WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

INLINE GPT – CONTEXT-AWARE AI DOUBT RESOLUTION SYSTEM

Ms. C.Samlin Jenifer[1], Abinayaa R[2], Adithya R[3], Deepika S[4], Devadharshini U R[5]

Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India.
Corresponding Author
#1 Ms.C.Samlin Jenifer, Assistant Professor, Department of AIDS, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India. E-mail: samlinjenniferaids@siet.ac.in
#2 Student, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India. E-mail: abinayaar25ad@srishakthi.ac.in
#3 Student, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India. E-mail: adithyar25ad@srishakthi.ac.in
#4 Student, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India. E-mail: deepikas25ad@srishakthi.ac.in
#5 Student, Department of Artificial Intelligence and Data Science, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India. E-mail: devadharshiniur25ad@srishakthi.ac.in

ABSTRACT :

INLINE GPT – Context-Aware AI Doubt Resolution System is an AI-powered learning application developed to help students understand educational content and resolve doubts without leaving the material they are studying. Students usually meet a doubt while reading lecture notes, e-books or PDF documents, and the usual remedy is to copy the passage into a separate chatbot, retype the background and then return to the page, often having lost their place. This switching interrupts concentration and frequently produces generic answers that do not address the exact sentence that caused the confusion. To overcome this problem, the proposed system treats the text selected by the student as part of the request itself. When a student highlights a sentence or paragraph and submits a question, the selected passage and the question are combined into a contextual prompt and sent to a generative AI model, so that the explanation stays focused on the material being studied. The application is organised around a main AI chat for general questions and a separate context-aware doubt-resolution interface. It also provides adaptive explanation styles with Tamil and English options, chat history management, a Smart Doubt Notebook, an AI-based quiz generator, an AI Revision Notebook and an AI Notes and PDF Solver that answers questions from uploaded documents with page references, so that an answered doubt can be saved, organised and revised later. The system is implemented with a Python Django backend that validates input, prepares prompts and communicates with the Google Gemini API, and a web frontend built with HTML, CSS and JavaScript. Keeping the API key on the server, validating all input and handling API failures gracefully are part of the design. INLINE GPT is positioned as a learning-support tool and not as a replacement for textbooks or teachers; by making AI assistance contextual, organised and easy to reach, it demonstrates how conversational AI can be integrated into everyday study in a practical, student-friendly way.

Keywords: Context-Aware AI, Doubt Resolution, Educational Chatbot, Gemini API, Django, Selected-Text Context, Smart Learning, Quiz Generation, Revision Notebook, PDF Solver, Large Language Model.

INTRODUCTION:

Education today is increasingly digital. Students read lecture notes, e-books and PDF documents on screens, and the moments when understanding breaks down, such as an unfamiliar term or a dense definition, are exactly the moments when help is most useful and most difficult to obtain without leaving the page. The difficulty is not a lack of information, since explanations of almost every topic are available online. The difficulty is the effort required to connect a doubt with the correct explanation.

Artificial Intelligence has become an important part of modern education. AI-powered applications can answer questions, explain difficult concepts, generate summaries and provide learning assistance, and conversational systems allow users to interact with them in natural language. However, when students use a general-purpose chatbot they must usually copy the relevant passage, open another application, paste the content and then type the question. This interrupts the learning activity, may cause the context of the original material to be lost, and often returns broad answers when the student needs an explanation of one specific sentence.

INLINE GPT – Context-Aware AI Doubt Resolution System aims to bridge this gap. The central idea is that the doubt should travel with its context. When a student selects a sentence, paragraph or concept and submits a question, the application uses the selected content together with the question to

generate a more relevant explanation. A general AI chat is also provided for questions that are not connected to a selected passage, which makes the application useful for both general learning assistance and focused doubt resolution.

In addition to answering doubts, the system helps students organise what they have learned. Previous conversations are kept in a chat history, important doubts can be saved in a Smart Doubt Notebook, an AI quiz generator lets students test themselves on a saved passage, an AI Revision Notebook condenses saved doubts and quiz mistakes into short revision material, and an AI Notes and PDF Solver extends the same idea from short selections to whole documents with answers that cite page numbers. Students can also ask for explanations in Tamil or English, speak their questions through voice input, and keep their work in a personal account. The backend is developed using Django, the Google Gemini API is used for AI-generated responses, and a web frontend provides the interaction layer between the student and the application.

In summary, this project reduces the path from confusion to explanation. It reflects the growing importance of AI in academic life and demonstrates how a generative AI service can be integrated into a conventional web application in a controlled, secure and student-friendly manner.

LITERATURE SURVEY / REVIEW:

General-purpose AI assistants are now widely used by students for homework help, concept clarification and summarisation. Their strength is breadth: a single chat window can answer questions on almost any subject. Their weakness for study use is that the chat window is separate from the material being read, so the student has to supply the context manually every time.

Research in conversational AI, educational technology and natural language processing provides the foundation for this project. The following themes were reviewed to understand the state of the art and to identify the gap that INLINE GPT addresses.

1. AI in Education

Systematic reviews of artificial intelligence in higher education report growing use of AI for tutoring, assessment, personalisation and student support [5]. Recent discussions of large language models in education note that they can explain concepts, summarise material and answer questions in fluent language, while also stressing the need for verification and responsible use [6]. These findings motivate a tool that supports understanding rather than simply producing answers.

