



School of Science and Engineering

Title: Intelligent Learning System

Capstone Design Report

FALL 2024

Student Name

Suhayb Daud

Supervised by:

Dr. Omar Iraqi

Student conduct and copyright

The supervisor and the student (the Capstoner) agree that:

1. Permission has been obtained for any third party content (eg Data, illustrations, photographs, charts or maps).
2. The results described in this report have not previously been published

Student's name and signature:

Subayb Daud



Supervisor's name and signature:

Omar Iraqi



Title: Intelligent Learning System

Capstone Report

Student Statement:

I have incorporated ethical principles into the design process and the selection of the final proposed solution, ensuring that public safety remains a top priority and is addressed throughout the design wherever relevant.



Suhayb Daud

Approved by the Supervisor



Dr. Iraqi

ACKNOWLEDGEMENTS

First and foremost, I would like to express my heartfelt gratitude to God for blessing me with the strength, patience, and guidance to reach this significant milestone. His presence has been a constant source of inspiration throughout my journey.

I extend my deepest thanks to my family for their unconditional love, support, and belief in my abilities. Their encouragement has been the foundation of my perseverance and success.

I am profoundly grateful to my academic supervisor, Dr. Omar Iraqi, for his invaluable mentorship, insights, and encouragement. His dedication to my growth has been instrumental in guiding me through this capstone project and my academic pursuits.

Lastly, I would like to thank everyone who has supported me throughout my university journey. Whether through encouragement, opportunities, or resources, your contributions have been vital to my success, and I am deeply appreciative of the role you have played in helping me achieve this milestone.

CONTENT

ACKNOWLEDGEMENTS.....	iii
LIST OF FIGURES	vi
LIST OF TABLES	vii
1 ABSTRACT (ENGLISH)	viii
2 INTRODUCTION.....	1
3 PROBLEM STATEMENT	1
4 PROJECT SPECIFICATIONS	1
4.1 Functional Requirements	1
4.2 Non-functional Requirements	2
5 LITERATURE REVIEW	2
5.1 Theoretical Approaches in Adaptive Learning	2
5.1.1 Macro-Adaptive Approaches	2
5.1.2 Micro-Adaptive Approaches:	3
5.2 Common Core State Standards	3
5.3 Addressing the Gap: A Structured Tagging Framework.....	3
6 STEEPLE ANALYSIS.....	4
7 ENGINEERING STANDARDS	5
7.1 IEEE/ISO/IEC 15288-2023.....	5
7.2 ISO/IEC 42001:2023	5
8 FEASIBILITY STUDY	6
9 LOGICAL MODEL FRAMEWORK	7
9.1 Target Population and Scope of the Project.....	7
9.2 Underlying Assumptions.....	7
9.3 Activities.....	8
9.4 Resources/Technology Enablers	9
9.4.1 Frontend: Angular	9
9.4.2 Backend: Django	9
9.4.3 Database: PostgreSQL.....	10
9.4.4 AI Models.....	10
9.4.5 Docker	11
9.5 Outputs of the Project.....	11
9.5.1 AI Pipeline.....	12
9.6 Outcomes.....	16
9.7 Challenges	16
10 METHODOLOGY & CAPSTONE DESIGN	17
10.1 Architecture	17
10.2 Design	17

10.2.1	Class Diagram.....	18
10.2.2	Entity-Relationship Diagram (ERD).....	18
10.3	Implementation Details.....	19
10.3.1	Environment Setup	19
10.3.2	Adding Features	24
10.3.3	Security.....	30
11	RESULTS/ UI OF THE APPLICATION	32
12	LEARNING STRATEGIES.....	35
13	FUTURE WORK	36
14	CONCLUSION.....	36
	REFERENCES	37
	APPENDIX A: CODE	38

