DEPARTMENT OF SOFTWARE ENGINEERING
Faculty of Computing & Information Technology (FCIT)
University of the Punjab, Lahore – PAKISTAN

FYDP-DSE Proposal

Pitch Pilot

Bachelors of Science in Software Engineering (2022-2026)

By

Hasnain Rasheed – BSEF22M519

Muhammad Zaid – BSEF22M522

Muhammad Zubair – BSEF22M548

Mutti-Ullah – BSEF22M549

Supervised by:

Primary supervisor,
Prof. Dr. Muhammad Kamran Malik,
Chairman Department of Data Science

Co-supervisor,
Dr. Omer Nawaz
Assistant Professor at Department of Software Engineering

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY,

UNIVERSITY OF PUNJAB, LAHORE

Proposal Approval Signatures (FAC)

We, the undersigned members of the assigned Faculty Advisory Committee (FAC), have reviewed the submitted project proposal. Based on our initial assessment, we acknowledge the group's submission and confirm that the proposal has been evaluated for completeness and alignment with FYDP objectives.

Group ID: FPSEF25X52

Project title: Pitch Pilot

Primary supervisor: **Sir Kamran Malik**

(Signature)

Co-supervisor: **Sir Omer Nawaz**

(Signature)

Signatures (Faculty Advisory Committee (FAC)

	FAC member#1	FAC member#2
Name	**Sir Farhan Ahmad Ch**	**Mam Muddassira Arshad**
Signature		

Head of FYDP Coordination Office: _____

(Signature)

Contents

1. Project Title and abstract

Project Title: Pitch Pilot - AI Solution to automate the pre-development phase.

Abstract:

The pre-development phase of the Software Development Life Cycle (SDLC) is important and problematic. It involves requirement engineering, feasibility analysis, planning, and documentation. But problems are that the clients are often novice and fail to express their needs clearly. Requirement gathering teams are mostly sales-oriented and lack domain knowledge. The problem is made worse by today's complex and interconnected software systems, which make requirements engineering more difficult. As a result, requirements are incomplete, ambiguous, conflicting, and thus remain changing. Human errors such as misinterpretation, assumptions, and miscommunication add more problems. Moreover, after gathering, requirements are not analyzed properly for feasibility, time, cost, and resources. Documentation, prioritization, and wireframe generation are also inadequate, causing project delays and increased costs.

To address these challenges, this project proposes the development of an AI agent to automate requirement gathering and analysis. The system will extract functional and nonfunctional requirements directly from client input, validate and confirm them iteratively, and analyze them for feasibility in terms of time, cost, and resource estimations. It will also prioritize requirements, generate structured documentation, and produce wireframes, ensuring accuracy and completeness throughout the process.

The proposed solution aims to minimize human bias in requirement gathering phase, saves valuable time, and ensures efficient use of resources. By automating the pre-development phase, it will support project completions within given time and budget constraints and enable software projects to deliver high-value returns to stakeholders.

2. Team Details

Name	Roll Number	Email	Role
Hasnain Rasheed	BSEF22M519	bsef22m519@pucit.edu.pk	Backend Developer Scrum Master
Muhammad Zaid	BSEF22M522	bsef22m522@pucit.edu.pk	Backend Developer Backend Lead
Muhammad Zubair	BSEF22M548	bsef22m548@pucit.edu.pk	Backend Developer Product Owner
Mutti Ullah	BSEF22M549	bsef22m549@pucit.edu.pk	Frontend Developer Frontend Lead

3. Problem Statement / Motivation

The pre development phase of SDLC (Requirement Engineering, Feasibility Analysis & Planning, Documentation) is a crucial phase. And this phase is very problematic. Often the clients are novice. The requirement gathering teams lack domain knowledge. Requirements are often incomplete, ambiguous, complex, conflicting and are not gathered adequately so often remain changing even after the development phase has started. Human biasness is also a contributing factor. After requirement gathered, they are not analyzed properly involving poor feasibility analysis, prioritization, time, cost & resource estimations cause project delays. There are also problems in the documentation, wireframe generation and prototyping of the requirements. The pre-development phase of SDLC is not only error-prone, but manually performing this consumes a lot of development time that makes it slow also.

Studies has shown that poor practices in pre-development phase leads to project failures. The Standish Group (2020) reported that only 31% of projects complete on time and within budget. Only 46% give high value to stakeholders. The Project Management Institute (2014) stated that requirement issues are the reason of 37% project failures. These issues also increase cost by 25%–40% of the initial budget. Carnegie Mellon's Software Engineering Institute (2023) reported that 68% of requirement defects are found late. Fixing them at this stage costs 5–10 times more than in the requirement phase.

These problems highlight the urgent need for improved practices in the pre-development phase. Automating requirement gathering, validation, analysis, and documentation can help us to reduce human bias, save time for development phases by reducing time from the predevelopment phase, optimize resources, and improve project success rates. This motivation drives the selection of this project, with the aim of automating pre-development phases so that software projects are completed within the given time and budget constraints, reducing cost of re-work due to miscommunications, while also deliver high-value returns to stakeholders.

