HO CHI MINH UNIVERSITY OF SCIENCE

VIETNAM NATIONAL UNIVERSITY

Falcuty of Information Technology

----------∞O∞----------

FINAL PROJECT REPORT

APPLIED NATURAL LANGUAGE PROCESSING

Instructor: Dr. Nguyễn Hồng Bửu Long
Dr. Lương An Vinh

Ho Chi Minh City, 20th May 2025

Table of Content

Acknowledgement 4

I. Project Title 5

II. Project member 5

III. Project overview 5

IV. Project purpose: 5

How do we solve: 5

V. The intended users or beneficiaries: 5

VI. Technologies used: 6

AI & NLP Technologies 6

Backend 6

Frontend 6

Tooling & DevOps 6

VII. Related diagram 7

1. Component diagram 7

a. Client-Side (Frontend) 7

b. Server-Side (Backend) 8

c. External Systems 8

d. Workflow Summary 8

2. Database schema 9

a. User & Authentication 9

b. Flashcard System 10

c. Chatbot Module 10

d. Learning History 10

e. Knowledge Graph 10

f. Administrative & Metadata 10

VIII. Project feature 10

IX. Installation guidelines 12

1. Prerequisites 12

2. Backend Setup (Django) 12

3. Frontend Setup (React + Next.js) 12
4. Database 12
5. Configuration 13
X. Usage Guidelines 13
1. How to Launch the Application 13
2. How to Use Its Core Functions 13
XI. Limitations 14
XII. Future work 14
XIII. References 15

Acknowledgement

We would like to express our sincere gratitude to all the lecturers who have guided us throughout the process of completing this project.

We especially appreciate the dedicated support of **Dr. Nguyễn Hồng Bửu Long**, who has accompanied us during the entire journey.

We are also thankful for the quiet but meaningful support from Dr. **Lương An Vinh** throughout the project.

The valuable knowledge shared by Dr. **Nguyễn Hồng Bửu Long** and **Dr. Lương An Vinh** during lectures, along with the detailed instructions in homework assignments, provided us with a solid foundation to complete this project.

Although the process was filled with challenges and difficulties, we managed to overcome them and reach the final result. This achievement would not have been possible without the dedicated support of our teachers.

I. Project Title

Intelligent Flashcard Generation System

II. Project member

Name	Student ID	Role	Contribution
Phan Lê Đức Anh	22127020	Member	Design, implement UX, UI
Trần Hoàng Linh	22127233	Member	Implement speaker, recording feature, learning history
Nguyễn Thị Ngọc Trang	22127421	Leader	Implement QnA with AI using RAG technique, Evaluation techniques for large language models
Nguyễn Gia Phúc	22127482	Member	Implement login/signup, knowledge graph, learn flashcard

III. Project overview

Memoria is an AI-powered learning platform that automatically generates intelligent flashcards from user-provided text or documents. By using **Named Entity Recognition (NER)** and **keyphrase extraction**, Memoria identifies important concepts, terms, and relationships to create personalized flashcards that enhance understanding and retention. With features like automatic flashcard generation, interactive learning, pronunciation practice, and AI chat support, Memoria transforms the way users study—making it smarter, faster, and more effective.

IV. Project purpose:

Students and self-learners often struggle to efficiently summarize and retain large volumes of study material in different file formats like pdf, pptx, txt, etc. Manually creating flashcards is time-consuming, inconsistent, and can lead to gaps in understanding. Additionally, traditional study methods may not leverage modern AI capabilities for personalized learning and reinforcement.

How do we solve:

Memoria addresses this by automating the flashcard creation process using **Natural Language Processing** techniques like **Named Entity Recognition** and **keyphrase extraction**. It provides AI-enhanced features such as interactive learning, pronunciation practice, and AI chat support tools to help users study more effectively, with less effort.

V. The intended users or beneficiaries:

- **Students:** High school, university, and graduate students who need to efficiently study textbooks, notes, or academic papers.

