Software Requirements Specification

for

GATE Notes Simplifier

Version 1.0 approved

Prepared by -:Mohammed Obaid,Deepak kumar Ojha

Indian Institute of Information Technology, Allahabad

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1. Purpose

This document specifies the software requirements for the GATE Notes Simplifier. It helps convert uploaded GATE notes into simplified Hinglish or plain English for better understanding and self-study.

1.2. Document Conventions

- ***Bold*** *is used for section titles and headings.*
- *Requirements are labeled with codes like **FR1**, **FR2** for functional requirements and **NFR1**, **NFR2** for non-functional ones.*
- *Numbering follows IEEE SRS format (e.g., 1.1, 2.3.1).*
- *Each requirement is treated separately and has its own priority.*
- *Plain and simple English is used throughout the document to make it easy to understand*

1.3. Intended Audience and Reading Suggestions

This document is meant for:

- ***Developers**: To understand what features to build.*
- ***Testers**: To know what needs to be tested.*
- ***Students and Users**: Primary users of the system so they can convert their work in their regional language..*
- ***Project Supervisors/Managers**: To track development progress and ensure goals met*

1.4. Product Scope

This software helps students by allowing them to upload their GATE notes in formats like PDF or plain text. Once uploaded, the system either automatically simplifies the content using language processing tools (like NLP) or gives the option to manually edit the notes. The main aim is to convert hard-to-understand technical content into simple, clear language — like Hinglish or easy English — so students can understand it better. After simplification, students can organize their notes by topic or subject and finally export them in popular formats like PDF or DOCX for studying or sharing

1.5. References

- IEEE Std 830-1998
- Gate Notes : https://study.madeeasy.in/category/cs-it/
- HuggingFace NLP APIs: https://ollama.com/

2. Overall Description

2.1. Product Perspective

The GATE Notes Simplifier is a standalone project made specially for students preparing for the GATE exam. It is not a part of any big software system or platform. The tool is designed as a complete and independent solution, where students can upload their notes, simplify them, edit if needed, and download the final version — all within this one application. It does not depend on any external system to work.

2.2. Product Functions

The main goal of this software is to help students simplify their GATE notes easily. It provides the following features:

- Students can upload their notes in PDF or text format.
- The system can simplify the uploaded content using language tools like NLP, or students can also choose to simplify it manually.
- The simplified notes can be edited as per the student's understanding.
- Students can organize their notes by topics or subjects for better clarity.
- Finally, students can export or download their personalized notes in formats like PDF or DOCX.

2.3. User Classes and Characteristics

- **Students**: These are the main users. They will upload their GATE notes, simplify them, edit the content if needed, and finally export the notes. Most students will have basic computer knowledge and should be able to use the system easily.
- **Supervisors or Reviewers**: These users may use the system to check the progress of students, especially if the software is being used in a college project or academic setting. They don't need to use all the features but may review the final output or usage.

2.4. Operating Environment

The software is designed to work on normal desktops or laptops that most students already use. It does not need any special hardware. It will run smoothly on common operating systems like **Windows** or **Linux**.This application will work in modern web browsers such as **Google Chrome** or **Mozilla Firefox**.

2.5. Design and Implementation Constraints

Since this project is being developed as part of an academic semester, it has a fixed time limit. The complete system must be designed, developed, and tested within a few months.Also, the software is planned to be lightweight and easy to run, so it won't depend on heavy machine learning models that need GPUs or high-performance servers.The design will focus on keeping things simple and easy to use, so that even students with basic computer knowledge can use it without any training.

2.6. User Documentation

The software will have a built-in help section that shows how to use each feature. It will guide users on how to upload notes, simplify them, and export the final version.Along with this, a simple **user manual** in PDF format will also be provided. It will include screenshots and step-by-step instructions to help new users understand the system easily

2.7. Assumptions and Dependencies

- It is assumed that users already have some basic understanding of GATE subjects, so they can decide what kind of simplification they want.
- The system will work best when there is a stable internet connection, especially if it's deployed online.
- If any external NLP tools or APIs (like HuggingFace) are used, they should have a free access tier or open-source license.
- It is also assumed that users will access the system through a laptop or desktop with a modern browser.

