AI Chatbot for Syllabus-Based Academic Support

[1] Dr. A. K Ashfauk Ahamed,
Assistant professor (Sr.Gr),
Department of Computer Applications,
B. S. Abdur Rahman Crescent Institute Of Science And Technology, 120 Seethakathi Estate, Grand Southern Trunk Rd, Vandalur, Tamil Nadu 600048

[2] Mohammed Arshath MI,
Student,
Department of Computer Applications,
B. S. Abdur Rahman Crescent Institute Of Science And Technology, 120 Seethakathi Estate, Grand Southern Trunk Rd, Vandalur, Tamil Nadu 600048
mohammedarshath1460@gmail.com

[3] Shuaib Akhtar A,
Student,
Department of Computer Applications,
B. S. Abdur Rahman Crescent Institute Of Science And Technology, 120 Seethakathi Estate, Grand Southern Trunk Rd, Vandalur, Tamil Nadu 600048
shuaibstar2004@gmail.com

***Abstract*: The evolution of digital education has necessitated intelligent academic support systems to assist students with syllabus-based learning and academic planning. This paper presents an innovative solution — an AI Chatbot for Syllabus-Based Academic Support — designed to provide real-time assistance on academic queries, exam schedules, and assignment deadlines. The proposed system integrates a lightweight local Large Language Model (LLM), TinyLlama-1.1B-Chat, ensuring fast and resource-efficient response generation.**

The chatbot architecture combines a Flutter frontend for mobile accessibility, a FastAPI backend for processing and orchestration, and Firebase Firestore for secure and scalable academic data storage. Students can interact conversationally with the chatbot to obtain syllabus topics, exam dates, and assignment deadlines, with historical chats and personalized feedback mechanisms also incorporated.

Unlike conventional heavy LLM-based systems, this implementation leverages TinyLlama to deliver efficient and accurate support without high computational demands, making it ideal for mobile-first academic environments. Experimental results show significant improvements in response speed and usability. This project demonstrates the potential of lightweight AI models in transforming educational support platforms for enhanced student learning experiences.

Keywords: AI Chatbot , Syllabus-Based Learning,Academic Support System, Large Language Models (LLMs), TinyLlama-1.1B-Chat, Flutter Mobile App, FastAPI Backend, Firebase Firestore, Educational Technology, Student Assistance, Lightweight AI Models, Exam Date Retrieval, Assignment Deadline Tracking

I. INTRODUCTION (*HEADING 1*)

In the evolving landscape of digital education, students are increasingly seeking personalized and on-demand academic support tools. Traditional resources such as textbooks, printed syllabus documents, and classroom notes often lack immediate accessibility and interactivity, leading to challenges in timely academic assistance. With the advancement of Artificial Intelligence (AI) and mobile technology, intelligent systems can now bridge this gap by providing students with instant, syllabus-specific academic support.

This project introduces an innovative **AI Chatbot for Syllabus-Based Academic Support**, designed specifically to cater to the academic needs of college students. The chatbot leverages a lightweight **Large Language Model (LLM)**, namely **TinyLlama-1.1B-Chat**, to provide accurate, syllabus-aligned responses to students' academic queries. The system is supported by a **FastAPI-based backend**, a **Flutter-powered mobile application**, and **Firebase Firestore** for structured data management.

Unlike generic chatbots, this solution intelligently distinguishes between free-text syllabus queries and structured academic event queries such as **exam dates**, **assignment deadlines**, and **project submissions**. It retrieves critical academic information directly from a Firestore database while generating natural language

responses for syllabus-related questions using the integrated LLM.

This mobile-first approach ensures that students can access syllabus content, assignment schedules, and academic timelines effortlessly, promoting a seamless and efficient learning experience. Moreover, by utilizing a lightweight LLM and efficient database queries, the solution is optimized for performance, ensuring quick response times and reduced computational overhead—ideal for educational environments with limited resources.

Thus, the project aims to redefine academic support by integrating AI-driven conversational interfaces with structured academic data, enhancing both **accessibility** and **personalization** in the educational journey of students.