LIST OF FIGURES

Figure 1: CCSS Coherence Map.....	8
Figure 2: LangSmith UI.....	13
Figure 3: LangSmith Capturing Model Output.....	14
Figure 4: Chapter 1 Processed Output by GPT-4o.....	15
Figure 5: ConceptAnalyzer Method with Concept Extraction Prompt.....	15
Figure 6: The Pipeline Evaluation Results.....	17
Figure 7: Application's Higher-Level Architecture.....	17
Figure 8: Class Diagram for the Application	18
Figure 9: ERD Diagram for the Application.....	18
Figure 10: The Angular Dockerfile.....	19
Figure 11: Frontend Structure	20
Figure 12: Django Dockerfile	21
Figure 13: Backend Structure	22
Figure 14: Application's Docker Compose File.....	23
Figure 15: Application's Docker Compose File.....	23
Figure 16: Django Admin Panel	25
Figure 17: Example of Django Model	25
Figure 18: Django Rest Framework Serializer	26
Figure 19: ModelViewSet Class.....	27
Figure 20: The Structure of the instructor App in Django Backend	27
Figure 21: Main Routing - Modules are Lazily loaded.....	29
Figure 22: Application's Authentication Actions	30
Figure 23: Interceptor that Includes Every Request the JWT Stored in the Local Storage	31
Figure 24: Login Guard that prevents Authenticated Users from the Login View	32
Figure 25: Failed Login	33
Figure 26: Instructor's Class View	33
Figure 27: Instructor's Generate Exercesise View	34
Figure 28: Student's Course Material View	34
Figure 29: Student's Exercise View	35
Figure 30: Student's Recomendation View	35

LIST OF TABLES

Table 1: Finetunned Dataset Descriptions	11
--	----

1 ABSTRACT (ENGLISH)

This Final report presents the development of an intelligent learning system that leverages natural language processing (NLP), computer vision, and recommendation technologies to enhance personalized education for students and optimize instructional support for educators. With a focus on the automatic tagging of content according to the Common Core Standards, set of academic standards in mathematics and English language arts/literacy (ELA), this project aims to streamline content delivery and improve student engagement. The system employs Angular for the frontend, Django for the backend, and integrates NLP, vision and large language models (LLMs) to extract, tag, and recommend content from uploaded mathematical documents. The architecture is based on the Model-View-Controller (MVC) design pattern, ensuring scalability, modularity, and ease of maintenance. Moreover, the implementation was successfully completed, achieving its objective of accurately tagging content with their corresponding standards, thereby enabling content-based recommendations. This report details the system's architecture, design, detailed implantation and future steps to build a comprehensive Intelligent learning environment.

Keywords: (keywords: Intelligent, NLI, Django, Education)

2 INTRODUCTION

This report presents the development of an intelligent learning system designed to enhance K-12 mathematics education through personalized learning experiences. The primary goal of the project is to recommend educational content tailored to each student's interactions and needs, thereby fostering a deeper understanding of mathematical concepts.

The system employs content-based recommendation algorithms to suggest relevant exercises, documents, examples, and prerequisite materials. Educational resources are tagged with skill-specific identifiers, enabling precise matching of content to areas where students require additional support. For instance, if a student encounters difficulties with fractions, the system recommends targeted resources to address this challenge.

To demonstrate feasibility within the scope of the project timeline, the focus is limited to mathematics education. The system leverages Angular for the front-end interface and Django for the back-end operations, ensuring a user-friendly and robust platform.

3 PROBLEM STATEMENT

Traditional learning management systems often struggle to provide immediate support and Intelligent learning experiences due to their inability to effectively extract, categorize, and recommend content from diverse educational materials. Consequently, Instructors face challenges in managing and tagging content according to standardized learning objectives, while students lack access to customized learning content that address their specific needs. The problem is further compounded by the difficulty of extracting meaningful information from mathematical documents, which include complex equations and varied problem types. This project aims to address these challenges by developing an intelligent learning that utilizes NLP, vision models, and recommendation techniques to automate content extraction, tagging, and recommendation, ultimately enhancing both the student and instructor experience.

4 PROJECT SPECIFICATIONS

4.1 Functional Requirements

1. Instructor Content Management

- The system shall allow instructors to upload course materials (e.g., slides, readings) so that students have centralized access to all learning materials.
- The system shall enable instructors to assign assignments manually or with system

assistance to tailor coursework to specific learning objectives.

- The system shall automatically tag documents, exercises, and examples with the required concepts and skills to ensure content aligns accurately with learning objectives.

2. Student Learning Management

- The system shall automatically generate exercises related to students' current course material and interactions to help them focus on areas needing improvement.
- The system shall recommend exercises and learning materials based on students' interactions and areas of need, reducing the effort required to manually search for content.

4.2 Non-functional Requirements

- The system shall be extensible. It should be able to have extensions such as student knowledge component modeling
- The user interface shall be intuitive, requiring minimal guidance for instructors and students
- The system shall include the following security features:
 - The application shall use data mappers (ORM/OGM) to prevent the use of raw database queries, serving as a safeguard against injection attacks.
 - The system shall validate the origin of data by verifying the identity of its provider.
 - The system shall employ CSRF tokens to ensure the integrity of data origins.

