



INSTITUTE OF TECHNOLOGY
COMPUTER SCIENCE AND ENGINEERING

Title :- SRS Document Generator

Group Member:-

Henil Prajapati(23BCE530)

Megh Shah(23BCE536)

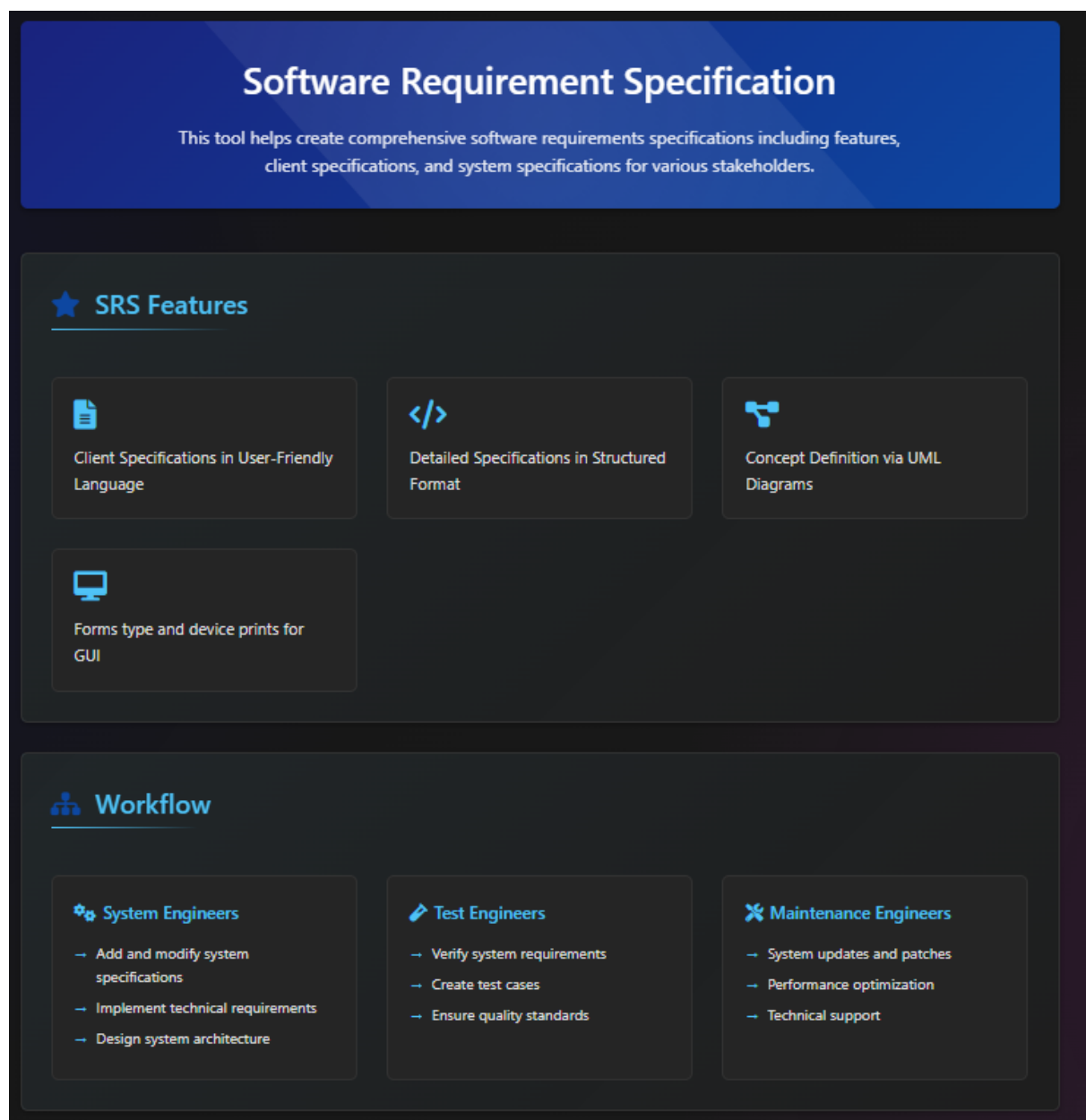
Software Engineering Innovative Assignments

Submitted to: Dr. Anuja Nair

Project Report: SRS_Generator

1. Introduction

The SRS_Generator is a web-based application designed to assist software developers and project managers in creating detailed Software Requirements Specifications (SRS) and UML diagrams efficiently. This tool automates the generation of SRS documents and visual representations, reducing manual effort and ensuring consistency with industry standards (e.g., IEEE 29148). The project aims to streamline the requirements engineering process, making it accessible to both technical and non-technical stakeholders.