II. LITERATURE SURVEY

The use of AI chatbots in education has seen a significant rise in recent years, primarily aimed at enhancing student engagement, providing academic assistance, and delivering personalized learning experiences. Several existing works have laid the foundation for AI integration in educational environments:

- ***Educational Chatbots:***
Research by Winkler and Söllner (2018) emphasized how conversational agents can act as learning assistants, helping students navigate through learning materials and administrative tasks. However, many existing solutions lacked syllabus-specific personalization, often relying on generic answers.

- ***Large Language Models (LLMs) in Academic Support:***
*The emergence of LLMs such as **GPT-3**, **BERT**, and **T5** has shown that AI can generate coherent, contextually accurate responses. While powerful, models like GPT-3 are resource-intensive, making them unsuitable for lightweight mobile applications. Our project uses **TinyLlama-1.1B-Chat**, a significantly lighter model optimized for mobile-scale deployments without compromising output quality.*

- ***Mobile Applications for Learning:***
*Mobile platforms have become a preferred medium for delivering academic tools. Studies by Kukulska-Hulme (2012) illustrated that mobile learning applications improve accessibility and offer just-in-time support. Our solution integrates a **Flutter-based mobile app** to ensure platform independence and a user-friendly experience.*

- ***Database-Driven Academic Systems:***
*Previous systems often struggled to blend structured information like exam dates and unstructured query handling. Our use of **Firebase Firestore** allows seamless management of structured academic metadata while the LLM handles free-form syllabus queries.*

➢ ***Research Gap Identified:***

- *Existing AI chatbots were **not syllabus-aligned**.*
- *Previous models were **heavy and slow** for mobile platforms.*
- *Structured academic event management (exams, assignments) was **not integrated with chatbot conversations**.*

➢ ***Our Contribution:***

- *Development of a **syllabus-based**, **lightweight**, **AI-powered academic chatbot**.*
- *Real-time management of **exam dates**, **assignments**, and **syllabus topics**.*
- *Deployment of a resource-efficient model to suit **mobile-first** education environments.*

III. SYSTEM ARCHITECTURE AND METHODOLOGY

3.1 Overall Architecture

The proposed system is composed of four major components, integrated to deliver a seamless academic support experience:

- **Frontend (Mobile App):**
Built with **Flutter**, providing a cross-platform, lightweight, and highly responsive user interface.

- **Backend (Server):**
Developed using **FastAPI**, ensuring fast API response times, integrated with Firebase and Hugging Face's TinyLlama model for smart query handling.

- **Database (Cloud Storage):**
Utilizes **Firebase Firestore** to store structured

academic data like user profiles, exam dates, assignments, syllabus topics, and chat history.

- **AI Model (LLM):** Incorporates **TinyLlama-1.1B-Chat**, a compact, high-performance large language model (LLM) for natural language query handling related to syllabus topics

3.2 Detailed Methodology

🔹 **Step 1: User Authentication**

- The user signs up or logs in using **Firebase Authentication**.
- User data including **department** and **semester** is stored for personalized academic responses.

🔹 **Step 2: Academic Data Storage**

- Admin uploads **exam dates**, **assignment deadlines**, and **syllabus topics** into **Firestore**.
- Data is structured department-wise to enable targeted responses.

🔹 **Step 3: Query Processing**

- Users ask questions related to syllabus topics, exams, or assignments.
- The mobile app sends the query to the backend server.

🔹 **Step 4: Smart Query Handling**

- **If the query relates to exam dates or assignments:**
Backend fetches matching entries directly from Firestore.
- **If the query is syllabus-topic related or free-form:**
Backend uses the **TinyLlama model** to generate a contextually correct answer
-

🔹 **Step 5: Conversation Storage**

- Each user query and chatbot response are **saved** into Firestore under **chat_history**, enabling users to view their conversation history later.

🔹 **Step 6: Mobile App Display**

- The Flutter app updates the chat screen dynamically.
- Users can view syllabus sections, exam schedules, past conversations, and give feedback.