- **Self-learners:** Individuals pursuing personal or professional development through online courses, articles, or eBooks.
- **Language learners:** Users learning new languages who benefit from vocabulary-based flashcards, pronunciation practice, and repetition.
- **Educators & Tutors: T**eachers who want to create flashcards for students from course materials or reading assignments.
- **Professionals preparing for exams:** Test-takers studying for standardized exams like GRE, TOEFL, IELTS, PMP, etc., where efficient revision is crucial.

VI. Technologies used:

AI & NLP Technologies

- **spaCy** – for NLP processing
- **LangChain** – for language model orchestration and chaining
- **ElevenLabs** – likely for text-to-speech capabilities
- **Python**, **NumPy**, **pandas**, **SciPy** – for data manipulation and scientific computing
- **OpenAI** (gpt-4o-mini) and **Gemini** (gemini-flash-2.0)

Backend

- **Django** – as the main web framework
- **Django REST Framework** – for building RESTful APIs
- **.env** – for environment variable management

Frontend

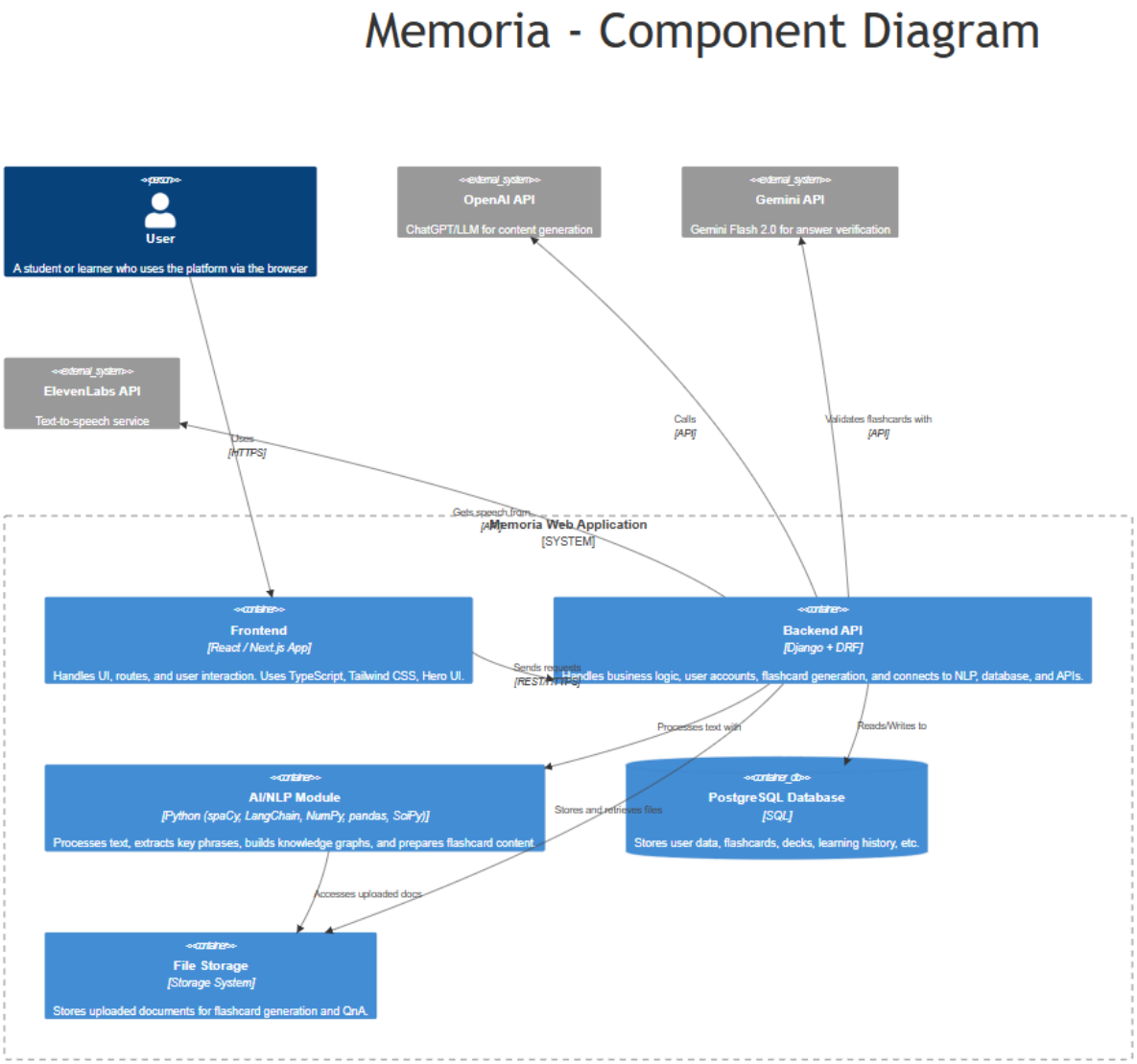
- **React** – for building dynamic user interfaces
- **Next.js** – for server-side rendering and routing
- **TypeScript** – for type safety
- **Tailwind CSS** – for styling