4. Project Objectives

Following are the clear, measurable and achievable objectives of our project:

i. **Taking Inputs from Client**
The system will take inputs directly from the client. It will help the client give inputs conveniently. It will guide the client even if he is a novice. Success will be measured if a novice client can also give inputs easily.

ii. **Extracting Requirements from Client Input**
The system will extract requirements from the client's input. It will extract functional requirements. It will extract non-functional requirements. Success will be measured if all the requirements that the client wants are successfully extracted.

iii. **Validation and Confirmation of Requirements**
The system will validate requirements at runtime. It will correct errors. It will ask for confirmation from the client. Success will be measured if the model can successfully find errors and ask validation questions from the client.

iv. **Identifying Conflicting and Infeasible Requirements**

The system will check requirements for conflicts. It will check requirements for feasibility. It will detect ambiguities. Success will be measured if the model can successfully identify conflicting and ambiguous requirements.

v. **Feasibility, Time, Cost, and Resource Analysis**
The system will analyze the feasibility of requirements. It will estimate time. It will estimate cost. It will estimate resources. Success will be measured if the model can successfully estimate all these factors.

vi. **Prioritizing and Planning Requirements**
The system will prioritize requirements. It will plan requirements based on importance, feasibility, and client needs. Success will be measured if the prioritization and planning align with expert expectations.

vii. **Documenting and Generating Wireframes**
The system will generate requirement documentation. It will generate structured documents. It will generate wireframes. Success will be measured if documents follow industry standards and wireframes are validated by clients with fewer revisions.

5. System Overview

The system will be a web-based platform designed to automate and improve the predevelopment phase of SDLC. It will include a client interface and a backend powered by AI models, including NLP models integrated through APIs, supported by a proper database. Both clients and developers will use this system directly. Initially, it will operate offline, but later it will be deployed online.

The system will provide step-by-step guidance to clients for entering requirements. It will process both functional and non-functional requirements and ask clarifying questions in real time. The processing will follow a structured sequence, where the output of one module becomes the input of the next, supported by a feedback loop to ensure accuracy and refinement at each stage. The system aims to reduce ambiguity, remove errors, save time, and increase project success rates.

i. **Requirement Gathering and Validation Module**
- Takes input directly from the client.
- Extracts functional and non-functional requirements.
- Validates, corrects, and confirms requirements at runtime.
- Detects ambiguous, unclear, and conflicting requirements.
- Highlights issues and asks clarifying questions.
- Resolves conflicts to make requirements crystal clear and stable.
- Removes human bias such as misconceptions, miscommunication, misinterpretation, and false assumptions.

ii. **Feasibility Analysis and Estimation Module**
- Analyzes and processes validated requirements from Module 1.
- Performs feasibility analysis. Estimates time, cost, and resources.
- Calculates overall project cost and company profit.

iii. **Planning and Prioritization Module**
- Arranges requirements based on project needs.
- Considers importance, urgency, and available time.
- Assigns priority levels to each requirement.
- Creates a structured and logical sequence for implementation.

iv. **Documentation and Prototyping Module**
- Generates structured requirement specifications and documentation.
- Produces wireframes to visually represent requirements.
- Provides clear outputs for validation by clients and developers.

6. Development Methodology

We have selected Agile (Scrum) as the development methodology for our project, as it aligns well with the project requirements. The methodology ensures flexibility, adaptability, and continuous feedbacks from supervisors.

Key Characteristics of Our Scrum Approach

i. **Sprint Duration**: Each deliverable (D1, D2, D3, D4) will be divided into the required number of sprints. Each sprint will have a duration of two weeks, during which the planned tasks will be executed under the guidance of the supervisors.

ii. **Scrum Roles**: We will assign clear roles to each of our team members for responsible and answerable execution of our project:
   a. **Product Owner** – He will be responsible for deciding what features the project should have, keeping track of the work list, and making sure it matches with supervisor expectations.
   b. **Scrum Master** – He will be responsible for guiding the team in meetings, helping to solve problems, managing scrum ceremonies and making sure the team follows the right process.
   c. **Frontend Lead & Frontend Engineer** – He will be responsible for designing, developing, and integrating the user interface.
   d. **Backend Lead & Backend Engineer** – He will be responsible for server-side development, handling APIs, managing databases, and ensuring overall system integration.

iii. **Collaboration Tools**:
   a. **Jira** – It will be used for managing the work list (backlog), planning sprints, and keeping track of tasks.
   b. **GitHub** – It will be used for saving code versions, working together as a team, and connecting with automated testing and deployment (CI/CD integrations).

iv. **Scrum Ceremonies**:
   a. **Sprint Planning** – In this meeting, we will set the goals for the sprint and decide which tasks from the backlog will be completed.
   b. **Daily Standups** – Every day, we will have a short meeting to check progress and solve any problems the team is facing.
   c. **Sprint Review/Demo** – At the end of the sprint, we will show the completed work and features to the supervisors.
   d. **Sprint Retrospective** – After the sprint, we will discuss what went well, what did not, and how we can improve in the next sprint.

v. **Supervisor Interaction**: We will have weekly meetings with both the supervisor and co-supervisor. The supervisor will be of primary importance and will guide us through every

problem. The co-supervisor will mainly help us in meeting deadlines as per the guidelines and instructions of the Software Engineering department.

vi. **Definition of Done (DoD)**: A feature will be marked as "done" only when it is fully built, tested successfully, and meets the agreed acceptance criteria.

