AI-Powered Student Productivity App

- Version 1.0
- **Prepared by:**
  Name: Porchezhian
  Reg no: 23BCE1619
  Email: porchezhian.r2023@vitstudent.ac.in
- **Instructor:** Dr. Nancy Parkavi
  Course: Software Engineering
  Lab Slot: L3+L4
  Teaching Assistant: NIL
  Date: 27/07/2025

Document Overview

- **Purpose**: To evaluate and justify the most appropriate SDLC process model for the project.
- **Scope**: Focused on selecting the lifecycle model based on project objectives, risk, team constraints, and feature complexity.

Project Summary

Problem Statement

In today's digital learning environment, students are overwhelmed by disorganized notes, scattered tasks, and fragmented study plans. Despite the availability of various productivity tools, most fail to provide contextual intelligence, seamless integration, and personalization. This results in poor time management, reduced focus, and lower academic performance. There is a growing need for a unified, intelligent system that can assist students in organizing, summarizing, and actively engaging with their academic material in an efficient and personalized manner.

Proposed Solution

We propose to develop an **AI-Based Student Productivity App** that uses artificial intelligence to analyze student inputs (text, notes, time blocks), organize them meaningfully, and assist in active recall through quizzes and intelligent nudges. The app will provide:

- Smart **note-taking** with summarization.
- AI-powered **quiz generation** from notes.
- A **grouping system** that logically connects notes across subjects.
- A **free-time detection system** that suggests when and what to revise.
- AI-based **progress tracking** and **productivity analytics**.
- A **minimalist UI** with features tailored for focus and academic enhancement.

The app will function both online and offline, with cloud sync capabilities when connected.

Impact

This system will:

- Enhance **learning retention** through intelligent quiz generation and nudging.
- Improve **academic time management** using free-time detection.
- Reduce **cognitive load** by automating organization and summarization.
- Provide **personalized feedback** and promote daily engagement.
- Increase productivity with minimal user effort.

Ultimately, the app bridges the gap between passive note-taking and active learning, improving student outcomes in a scalable, tech-driven manner.

Methodology

The project will be developed following a suitable **Software Development Life Cycle (SDLC)** approach, tailored for iterative and requirement-driven updates. The steps include:

1. **Requirement Analysis**
    - Identify key features and user needs.
    - Develop SRS, use case models, and diagrams.
2. **Process Model Selection**
    - Compare Waterfall, Agile, Spiral, V-Model, RAD, etc., and adopt a **Hybrid Agile-Incremental Model** for flexibility and rapid prototyping.
3. **System Design**
    - Use UML diagrams: Class, Use Case, ER, DFD, State Transition.
    - Define UI using storyboarding and wireframes.
4. **Implementation**
    - Build the front end using **React Native** (cross-platform).
    - Use **Python (Flask/FastAPI)** or **Node.js** for the backend.
    - Integrate AI features using **OpenAI APIs**, **Hugging Face**, or **custom NLP models**.
5. **Testing**
    - Functional and non-functional testing.
    - User testing for feedback.
6. **Deployment**
    - Deploy a demo version with cloud sync support (Firebase/GCP).
7. **Evaluation**
    - Gather feedback via in-app surveys.
    - Measure user retention, time-on-app, and learning improvements.

SDLC Models Evaluated

SDLC Model	Overview	Pros for This Project	Cons for This Project	Suitability
1. Waterfall	Linear & sequential model; one phase completes before next begins	Simple to plan and document	No flexibility for changes, poor fit for AI experimentation and offline-sync evolution	Low
2. V-Model	Waterfall with parallel validation/verification per phase	Better testing coverage, more discipline	Still rigid; no adaptability to evolving AI/user needs	Low
3. RAD (Rapid App Development)	Emphasizes fast prototyping with component reuse	Fast UI prototyping; good for note/UX testing	Not suitable for deep backend logic, AI pipelines, or complex sync systems	Medium (good for UI-only prototyping)
4. Spiral	Combines iterative dev + risk analysis at each loop	Good for projects with high risks (AI accuracy, sync conflicts); iterative nature helps AI training	Complex, heavyweight for a student team; high management overhead	Moderate to High

5. Incremental Model	Build software in independent chunks (modules)	Breaks the app into useful modules (notes, projects, rewards, etc.), easier for distributed team	Needs careful interface planning; limited feedback loop early	Moderate
6. Agile (Scrum)	Iterative, incremental, feedback-driven cycles	High flexibility, user feedback, MVPs, AI can evolve, great for multi-feature complex tools	Needs discipline in sprint planning & testing, might require tooling	Very High
7. Iterative Model	Build quickly with partial functionality, then refine repeatedly	Good for early versioning, refining AI modules; aligns with summary → quiz → analysis pipeline	Can accumulate technical debt, needs refactoring discipline	High
8. Big Bang	No clear process; start coding, adjust as needed	Fun and fast for prototypes	Chaos in large apps, no scalability, hard to manage AI logic, bad for teamwork	Very Low

Selected Model: Agile Development (Scrum)

Justification:

- **User Feedback Ready**: AI features can evolve based on student feedback during sprints.
- **Flexible Planning**: Adaptive to changing requirements, UI tweaks, and AI model shifts.
- **Parallel Work Streams**: Backend AI and frontend UI feature development happen concurrently.
- **Deliver Value Early**: MVPs (note module, summarizer) can be shipped early for real user testing.
- **Team Collaboration**: Sprint ceremonies support coordination, sprint reviews, retrospectives.