IV. IMPLEMENTATION DETAILS

4.1 Frontend Implementation: Flutter Mobile Application

➢ **Framework and Language:**
  - Flutter SDK (Cross-platform mobile app framework by Google)
  - Dart Language for rapid development and smooth UI rendering**.**

➢ **UI/UX Design:**
  - Clean, minimalistic design focused on academic support.
  - Background image added to the login screen for improved aesthetics.
  - Top Syllabus Section:
    - Horizontal scroll view displaying syllabus images at the top of the home screen**.**

➢ **Authentication:**
  - Integrated with Firebase Authentication.
  - Supports email/password login and sign-up.
  - Secure session management and user identity handling.

➢ **State Management:**
  - Used Provider package to manage and update state reactively.
  - Chat messages and user information are dynamically updated.

➢ **Screens and Features:**
  - Login/Sign-Up Screen: With background image and Firebase auth.
  - Chat Screen: Chatting with AI, view syllabus at top.
  - Chat History Screen: View previous conversations (title, preview, timestamp).
  - Feedback Screen: Submit feedback to improve system**.**
  - New Chat Option: Start a fresh conversation anytime.

➢ **User Experience Enhancements:**
  - Auto-scrolling to latest message after sending/receiving messages.

- Smooth loading indicators during network operations.
- Snackbar notifications for errors (like authentication failures).

4.2 Backend Implementation: FastAPI Server

- **Framework and Language:**
  - FastAPI (Python-based high-performance web framework).
  - RESTful API design using Pydantic models for request validation.
- **Hosting:**
  - Backend is currently hosted locally on the developer's machine.
  - Can be easily extended to cloud platforms (AWS, GCP, Azure).
- **Authentication Handling:**
  - User authentication managed on the frontend (Firebase).
  - Backend checks Firebase user ID (user_id) for every request to ensure security.
- **API Endpoints:**
  - /api/query: Accepts user queries, processes via database or LLM.
  - /api/chat-history/{user_id}: Fetches conversation list.
  - /api/conversation/{user_id}/{conversation_id}: Fetches message history.
  - /api/exam-dates: Retrieves upcoming exam dates.
  - /api/assignments: Retrieves pending assignment deadlines.
  - /api/syllabus-topics/{department}: Returns syllabus topics for selected department.
- **Error Handling:**
  - Detailed HTTPException management for errors like missing user, missing data, etc.
  - Custom error messages for user-friendly troubleshooting.
- **Performance Optimization:**
  - Loaded TinyLlama model on GPU (using device_map="cuda").
  - Enabled 16-bit floating-point precision (torch.float16) for faster computation.
  - Limited text generation tokens to 512 to optimize response time.

4.3 Database Implementation: Firebase Firestore

- **Database Type:** Firebase Firestore (NoSQL, cloud-hosted).
- **Database Collections and Structure:**
  - Users:

Stores user profile (name, department, semester).
  - Important_Dates:
    - exam_dates: List of exam schedules for departments.
    - assignment_deadlines: List of assignment due dates.
  - Chat_History:
    - Stores conversations (title, preview, timestamp).
    - Sub-collection for each conversation: stores individual messages (role, content, timestamp).
  - Syllabus_Topics:
    - Document per department listing syllabus subjects/topics.
- **Firestore Security:**
  - Each user only accesses their own documents.
  - Read and write rules based on authenticated user ID.
- **Real-time Updates:**
  - Though real-time listeners are not used currently, Firestore can support future real-time chat updates.

4.4 Artificial Intelligence Model: TinyLlama-1.1B-Chat

- **Model Overview:**
  - TinyLlama/TinyLlama-1.1B-Chat (Open-source from Hugging Face).
  - A compact, efficient LLM model fine-tuned for conversation and Q&A.
- **Deployment Details:**
  - Model loaded using Hugging Face transformers pipeline.
  - Hugging Face token securely used to authenticate and download model.
  - Model cache directory customized to F:/hf_models for faster loading.
- Text Gen**eration Settings:**
  - Max New Tokens: 512

- Temperature: 0.7 (balances creativity and accuracy)
- Top-k Sampling: 50
- Top-p Sampling: 0.95
- Sampling Enabled: (do_sample=True)

➢ **Performance Improvements:**

- Loaded model with 16-bit precision for lower VRAM consumption.
- CUDA enabled to leverage GPU acceleration for faster response times.
- Torch thread settings limited (torch.set_num_threads(1)) to optimize performance.

