vulnerabilities.

Incident Response Plan:

In the event of a security incident,

- Detection and Containment: Monitoring tools will alert the team of unusual activity (e.g., multiple failed login attempts). Affected systems will be isolated immediately to prevent further damage.

- Assessment and Remediation: The nature and scope of the breach will be analysed. Vulnerabilities will be patched, and compromised accounts will be secured.

- Notification and Documentation: Users affected by a breach will be notified promptly. Incidents will be documented, and learnings will be applied to prevent recurrence.

—----------

10. Legal and Compliance

Licensing:

- Software Components: Open-source libraries (e.g., Flask, React) used in the project will comply with their respective licences (e.g., MIT, Apache 2.0). Proper attribution will be provided in line with licensing requirements.

- Hardware/Cloud Services: PythonAnywhere services will be used under a valid subscription, with adherence to GCP's terms and conditions.

Regulatory Compliance:

- GDPR Compliance: Users will have the ability to opt-in to data collection, access their data, and request deletion. A Data Protection Officer (DPO) will oversee GDPR adherence.

- ISO Standards: The project will follow ISO/IEC 27001 for information security management and ensure secure handling of data.

Intellectual Property:

- Ownership: Code and designs developed by the team will be registered under the project's intellectual property rights. Contributions will be documented to ensure proper credit to team members.

- Protection: Code repositories on GitHub will be private, with strict access control. Watermarking and copyright notices will be used to protect documentation and digital assets.

—----------

11. Environmental Impact Assessment

Sustainability:

- Energy Consumption: The project will use PythonAnywhere's energy-efficient data centres, which are powered by renewable energy sources. Servers will be optimised to minimise idle time and reduce energy usage.

- Waste Reduction: The project's operations will avoid unnecessary print materials by adopting fully digital workflows.

Green IT Practices:

- Virtualized Infrastructure: PythonAnywhere's virtualized infrastructure will minimise hardware dependency, reducing electronic waste.

- Sustainable Development Practices: Efficient algorithms will be designed to reduce computational resource usage, ensuring optimal system performance with minimal environmental impact.

—----------

12. User Documentation

User Manuals:

- Comprehensive Guides: Detailed user manuals will cover every feature of the platform, from registration to request tracking and admin dashboard usage. Manuals will include screenshots and step-by-step instructions for clarity.

- Language Support: The manual will be available in multiple languages to cater to a diverse user base.

Online Help and Tutorials:

- FAQs and Tutorials: A dedicated FAQ section will address common user queries. Tutorials (text-based and video) will guide users through platform operations, such as creating and cancelling requests.

- Help Desk Support: A live chat or ticket-based support system will be available for real-time assistance.

User Interface Design:

- Intuitive Navigation: The platform's interface will be designed with usability testing to ensure a smooth user experience. Icons and tooltips will enhance navigation and help users understand system functions.

- Accessibility: The interface will adhere to accessibility standards (e.g., WCAG 2.1) to accommodate users with disabilities.

—----------

13. Project Planning

Roadmap and Phases

- **Planning Phase:** Involves requirements gathering, system design, and initial setup.
- **Development Phase:** Focuses on coding, implementing features, and integrating components.
- **Testing Phase:** Includes quality assurance, resolving bugs, and conducting user acceptance testing.
- **Deployment Phase:** Encompasses the final release and system launch.

Timelines and Milestones

- Define a detailed timeline for each phase, establishing clear schedules and milestones to monitor progress.
- Set achievable goals for task completion while allowing buffer time to manage unforeseen delays.

Resource Allocation

- Assign essential resources such as developers, tools, and infrastructure to each project phase to ensure smooth execution.
- Utilize collaboration tools like Trello, JIRA, or Asana for task assignments, progress tracking, and team coordination.

Risk Management

Identifying Risks

- Perform a risk assessment to pinpoint potential issues, such as:
    - Delayed deadlines or incomplete requirements.
    - Insufficient server capacity or scalability challenges.
    - Security vulnerabilities or non-compliance concerns.

Mitigation Strategies

- Develop contingency plans for identified risks:
    - Implement backup systems to maintain data integrity during failures.
    - Conduct regular progress reviews to address delays or issues promptly.
    - Organize training sessions to address skill gaps within the team.

Ongoing Risk Monitoring

- Continuously evaluate risks throughout the project using assessment tools or regular team review meetings.
- Update mitigation strategies to address new risks or changes in the project scope.

---

14. Testing and Quality Assurance

Unit Testing

- **Focus Areas for Unit Testing:**

    - **Views:** Test individual Django views for proper rendering and functionality, such as:
        - Dashboard views to ensure accurate data display.
        - Task assignment views to confirm correct task management.
    - **Models:** Validate data handling for key models, including:
        - Booking Model: Verify accurate storage and retrieval of booking data.
        - User Model: Ensure proper management of user registration and authentication details.
    - **Utility Functions:** Test utility functions like:

        - Email Notifications: Confirm that emails are triggered correctly for booking status updates.
- **Testing Tools:**
    - Use Django's built-in testing framework for automating view, model, and utility function tests.
    - Perform manual testing for edge cases and complex scenarios.

Integration Testing

- **Focus Areas for Integration Testing:**
    - **Frontend and Backend Interaction:** Verify proper communication between frontend forms and backend APIs.
    - **Database Operations:** Ensure accurate saving and updating of records, such as bookings and administrative tasks.
    - **Email Notifications:** Confirm that status updates trigger email notifications as expected to keep users informed.
- **Testing Tools:**
    - Leverage Django's testing framework for end-to-end testing of component interactions and database operations.
    - Conduct manual integration testing for intricate workflows, validating data flow between frontend and backend systems.