7. Technology Stack

Layer	**Technologies & Tools**
Frontend	HTML+CSS, Tailwind, React Js
Backend	Node.js + Python
Database	MongoDB
Dev Tools	Git, GitHub, Docker, Jira, LLM Models, Prompt Engineering, Postman

8. Project Timeline / Milestones

Project Deadlines & Timelines	**Milestones & Work to be Completed**
6th Oct 2025 **(Before D1 – 10th Oct 2025)**	Set up GitHub repository and configure initial tools. Define product vision, epics, and initial backlog items. Starting working on Module 1 (Requirement Gathering & Validation). Perform backlog grooming with supervisor feedback. Conduct internal review of progress and tasks.
15th Dec 2025 **(Before D2 – 19th Dec 2025)**	Refine Module 1 by implementing runtime validation and ambiguity/conflict detection mechanisms. Design functional workflows to support structured requirement entry. Integrate baseline AI/NLP components for automated requirement analysis. Perform continuous sprint testing, including unit and integration tests. Deliver the first working prototype for supervisor review. Conduct

	sprint catch-up sessions and finalize testing outcomes.
End of Jan 2026 **(Before D3 – 1st week of Feb 2026)**	Develop Module 2 (Feasibility Analysis & Estimation). Expand AI integration to support cost, time, and resource estimation. Set up CI/CD pipeline for automated builds and testing. Perform continuous testing and iterative refinement. Apply final polishing and bug fixing.
Late Mar 2026 **(Before D4 – 6th week of Spring 2026)**	Implement Module 3 (Planning & Prioritization). Implement Module 4 (Documentation & Prototyping), including requirement specifications and wireframes. Perform full system integration (Frontend + Backend). Configure deployment pipeline with Docker and CI/CD. Conduct comprehensive testing and evaluation. Deliver user interface demo. Prepare user manuals, guides, and deployment documentation. Conduct acceptance testing and sprint review. Execute dry-run deployment and supervisor review. Deliver final presentation.

9. Market Fit / Commercial Potential

Usefulness of Pitch Pilot:

- Studies show that poor requirements engineering (RE) causes about 37% of project failures and increases costs by 25–40%**.**
- The manual pre-development phase is slow, error-prone, and resource-intensive, which makes automation highly desirable.
- Companies lose significant time and revenue because 68% of requirement defects are detected in later stages, where fixing them costs up to 10 times more.

Uniqueness of Pitch Pilot:

Pitch Pilot stands out from generic project management tools by focusing entirely on automating RE and pre-development tasks. Key features include:

- Automated extraction and validation of functional and non-functional requirements.
- AI-powered analysis for feasibility, time, cost, and resources in real-time.
- Automated wireframe generation and structured documentation.
- Reduced bias and errors through iterative client input validation.

- **Supporting Research:** Studies confirm that Gen AI and LLMs can detect ambiguities, validate requirements, and assist with documentation. However, there is currently no integrated product that performs all steps end-to-end.

Target Market:

Our main users will be small and medium businesses, startups, and large software companies that want to speed up their Software Development Life Cycle (SDLC) and reduce project failures. Consulting firms can also use our system to make requirement workshops shorter and more effective.

Monetization Plan:

We plan to earn revenue mainly through a SaaS (Software as a Service) subscription. Users will pay a monthly or yearly fee, either per user or per project. This model is simple, flexible, and works well for startups, small businesses, and growing companies.

Open-Source Plan:

We plan to release an open-source version of the product. This will let developers, students, and organizations use it for free, share feedback, and add improvements. It will help more people adopt the product and make it grow stronger over time.

10. Competitor Analysis

The table below summarize the key competitors in the field of Requirement Engineering, but no tool provide true end to end automation:

Tool / Product	What it Does	Limitations	Advantages
VISURE	It uses AI to gather requirements, make sure they follow the rules, and keep track of any changes.	It focuses mostly on tracking and does not create wireframes or check if things are feasible.	Our model provides complete automation, including feasibility checks, prioritization, documentation and UI creation.
AUTOSAD	It automatically updates the system design whenever the requirements change.	It only focuses on system design and does not validate with clients or provide proper documentation.	Our model covers the full cycle, including validation, feasibility checks, documentation, and wireframe generation.
Req2Spec	It automatically converts natural language requirements into formal specifications.	It does not address frontend design or UI creation.	Our model provides end-to-end automation, including UI creation and alignment with specifications.
UX Pilot	It automatically assists in creating wireframes, UI designs, and prototypes.	It has issues with Figma integration and design quality.	Our model provides seamless Figma integration, greater customization, and AI-driven insights.