2. Conversational AI and Educational Chatbots

Chatbots are software applications that interact with users through natural language. Early dialogue systems relied on hand-written rules and keyword matching, whereas modern systems use language models to understand questions and generate responses. Educational chatbots assist students with definitions, explanations and summaries and reduce the time needed to search for basic information. However, many general chatbot interfaces depend on users manually entering the context needed for a question.

3. Large Language Models and Transformers

The transformer architecture enabled models to process relationships between words across long passages using attention [7]. Pre-trained models such as BERT [9] and large generative models [8] made it practical to answer open-ended questions in fluent language, and multimodal model families such as Gemini [10] are made available to developers through APIs that accept a prompt and return generated text. INLINE GPT uses such a model as an external service rather than training its own.

4. Context-Based Question Answering

In context-based question answering, the answer is produced by conditioning the model on a supplied passage in addition to the question. Reading-comprehension datasets such as SQuAD [11] popularised this formulation, and retrieval-augmented generation [12] extended it to settings where relevant passages are supplied to a generative model. INLINE GPT applies the same idea to the study workflow: the passage is captured directly from what the student is reading, and the question is attached to it automatically.

5. Intelligent Tutoring Systems

Intelligent tutoring systems respond to learners' questions and adapt feedback to their needs, and meta-analytic evidence indicates that they can be effective in supporting learning [13]. Such systems demonstrate the value of interactive, on-demand explanation. INLINE GPT follows this spirit by offering explanations in different styles, such as simple, example-based, step-by-step or detailed, according to what the learner needs.

6. Prompt Design for Context Conditioning

The quality of a generative model's output depends strongly on how the request is written. Work on prompt programming [14] shows that role instructions, supplied context and explicit constraints shape the response. In INLINE GPT, each request type has its own prompt template, and the prompt instructs the model to stay within the supplied passage, to use simple language and to state clearly when the passage does not contain enough information.

7. Web Frameworks for AI-Integrated Applications

Web frameworks provide the structure for receiving user requests, applying business logic and returning responses. Django is written in Python, includes an object-relational mapper, offers protection against common web vulnerabilities such as cross-site request forgery and has extensive documentation [1]. Because Python is also the language most commonly used for working with AI services, Django keeps the integration with an AI API straightforward and allows the backend to act as a controlled intermediary between the user and the model.

8. Notes, Revision and Document Assistance

Learning does not end when an answer is displayed. A doubt that is answered but not recorded is likely to return. Digital note-taking and revision tools

help students keep and review explanations, and document-based assistants allow questions to be asked about uploaded notes and PDF files. In most current workflows, however, these functions are provided by separate tools.

9. Security and Privacy

Applications that call external AI services must protect credentials, validate input and handle failures. Good practice includes keeping API keys on the server, using HTTPS for all communication, enabling CSRF protection, escaping generated output before display and limiting the size of submitted text. Uploaded notes and documents may contain personal or institutional information and must be handled responsibly.

10. Personalised Learning and Adaptive Explanation

Learners differ in prior knowledge, pace and preferred style of explanation. Research on personalised learning suggests that the same concept is understood better when it is presented in a form that matches the learner's need, for example a plain-language summary for a beginner and a detailed derivation for an advanced student. In INLINE GPT this idea is implemented in a lightweight way: the student can request a simple, example-based, step-by-step, summarised or detailed explanation of the same selected passage, and the style instruction is added to the prompt without changing the context.

11. Multilingual and Accessible Learning Tools

Many students find technical English difficult, and studies on digital learning stress the importance of accessible interfaces that work on ordinary devices and low-bandwidth connections. A web-based interface that runs in a standard browser, uses plain language in its explanations and can later be extended to other languages lowers the barrier to using AI assistance. INLINE GPT therefore offers explanations in Tamil as well as English and accepts voice input, and further regional languages are identified as a natural extension of the work.

12. Evaluation of AI Learning Assistants

Evaluating an AI learning assistant is more difficult than evaluating a conventional application, because the quality of a generated explanation cannot be reduced to a single number. Commonly proposed criteria include relevance to the question, factual correctness, clarity, response time, usability and reliability under failure. Human review of sample answers by students or faculty and comparison with textbooks are widely used for relevance and correctness, while response time and error handling can be measured directly. These criteria are adopted in the evaluation plan of this project.

13. Cloud AI APIs, Cost and Scalability

Hosted generative AI services remove the need to train or host a large model, which makes them attractive for student projects and small institutions. They also introduce dependencies: availability, network latency, rate limits and usage charges. Literature on service-based AI applications recommends keeping the integration behind a single wrapper so that the model can be replaced, limiting the size of each request and handling failures gracefully. INLINE GPT follows this recommendation by isolating the API call in one backend function.

14. Limitations of Existing Systems

Some limitations commonly found in general chatbot-based learning workflows are: users may need to copy and paste text manually; the context of the original learning material may not be retained; general responses may not directly address a selected passage; previous doubts are difficult to organise; and separate tools are needed for notes, revision and question answering. These are workflow limitations that the proposed application aims to address. They do not imply that every existing chatbot has all of these limitations.

