School of Computing
SRM IST, Kattankulathur – 603 203

Course Code: 21CSC303J

Course Name: Software Engineering and Project Management

Experiment No	**1**
Title of Experiment	**To identify the Software Project, Create Business Case, Arrive at a Problem Statement**
Name of the candidate	**SAATVIK AGNIHOTRI**
Team Members	**RIYA RAI, MD. DILSHAD ALAM**
Register Number	**RA2211003011922, RA2211003011919, RA2211003011917**
Date of Experiment	**24-01-2025**

Mark Split Up

S.No	**Description**	**Maximum Mark**	**Mark Obtained**
1	Exercise	5	
2	Viva	5	
	Total	**10**	

Staff Signature with date

Aim : To Frame a project team, analyze and identify a Software project. To create a business case and Arrive at a Problem Statement for the **AI Technical Document Generator**

Team Members:

S. No	Register No	Name	Role
1	RA2211003011922	SAATVIK AGNIHOTRI	Lead
2	RA2211003011919	RIYA RAI	Member
3	RA2211003011917	MD. DILSHAD ALAM	Member

Project Title: **AI Technical Document Generator(AutoDocX)**

Project Description :

The AI-Based Technical Documentation Generator is an automated system designed to generate human-readable, concise, and accurate documentation for software projects. This system integrates directly with version control tools like GitHub, triggering the creation of documentation each time new code is committed. It analyzes the changes in the codebase (including new functions, classes, and modifications to existing ones), generates summaries, and organizes them in a centralized documentation file (e.g., Markdown or HTML). This eliminates the need for manual documentation, ensuring the documentation is always up-to-date and aligned with the latest code changes.

BUSINESS CASE TEMPLATE

DATE	24/01/2025
SUBMITTED BY	SAATVIK AGNIHOTRI, RIYA RAI, MD DILSHAD ALAM
TITLE	AI TECHNICAL DOCUMENT GENERATOR (AutoDocX)

THE PROJECT

- **Manual Documentation is Time-Consuming**: Writing and updating documentation manually can be tedious and error-prone.
- **Documentation Often Becomes Outdated**: Code changes frequently, causing documentation to fall out of sync and leading to confusion.
- **Inconsistent Documentation Quality**: Different developers may write documentation in varying formats and levels of detail.
- **Opportunity for Automation**: Automating documentation generation ensures it stays up-to-date and accurate with every code commit.
- **Increased Developer Productivity**: Developers can focus more on coding and less on maintaining documentation, improving efficiency.

THE HISTORY

- **Manual Documentation Process**: Most software projects rely on developers manually writing documentation, which is time-consuming and often neglected.
- **Outdated Docs**: As code evolves rapidly, documentation often fails to keep up, leading to discrepancies between the codebase and the documentation.
- **Lack of Standardization**: Documentation varies in quality and format, depending on the individual developer, resulting in inconsistency across the project.
- **Increased Developer Overhead**: Developers spend valuable time maintaining documentation instead of focusing on writing code or developing features.
- **Difficulty in Managing Large Codebases**: As projects grow, it becomes increasingly difficult to maintain comprehensive and accurate documentation manually.

LIMITATIONS

- **Integration Challenges**: Difficulty in seamlessly integrating with various version control tools (e.g., GitHub, GitLab) or code repositories with different structures.
- **Complexity of Code Analysis**: The AI may struggle to understand highly complex or poorly written code, leading to inaccurate or incomplete documentation.
- **Diverse Documentation Styles**: Variations in how different developers write and comment their code could make automated documentation generation less consistent.
- **Performance Issues**: Handling large codebases or frequent commits could strain system performance or slow down the documentation generation process.
- **Lack of Domain-Specific Knowledge**: The AI might need training or customization to understand specific programming languages or frameworks used in different projects.
- **Resource Constraints**: The need for sufficient computing power or cloud resources to analyze large codebases and generate documentation in real-time.

APPROACH

- **Integration with Version Control Systems**: Set up seamless integration with platforms like GitHub to trigger documentation generation on every commit.
- **AI Model for Code Analysis**: Develop or integrate an AI model capable of understanding code changes (functions, classes, etc.) and generating accurate documentation.
- **Documentation Format Setup**: Decide on output formats (e.g., Markdown, HTML) for the generated documentation and ensure they are developer-friendly.
- **Customization Options**: Allow for configuration of documentation styles or templates to suit different project needs.
- **Testing and Refinement**: Continuously test the system with various codebases, fixing bugs, and improving the AI's understanding and accuracy.
- **User Interface for Monitoring**: Create an interface for developers to monitor and review generated documentation before it's finalized, allowing manual adjustments if necessary.

BENEFITS

- **Time-Saving**: Automates the documentation process, freeing up developers to focus more on coding.
- **Always Up-to-Date**: Documentation remains in sync with code changes, ensuring accuracy and relevance.
- **Consistency**: Provides a standardized approach to documentation, eliminating variations in style

and quality.

- **Improved Developer Onboarding**: New developers can easily understand the project by reviewing accurate, up-to-date documentation.
- **Reduced Documentation Overhead**: Significantly reduces the manual effort required to maintain and update documentation, leading to increased efficiency.

Result:

Thus, the project team formed, the project is described, the business case was prepared and the problem statement was arrived