Tooling & DevOps

- **Docker** – for containerized deployment
- **npm**, **pipenv** – for dependency management

VII. Related diagram

1. Component diagram



The component diagram illustrates the high-level architecture of Memoria, a web-based flashcard learning platform. It visualizes the interaction between users, internal components, and external services.

a. Client-Side (Frontend)

- **User**: A student or learner who accesses the platform through a web browser.
- **Frontend (React / Next.js App)**: Responsible for the user interface (UI), routing, and user interaction. Built with TypeScript, Tailwind CSS, and Hero UI, it

communicates with the backend over HTTPS. It allows users to upload files or text, view flashcards, and engage in learning tasks.

b. Server-Side (Backend)

- **Backend API (Django + DRF)**: The core of the application's business logic. It handles:
  - User authentication and account management
  - Flashcard and deck creation
  - Pronunciation and chatbot services
  - Communication with the AI/NLP module, file storage, and database
  - Integration with external APIs for advanced features
- **AI/NLP Module**: Implements text processing using libraries like spaCy, LangChain, NumPy, and pandas. It:
  - Extracts entities and keywords
  - Builds knowledge graphs
  - Prepares intelligent flashcard content
- **File Storage**: Temporary storage for uploaded documents used in flashcard generation and AI-powered QnA.
- **PostgreSQL Database**: A relational database for storing persistent data such as:
  - User accounts and session data
  - Flashcard decks and learning progress
  - Knowledge graphs