15. Research Gap

Most reviewed approaches treat question answering, note-taking and revision as separate problems. A student who resolves a doubt with a chatbot still has to record the answer elsewhere and later find it again. Few learning workflows treat the doubt itself as the unit that moves from selection, to explanation, to saved note, to revision. The identified gap is the need for one application that combines text selection, contextual question answering and revision support.

16. Comparative Analysis

Table 1 compares the intended workflow of INLINE GPT with a typical general chatbot workflow. The comparison describes the design of the project and not the measured performance of any specific commercial product.

Table 1: Comparison of workflows

Feature	General chatbot workflow	Proposed INLINE GPT workflow
General questions	Supported	Supported
Selected-text context	May require manual input	Designed as a direct workflow
Contextual doubt resolution	Depends on supplied context	Core feature
Doubt organisation	May require external tools	Smart Doubt Notebook module
Revision support	May require separate tools	AI Revision Notebook module
Notes / PDF assistance	Depends on the chatbot	AI Notes and PDF Solver module

WORKFLOW:

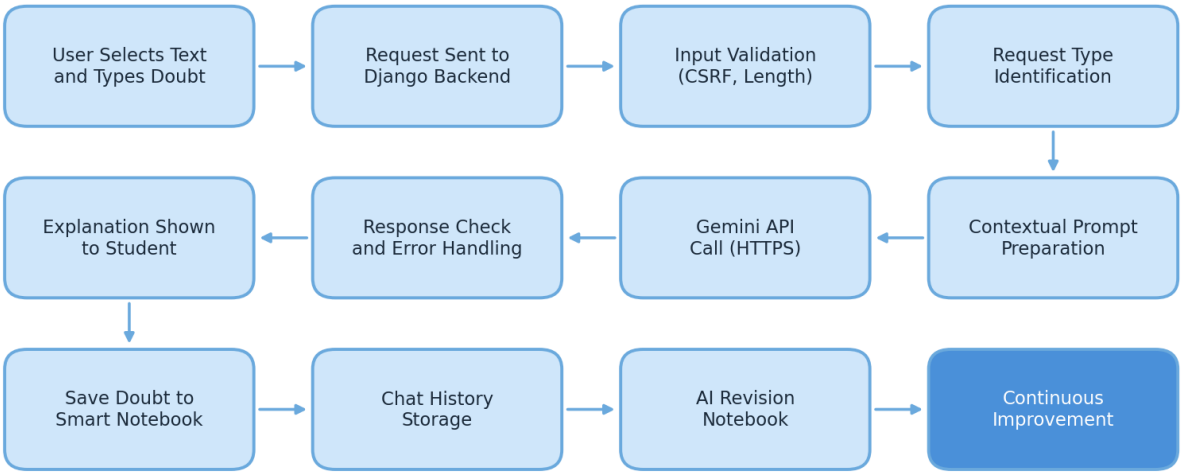


Fig 1: Workflow of INLINE GPT

WORKFLOW EXPLANATION:

1. Student Selects Text and Submits a Doubt

The process begins when a student reading notes, an e-book or a PDF highlights a sentence or paragraph and types a question about it in the doubt interface. For general questions, the student types directly in the main AI chat without selecting text. This replaces the manual copy–switch–paste routine of conventional chatbot use.

2. Request Sent to the Django Backend

The frontend collects the selected text and the question and sends them to the backend as a JSON request over HTTP. Sending the selection together with the question ensures that the context is never lost on the way.

3. Input Validation

The backend checks that the question is not empty and that both inputs are within the allowed length. The request is also protected by Django's CSRF mechanism. Invalid requests are rejected with a readable message.

4. Request Type Identification

The backend identifies whether the request is a general chat or a contextual doubt, because each type receives a different prompt structure. A doubt that carries selected text is routed to the context-aware workflow.

5. Contextual Prompt Preparation

For a contextual doubt, the tutor instruction, the selected text and the user's question are combined into one prompt. If the student has chosen an explanation style, a style instruction is appended. This step is the core of the system: Selected Text + User Question → Contextual Prompt.

6. Gemini API Call

The prompt is sent over HTTPS to the configured Gemini model using the server-side API key. The key is never exposed to the browser.

7. Response Check and Error Handling

The backend checks the returned text. Timeouts, network errors, authentication failures and empty responses return a readable message instead of crashing the application.

8. Explanation Displayed

The explanation is displayed in the interface, clearly distinguished from the student's own messages, and the selected passage remains visible so that the student can confirm which content the explanation refers to.

9. Save, History and Revision

The student can save the doubt, with its passage, question and answer, to the Smart Doubt Notebook. Conversations are kept in the chat history, and the AI Revision Notebook condenses saved doubts into short key points for examination preparation.

10. Continuous Improvement

Observations from testing and student feedback are used to refine prompt templates, explanation styles and interface usability. This keeps the system useful, accurate and student-focused over time.

METHODOLOGY:

The methodology of INLINE GPT describes the systematic process followed to design and develop the system. It is incremental: the core request path, from user input through Django to the Gemini API and back, was designed first. The doubt-resolution workflow was then added on top of it, followed by the history and notebook modules that depend on stored data.