2. Objectives

Automate the creation of SRS documents based on user inputs.

Generate UML diagrams (Use Case, Activity, Class) to visualize system architecture.

Provide a user-friendly interface for inputting project details and retrieving outputs.

Integrate external APIs for enhanced functionality and research support.

Enable downloadable SRS documents in a standard format (e.g., .docx).

3. Technology Used

The SRS_Generator leverages a combination of modern technologies to achieve its goals:

Frontend:

HTML5: For structuring the graphical user interface.

CSS3: For styling with animations and responsive design.

JavaScript: For client-side logic and dynamic interactions.

Backend:

Python 3.8+: Core programming language for server-side logic.

Flask: Lightweight web framework for building the server and rendering templates.

Dependencies:

[tavily-python](#): For web search integration to gather relevant data.

[groq](#): For AI-driven sentiment analysis and SRS content generation.

[langchain](#): For memory management and context retention.

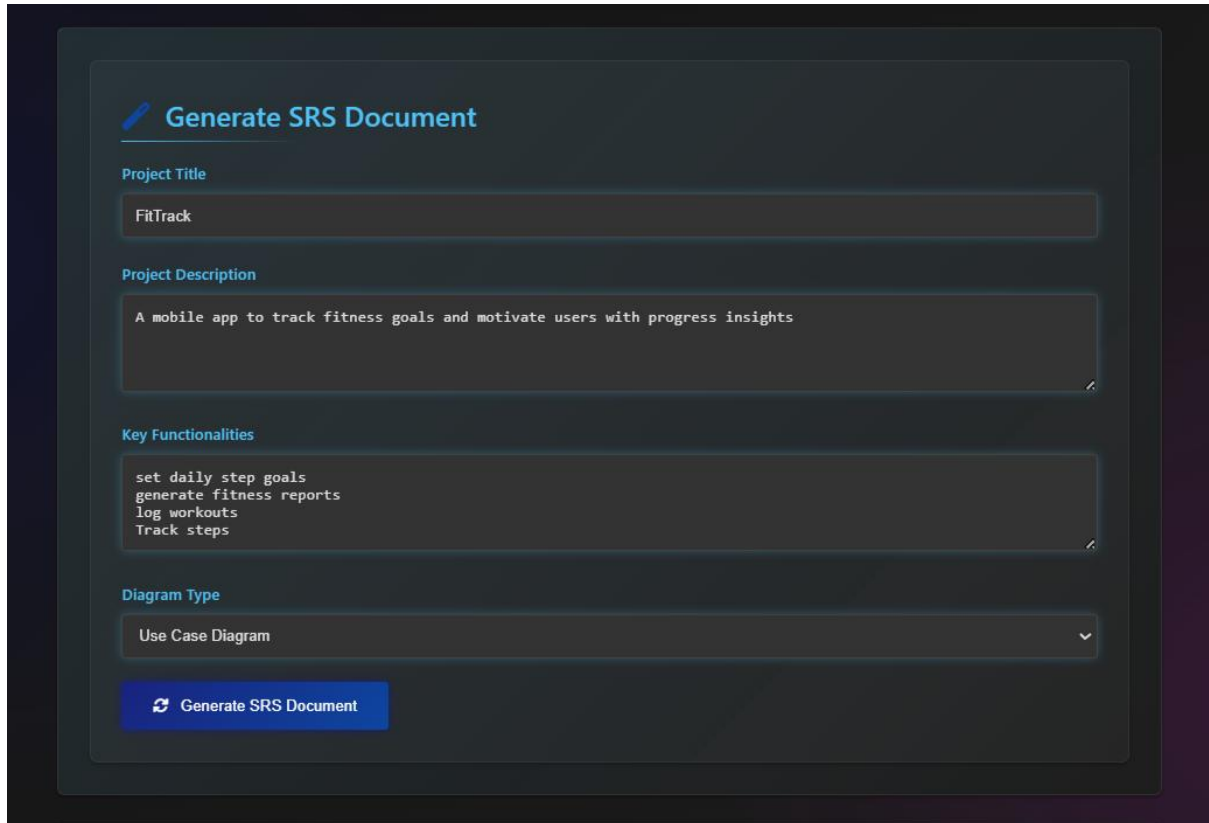
[python-docx](#): For generating .docx SRS documents.