c. External Systems

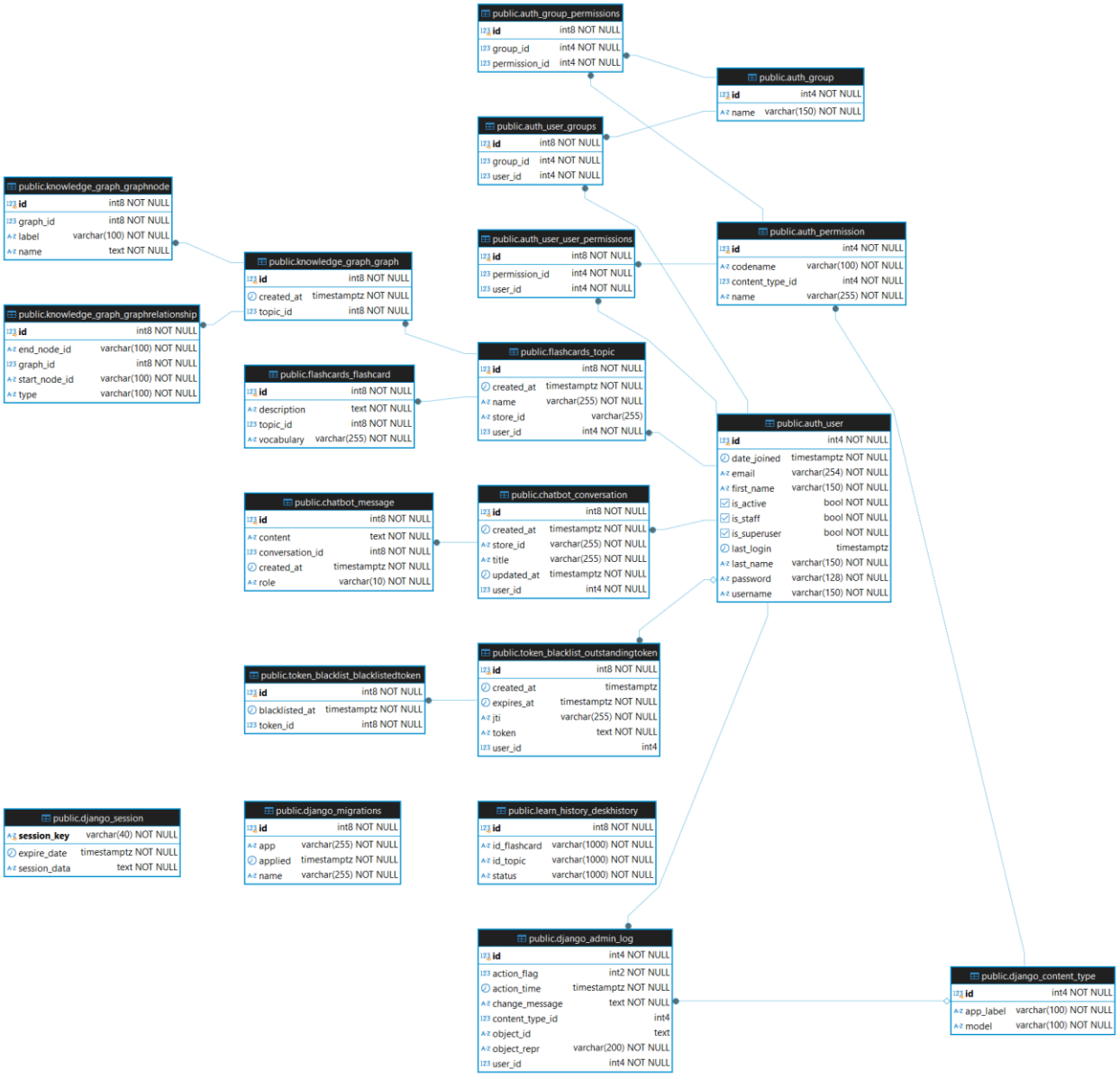
- **OpenAI API**: Integrates GPT models for generating content like flashcards or explanations.
- **Gemini API**: Provides verification and evaluation for flashcards using Gemini Flash 2.0 models.
- **ElevenLabs API**: Supports text-to-speech functionality, allowing pronunciation playback of flashcard vocabulary.

d. Workflow Summary

1. The **user** interacts with the **frontend**, submitting files or text.
2. The frontend sends requests to the **backend API**, which forwards them to the **AI/NLP module** or **external APIs**.
3. The backend stores or retrieves data from the **PostgreSQL database** and manages files via the **file storage system**.

4. **Flashcards**, generated and validated, are returned to the user for study and interaction.

2. Database schema



The diagram above presents the **relational schema** of the *Memoria* platform, designed to support user accounts, flashcard management, chatbot interactions, learning history, and knowledge graph storage. The schema is structured across several interconnected tables:

a. User & Authentication

- **auth_user**: Stores core user credentials and profile info.

