Plagiarism Scan Report

0% Plagiarism

0% Exact Match

0% Partial Match

100% Unique

Words	924
Characters	6590
Sentences	52
Paragraphs	28
Read Time	5 minute(s)
Speak Time	7 minute(s)

Content Checked For Plagiarism

Chapter 3: Design Methodology
3.1 Proposed System Design
The proposed system is designed to create an efficient and user-friendly interface for managing the process of requesting new assets in an industrial environment. This system leverages a mobile application that enables users to submit asset requests through a structured form, with built-in validation checks to ensure data accuracy. The system comprises two main components: the frontend, which serves as the user interface, and the backend, responsible for data processing and storage.
The frontend is designed to provide an intuitive and responsive user experience, allowing users to submit requests, track their status, and receive updates. The backend is responsible for managing request submissions, approvals, and data storage in a secure and efficient manner. The integration between the frontend and backend ensures seamless communication and data synchronization. Furthermore, the system incorporates features such as real-time data validation, push notifications for request status updates, and user authentication to ensure the system's integrity and reliability.

Figure 3.1: Work flow chart of Request New Asset

Requirement Gathering
This initial and crucial phase of any project workflow focuses on gathering and analyzing the requirements and expectations of stakeholders, such as clients, end-users, and project teams.

Activities in this phase include meetings, interviews, surveys, and reviewing existing documentation. The aim is to outline the project's objectives, establish its scope, and record detailed requirements. A well-defined requirement document acts as the cornerstone of the project, helping to reduce misunderstandings and prevent scope creep.

Design
Once the requirements are clearly outlined, the project moves forward into the design phase.
This step involves translating the requirements into a blueprint for the solution. System architects and designers create detailed plans, including system architecture, database design, user interfaces, and

workflows. Design documents, wireframes, or prototypes are often created to visualize the system. This phase ensures that all components work cohesively, laying a solid groundwork for the development phase.

Development

The development phase is the process where the application's core features and functionalities are constructed through programming.

Developers use the design documents to write the software code, build databases, and integrate APIs or other system components. This phase may include sub-phases such as front-end and back-end development. Developers often follow coding standards, version control practices, and Agile or Waterfall methodologies. Regular code reviews and team collaborations ensure that the development aligns with the design specifications.

Testing

Testing is essential to verify that the product aligns with the defined requirements and performs as expected. This phase involves identifying and fixing bugs, verifying functionality, and validating performance under various conditions. Testing involves several approaches, such as unit testing, integration testing, system testing, and user acceptance testing (UAT). Comprehensive testing ensures the product is robust, reliable, and ready for deployment.

Deployment

The development phase is when the actual implementation and coding are carried out.

This step involves making the product available for use, either by installing it on client systems or deploying it to a live production environment. Deployment may be done in stages, such as pilot launches or phased rollouts, to ensure a smooth transition. Proper planning, including training and support, helps users adapt to the new system seamlessly.

Maintenance

The final phase ensures the longevity and optimal performance of the product. Maintenance activities include monitoring the system, addressing user feedback, fixing bugs, and implementing updates or enhancements. Regular maintenance ensures the system continues to meet user needs, remains secure, and evolves alongside changing business requirements. This phase often extends throughout the product's lifecycle.

Each step is interconnected, and attention to detail at every phase ensures a successful and sustainable project.

3.2 System Architecture

The system architecture follows a client-server model, where the frontend mobile application acts as the client, and the backend, built using PHP and MySQL, functions as the server. The mobile app, developed using Flutter and Dart, communicates with the backend through RESTful API calls. These APIs, developed in PHP, handle various requests such as form submission, data retrieval, and status updates, ensuring smooth interaction between the client and server.

On the backend, MySQL is used as the database management system to store all asset request data, including user information, request details, and approval statuses. The backend logic is structured to handle data processing, security checks, and response generation, ensuring that the system can handle multiple requests simultaneously without compromising performance. The backend also includes validation mechanisms to ensure that only authorized personnel can submit or approve asset requests. The architecture ensures scalability, allowing the system to be adapted to different organizational needs and expanded with additional features as required.

3.3 Tools and Technologies Used

This project utilizes a combination of cutting-edge tools and technologies to build a robust and scalable system. Flutter and Dart are used for developing the frontend of the application, offering a modern and responsive interface across multiple platforms. Gradle is employed as the build automation tool to manage dependencies and ensure smooth project builds in Android Studio, the integrated development environment (IDE) used for coding and debugging. PHP serves as the backend scripting language, responsible for handling API requests and connecting the frontend with the database. The database management system used is MySQL, chosen for its reliability and ability to handle complex data structures required for asset

management.
For development and testing, an Android emulator is used to simulate the app on various Android devices, ensuring compatibility and performance optimization across different screen sizes and hardware specifications. This setup provides a comprehensive development environment that allows for rapid prototyping, testing, and iteration, ensuring that the system performs efficiently and meets user expectations.

Matched Source

No plagiarism found