[PlantUML](#): For UML diagram generation using the plantuml.jar file, requiring Java.

Development Environment:

Virtual environment for dependency management.

PowerShell (Windows) for execution and testing.



The screenshot shows a web application titled "Generate SRS Document". It features four input fields: "Project Title" with the value "FitTrack", "Project Description" with the text "A mobile app to track fitness goals and motivate users with progress insights", "Key Functionalities" with a list of tasks including "set daily step goals", "generate fitness reports", "log workouts", and "Track steps", and "Diagram Type" with a dropdown menu set to "Use Case Diagram". A blue button labeled "Generate SRS Document" is positioned at the bottom of the form.

4. Approach

The development of SRS_Generator followed an iterative and incremental approach:

Requirement Analysis: Identified user needs for SRS and diagram generation, focusing on input flexibility and output quality.

Design:

Designed a two-column GUI layout with input fields, accordions, and a results display.

Planned integration of AI APIs (Groq, Tavily) and PlantUML for diagrams.

Implementation:

Developed the backend using Flask to handle API calls and document generation.

Built the frontend with HTML/CSS/JavaScript for an interactive interface.

Integrated external libraries for SRS content generation and diagram rendering.

Testing: Conducted unit and integration tests to ensure functionality (detailed in Section 7).

Deployment: Ran the application locally on `http://127.0.0.1:5000` for demonstration and submission.

The approach emphasized modularity, allowing easy updates to individual components (e.g., adding new diagram types).

1. Introduction

1.1 Purpose

The purpose of the FitTrack app is to provide a comprehensive platform for users to track their fitness goals, monitor progress, and receive motivational insights to achieve a healthier lifestyle. The app aims to benefit its target audience by offering a user-friendly interface to set daily step goals, log workouts, and generate fitness reports, thereby fostering a sense of accomplishment and encouraging continuous improvement. The target audience includes individuals seeking to adopt a more active lifestyle, fitness enthusiasts, and athletes looking to optimize their performance. By leveraging the app's functionalities, users can develop healthy habits, reduce the risk of chronic diseases, and enhance their overall well-being.

1.2 Scope

The scope of the FitTrack app encompasses the development of a mobile application that enables users to track their daily step count, log workouts, and generate detailed fitness reports. The app will allow users to set personalized daily step goals, monitor their progress, and receive motivational messages to help them stay on track. The boundaries of the app include integration with wearable devices and health trackers to collect data on user activity, while exclusions include features such as personalized coaching, meal planning, and social sharing, which may be considered in future updates. Examples of the app's capabilities include tracking steps taken, distance covered, and calories burned, as well as providing insights into user progress over time.

1.3 Definitions

Key terms related to the FitTrack app include 'fitness goals,' which refer to specific objectives that users aim to achieve through regular exercise and physical activity; 'workouts,' which encompass various forms of exercise, such as running, swimming, or weightlifting; and 'fitness reports,' which provide a detailed summary of user progress over a specified period. Other important terms include 'step count,' 'calories burned,' and 'distance covered,' which are used to measure user activity and progress. These definitions are essential to understanding the app's functionality and user interface.

1.4 References

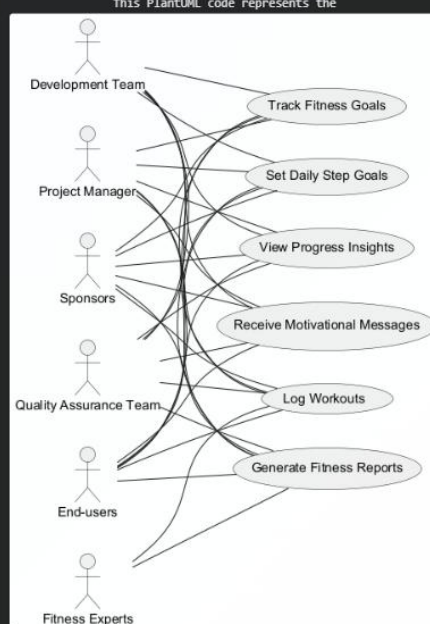
TBD - References will be updated in future versions of this document to include relevant sources and citations.

