	Hacettepe University Computer Engineering Department **BBM479/480 End of Project Report**

Project Details

Title	Beytepe Information Assistant
Supervisor	Assoc. Prof. Aydın Kaya

Group Members

	Full Name	**Student ID**
1	Ali Diren Şenyurt	2200356041
2	Kazım Halil Kesmük	2200765031
3	Murat Eren Güven	2200356059

Abstract of the Project (/ 10 Points)

In recent years, students attending large university campuses have been increasingly burdened by the widespread dispersion of essential information across various digital platforms and physical locations. Whether it's accessing daily dining hall menus, checking shuttle bus schedules, or staying updated with departmental announcements, students often find themselves navigating a disjointed and fragmented system that complicates their campus experience.

In response to this ongoing challenge, we developed the Beytepe Information Assistant, a comprehensive mobile application specifically designed to serve the needs of students at Hacettepe University. The primary aim of this project is to consolidate vital campus-related information into a single, user-friendly platform while also introducing an intelligent assistant feature that utilizes Retrieval-Augmented Generation (RAG) techniques to provide accurate and helpful answers to frequently asked academic questions.

The architecture of the system is composed of three core modules: a cross-platform mobile application developed using the Flutter framework, a backend server built with FastAPI, and an intelligent assistant module that operates based on the RAG paradigm. The backend is responsible for collecting and updating real-time data from official university sources, including the university dining hall website and the announcement page of the Computer Engineering Department. For the shuttle bus schedules, which are not frequently updated, the data is integrated as static content.

The RAG component relies on a carefully curated and manually preprocessed collection of documents sourced from the official website of the department. These documents are segmented into overlapping chunks of 500 characters with a 200-character overlap to maintain context, and then indexed into a vector database using ChromaDB. When a query is submitted, the system retrieves the top three most relevant text segments based on similarity (top_k=3) and forwards them to a language model via API for response generation. To further enhance the accuracy and relevance of the system's responses, an additional question-and-answer dataset was incorporated.

Through comprehensive testing and evaluation, we observed that the inclusion of this supplementary Q&A dataset significantly improved the quality and relevance of the answers provided. Furthermore, we found that the effectiveness of the system heavily depended on the formulation of the system prompt and the capabilities of the selected large language model. The final application offers a visually appealing and intuitive user interface, which incorporates university-themed colors and branding to create a modern, cohesive, and engaging experience for students.

Our overall findings underscore the critical importance of thoughtful RAG architecture, meticulous data preprocessing, and seamless system integration when building intelligent assistant systems. The success of the Beytepe Information Assistant suggests that this model can be scaled to support other departments within the university and enhanced with additional features, such as real-time push notifications or customizable modules tailored to specific departmental needs.

Introduction, Problem Definition & Literature Review (/ 20 Points)

Introduction

In modern higher education environments, the accessibility and timely delivery of information represent critical factors that significantly influence both the day-to-day experiences of students and their long-term academic performance. Universities, by their nature, operate through a variety of decentralized and often disconnected channels, such as institutional websites, physical notice boards, email systems, and third-party mobile applications. As a result, students are frequently required to monitor multiple sources to access essential campus-related information. Given the increasing dependence of students on mobile devices for organizing and managing their academic and personal responsibilities, there is a clear and growing demand for intelligent, centralized platforms that can effectively aggregate, personalize, and streamline access to such information.

The primary goal of this project is to address this systemic fragmentation by developing an intelligent campus information assistant tailored specifically for Hacettepe University. This assistant is designed to function not only as a centralized aggregator of critical, dynamic information—such as daily dining hall menus, shuttle bus schedules, and department-specific announcements—but also as a natural language interface capable of understanding and answering academic questions. By leveraging recent breakthroughs in large language models (LLMs), the system aims to provide students with a seamless, intuitive, and personalized experience.

Problem Definition

At the core of this project lies a clearly defined problem: the dispersed and fragmented distribution of student-related information across Hacettepe University's various digital and physical platforms. This lack of integration often results in inefficiencies, such as missed announcements or outdated knowledge, and leads to reduced engagement with available academic resources. For example, a typical student may need to consult multiple departmental and institutional websites on a daily basis just to stay informed about basic services and responsibilities. Moreover, students frequently encounter recurring questions—such as queries related to course prerequisites, registration procedures, or advisor availability—that are not always easily answered through existing systems.