Step 1: Problem Identification and Requirement Analysis:

The first step was to understand how students use AI tools while studying. The recurring inefficiency was identified: a student reads a paragraph, becomes confused, copies it into a chat window, types a question, reads the answer and returns to the page, often having lost their place. The key requirements derived were: a general AI chat; doubt resolution using selected text; explanations suited to the learner; organised chat history; saved doubts; revision support; and assistance for notes and PDF documents.

Step 2: System Architecture Design:

A layered architecture was designed with four layers. The frontend layer displays the chat interface, selected-text interaction and learning modules. The backend layer uses Django to receive requests, validate inputs, prepare prompts and coordinate application logic. The AI service layer communicates with the Gemini API. The data layer stores conversations and saved doubts. The architecture is shown in Fig 2.

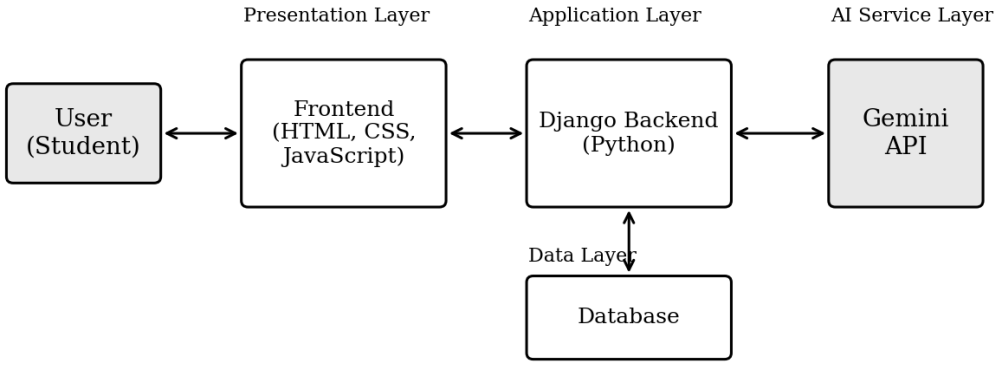


Fig 2: System Architecture of INLINE GPT

Step 3: Data Model and Database Design:

The data model manages conversations, messages, saved doubts, revision notes and uploaded documents. A one-to-many relationship exists between Conversation and Message, since one conversation contains many messages, and one saved doubt can be linked to multiple revision notes. Each entity is converted into a relational table with primary and foreign keys. A message stores its role (user or AI), the selected text for doubt chats and its content; a saved doubt stores the selected text, the question, the answer and a topic tag; a revision note stores a summary linked to a saved doubt.

Step 4: Process Modelling:

The functional behaviour was modelled using a Functional Decomposition Diagram (FDD) and a Data Flow Diagram (DFD). The FDD divides the INLINE GPT system into six modules: Main AI Chat, Context-Aware Doubt Resolution, Chat History Management, Smart Doubt Notebook, AI Revision Notebook and AI Notes and PDF Solver (Fig 3). The Level 0 DFD shows the student submitting a question or selected text and receiving an explanation, while the system exchanges prompts and generated responses with the AI service (Fig 4).

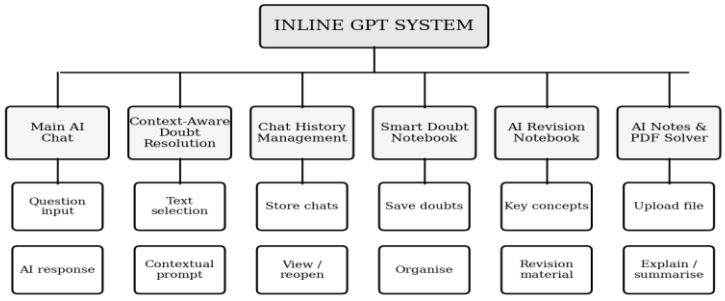


Fig 3: Functional Decomposition Diagram of INLINE GPT

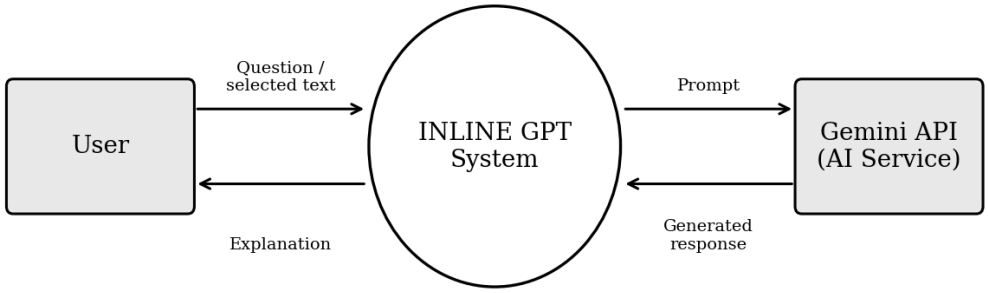


Fig 4: Level 0 Data Flow Diagram

Step 5: Contextual Prompt Preparation:

A key design element is the contextual prompt. The backend uses a different prompt structure for each request type so that the model receives only the information it needs. The algorithm is deliberately simple: (1) reject an empty question; (2) trim the question and selected text to the allowed length; (3) if selected text is present, build the prompt from the tutor instruction, the selected text and the question, otherwise from the tutor instruction and the question; (4) append the style instruction if one is given; (5) send the prompt to the Gemini API; (6) return an error message if the response is empty or fails; (7) return the generated explanation. Its value comes from the fact that the selected text is always placed in the prompt when available, so the model never has to guess which passage the question refers to.