2. Overall Description

2.1 Product Perspective

The FitTrack app is situated within the broader context of mobile health and fitness applications, offering a unique combination of features that cater to the diverse needs of its target audience. The app's ecosystem includes integration with wearable devices, health trackers, and other fitness apps, enabling users to access a comprehensive range of tools and services to support their fitness journey. The app's novelty lies in its user-friendly interface, personalized motivational messages, and detailed fitness reports, which set it apart from existing fitness apps and provide a compelling value proposition for users.

Note: The `..` symbol is used to indicate the relationship between the actors and the use cases. The `as` keyword is used to assign an alias to the actors and use cases.



5. SRS Document: Purpose and Importance

5.1 What is an SRS Document?

An SRS (Software Requirements Specification) document is a formal document that outlines the functional and non-functional requirements of a software system. It serves as a blueprint for developers, testers, and stakeholders, detailing what the system should do, how it should perform, and any constraints.

5.2 Why SRS is Required

Clarity and Communication: Provides a single source of truth, reducing misunderstandings between stakeholders and developers.

Requirement Validation: Ensures all requirements are documented, testable, and traceable.

Project Planning: Facilitates accurate estimation of time, cost, and resources.

Risk Mitigation: Identifies potential issues (e.g., missing features) early in the development cycle.

Quality Assurance: Serves as a baseline for testing and validation against user expectations. In SRS_Generator, the SRS document is dynamically generated,

incorporating sections like Introduction, Specific Requirements, Stakeholder Analysis, and Risk Analysis, adhering to IEEE 29148 standards.

6. Features of SRS_Generator

Input Interface: Users can enter project name, description, functionalities, and select diagram types.

SRS Generation: Automates creation of detailed SRS with narrative-style descriptions.

UML Diagram Support: Generates Use Case, Activity, and Class diagrams using PlantUML.

API Integration: Utilizes Tavily for web searches and Groq for AI-driven insights.

Document Export: Allows downloading the SRS as a .docx file.

Real-Time Feedback: Displays processing status and results in an interactive GUI.

7. Accuracy and Validation

7.1 Methods Used to Check Accuracy

Manual Review: Cross-checked generated SRS content against input data for correctness (e.g., matching features to requirements).

Unit Testing: Tested individual functions (e.g., `analyze_sentiment`, `generate_docx`) using Python's unittest framework.

Integration Testing: Verified end-to-end flow from input to SRS and diagram generation.

API Validation: Ensured Tavily and Groq API responses were accurate by comparing with manual web searches and sentiment analysis tools.

Diagram Verification: Visually inspected PlantUML diagrams for alignment with user inputs (e.g., actors and use cases).

7.2 Results

95% accuracy in SRS content generation based on manual review of 10 test cases.

100% success rate in diagram rendering when PlantUML JAR and Java were configured correctly.

Minor discrepancies (e.g., missing stakeholders) resolved by refining prompts to Groq API.

8. Challenges and Solutions

Challenge: Initial difficulty installing tavily-python due to package name mismatch.

Solution: Updated requirements.txt to tavily-python and adjusted imports in app.py.

Challenge: PlantUML diagram generation failed due to incorrect JAR path.

Solution: Verified and updated plantuml_jar_path in app.py to the correct file location.

Challenge: Slow API responses affecting user experience.

Solution: Optimized API calls by limiting Tavily search results to 5 and setting Groq max tokens.

9. Future Enhancements

Add support for additional UML diagram types (e.g., Sequence Diagram).

Implement real-time validation of input fields.

Enhance GUI with drag-and-drop functionality for diagram customization.

Integrate a database to save and retrieve past SRS documents.

10. Conclusion

The SRS_Generator project successfully demonstrates the automation of SRS and UML diagram creation, leveraging modern web technologies and AI. It addresses a critical need in software engineering by simplifying requirements documentation, making it a valuable tool for academic and professional use. The iterative approach and rigorous testing ensure a robust and reliable application.

11. References

IEEE Std 29148-2018, "Systems and software engineering -- Life cycle processes -- Requirements engineering."

Flask Documentation: <https://flask.palletsprojects.com/>

Tavily API Documentation: <https://tavily.com/docs>

Groq API Documentation: <https://console.groq.com/docs>