Software Requirements Specification
AI-Based Student Productivity Application

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: 28/07/2025

Contents

- 3. Specific Requirements
    - 3.1 External Interface Requirements
    - 3.2 Functional Requirements
    - 3.3 Use Case Model
- 4. Other Non-Functional Requirements
    - 4.1 Performance Requirements
    - 4.2 Safety and Security Requirements
    - 4.3 Software Quality Attributes
- 5. Other Requirements
- Appendix A: Data Dictionary

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Porchezhian	Initial Draft	28/07/2025

1. Introduction

This document specifies the Software Requirements Specification (SRS) for the AI-based Student Productivity Application. It outlines the features, architecture, design constraints, and user interface requirements for the system, which aims to enhance productivity for students by using AI for notes, assignments, and schedule management.

1.1 Document Purpose

This document serves to define the software requirements for the AI-based Student Productivity Application (Version 1.0). It outlines the complete functional and non-functional requirements to be implemented in the application. It is intended to serve as the basis for the design, development, and validation of the software.

1.2 Product Scope

The application will function as a downloadable desktop software, working offline-first with cloud sync. Its key features include AI-generated summaries of notes, quiz generation from notes, schedule-aware nudges, reward tracking, and productivity analysis. It supports modular features that evolve over time.

1.3 Intended Audience and Document Overview

This document is intended for instructors, project evaluators, developers, testers, and future maintainers of the project. It includes high-level overviews, detailed requirements, use cases, and data definitions. It is structured to provide clarity from conceptual to implementation levels.

1.4 Definitions, Acronyms and Abbreviations

- AI – Artificial Intelligence
- UI – User Interface
- UX – User Experience
- DB – Database
- CRUD – Create, Read, Update, Delete
- MVP – Minimum Viable Product
- SRS – Software Requirements Specification
- JSON – JavaScript Object Notation

1.5 Document Conventions

- Font: Arial, size 11
- Section titles bolded and numbered as per IEEE SRS standards
- Italics used for TODOs or editable fields

1.6 References and Acknowledgments

- IEEE SRS Format Guidelines
- Agile Manifesto (https://agilemanifesto.org/)
- ChatGPT & HuggingFace Transformers Documentation
- SQLite & REST API Integration Guides

2. Overall Description

2.1 Product Overview

The product is a desktop productivity suite focused on students. It allows note-taking, assignment tracking, AI-assisted summaries, quiz creation, and rewards. The application supports offline use with cloud-based sync once online.

[Diagram Placeholder: High-level System View]
[User] <--> [Frontend App] <--> [Local DB] <--> [Sync Service] <--> [Backend AI Server + Cloud DB]

2.2 Product Functionality

- Note-taking with markdown/rich text editor
- AI-generated summaries for each note
- Grouping and compiling summaries
- Quiz generation from summaries
- Assignment/project tracker with deadlines
- Timetable/Calendar based free time detection
- AI-generated productivity nudges
- Reward points, streaks, and gamification
- Works offline, syncs when online

2.3 Design and Implementation Constraints

- Desktop-based (Electron.js or Tauri suggested)
- SQLite for local database
- REST API for cloud sync
- Python-based AI models for summaries/quizzes
- Must follow UML and COMET modeling methods

2.4 Assumptions and Dependencies

- User has periodic internet access for sync
- AI models hosted on external cloud (FastAPI or Flask)
- External libraries for NLP (Transformers, spaCy)
- Offline data must be encrypted

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

- UI includes dashboard, note editor, summary viewer, quiz screen, and reward tracker
- Interface is designed to be touch/mouse-friendly with collapsible panels

3.1.2 Hardware Interfaces

- Desktop system with basic storage and internet capabilities
- No special sensors or devices needed

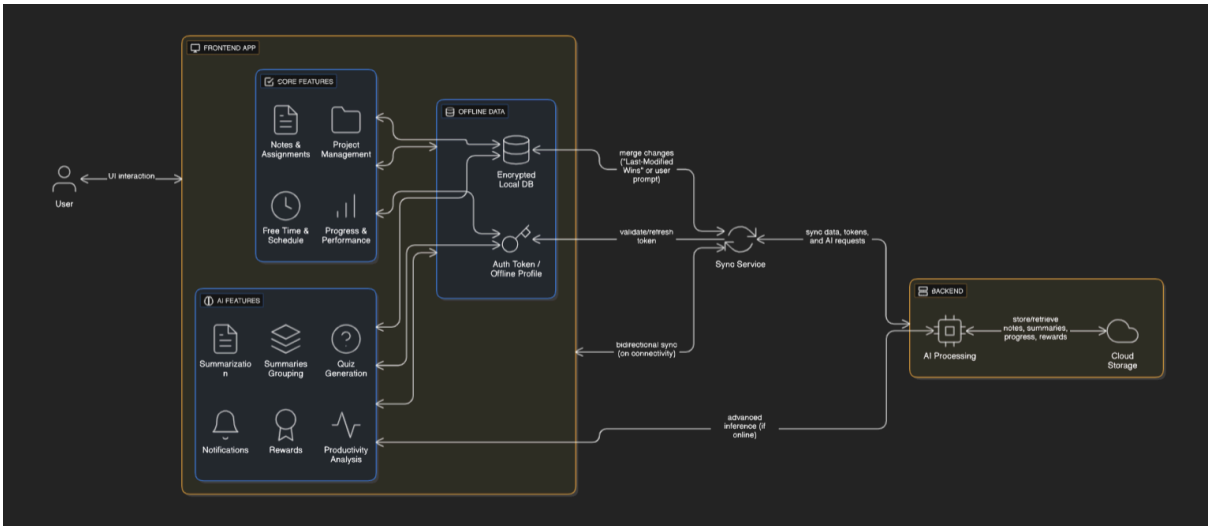
3.1.3 Software Interfaces

- Local app communicates with remote sync API
- Backend interacts with AI microservices and cloud database