5 LITERATURE REVIEW

My project comes under the umbrella of Adaptive Learning systems. Adaptive learning systems have transformed traditional education by offering personalized learning experiences. These systems use technology to analyze each student's behavior and needs, adjusting the content to improve understanding and retention.

5.1 Theoretical Approaches in Adaptive Learning

Adaptive learning systems are based on different theoretical approaches:

5.1.1 Macro-Adaptive Approaches

Macro-adaptive approaches adjust the overall learning path based on broad characteristics of the learner but may not adapt in real-time (Kabudi et al, 2021). Moreover, they are less personalized and typically caters to a group of learners with similar traits.

5.1.2 Micro-Adaptive Approaches:

These approaches respond dynamically to real-time inputs, adjusting content delivery to address immediate learning needs (Kabudi et al, 2021).

5.1.2.1 Content-Based Recommendation Systems in Education

Content-based recommendation systems are key components of micro-adaptive learning environments. They analyze the features of educational materials and match them with student profiles to suggest relevant exercises, documents, and prerequisite content (Ennouamani & Mahani, 2017). In mathematics, where concepts build upon each other, these systems are crucial. They help ensure students receive the right support. What this also implies is that the content is in the first place tagged with metadata that enables it for a recommendation task.

5.2 Common Core State Standards

The Common Core State Standards (CCSS) are educational benchmarks for K-12 students in subjects like mathematics and English Language Arts. They define clear learning objectives and skills for each grade level, helping educators structure curricula and assess student progress effectively (Common Core State Standards Initiative, 2010).

By integrating the CCSS into adaptive learning systems, they could serve as our foundational metadata or attributes for educational content with standardized learning goals. Tagging materials—like exercises, problems, and documents—with specific CCSS identifiers allows the system to recommend content that addresses the exact standards a student is working. For example, if a student is having trouble with a problem tagged under the standard for understanding fractions, the system can suggest additional resources on equivalent fractions or prerequisite material to which they might be lacking (Common Core State Standards Initiative, 2010).

5.3 Addressing the Gap: A Structured Tagging Framework

Despite progress in adaptive learning, there is still a significant gap in how educational content is organized. Many current systems lack a cohesive framework that ensures content is appropriate and aligned with teaching goals.

This project addresses this gap by implementing a structured tagging framework based on the CCSS. By organizing and categorizing educational content according to these standards, the

system can recommend related exercises, prerequisite materials, or advanced content based on student interactions.

6 STEEPLE ANALYSIS

- **Social** · Impact of the project: The Intelligent learning system has the potential to reduce learning disparities by providing customized learning experiences for students, regardless of their starting ability. It can empower students by tailoring content to their needs, thus improving engagement and academic performance. · Impact on the project: The social perception of Intelligent learning systems may influence the system's adoption. Students, parents, and teachers must trust the system for it to be widely used. Resistance to technology in education could be a barrier.
- **Technological** · Impact of the project: The project pushes the boundaries of Intelligent learning technology by using advanced techniques like GPT-4o.
- **Economic** · Impact of the project: The Intelligent learning system could lead to cost savings in education by reducing the need for one-on-one tutoring and improving the efficiency of learning resources. Schools may need fewer human resources to track student progress, and learning materials can be customized without added costs.
- **Environmental** · Impact of the project: By digitizing learning content and assessments, the system could reduce the need for printed textbooks and paper-based tests, contributing to environmental sustainability.
- **Political** ·
 - Impact of the project: Education is often influenced by political policies. If governments or local authorities support digital learning solutions, the project could receive funding or recognition, leading to broader adoption. ·
 - Impact on the project: Changes in education policies or government regulations may affect how the system can be implemented, especially if there are requirements for data privacy, accessibility, or standardized curricula.
- **Legal**
 - Impact on the project: The system must comply with laws and regulations related to data privacy and protection, such as GDPR or FERPA, since it will handle sensitive student data. Any violation could lead to legal ramifications or hinder its adoption in certain regions. ·
 - Impact of the project: The system could influence the legal landscape by setting new standards for Intelligent learning technologies. It may drive the need for

regulations on fairness and transparency in AI-driven education tools.

