SDLC

DOCUMENTATION FOR FEEDBACK SYSTEM

Requirements Gathering: In this stage, the project team will gather the requirements for the feedback/suggestion application. This may involve gathering user stories, interviewing stakeholders, and analyzing existing feedback systems. The team will create a detailed specification document that outlines the requirements for the application.

Design: In this stage, the team will create the design for the feedback/suggestion application. This will involve creating a high-level architecture of the system, including the software structure, interfaces, algorithms, and data models. The team will also design the user interface and user experience (UI/UX) for the application.

Implementation: In this stage, the team will develop the feedback/suggestion application. This will involve writing the code, creating the database, and implementing the UI/UX design. The team will also conduct unit testing to ensure that the code is functioning correctly.

Testing: In this stage, the team will test the feedback/suggestion application for functionality, performance, and usability. This will involve creating test cases, executing tests, and reporting and fixing any bugs or issues that are identified. The team will also conduct user acceptance testing (UAT) to ensure that the application meets the requirements and is easy to use.

Requirements Gathering

Purpose and scope:

Purpose of using the Feedback/ Suggestion system is to create an open culture where employees can share their thoughts anonymously or by being known. Problems, Complaints , Questions or Suggestions would be properly addressed with this development.The system allows employees to share their insights and knowledge, and encourages them to contribute to the continuous improvement of the company.

Functional Requirements:

User registration and login: The suggestion system will use the Email and Password. It simplifies the login process for users and enables administrators to track and analyze login activity and system usage.

Submission form: The suggestion system should provide a form for employees to submit their ideas and suggestions, including a title, description(optional), and any relevant attachments(optional).

Review process: The suggestion system will have a process for reviewing submitted suggestions, the admin will review the issue and pass on to the respective department manager. The answer of the department manager will be reviewed by the admin to publish globally.

Tracking and reporting: The suggestion system should have a tracking and reporting mechanism to monitor the progress of each suggestion, including its status and feedback received.

Technical Requirements:

Web-based platform: The suggestion system should be accessible through a web-based platform that allows employees to submit and view suggestions from any location with internet access.

Database: The suggestion system will have two databases-First, For the employee information and Second to store the data related to queries.

User Requirement:

User-friendly interface: The suggestion system should have a user-friendly interface that is easy to navigate, with clear instructions for submitting suggestions.

Accessibility: The suggestion system should be accessible to all employees.

Feedback mechanism: The suggestion system should provide feedback to employees on the status of their suggestions, such as when a suggestion has been accepted, rejected, or under review.

Transparency: The suggestion system should be transparent, with clear guidelines on how suggestions are reviewed and evaluated, and how decisions are made on which suggestions to implement.

Confidentiality: The suggestion system should ensure the confidentiality of employees' suggestions, with appropriate measures in place to protect sensitive or confidential information.

User support: The suggestion system should provide user support to help employees with any issues or questions they may have about the system or the suggestion process.

Data Requirements:

Suggestion data: The suggestion system should store data about each suggestion submitted, including the title, description, date submitted, status, and feedback received.

Workflow data: The suggestion system should store data about the workflow process used to review and evaluate suggestions, including the tasks assigned to reviewers and the status of each suggestion.

DESIGN

Architecture:

Front-end interface: This component is responsible for presenting the user interface that allows employees to submit suggestions. We built it using HTML, CSS, and JavaScript, and used a front-end framework: Bootstrap.

Database: This component provides the persistent storage for suggestion data. Depending on the specific requirements of the system, We implemented using a relational database: SQLite 3.

User Interface and User Experience (UI/UX):

Data Model:

Data Structure: The database model should be designed to store all the data related to the suggestion system, including user data, feedback data, and any other relevant information. The data structure should be optimized for fast access and retrieval of data, with appropriate indexing and normalization to avoid redundancy and improve performance.

Relationships: The database model should reflect the relationships between the different types of data, such as users and their feedback, and feedback categories and subcategories. This can be achieved through the use of foreign keys and joins to establish relationships between tables in the database.

Database Management System: The choice of database management system (DBMS) is critical to the design of the database model. The DBMS should be chosen based on factors such as scalability, reliability, performance, and cost. Common DBMS options for suggestion systems include MySQL, PostgreSQL, and MongoDB.

Security and Data Privacy: The database model should be designed with security and data privacy in mind. This includes measures such as data encryption, access control, and backup and recovery procedures to prevent data loss and ensure the confidentiality and integrity of user data.

System Integration:

Data Integration: Developers must integrate the suggestion system with other data sources, such as user data and feedback data from other systems or databases. This may involve using data transformation tools to ensure that the data is properly formatted and integrated into the suggestion system.

Backend Integration: Developers must integrate the backend components of the suggestion system, such as the business logic and data processing components, with the rest of the system. This may involve using messaging systems or other communication protocols to ensure that the backend components are working together seamlessly.

User Interface (UI) Integration: Developers must integrate the UI components of the suggestion system with the backend components to ensure that they are working together as intended. This may involve using JavaScript frameworks such as React or Angular to build responsive and interactive UI components.

Performance:

: The suggestion system should be designed to handle large volumes of data and user requests. This may involve using techniques such as load balancing, clustering, or sharding to distribute the load across multiple servers.