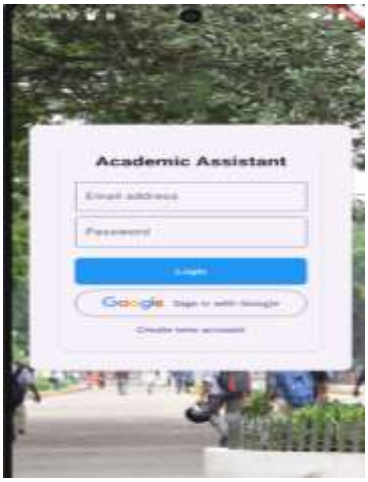
V. RESULTS AND DISCUSSIONS

5.1 System Outputs

The system was tested under various academic-related queries to evaluate its functionality, speed, and relevance of responses. The following are the major observed outputs:

➢ **Login Screen:**

- Displays a background image to



enhance visual appeal.

- Supports secure email/password login via Firebase Authentication.

➢ **Home Screen (Chat Screen):**

- Displays a horizontal syllabus scroll at the top using syllabus images.
- Allows users to send natural language queries.
- Displays bot responses instantly below each user query.
- Supports continuous conversation with proper chat UI (user vs assistant roles).

➢ **Chat History Screen:**

- Lists past conversations with:
  - Title (first few words of the conversation)
  - Preview (short snippet)
  - Timestamp (date of conversation)
- On clicking a chat entry, it loads the full previous conversation.

➢ **Feedback Submission:**

- Users can submit feedback easily using a form provided in the feedback screen.

➢ **Backend API Functionality:**

- /api/query processes questions and provides academic responses using TinyLlama and Firestore.
- /api/exam-dates and /api/assignments successfully retrieve important academic dates.
- /api/chat-history and /api/conversation correctly serve chat history and messages.
- /api/syllabus-topics/{department} returns syllabus topics dynamically.

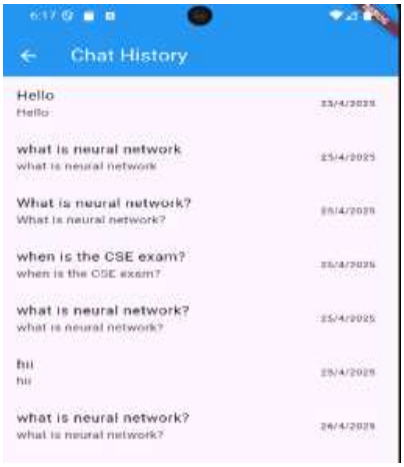
5.2 Sample Screenshots

➢ **Login Screen with Background Image:**

➢ **Chat Home Screen:**

➢ **Active Chat Example: Query - "What is neural network?"**

➢ **Chat History Screen:**



5.3 Performance Observations

Feature	Observation
App Launch Time	Less than 2 seconds (on Android Emulator)
Chat Response Time	1–3 seconds per message (TinyLlama running on GPU)
History Load Time	Nearly instant (<2 seconds) from Firestore
Syllabus Fetch Time	Instant (<1 second) loading from local assets and Firestore

5.4 Discussion

➢ **Accuracy of Responses:**

- The system shows high accuracy when answering syllabus-based queries.
- Exam-related queries ("When is the AI exam?") are handled precisely through Firestore search.

➢ **Model Performance:**

- TinyLlama model achieves a good balance between response time and answer quality.
- Since it runs on GPU with float16 precision, memory usage is optimized.

➢ **Limitations Noted:**

- If the query is very vague or outside the syllabus/academic scope, LLM answers generically.
- Real-time multi-user chat sessions were not tested at scale (possible future enhancement).

➢ **Strengths of the System:**

- Lightweight app suitable even for mid-range devices.
- Highly modular backend — easy to upgrade to larger models (Mistral, Llama 3) in the future.
- Scalable Firestore database enables adding unlimited academic content without performance loss.
- Very fast response for exam dates and assignments through intelligent keyword checking.