- **auth_group, auth_permission, auth_user_groups, auth_user_permissions**: Django's built-in tables for managing user roles and permissions.
- **token_blacklist_blacklistedtoken, token_blacklist_outstandingtoken**: Tracks invalidated and active JWT tokens for secure authentication.
- **django_session**: Handles session data for non-token-based authentication.

b. Flashcard System

- **flashcards_topic**: Represents individual flashcard topics (decks), including a store_id used to associate with uploads or chat context.
- **flashcards_flashcard**: Each flashcard is linked to a topic and contains a vocabulary term and a description.

c. Chatbot Module

- **chatbot_conversation**: Captures the conversation session metadata (user, title).
- **chatbot_message**: Stores individual messages per conversation, including user/AI roles.

d. Learning History

- **learn_history_deckhistory**: Tracks a user's learning progress per flashcard per topic, recording learning status.

e. Knowledge Graph

- **knowledge_graph_graph**: The root entity for each topic's knowledge graph.
- **knowledge_graph_graphnode**: Stores the nodes in the graph (e.g., concepts, entities).
- **knowledge_graph_graphrelationship**: Stores labeled edges/relationships between nodes.

f. Administrative & Metadata

- **django_admin_log, django_content_type, django_migrations**: Django system tables used for tracking admin actions, content types, and migration history.

This schema reflects a well-structured separation of concerns between **authentication**, **user-generated content**, **AI-powered features**, and **semantic knowledge representation**, allowing flexible and scalable growth of the platform.

VIII. Project feature

Feature	Purpose	How it Was Implemented	Special Techniques or Notes
AI Flashcard Generator	To create flashcards automatically from text or files	Used NLP tools like spaCy and LangChain to find key terms and important info	- Uses Named Entity Recognition (NER) and keyphrase extraction - Cards are double-checked using Gemini API to make sure the content is correct and meaningful
Flashcard-study	Let users learn at their own pace with flashcards	Built with React pages and components to display and flip cards	- Tracks user progress - Includes a knowledge graph that shows related words/concepts using NLP and simple graph logic- Helps users see connections between terms in the same topic
Pronunciation Practice	To help with learning how words sound	Uses text-to-speech from ElevenLabs to read words out loud	
AI Chat	Let users ask questions about topics	Uses OpenAI API and a chatbot interface built with React	Keeps chat history and gives helpful answers, but users can only ask questions related to the topic due to RAG.
Deck Management	Organize flashcards into topics or subjects	Flashcard decks are created and stored in Django backend	Users can rename, delete cards easily
Learning History	Shows users what they've studied and how much	Backend saves quiz results and time spent on decks	Helps users review old topics
User Authentication	Allows login, sign-up, and secure access	Built with Django and React forms with validation	Includes email verification and password reset

IX. Installation guidelines

1. Prerequisites

- Python 3.11 or later
- Node.js and npm/yarn installed
- pipenv for Python virtual environment management

2. Backend Setup (Django)

- Open terminal and go to the backend directory: `cd server`
- Install **pipenv** if not installed: `pip install pipenv`
- Launch a **pipenv** environment (create if not existed): `pipenv shell`
- Install Python dependencies: `pipenv sync`
- Create tables in database: `python manage.py migrate`
- Run the development server: `python manage.py runserver`

3. Frontend Setup (React + Next.js)

- Open a new terminal and go to the frontend directory: `cd client/memoria`
- Install npm packages: `npm install`
- Start the development server: `npm run dev`
- Open your browser and visit: http://localhost:3000

4. Database

- Ensure PostgreSQL is installed and running.
- Create a new database (e.g., memoria), user, and password.
- Update your database settings in `server/settings.py`:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'memoria',
        'USER': 'your_db_user',
        'PASSWORD': 'your_db_password',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

- Install PostgreSQL drivers for Python if prompted (`pipenv install psycopg2`).

5. Configuration

- Ensure the backend and frontend servers are running simultaneously.