Thus, the central research question can be formulated as follows: *How can we design and implement a mobile-based, centralized system that autonomously collects, structures, and delivers dynamic campus-related data while simultaneously offering intelligent, context-aware question-answering capabilities through a Retrieval-Augmented Generation (RAG) approach powered by large language models?*

Literature Review

Recent developments in the field of natural language processing (NLP) have led to substantial progress in building more effective dialogue systems and intelligent digital assistants. One of the foundational breakthroughs in this area was the introduction of BERT by Devlin et al. (2019), which enabled contextualized word embeddings and significantly advanced question-answering (QA) tasks. Later, Lewis et al. (2020) introduced the RAG framework, which strategically combines dense retrieval systems with generative transformers, resulting in improved factual consistency and robustness in open- and closed-domain settings.

Subsequent research, such as the work by Karpukhin et al. (2020), emphasized the benefits of dense passage retrieval for improved document-level relevance, while Guu et al. (2020)

demonstrated that incorporating domain-specific fine-tuning techniques greatly enhances the semantic accuracy and specificity of generated responses. In the context of educational support tools, studies like those conducted by Zhang et al. (2018) illustrated how chatbots could be effectively integrated into student advisory systems, providing benefits in terms of scalability, cost-efficiency, and user satisfaction. However, these implementations often suffered from limitations, such as reliance on static FAQ-style data and a lack of real-time integration with evolving institutional information.

Building on the aforementioned studies, this project proposes a hybrid solution that combines real-time web scraping, mobile-first development using the Flutter framework, and a custom Retrieval-Augmented Generation pipeline. This combination aims to deliver a solution that not only understands user intent in natural language but also dynamically retrieves relevant and up-to-date institutional information.

Hypothesis and Proposed Solution

We hypothesize that the implementation of a mobile-based intelligent assistant—integrating dynamic data retrieval, centralized access to campus information, and natural language understanding via RAG-powered large language models—will lead to a significant improvement in students' ability to access timely and relevant academic and administrative information. Furthermore, we expect this system to reduce the volume of repetitive and unanswered student inquiries, thereby enhancing both the efficiency and satisfaction of student-institution interactions.

To realize this hypothesis, we propose a solution composed of several interdependent modules. The data collection layer includes web scraping mechanisms and API-based data retrieval from university sources. The data storage and retrieval component utilizes a vector database that indexes preprocessed documents to enable efficient semantic search. The reasoning layer integrates an LLM that operates within a RAG architecture to generate contextually accurate responses. Finally, the user interaction layer is delivered through a modern, intuitive mobile application developed using Flutter, ensuring platform independence and a cohesive user experience.

Methodology (/ 25 Points)

1. Data Collection and Preprocessing

The data utilized within this project was primarily sourced from two central platforms affiliated with Hacettepe University: the official dining hall website and the website of the Computer Engineering Department. While both the announcements and dining hall menus were publicly accessible in standard HTML format—thus allowing for automated data extraction—shuttle bus schedules lacked a structured online representation and were therefore curated manually due to their static and infrequently updated nature.
To automate the extraction process for suitable sources, a combination of web scraping tools was employed. Specifically, the BeautifulSoup library was used for parsing HTML content, and Selenium was integrated for interacting with dynamic web elements when required.
For the Retrieval-Augmented Generation (RAG) component, a distinct dataset of academic and administrative documents was manually collected from the cs.hacettepe.edu.tr domain. These documents underwent a detailed preprocessing pipeline to enhance clarity and consistency. Outdated or irrelevant content was filtered out, and the remaining materials were segmented into overlapping text chunks, each consisting of 500 characters with an overlap of 200 characters. This chunking strategy was selected to preserve context between segments. The resulting text fragments were then transformed into numerical embeddings and indexed using ChromaDB, a vector database tailored for efficient similarity-based retrieval.