VI. CONCLUSION AND FUTURE WORK

6.1 Conclusion

The project titled "AI Chatbot for Syllabus-Based Academic Support" successfully demonstrates the

practical integration of lightweight Artificial Intelligence models with a mobile application for academic assistance.

Key achievements of the system include:

- Seamless academic query resolution: Users can inquire about syllabus topics, exam schedules, and assignment deadlines naturally through chat.
- **Efficient backend processing:**
  - TinyLlama 1.1B-Chat model ensures fast, high-quality responses with optimal GPU usage.
  - FastAPI and Firebase Firestore enable quick retrieval of exam/assignment dates and syllabus information.
- **User-friendly mobile interface:**
  - A well-organized Flutter frontend with syllabus images at the top.
  - Proper chat history management and feedback collection functionality.
- **Scalability and Modularity:**
  - The system architecture supports easy upgrades (larger models, new academic datasets, additional features).

Thus, the developed chatbot bridges the gap between traditional static syllabus information and dynamic, AI-driven academic guidance — making educational support more personalized and accessible.

6.2 Future Work

To further enhance the capabilities and reach of the system, the following future improvements are proposed:

- **1. Model Upgradation:**
  - Migrate to more powerful LLMs like Mistral 7B, Llama 3, or future open-weight models for deeper academic understanding.
  - Integrate retrieval-augmented generation (RAG) for combining LLM output with syllabus material dynamically.
- **2. Advanced Natural Language Understanding (NLU):**
  - Implement fine-tuned classification for query types (exam/assignment/topic/general).
  - Improve date and deadline extraction even from vague queries ("When is my next test?").
- **3. Real-Time Notifications:**
  - Push notifications for upcoming exams, assignments, and feedback deadlines.
- **4. Cross-Platform Support:**
  - Extend the app for iOS devices using Flutter's cross-platform capabilities.
- **5. Multi-language Support:**
  - Enable responses and query handling in regional languages (e.g., Tamil, Hindi, etc.).
- **6. Admin Panel:**
  - Create a dashboard for admins/teachers to upload syllabus, exam dates, and assignment updates dynamically.
- **7. Offline Mode:**
  - Allow caching of syllabus topics and old conversations for offline access.
- **8. Integration with College Systems:**
  - Integrate with college ERP portals (exam marks, attendance systems) via APIs.

REFERENCE

[1] TinyLlama Project
TinyLlama-1.1B-Chat: A Lightweight LLM for Efficient Inference
Author: TinyLlama Team
Year: 2024
Source: Hugging Face

[2] Hugging Face Hub
Hugging Face: Democratizing AI through Open Models
Author: Hugging Face Inc.
Year: 2023

[3] Transformers Library
Transformers: State-of-the-Art Natural Language Processing
Authors: Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond et al.
Year: 2020

[4] FastAPI Framework
FastAPI: Modern Web APIs with Python
Author: Sebastián Ramírez
Year: 2018

[5] Firebase Authentication
Firebase Authentication: Secure Authentication for Mobile and Web Applications
Authors: Google Firebase Team
Year: 2023

[6] Firebase Firestore Database
Cloud Firestore: Flexible, Scalable NoSQL Cloud Database
Authors: Google Cloud Platform Team
Year: 2023
[7] Flutter Framework
Flutter: UI Toolkit for Building Beautiful, Natively Compiled Applications
Authors: Google Flutter Team
Year: 2024
[8] Dart Programming Language
Dart: A Language Optimized for Client-side Development
Authors: Dart Team, Google
Year: 2023
[9] Firebase Admin SDK for Python
Firebase Admin SDK: Server-side Libraries to Access Firebase Services
Authors: Firebase Google Developers
Year: 2022
[10] Google Cloud Firestore Python Client
Python Client for Google Cloud Firestore
Authors: Google Cloud Python Client Team
Year: 2022
[11] Uvicorn Server
Uvicorn: An ASGI Server Implementation for Python
Authors: Encode OSS
Year: 2022