Caching: Developers should implement caching mechanisms to reduce the number of requests to the database and improve performance. This can be achieved using technologies such as Memcached or Redis to cache frequently accessed data.

Indexing: Developers should create appropriate indexes on the database tables to improve query performance. This can be achieved by analyzing the query patterns and creating indexes on the most frequently queried fields.

Query Optimization: Developers should optimize database queries to ensure that they are efficient and performant. This can be achieved by using techniques such as query profiling, query optimization, and index tuning.

Code Optimization: Developers should optimize the application code to improve performance. This may involve using techniques such as lazy loading, reducing database calls, and optimizing code execution time.

Testing: Developers should thoroughly test the suggestion system to ensure that it is performing as expected. This may involve using load testing tools to simulate a large number of user requests and measure system performance.

Security:

Authentication and authorization: The suggestion system should implement strong user authentication and authorization mechanisms to ensure that only authorized users can access and submit suggestions.

Encryption: All sensitive data, including user login credentials and suggestion data, should be encrypted using strong encryption algorithms to protect it from unauthorized access.

Logging and monitoring: The suggestion system should implement logging and monitoring to detect and respond to security incidents and unauthorized access attempts.

Access control: Access to suggestion data should be limited to authorized personnel only, and access control policies should be implemented to ensure that only authorized users can access and modify suggestion data.

Implementation

TimeFrame:

It should be completed under 2 months of period.

Budget:

Development team size and cost: The size of the development team and their hourly or monthly rate will have a significant impact on the budget. A larger team with more senior developers will generally be more expensive than a smaller team with more junior developers.

Technology and tools: The cost of technology and tools needed to develop the system can vary widely depending on the complexity of the system and the requirements for hardware, software, and licenses.

Resources:

Django and SQLite are both popular choices for creating suggestion systems. Django is a web framework that makes it easy to build and deploy web applications, while SQLite is a lightweight database that is easy to set up and use.

Here are some of the advantages of using Django and SQLite for a suggestion system:

Django is a mature and well-tested framework. It has been used to build many successful websites and web applications, including Instagram, Pinterest, and Disqus. This means that there is a large community of developers who are familiar with Django and can help you if you run into any problems.

SQLite is a fast and efficient database. It is also very lightweight, which makes it a good choice for small to medium-sized suggestion systems.

Django and SQLite are both open source. This means that they are free to use and there are many resources available online to help you get started.

Overall, Django and SQLite are a good choice for creating a suggestion system. They are both mature, well-tested, and open source technologies that can be used to create a fast, efficient, and scalable recommendation system.

Infrastructure:

The cost of the infrastructure required to support the system, such as servers, databases, and networking equipment, can also have a significant impact on the budget.

User training:

Accessing the system: Users will need to know how to access the suggestion/feedback system, which may involve logging in with a username and password or using a single sign-on (SSO) process.

Submitting feedback/suggestions: Users will need to know how to submit feedback/suggestions through the system, including how to navigate the user interface and use any required fields or input forms.

Reviewing feedback/suggestions: Users who are responsible for reviewing feedback/suggestions will need to know how to access and review the feedback/suggestions submitted through the system, including how to filter and sort the data.

Responding to feedback/suggestions: Users who are responsible for responding to feedback/suggestions will need to know how to access and respond to the feedback/suggestions through the system, including how to use any required response templates or workflows.

System administration: Users who are responsible for administering the suggestion/feedback system will need to know how to perform tasks such as user management, configuration management, and system maintenance.

Compliance and data privacy: Users will need to be trained on any relevant compliance and data privacy requirements related to the use of the suggestion/feedback system.

Change management:

Stakeholder communication: Effective communication is crucial to ensure that all stakeholders are aware of the upcoming changes and how they will be affected. Communication can take the form of emails, newsletters, meetings, or other

methods, and should provide clear and concise information about the benefits of the new system and how it will work.

Training: As mentioned earlier, training is essential to ensure that users are able to use the system effectively. The training program should cover all aspects of the system, including how to submit and review feedback/suggestions, how to respond to feedback/suggestions, and how to access reports and analytics.

Resistance management: Some people may be resistant to the new system, either because they are unfamiliar with it or because they are comfortable with the old way of doing things. Resistance can be addressed by providing clear communication about the benefits of the new system, addressing concerns and questions, and involving stakeholders in the implementation process.

Process redesign: The introduction of a suggestion/feedback system may require changes to existing processes, such as how feedback is collected, reviewed, and responded to. It is important to ensure that the new processes are clearly defined and communicated to all stakeholders.

Metrics and reporting: The suggestion/feedback system should be designed to collect relevant data and provide meaningful reports and analytics. This can help to demonstrate the value of the new system and identify areas for improvement.

Continuous improvement: The suggestion/feedback system should be continuously reviewed and improved to ensure that it is meeting the needs of stakeholders and delivering value to the organization. This can involve gathering feedback from users, monitoring system usage, and conducting periodic reviews and assessments**.**

Testing

Unit Testing:

Input validation: Ensure that the system validates user inputs, such as text and data formats, to prevent unexpected behavior or errors.

Boundary testing: Test the system's response to extreme values, such as large amounts of data or user inputs, to ensure the system can handle such scenarios without crashing or producing incorrect results.

Error handling: Test the system's response to errors, such as invalid inputs or missing data, to ensure it produces appropriate error messages or exceptions.