Overview

A design document serves as a comprehensive plan for system development. It outlines the system's functionality, architecture, user interfaces, and other key aspects. To complete your assignment for ISYS3416 – Software Engineering Fundamentals, specifically the Software Design Document for Assessment 3, which involves a series of steps and considerations. Here's an overview to guide you based on design proposal template document

Guideline for writing a design document of the functionality described for The Music Emoji App

Project: Music Emoji App
Project Description: Music can change the mood of the listener however there are some days that you feel low or happy but don't know what music is suitable for your mood. This project will help you choose music based on your mood. This project aims to create a mobile app that will suggest music to play based on the user's mood. Using face recognition, the system will analyze the image and suggest music according to the mood shown in the image. This will also analyze the trend of the moods to give suggestions or tips on improving good mood and health.

1. Non-Functional Requirements

Capturing non-functional requirements in customer-friendly language is a process of translating technical specifications into terms that emphasize the user experience and impact. Here's a step-by-step guide:

Step 1: Understand the Concept of Non-functional Requirements

- Define Non-functional Requirements: Understand that non-functional requirements describe how the system works, rather than what it does. They typically cover areas such as performance, security, usability, reliability, and maintainability.
- Recognize the Importance: Acknowledge that non-functional requirements are critical for customer satisfaction and overall system quality.

Step 2: Identify and Categorize Non-functional Requirements

- List Non-functional Requirements: Start by listing all non-functional requirements for your project. Common categories include:

    - Performance (speed, response time, throughput)
    - Reliability (uptime, error rate)
    - Usability (user interface design, user experience)
    - Security (data protection, authentication)
    - Scalability (handling load increase)
    - Maintainability (ease of updates, troubleshooting)
- Categorize Based on User Impact: Group these requirements based on how they affect the user experience. For example, performance and usability will directly impact how the user interacts with the app.

Step 3: Draft and Validate

- Write Draft Descriptions: Create descriptions for each non-functional requirement using the customer-friendly language developed in the previous steps.
- Validation: Have non-technical stakeholders review the descriptions to ensure they are understandable and effectively communicate the intended message.

Example

- **Performance**
    - The app will be fast and responsive. When a user requests a song based on their mood, the app will display suggestions within a few seconds, ensuring a smooth and enjoyable experience.
- **Usability**
    - The app will be user-friendly, with intuitive navigation and a clear layout, making it easy for users of all ages and technical backgrounds to enjoy its features without confusion or frustration.
- **Reliability**
    - The app will function reliably, with minimal downtime or errors. Whether the user is accessing their mood history, listening to music, or exploring new features, they can expect a consistently stable experience.
- **Scalability**
    - As the number of users grows, the app will effortlessly handle increased traffic and data. This ensures that the app's performance remains stable and reliable, even during peak usage times.
- **Security**
    - User data, especially sensitive facial recognition information and music preferences will be protected with robust security measures. This ensures that users' personal information is safe and not susceptible to unauthorized access.

- **Compatibility**
    - The app will be compatible with a wide range of smartphones and tablets, ensuring that as many users as possible can enjoy it, regardless of their device.
- **Maintainability**
    - The app will be designed for easy updates and maintenance, ensuring that it stays up-to-date with the latest features and security standards, offering an ever-improving experience to its users.
- **Data Privacy and Compliance**
    - User privacy will be a top priority. The app will only use facial recognition data for mood detection and not store any personal images. Users will have control over their data and can choose what information is saved and for how long.

1.1. User Interface (individual)

Creating a section on the User Interface (UI) for your Software Design Document (SDD) based on your app's use cases involves several steps from conceptualizing the design to justifying your choices. Here's a step-by-step guide:

Step 1: Understand the Use Cases

Review Use Cases: Start by thoroughly reviewing the use cases you have written in your SRS. Understand the user tasks and goals that the UI must support.
Identify Key Interactions: Note the primary interactions that the user will have with your app based on these use cases.

Step 2: Conceptualize the Design

Sketch Ideas: Begin with rough sketches on paper to explore different layouts and arrangements for the UI elements that will support the use cases.
Consider UI Principles: Apply principles of good UI design, such as consistency, simplicity, and user feedback, to ensure the interface is intuitive and easy to use.

Step 3: Create Mockups

Select a Tool: Choose a mockup tool that you are comfortable with, such as Balsamiq, Adobe XD, Sketch, or even PowerPoint.

Develop Mockups: Create more refined UI mockups that represent the "look and feel" of the app. Ensure that the design aligns with the functionality required by the use cases.

Step 4: Generate Screenshots

Capture Screenshots: Once your mockups are ready, take screenshots of each key interface.
Label Screenshots: Give each UI screenshot a clear title that relates to its function or the use case it supports.

Step 5: Provide Descriptions and Explanations

Write Descriptions: For each screenshot, write a description that explains what the user can do on that screen and how it supports the use case.
Explain Design Choices: Justify your design decisions by explaining why the layout, colors, fonts, and controls were selected in terms of usability and user experience.

Step 6: Name Each User Interface

Assign Names: Give each distinct UI screen a descriptive name that captures its essence, such as "Mood Detection Home Screen" or "Music Recommendation Playlist View."
Reference Use Cases: Relate each named UI back to the specific use cases they are designed to support.

Step 7: Document in the SDD

Insert into Document: Place the screenshots, titles, descriptions, and explanations into the SDD under the "User Interface (Individual)" section.
Organize Logically: Arrange the UI elements in a logical order, following the user's journey through the app.

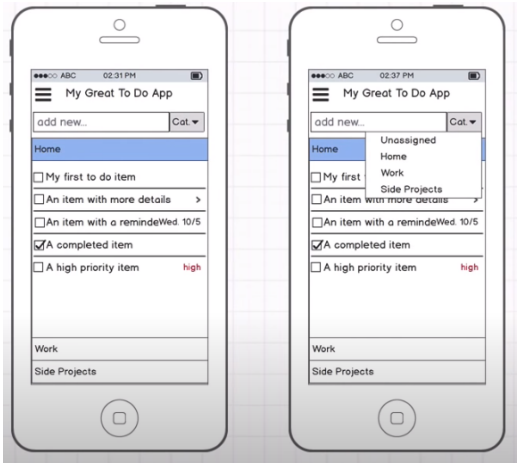
Step 8: Review and Revise

Peer Review: Have team members or peers review this section to get feedback on the clarity of your explanations and the effectiveness of your designs.
Revise: Make any necessary revisions based on the feedback to improve both the UI design and the explanations.

Step 9: Finalize

Proofread: Ensure that the text is free of errors and that the images are clear and properly formatted.
Consistency Check: Verify that the naming convention and style are consistent throughout the document.

Example:



>

1.2. Performance

When documenting the performance section of your Software Design Document (SDD), you'll need to describe how the system is expected to perform regarding certain non-functional criteria. Here's a step-by-step guide

Step 1: Understand the Performance Criteria

Define Each Criterion:

- Availability: The proportion of time the system is operational and accessible.
- Response Time: The time taken for the system to react to a given input.
- Security: The measures in place to protect against unauthorized access or alterations.
- Mobility: The ability of the system to be used across various mobile devices and platforms.
- Maintainability: The ease with which the system can be updated, modified, or fixed.