3. External Interface Requirements

3.1. User Interfaces

The software will have a clean and user-friendly interface, so that students can easily upload, simplify, and export their notes.The main screen will include:

- An **Upload button** to select PDF or text files
- A **text editor area** where students can view and edit simplified content
- A **Simplify button** to run automatic conversion using NLP
- An **Export option** to save the final notes as PDF or DOCX

The interface will also include standard buttons like **Help**, **Clear**, and **Save**. Basic error messages (like file not supported, or no content found) will be shown in simple English.More technical details of the UI design (like colors, fonts, and layout grids) will be provided in a separate UI specification document

3.2. Hardware Interfaces

The software is designed to work on **standard desktop or laptop computers**. It does not require any special hardware components or embedded systems.The user needs:

- A **keyboard and mouse** for input
- A **monitor or screen** for displaying the user interface
- At least **4 GB RAM** and a **basic processor** (Intel i3 or better) for smooth performance

There is no direct interaction between the software and any external hardware like sensors, cameras, or microcontrollers.

3.3. Software Interfaces

This software is designed to work on normal desktop and laptop computers. It does not require any special hardware or external devices.The system will run smoothly on devices that have:

- A working **keyboard and mouse**
- Basic **file storage** for uploading and saving notes
- A **modern processor** (Intel i3 or above) and at least 4 GB of RAM

Since it is a lightweight web-based application, no special control systems or sensors are needed. Also, no direct hardware communication protocols (like USB or serial ports) are used in this project.

3.4. Communications Interfaces

If the software is deployed online, it will use standard **HTTP/HTTPS protocols** for communication between the user's browser and the server.All data transfers, such as uploading notes and downloading simplified content, will happen securely through these web protocols.The system does not require any other network protocols like FTP, Bluetooth, or email-based communication. It is designed to be simple and browser-based

4. System Features

4.1. Upload Notes

Description:
This feature allows the user to upload GATE notes in either PDF or plain text format.

Stimulus/Response:
User clicks the "Upload" button → system opens a file picker → selected file gets uploaded and its content is extracted.

Functional Requirements:

- FR1: System should allow uploading of PDF or .txt files
- FR2: Extract raw text content from the uploaded file
- FR3: Display extracted text in the editor area

4.2. Simplify Notes

Description:
This feature processes the uploaded notes and simplifies them using NLP or manual editing.

Stimulus/Response:
User clicks "Simplify" → system uses NLP model or prompts for manual input → simplified version appears in the editor

Functional Requirements:

- FR4: Convert technical content into simplified Hinglish or plain English
- FR5: Allow user to edit simplified content manually

4.3. Organize Notes

Description:
This feature allows the user to organize notes by topic or subject name.

Stimulus/Response:
User adds tags or selects a subject → system groups notes accordingly

Functional Requirements:

- FR7: Let users assign tags or categories to their notes
- FR8: Filter or group notes based on subject/topic

4.4.Export Notes

Description:
Allows users to download their simplified notes in PDF or DOCX format.

Stimulus/Response:
User clicks "Export" → system generates document → download starts

Functional Requirements:

- FR9: Convert final notes to PDF
- FR10: Optionally convert to DOCX format
- FR11: Ensure exported file keeps formatting and supports Unicode

5. Other Nonfunctional Requirements

5.1. Performance Requirements

- The system should be able to handle **notes of at least 20 pages (PDF or text)** without slowing down.
- All basic operations (upload, simplify, edit, and export) should complete in **less than 5 seconds** under normal internet and system conditions.
- The text simplification process using NLP should not take more than **10 seconds** even for large input files.
- The user interface should remain responsive while processing, and any delays should be clearly shown using a **progress**

5.2.Safety Requirements

- The system should include an **auto-save feature** that saves the user's progress regularly while editing notes, so that no data is lost in case the browser is accidentally closed or there is a power failure.
- If the user tries to close the tab or browser with unsaved changes, a **warning message** should be displayed.
- The software should also have a **reset button** to clear the current session, but only after confirming with the user.

5.2. Security Requirements

- If the system includes login, each user should only be able to see, edit, and export their own notes.
- All user data (notes and uploads) should be stored securely and should not be shared with anyone without permission.
- If the software is hosted online, it should use **HTTPS** to protect data during upload and download.
- Uploaded files and simplified notes should be automatically deleted after the session ends, unless the user chooses to save them.
- The system should check and block any harmful files during upload (like scripts or executables) to avoid security risks.

5.3. Software Quality Attributes

Usability: The system should be easy to use for all students, even if they are not very technical. Clear buttons, tooltips, and a simple layout will be used.

Reliability: The software should work consistently without crashing. It should autosave progress and handle common issues like large file uploads or slow internet.

Portability: The system should work smoothly on different devices and operating systems (like Windows and Linux) using any modern web browser.

Maintainability: The code should be written in a clean and modular way so that future developers can easily understand and update the system if needed.

Performance: The software should respond quickly and handle large notes without lagging.

5.4. Business Rules

- Each user will have access only to their own uploaded and simplified notes. They cannot view or edit notes uploaded by others.
- Only the owner of the notes can delete or export them.
- If login is used, users must register with a valid email ID.
- Uploaded content should follow academic use only — no unrelated or copyrighted material should be allowed.
- Notes should be used only for study purposes and not for commercial distribution.