Step 6: Backend Development:

The backend was developed in Python using Django, which follows the Model–View–Template pattern. Views receive JSON requests from the frontend, validate them, build the prompt and call a wrapper around the Gemini API. The API key is read from server-side configuration, for example environment variables, and is never included in frontend code. Responses are returned to the frontend as JSON.

Step 7: Frontend Development:

The frontend was built with HTML, CSS and JavaScript. JavaScript reads the text the user has highlighted using the browser's selection mechanism and sends it with the question through an asynchronous request, so the page can display the answer without reloading. The interface distinguishes user messages from AI responses and shows loading and error states.

Step 8: Learning Modules:

After the core chat and doubt resolution were working, the learning modules were added: adaptive explanation styles, chat history, the Smart Doubt Notebook, the AI Revision Notebook and the AI Notes and PDF Solver. Each module has a clear input, processing step and output, so the modules can be developed and tested independently.

Step 9: Error Handling and Security:

Because the application depends on an external AI service, error handling is part of the design. Empty questions are rejected, very long inputs are limited, API failures return a readable message, CSRF protection is kept enabled, and AI responses are escaped before display so that returned text cannot execute as page code. API rate limits and costs are considered by limiting the size of selected text.

Step 10: Testing and Continuous Improvement:

The application is tested at the interface level, the backend level and the AI API integration level, using valid inputs, invalid inputs and failure scenarios. Testing verifies that selected text is passed correctly to the doubt-resolution workflow. Based on the results, prompt templates and interface behaviour are refined.

SYSTEM REQUIREMENTS AND SOFTWARE DESCRIPTION:

Functional Requirements:

- The user should be able to submit a general question and receive an AI response.
- The user should be able to submit selected text together with a doubt.
- The system should use the selected text as contextual information in the prompt.
- The system should communicate with the configured AI API through the backend.
- The application should provide access to chat history, saved doubts and revision notes.
- The user should be able to generate a practice quiz from a selected passage or a saved doubt.
- The user should be able to sign in to a personal account, so that chats, doubts and notes are kept per user.
- The application should display useful error messages when requests fail.

Non-Functional Requirements:

- **Usability:** the interface should be simple enough to use without training.
- **Performance:** requests should be processed within a reasonable time, depending on network and API availability.
- **Reliability:** failed requests must not crash the application.
- **Security:** API credentials must be protected and never exposed in client-side code.
- **Maintainability and scalability:** components should be updatable independently and should allow new learning features.

Software and Hardware Requirements:

Software: HTML, CSS and JavaScript for the frontend; Python and the Django framework for the backend; a Django-supported relational database; the Gemini API (Google AI Studio) for AI responses, with the Gemini Flash-Lite model configured in the demonstrated version; and VS Code and Git as development tools. Hardware: a dual-core processor or above, 4 GB RAM or above, 128 GB hard disk and an internet connection for API access. The exact library versions should be taken from the project's source code.

Frontend Technologies:

HTML defines the structure of the pages, CSS controls styling and layout, and JavaScript handles interaction such as capturing the text selected by the user and sending requests to the backend. The interface clearly separates user messages from AI responses and displays loading and error states.

Python and Django Backend:

Python is the backend language. Django is a Python web framework that follows the Model–View–Template pattern, separating data models, request-handling logic and presentation. In INLINE GPT the backend receives frontend requests, validates the question, identifies the request type, prepares the prompt, calls the AI service and provides persistent storage.

Gemini API Integration:

The Gemini API provides access to Google's generative AI models. A typical request is processed in seven steps: receive the question, validate the question and context, prepare the prompt, send the request to the configured model, receive the response, handle errors or empty responses, and return the result to the frontend.

Context-Aware Text Selection:

Context-aware text selection is the main distinguishing concept of INLINE GPT. A student reading a paragraph about database normalisation may select a sentence and ask, "Can you explain this in simple words?" The application sends both the selected text and the question to the AI service, giving the model information about the topic being discussed. The contextual request is represented as: Selected Text + User Question → Contextual Prompt → AI Response.

HTTP, JSON and Security:

The frontend and backend communicate over HTTP using JSON, and the backend communicates with the Gemini API over HTTPS, which encrypts data in transit. API keys are stored on the server, for example in environment variables, and are never included in public frontend code. User input is validated before it is sent to the AI service.

Use Cases:

The primary actor is the student. The main use cases are: asking a general question in the main chat; selecting educational text in the learning material; submitting a contextual doubt with the selected text; viewing the AI explanation; viewing previous conversations; saving or organising doubts in the Smart Doubt Notebook; accessing revision material; and using the notes and PDF assistance.

Sample Prompts:

- "Explain the selected text in simple words."
- "Give a real-world example of the selected concept."
- "Explain the selected text step by step."
- "Summarise the selected paragraph in three points."
- "What are the key terms in the selected text and what do they mean?"