- **Ethical**
 - Impact on the project: Ethical considerations, such as ensuring that the system treats all students fairly and without bias, are critical. The algorithms must be transparent, and the decisions they make (like determining knowledge gaps) must be explainable.
 - Impact of the project: The Intelligent learning system could contribute to more ethical learning environments by ensuring that every student has an equal opportunity to succeed, regardless of their initial proficiency or background. However, there may be ethical concerns if the system fails to account for students with special learning needs.

7 ENGINEERING STANDARDS

In developing the application, I adhered to key international standards to ensure quality, reliability, and alignment with industry best practices. These standards guided the project's methodologies, tools, and management processes to deliver a robust and secure system. Specifically, the following standards were implemented:

7.1 IEEE/ISO/IEC 15288-2023

The ISO/IEC/IEEE 15288:2023 standard provides a comprehensive framework for the entire system life cycle, from conception to retirement. By adhering to this standard, the application follows well-defined processes that ensure the system is developed, delivered, and maintained with high quality and reliability. Key aspects of compliance include:

- **System Life Cycle Management**
 - The project lifecycle is managed through defined phases, including concept, development, production, utilization, support, and retirement.
 - Each phase is clearly documented and reviewed to ensure alignment with project goals and user requirements.

7.2 ISO/IEC 42001:2023

ISO/IEC 42001:2023 - Information Technology — Artificial Intelligence — Management System

This newly established standard outlines requirements for managing artificial intelligence (AI) systems to ensure safety, accountability, and ethical considerations. Its application to the project includes:

- **Risk Management:** Identification, evaluation, and mitigation of risks associated with the AI functionalities in the system.
- **Transparency:** Documentation and auditing of AI decision-making processes to ensure they are explainable and understandable to stakeholders.
- **Security and Privacy:** Safeguarding data used and produced by AI systems, ensuring compliance with relevant privacy laws and protecting user data from unauthorized access.

By following these standards, the application follows software engineering lifecycle phases and integrates AI responsibly and ethically, ensuring trust and reliability in its recommendations and functionalities.

8 FEASIBILITY STUDY

Technical Feasibility

The main building blocks of the functional requirements are as follows:

- **Content Tagging:** Automatically tagging content with the relevant skills and competencies it covers.
- **Content Retrieval:** Retrieving appropriate content based on the material the student is interacting with.
- **Assignment Generation:** Automatically generating assignments based on content and the skills/competencies associated with it.

The technologies chosen for this project are well-suited to meet these functional requirements:

- **Django and Angular:** I have extensive experience working with Django and its ecosystem, and I have exposure to Angular, which I am using for the frontend. Both frameworks are well-documented, ensuring ease of development.
- **PostgreSQL:** PostgreSQL integrates seamlessly with Django, providing a reliable, relational database solution for storing user information, tagged content, and interaction data.
- **Vision Models (GPT-4o):** Vision models will be employed to extract information from mathematical documents, allowing the system to categorize and tag content effectively.

These technologies provide a solid basis for the project's technical feasibility. Django, Angular, and PostgreSQL form a mature and scalable stack for developing intelligent learning system. The use of NLP and vision models further support the automatic extraction and tagging of content, which is crucial for providing personalized recommendations. The existing technology choices and my background in these areas ensure that the system can meet its goals effectively.

Temporal Feasibility

The project timeline is structured into distinct phases to ensure feasibility within the capstone period. During the first, second, and third weeks, the focus will be on finalizing project requirements, architecture, and design. By the fifth week, the evaluation of the machine learning pipeline will be completed. The remaining time will be dedicated to building and refining the application, ensuring a smooth progression and timely delivery of the project.

9 LOGICAL MODEL FRAMEWORK

9.1 Target Population and Scope of the Project

The target population for my intelligent learning system includes two key groups: instructors and students. Due to time constraints and to ensure technical feasibility as a proof of concept, the project's scope will focus exclusively on mathematics. Instructors and students, primarily those teaching K-12 math, will benefit from the system's ability to generate tailored exercises based on the specific skills and competencies being taught. This tool will save instructors time by automating the creation of relevant math exercises, ensuring that the content aligns with students' learning goals.

9.2 Underlying Assumptions

One of the key underlying assumptions of this project is that educational standards provide a structured foundation that can be used to extract and organize knowledge components. Educational content is typically designed to build upon interconnected skills and competencies, forming a cohesive and logical progression for learners. By leveraging these standards, I assume that I can accurately identify relationships between different skills and use this information to guide learning paths and recommendations.