2. Retrieval-Augmented Generation (RAG)

The RAG pipeline implemented in this system is composed of two fundamental components: a retriever and a generator. The retriever's function is to take an input query, embed it into a vector representation, and then search the ChromaDB index to identify and return the three most semantically relevant text chunks. These retrieved segments are then appended to the system prompt, which is subsequently forwarded to the generator—a large language model accessed via API—for producing a final response.
To enhance the quality and contextual accuracy of the answers generated, a supplementary dataset consisting of manually constructed question-answer pairs was created. These QA pairs were embedded and indexed alongside the document chunks within the same ChromaDB instance. Various configurations of the RAG parameters—such as chunk size, character overlap, and the number of top results (top_k)—were tested. The combination of 500-character chunks with a 200-character overlap and top_k set to 3 was found to yield the most coherent and contextually appropriate outputs.

3. Backend Development

The backend infrastructure of the application was developed using FastAPI, a modern and high-performance web framework for building APIs with Python. The backend is responsible for several core functionalities, including the scheduled or user-triggered scraping of dining hall menus and departmental announcements. In addition, it serves the manually curated shuttle schedule data, which is relatively static.
Furthermore, the backend acts as a communication bridge between the mobile client and the RAG-based language model pipeline. This modular and service-oriented design approach ensures that individual system components—such as data retrieval scripts or the LLM interface—can be updated or replaced without necessitating changes to the overall system architecture, thereby improving maintainability and scalability.

4. Mobile Application

The mobile application's frontend was implemented using Flutter, a UI toolkit known for its ability to create visually consistent, high-performance applications across multiple platforms. The user interface was designed with a particular emphasis on usability and adherence to Hacettepe University’s visual identity, which includes the use of purple-themed color palettes. Core features of the mobile app include clearly navigable menus that allow users to access dining hall menus, shuttle schedules, and departmental announcements. Additionally, the application incorporates a chat interface that enables users to interact with the intelligent assistant via API.

During the development process, extensive user testing was conducted with peers and potential end users. Feedback from this testing highlighted the importance of a minimalist interface design, as well as the critical role of fast response times in maintaining user engagement and satisfaction.

5. LLM Integration

For the language generation component of the system, a commercial large language model with accessible API support was employed. Considerable effort was dedicated to iteratively refining the system prompts, striking a balance between informativeness and brevity to ensure responses were both helpful and concise.

In order to improve system robustness and user experience, mechanisms for response caching and comprehensive error handling were implemented. These additions ensure that the application can maintain functionality and responsiveness even under conditions of high load or intermittent connectivity, thereby enhancing the reliability of the assistant.

Results & Discussion (/ 30 Points)

1. Performance Evaluation of the RAG Pipeline

To assess the effectiveness and robustness of the Retrieval-Augmented Generation (RAG) pipeline, a series of configurations were systematically tested with the goal of evaluating both the precision of the retrieval mechanism and the quality of the generated responses. These experiments led to several key observations:

- The inclusion of a supplementary dataset composed of manually prepared question-answer (Q&A) pairs resulted in a notable improvement in the relevance of the system's responses. Specifically, the addition of this data improved answer precision by approximately 20–25%, as determined through comparative analysis of user queries and output consistency.
- The use of a 200-character overlap during document chunking contributed to preserving contextual continuity across segments without introducing excessive redundancy. This overlap ensured that semantically connected phrases were not split in ways that would confuse the language model.
- Setting the top_k parameter to 3 during retrieval provided a favorable balance between precision and breadth. This configuration successfully minimized the inclusion of irrelevant information ("noise") while maintaining enough diversity to allow the language model to produce context-rich responses.

Sample queries, including but not limited to "advisor office hours" and "graduation requirements," consistently produced highly relevant and accurate answers. These results indicate that the vector space constructed through ChromaDB indexing was well-structured and effective in preserving semantic relationships between queries and document content.

2. Assistant Quality

The intelligent assistant demonstrated the capability to handle a wide range of user queries, including both straightforward, factual inquiries (e.g., "What is today's menu?") and more nuanced, context-sensitive questions (e.g., "Are there vegetarian options available today?"). The system showed particular sensitivity to how prompts were phrased, which had a direct impact on the quality and accuracy of the generated responses.
High-quality, contextually appropriate answers were only produced consistently after extensive tuning of the system-level prompts. These adjustments involved explicitly embedding relevant contextual information into the prompt template, which helped guide the large language model toward more informative and coherent outputs.