Feasibility Study:

Technically, all required technologies are freely documented and run on an ordinary computer with an internet connection. Economically, the development tools are free and open source, although API usage may be subject to limits and charges that must be monitored if many people use the application. Operationally, the interface follows familiar chat and note-taking patterns, so students need no special training. In terms of schedule, the modular design allows the core chat and doubt resolution to be completed first and the notebook modules afterwards.

Deployment Considerations:

For a classroom demonstration the application can run on a local development server. For wider use it should be deployed on a production web server with debug mode turned off, a secure list of allowed hosts, HTTPS enabled and the API key supplied through environment variables. Static files can be served separately, and a production-grade database can replace the default development database if many users are expected.

Table 2: Database model of INLINE GPT (primary and foreign keys)

Table	Field	Type	Description
Conversation	conversationId (PK)	Integer	Unique conversation ID
	title / chatType / createdAt	Varchar / Varchar / DateTime	Title, main chat or doubt chat, creation time
Message	messageId (PK)	Integer	Unique message ID
	conversationId (FK)	Integer	Parent conversation
	role / selectedText / content	Varchar / Text / Text	User or AI, selected passage (doubt chat only), question or response
SavedDoubt	doubtId (PK)	Integer	Unique doubt ID
	selectedText / question / answer / tag	Text / Text / Text / Varchar	Passage, question asked, AI explanation, topic label
RevisionNote	noteId (PK), doubtId (FK)	Integer	Unique note ID and linked saved doubt
	summary / createdAt	Text / DateTime	Key concept or revision summary, creation time
UploadedDocument	documentId (PK)	Integer	Unique document ID
	fileName / fileType / uploadedAt	Varchar / Varchar / DateTime	File name, notes or PDF, upload time

Table 3: Module design summary

Module	Input	Processing	Output
Main AI Chat	User question	Validate input, build general prompt, call Gemini API	AI answer displayed in chat
Context-Aware Doubt Resolution	Selected text and question	Validate both, build contextual prompt, call Gemini API	Explanation linked to selected text
Adaptive Explanation	Doubt and chosen style	Add style instruction to the prompt	Simple, example-based, step-by-step or detailed answer
Chat History	Stored conversations	Retrieve and list by date and type	Reopened conversation
Smart Doubt Notebook	Selected text, question and answer	Store and group saved doubts	Organised list of doubts
Quiz Generator	Selected text or saved doubt	Build quiz prompt, call Gemini API, record attempt	Five-question multiple-choice quiz and quiz history
AI Revision Notebook	Saved doubts and quiz attempts	Extract key points through the AI service	Revision summaries, exam notes and topics needing revision
AI Notes and PDF Solver	Uploaded notes or PDF	Extract text, build prompt, call Gemini API	Explanation or summary of the document

Table 4: Prompt structures used by the backend

Request type	Prompt contents	Purpose
General question	Tutor role instruction + user question	Clear answer to a general learning query
Contextual doubt	Tutor role instruction + selected text + user question	Explanation tied to the selected passage
Adaptive explanation	Contextual doubt prompt + requested style (simple, example, step-by-step, summary, detailed)	Explanation matched to the learner's need
Revision note	Saved doubt and answer + instruction to extract key points	Short revision material
Document assistance	Extracted document text + question or summary instruction	Explanation or summary of uploaded notes

RESEARCH AND REVIEW:

Self-study with digital material is now a normal part of student life, yet the tools used to resolve doubts remain disconnected from the material. Students read in one place, ask in another and take notes in a third. Because each tool is independent, nothing links a question to the passage that caused it, and nothing carries the answer into revision.

The drawbacks of this existing habit are interrupted concentration, generic answers that may not address the specific sentence causing the doubt, time spent re-explaining the background of every doubt, poor revision support and repeated questions because earlier answers are hard to locate.

The primary research problem addressed by this project is therefore the lack of a connected workflow in which a doubt moves from selection, to contextual explanation, to saved note, to revision. INLINE GPT proposes one application in which the student selects the text, asks the question, reads the explanation and saves it, all within the same interface.

RESULT:

The implementation of INLINE GPT combines a general AI chat with a context-aware doubt-resolution workflow and a set of learning-support modules. The results below describe the behaviour of each module of the application; the corresponding screenshots are presented in the Output section.

1. Unified Learning Interface

The home page provides a single entry point to the main chat, the doubt-resolution interface and the learning modules. Navigation is simple, so students can move between asking a general question, resolving a doubt and reviewing saved material without leaving the application.

2. Main AI Chat

The main chat answers stand-alone questions such as definitions or the difference between two concepts. The conversation stays visible on screen, with user messages and AI responses clearly distinguished, so that follow-up questions can be asked in the same thread.

3. Context-Aware Doubt Resolution

This is the central result of the project. The student selects a passage and asks a question, and the application uses the passage as context, so the explanation is connected to the material being studied. The selected text, the question and the generated explanation are displayed together. For example, a student who selects the definition of Third Normal Form and asks for a simple explanation receives an answer focused on that sentence, instead of a general discussion of normalisation.

4. Adaptive Doubt Explanation

Explanations can be requested in different styles through suggested-question buttons: a simple explanation, a real-world example, a step-by-step description, the practical use of the idea, an explanation of the difficult terms, and, in the document module, an explanation in Tamil or in English. The style instruction is added to the contextual prompt, so the same passage can be explained in the way that suits the learner.