- Environment variables should be set properly in the `.env` file (if applicable).

X. Usage Guidelines

1. How to Launch the Application

Backend Setup (Django)

1. Open terminal and navigate to the backend folder: cd server
2. Install dependencies: pip install pipenv pipenv install
3. Run the backend server: pipenv run python manage.py runserver

Frontend Setup (React + Next.js)

1. Open another terminal and move to the client directory: cd client/memoria
2. Install packages: npm install
3. Run the app: npm run dev

Access the app in your browser at: http://localhost:3000

2. How to Use Its Core Functions

User Authentication

- Sign Up: Register with your email and verify it via the email confirmation page.
- Login: Log in using your registered credentials.
- Forgot Password: Use the "Forgot password" feature to reset credentials via email.

Create Flashcards

- Go to the Create Flashcard page.
- Upload a .txt, .pdf, .pptx or .docs file.
- The system will extract keywords and generate flashcards using NLP tools like SpaCy.
- You can review and edit flashcards before saving them.

Learn Flashcards

- Navigate to the Learn section.
- Select a flashcard deck to study.
- Flashcards are displayed with interactive buttons (show answer, mark as known/unknown).

View Learning History

- Head to the Learning History tab.
- See your past performance and progress in different flashcard decks.

Chat with AI

- Visit the AI Chat page.
- Ask the AI assistant anything related to the uploaded content or topic-specific questions.
- The AI provides helpful and personalized responses using LangChain integration.

XI. Limitations

While the Memoria application offers a wide range of useful features, there are still some limitations to be aware of:

- The system currently lacks proper Quality Assurance (QA) control due to the integration of various third-party applications and services. This makes it harder to ensure consistent performance across all components.
- Some features, such as AI-generated flashcards or chat responses, may produce inaccurate results depending on the input quality or external API stability.
- There might be minor performance issues when handling very large files or datasets, especially during NLP processing.

XII. Future work

- **Scalability**:
  As the number of users grows, we plan to migrate to a cloud environment such as AWS or Firebase to handle larger traffic and improve uptime. Backend services may also be containerized using Docker and deployed with orchestration tools like Kubernetes.

- **Performance Improvements**:
  We will optimize flashcard generation speed for large documents by caching and improving NLP processing efficiency. Also, frontend rendering and routing will be fine-tuned for a smoother experience on low-end devices.
- **User Experience Enhancements**:
  We're planning to redesign the UI for mobile-first usage, add theme-switching (dark/light mode), and integrate more interactive learning elements like quizzes and AI-powered feedback

XIII. References

〔1〕 LangChain (2024) – How to construct knowledge graph [Online] – Available: https://python.langchain.com/docs/how_to/graph_constructing/

〔2〕 LangChain (2024) – Build a Question Answering application over a Graph Database [Online] – Available: https://python.langchain.com/docs/tutorials/graph/

〔3〕 Tomaž Bratanič, neo4j (March, 2024) – Using a knowledge graph to implement a RAG application [Online] – Available: https://neo4j.com/blog/developer/knowledge-graph-rag-application/

〔4〕 OpenAI Platform (May, 2025) – OpenAI developer platform [Online] – Available: https://platform.openai.com/docs/overview

〔5〕 Django REST framework (May, 2025) – Django REST framework [Online] – Available: https://www.django-rest-framework.org/

〔6〕 Coqui, GitHub (August, 2025) – a deep learning toolkit for Text-to-Speech, battle-tested in research and production [Online] – Available: https://github.com/coqui-ai/STT

〔7〕 James Briggs, YouTube (2023) – OpenAI's New GPT 3.5 Embedding Model for Semantic Search [Online] – Available: https://www.youtube.com/watch?v=ocxq84ocYi0

〔8〕 Geaux-EDU, GitHub (June, 2024) – Text_embedding_ada-002 [Online] – Available: https://github.com/Geaux-EDU/Text_embedding_ada-002