3.2 Functional Requirements

- F1: The system shall allow users to create, update, delete, and group notes
- F2: The system shall generate summaries using AI for each note
- F3: The system shall generate quizzes from compiled summaries
- F4: The system shall detect free time using user schedule
- F5: The system shall send nudges during free periods
- F6: The system shall track and visualize rewards and streaks
- F7: The system shall support offline mode with sync-on-connect

3.3 Use Case Model



Use Case Diagram for the software

Use Case 1: Take Notes

- **Actor(s)**: Student
- **Preconditions**: User is logged in
- **Main Flow**:
    - User opens note-taking module
    - Creates new note (text/audio/image)
    - Saves note locally
- **Alternate Flow**:
    - User edits or deletes a previous note
- **Postconditions**: Note saved locally and marked for cloud sync
- **Exceptions**: If storage is full, alert user

Use Case 2: AI-Generated Note Summarization

- **Actor(s)**: Student, AI Backend
- **Preconditions**: Note exists
- **Main Flow**:
    - User taps "Summarize"
    - Note sent to server
    - AI processes and returns a summary
    - Summary displayed at top of note
- **Alternate Flow**:
    - Auto-summarization upon saving the note
- **Postconditions**: Summary stored alongside note
- **Exceptions**: No internet — queue for sync later

Use Case 3: Group Notes and Compile Summaries

- **Actor(s)**: Student
- **Preconditions**: At least two notes exist
- **Main Flow**:
    - User selects multiple notes
    - Taps “Group & Compile”
    - AI compiles unified summary
- **Alternate Flow**:
    - User manually edits the group summary
- **Postconditions**: Compilation saved as new document
- **Exceptions**: Notes not compatible in format

Use Case 4: Generate Quiz from Summaries

- **Actor(s)**: Student, AI Backend
- **Preconditions**: Valid summary available
- **Main Flow**:
    - User selects summary
    - Taps “Generate Quiz”
    - AI sends multiple-choice questions
    - User answers and gets feedback
- **Alternate Flow**:
    - User saves quiz for later
- **Postconditions**: Quiz score saved
- **Exceptions**: Server timeout

Use Case 5: Manage Assignments & Projects

- **Actor(s)**: Student
- **Preconditions**: User is logged in
- **Main Flow**:
    - User adds task details
    - Sets priority, deadline, category
    - System adds to calendar
- **Alternate Flow**:
    - User marks task as completed
- **Postconditions**: Task saved & synced
- **Exceptions**: Deadline in the past

Use Case 6: Free Time Detection & Notification

- **Actor(s)**: System
- **Preconditions**: Schedule must be populated
- **Main Flow**:
    - System monitors calendar
    - Finds free time slots
    - Sends motivational/productive nudges
- **Alternate Flow**:
    - User opts out of nudges
- **Postconditions**: Log created for nudges sent
- **Exceptions**: Notification permissions disabled

Use Case 7: Reward System for Productivity

- **Actor(s)**: System, Student
- **Preconditions**: User completes tasks or maintains streak
- **Main Flow**:
    - System analyzes daily/weekly progress
    - Awards points or badges
    - User views reward history
- **Alternate Flow**:
    - System gives motivational quote instead of reward
- **Postconditions**: Reward saved to user profile
- **Exceptions**: Corrupted local log

Use Case 8: Offline Mode & Sync

- **Actor(s)**: Student, System
- **Preconditions**: Device is offline
- **Main Flow**:
    - User takes notes or completes tasks
    - App stores data locally
    - On reconnection, syncs to cloud
- **Alternate Flow**:
    - Partial sync when network is unstable
- **Postconditions**: All data synced and merged
- **Exceptions**: Sync conflict resolution

4. Other Non-Functional Requirements

4.1 Performance Requirements

- P1: The app should load in <3 seconds
- P2: Summary generation must complete within 10 seconds of user request

4.2 Safety and Security Requirements

- S1: All local data should be encrypted
- S2: Sync operations must use HTTPS
- S3: User login must require secure authentication

4.3 Software Quality Attributes

4.3.1 Reliability

- Auto-backup of notes before sync
- Sync logs for tracking history

4.3.2 Usability

- Minimalistic UI with onboarding tutorial
- Responsive layout for different screen sizes

4.3.3 Maintainability

- Modular code structure with well-documented components
- Separate configuration files for sync, AI, and local storage

4.3.4 Portability

- Cross-platform packaging (Windows, macOS, Linux)
- Electron/Tauri enables consistent codebase

5. Other Requirements

- Language: English
- Internationalization support planned in future versions
- Open-source licensing (MIT or GPL suggested)

Appendix A – Data Dictionary

Variable	Description	Type	Source	Constraints
note_id	Unique identifier for note	UUID	Generated	Read-only
summary	AI-generated summary	Text	AI Engine	None
reward_points	Tracks gamification progress	Integer	App Logic	>= 0
quiz_set	Set of generated questions	JSON	AI Engine	None