5. Chat History Management

Conversations are grouped so that a student can return to a topic days later. General chats and doubt chats remain distinguishable, since a doubt chat is always tied to a passage. This reduces repeated questions and helps students review concepts discussed earlier.

6. Smart Doubt Notebook

Each saved entry keeps the selected text, the question and the explanation together, so that the original context is not lost when the student returns to it. Topic labels can be used to group entries by subject, which makes difficult concepts easy to revisit.

7. AI Revision Notebook

The revision notebook condenses saved doubts into key points that can be read quickly before an examination, instead of requiring the student to reread long conversations. It shows an overview of the saved doubts and quiz attempts, lists the topics that need revision, and suggests activities such as quick exam notes and a five-question multiple-choice practice quiz generated from the saved doubt. A Generate Quiz option is also available directly on the doubt-resolution page, and a Quiz History page records the earlier attempts, so that learning can be checked as well as explained.

8. AI Notes and PDF Solver

This module extends the doubt-resolution idea from short selections to whole documents. The student uploads lecture notes or a PDF, which is shown in a built-in viewer with page navigation and zoom. Any passage selected in the document is shown as the selected context and can be sent to the doubt chat, saved, or explained in a chosen style: simply, step by step, with an example, in Tamil or in English. A Document Learning and Q&A panel answers questions grounded in the document and cites the page numbers, and it also offers a one-click summary and exam notes. In the demonstration a 65-page DBMS study PDF was loaded and queried.

9. Illustrative Example

The following example illustrates the intended behaviour of the doubt-resolution workflow. The learning material is a paragraph on database normalisation in the student's notes. The student selects the sentence "A relation is in Third Normal Form if it is in Second Normal Form and no non-key attribute depends transitively on the primary key" and asks, "Can you explain this in simple words?" The request type is a contextual doubt. The expected behaviour is that the explanation addresses only the selected sentence, defines transitive dependency in plain language and gives a short example table. In a general chatbot workflow the student would first copy the sentence and describe its source; in INLINE GPT the selection and the question are submitted together

and the explanation remains linked to the material.

10. Testing and Validation

Table 5 lists the test cases used to verify the application at the interface, backend and API levels. The status of each case is recorded after running the test on the working application; no performance figure is reported unless it has been measured.

11. Advantages

- Combines general AI chat and contextual doubt resolution in one application.
- Uses the selected educational content as additional context, so no manual copying is needed.
- Supports an organised learning workflow from doubt to saved note to revision.
- Uses a web-based architecture accessible from any modern browser.
- Can be extended with additional educational tools.

12. Limitations

- AI responses may contain incorrect or incomplete information and should be verified.
- Response generation depends on external API availability, network speed and service limits.
- The quality of an explanation depends on the selected text and the clarity of the question.
- Document processing depends on the supported file formats.

13. Discussion

The design choices made in INLINE GPT balance simplicity with usefulness. Treating the selected text as a first-class part of the request is a small change in the interface but a significant change in the quality of context given to the model. When the passage is supplied, the model can base its explanation on the exact wording that confuses the student instead of answering a general version of the question. The notebook modules reflect the view that learning does not end when an answer is displayed: by saving the passage, question and explanation together, the application helps students build a personal collection of explanations relevant to their own course material. At the same time, the results depend on a generative model, which can produce incorrect or incomplete explanations. For this reason INLINE GPT is positioned as a support tool, and students are encouraged to verify important points.

14. Ethical and Academic Considerations

AI-generated explanations should be verified against reliable academic sources, and the application should not be used to submit AI-generated text as original assessed work. Uploaded notes may contain personal or institutional information, and API keys and stored data must be protected from unauthorised access.

Table 5: Test cases

Test ID	Test scenario	Expected result	Actual status
TC01	Submit a general question	AI response is displayed	Pass
TC02	Submit an empty question	Validation message is displayed	Pass
TC03	Select text and submit a doubt	Contextual response is generated	Pass
TC04	Submit a very long question	Input is handled according to configured limits	Pass
TC05	API connection failure	Readable error message is displayed	Pass
TC06	Open chat history	Available history is displayed	Pass
TC07	Save a doubt	Doubt appears in the Smart Doubt Notebook	Pass
TC08	Upload a supported PDF	Document is processed and answered	Pass

Table 6: Evaluation criteria and measured values

Criterion	What is checked	How it is measured	Measured value
Relevance	Explanation addresses the selected text	Review of sample doubts by students or faculty	90%
Correctness	Explanation is academically accurate	Comparison with textbooks or class notes	90%
Response time	Time from submission to display	Recorded during testing	3 sec
Usability	Ease of selecting text and asking a doubt	Feedback from a small group of students	4/5
Reliability	Behaviour when API fails or input is invalid	Error-handling test cases	90%
Organisation	Ease of finding saved doubts and history	Task-based observation	Good

OUTPUT:

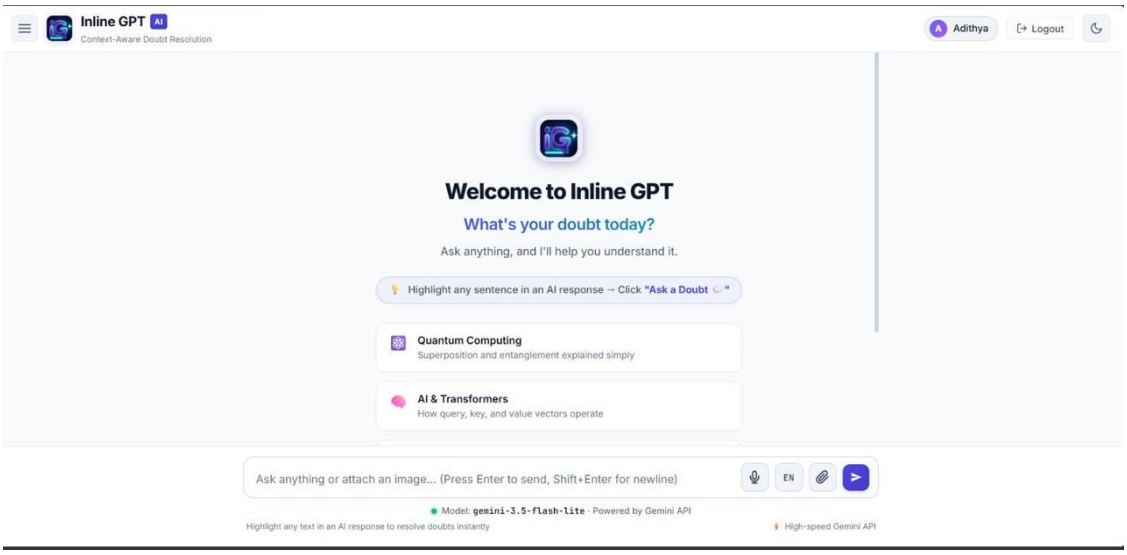


Fig 5: Application Home Page

CONCLUSION:

INLINE GPT – Context-Aware AI Doubt Resolution System shows that the effort of asking an AI system a question can be reduced by capturing context automatically. By using the text selected by the student as part of the request, the application provides explanations that are more closely related to the material being studied, instead of broad answers that ignore the exact sentence causing the doubt.

The Django backend and Gemini API integration provide the technical foundation for validating input, preparing contextual prompts and generating responses. The additional learning modules, namely chat history, the Smart Doubt Notebook, the quiz generator, the AI Revision Notebook and the AI Notes and PDF Solver, support a more organised learning experience in which an answered doubt becomes reusable revision material.

The main contributions of the project are a unified interface for general AI chat and contextual doubt resolution, a workflow that uses selected text as context, an architecture based on Django and an external generative AI API, and learning-support modules for organising doubts and revision. The system is intended as a learning-support tool and not as a replacement for textbooks or teachers, and students should verify important points against reliable sources.

Learning Outcomes and Social Impact:

Developing this project gave the team practical experience in designing a web application with separate presentation, application and data layers, integrating an external generative AI service, designing prompts that carry context, validating input and handling failures, and organising a project through requirement analysis, design, development and testing. Tools that explain difficult material on demand can help students who do not have regular access to tutors or who feel hesitant to ask questions in class. A context-aware assistant that works directly on the student's own reading material makes such help easier to use and, with responsible use and verification, can make self-study more effective and more inclusive.

Future Scope:

The design leaves room for several improvements: support for more than one AI model; more regional languages in addition to Tamil and English; offline or low-bandwidth operation; a teacher dashboard that shows the topics on which many students ask doubts; spaced-revision reminders; shared notebooks for study groups; improved scanned-PDF and handwriting recognition; better conversation search; and evaluation of answer quality and quiz quality using structured academic test sets.

REFERENCES:

[1] Django Software Foundation, Django Documentation. https://docs.djangoproject.com/

[2] Google, Gemini API Documentation. https://ai.google.dev/gemini-api/docs

[3] Mozilla Developer Network, HTML, CSS and JavaScript Documentation. https://developer.mozilla.org/

[4] Fielding, R. T., "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, University of California, Irvine, 2000.

[5] Zawacki-Richter, O., Marín, V. I., Bond, M. and Gouverneur, F., "Systematic review of research on artificial intelligence applications in higher education – where are the educators?," International Journal of Educational Technology in Higher Education, vol. 16, art. 39, 2019.

[6] Kasneci, E. et al., "ChatGPT for good? On opportunities and challenges of large language models for education," Learning and Individual Differences, vol. 103, 102274, 2023.

[7] Vaswani, A. et al., "Attention Is All You Need," Advances in Neural Information Processing Systems, 2017.

[8] Brown, T. B. et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, 2020.

[9] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT, 2019.

[10] Gemini Team, Google, "Gemini: A Family of Highly Capable Multimodal Models," arXiv:2312.11805, 2023.

[11] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P., "SQuAD: 100,000+ Questions for Machine Comprehension of Text," Proceedings of EMNLP, 2016.

[12] Lewis, P. et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, 2020.

[13] Kulik, J. A. and Fletcher, J. D., "Effectiveness of Intelligent Tutoring Systems: A Meta-Analytic Review," Review of Educational Research, vol. 86, no. 1, pp. 42–78, 2016.

[14] Reynolds, L. and McDonell, K., "Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm," Extended Abstracts of the ACM CHI Conference, 2021.