3. App Usability

The mobile application, developed using Flutter, benefited greatly from the framework's built-in UI components, which allowed for the implementation of a fast, responsive, and aesthetically pleasing user interface. The visual design adhered closely to Hacettepe University's branding guidelines, with a dark-themed layout incorporating the institution's signature purple tones.
User testing revealed that the application's tab-based navigation structure was intuitive and user-friendly. Among the various features provided, the chat interface and the dining menu

section emerged as the most frequently accessed, indicating strong demand for both real-time assistance and up-to-date daily services.

4. Backend Reliability

The backend, built using FastAPI, exhibited a high degree of reliability and modularity, which proved advantageous for both testing and maintenance. Its component-based structure facilitated targeted debugging and allowed for isolated testing of individual modules. However, one of the recurrent technical challenges involved the web scraping modules. These components occasionally failed due to changes in the HTML structure of the source websites. To mitigate the impact of such disruptions, fallback mechanisms were implemented. These ensured that in the event of a scraping failure, the application could still function using previously cached or static data, thereby maintaining a basic level of usability.

5. Challenges Faced

Throughout the development and implementation phases of the project, several challenges emerged:

- The process of document preprocessing for integration into the RAG system was labor-intensive and required considerable manual effort. Tasks such as filtering, formatting, and segmenting documents demanded attention to detail and consumed significant development time.
- While the large language model exhibited strong performance under structured inputs, it demonstrated limitations when dealing with poorly formulated or ambiguous queries. In such cases, the output could become inconsistent or lack relevance.
- Managing the trade-off between performance and cost posed a notable difficulty, particularly when relying on commercial LLMs accessed via external APIs. Balancing response quality, latency, and API usage costs required careful monitoring and tuning.

6. Comparison with State-of-the-Art

In comparison to existing systems deployed in similar educational contexts—particularly those used by other universities—this project introduces several distinguishing features. Most institutional assistant systems rely heavily on static, pre-defined FAQ databases, which are inherently limited in their ability to handle dynamic content or nuanced queries.
By contrast, the system presented in this project combines real-time data scraping with a Retrieval-Augmented Generation framework, enabling it to deliver not only accurate and timely responses but also a deeper and more context-aware understanding of user queries. This hybrid approach allows for both freshness of content and semantic depth, setting it apart from conventional university information bots.

The Impact and Future Directions (/ 15 Points)

The **Beytepe Information Assistant** represents a promising step toward enhancing the daily experiences of students by alleviating the cognitive burden caused by the fragmented distribution of university-related information. By centralizing essential resources—such as dining hall menus, shuttle schedules, and departmental announcements—within a single, user-friendly mobile platform, the assistant provides immediate and reliable access to frequently needed information. Moreover, the system's ability to answer academic and administrative queries through natural language interaction further empowers students to make informed decisions quickly and efficiently.

Looking forward, several directions for the system’s enhancement and expansion have been identified:

- **Scalability across departments**: The architecture allows for adaptation to other faculties and departments within the university by integrating department-specific datasets. With appropriate content curation, the assistant can be tailored to reflect the unique informational needs of diverse academic units.
- **Push notification integration**: Future iterations of the system could incorporate real-time alerts and event notifications, enabling the assistant to actively inform students about last-minute changes, deadlines, or campus activities without requiring them to manually check the application.
- **Multilingual support**: To better serve the international student population at Hacettepe University, multilingual functionality could be added. This would make the system more inclusive and accessible by providing interfaces and assistant responses in multiple languages.
- **Self-hosted, fine-tuned language model deployment**: To reduce long-term costs and dependency on commercial API services, the system could be enhanced by deploying a fine-tuned, locally hosted large language model. This would also enable more domain-specific customization and control over data privacy.

In addition, the backend infrastructure could be further strengthened by integrating directly with official university APIs, which would open the door to a range of authenticated services. Such integration would allow the assistant to support personalized operations, including but not limited to course registration, transcript viewing, or advisor appointment scheduling—features that would significantly increase the assistant's value to students.

Given its modular and extensible design, the system lays a solid foundation for evolving into a comprehensive, university-wide digital assistant. Its current success in aggregating, contextualizing, and delivering information indicates strong potential for broader deployment and continuous enhancement aligned with institutional needs and technological advancements.