6. Other Requirements

- The NLP tools or APIs used in this software (such as HuggingFace models) should either be **open-source** or have a **free access tier** so that the system remains affordable and usable for students.
- The export feature should fully support **Unicode text**, so that Hinglish or regional characters are not lost or misformatted in the final PDF/DOCX files.
- The software should be lightweight enough to **run without a dedicated GPU**, so that students can use it on normal laptops.
- In future versions, the software can also support **drag-and-drop uploading**, support for **images**, and optional **login-based dashboards**.
- The system should be easy to deploy both **online (web-based)** and **offline (local desktop use)**.

Appendix A: Glossary

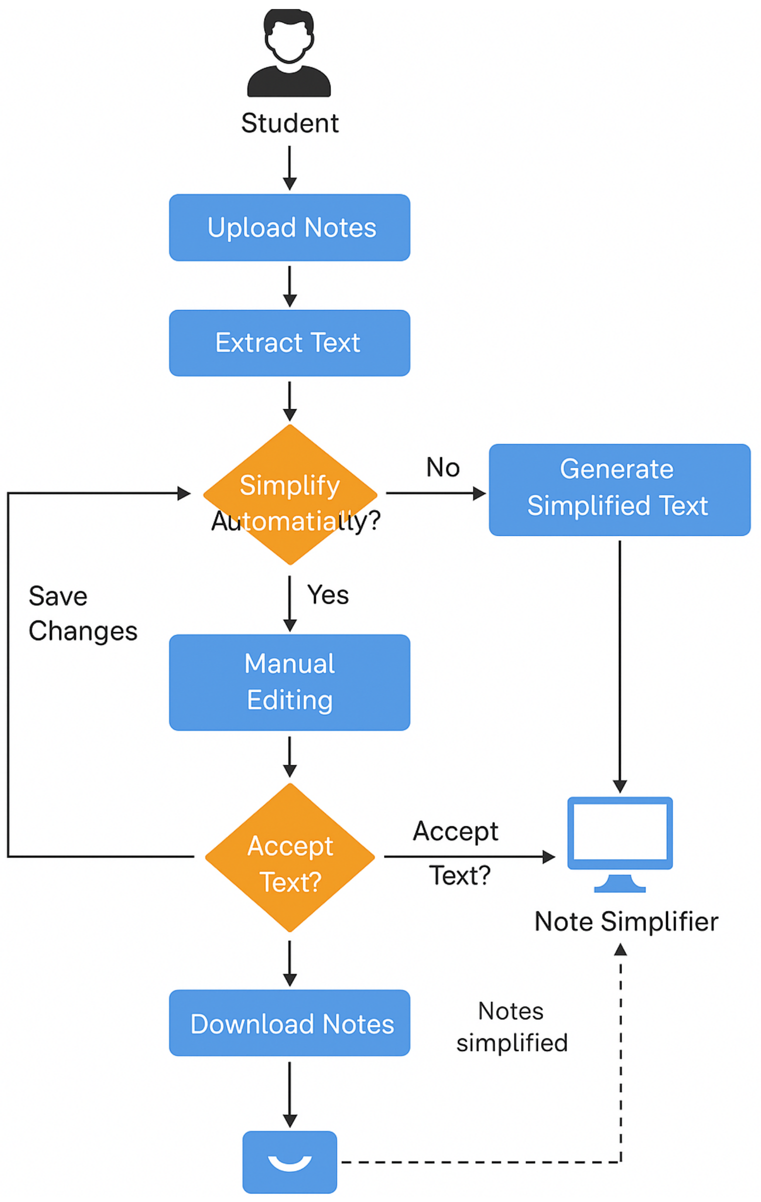
This section defines the key terms and abbreviations used in the document:

- **GATE** – Graduate Aptitude Test in Engineering
- **NLP** – Natural Language Processing
- **PDF** – Portable Document Format
- **DOCX** – Microsoft Word Document Format

- **GUI** – Graphical User Interface
- **Unicode** – A universal character encoding standard that supports multiple languages
- **HTTPS** – Secure version of HTTP for web communication
- **API** – Application Programming Interface

Appendix B: Analysis Models

Workflow chart



Appendix C: To Be Determined List

These are the decisions or details that are yet to be finalized:

- Exact **NLP model** to be used for text simplification
- Whether **user login system** will be implemented or not
- What **color scheme or layout** will be followed for the final GUI
- Whether support for **regional language translation** (other than Hinglish) will be added
- Cloud vs. local **deployment approach** for final release
- Optional support for **user feedback/rating system** after export