Specifically, this project assumes that we can effectively extract knowledge component dependencies or relationships from predefined curricula or coherence maps, particularly those developed by the Common Core Standards. The Common Core Standards provide a well-defined framework that outlines essential skills and competencies for each subject area, allowing the mapping of educational content to specific learning objectives.

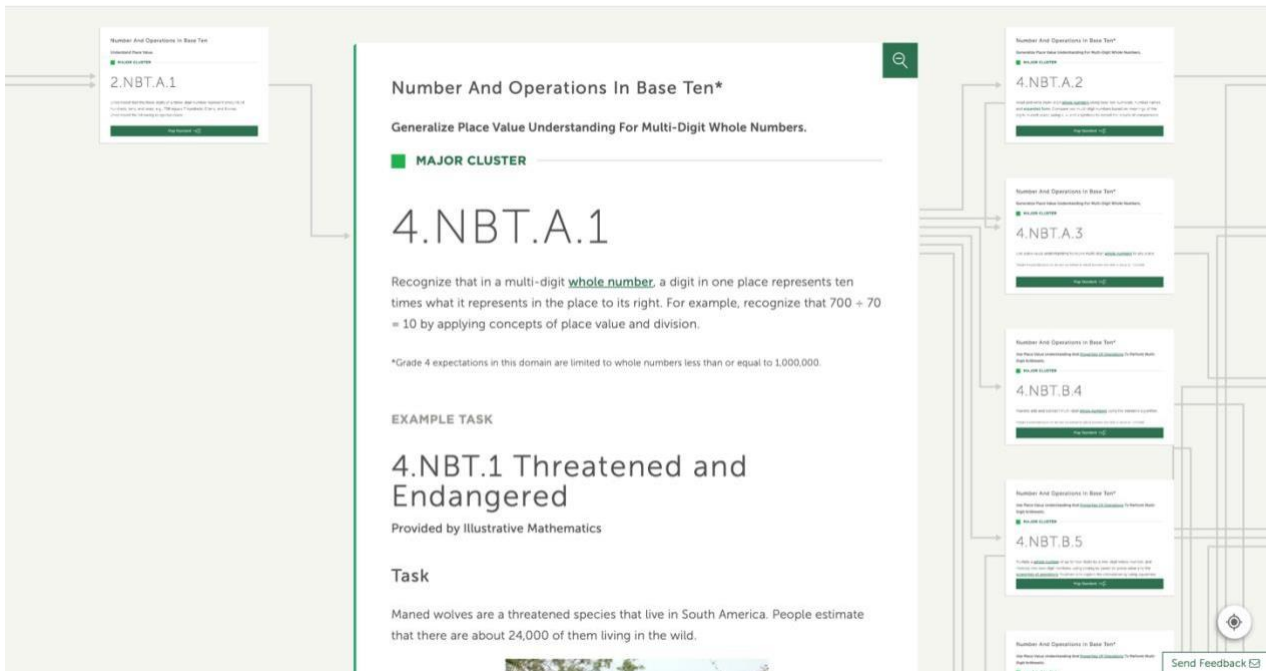


Figure 1: CCSS Coherence Map

By leveraging pre-existing dependencies, the system utilizes Natural Language Processing (NLP) techniques to tag content and recommend appropriate resources to students, ensuring a logical and effective progression in their learning experience. The use of Common Core Standards as a basis for tagging and categorizing content is essential to achieving the goal of personalized, standards-aligned recommendations, and forms a key part of the system's underlying design philosophy.

9.3 Activities

The project involves several key activities aimed at building and validating an intelligent learning system for K-12 mathematics. These activities include:

1. **Backend Development:** Designing a robust backend using Django and Django REST Framework, integrated with PostgreSQL for data management and JWT for secure authentication.
2. **Frontend Development:** Creating a dynamic and responsive user interface with Angular and styled using Tailwind CSS.
3. **AI Integration:** Leveraging advanced AI models, OpenAI's GPT-4o, for extracting concepts, exercises, and examples, and a Natural Language Processing (NLP) model for tagging content based on standards.

4. **Evaluation:** Testing the pipeline on annotated textbooks to ensure the accuracy of content extraction and tagging before full application integration.
5. **Containerization:** Dockerizing the application to streamline development, testing, and deployment, ensuring consistency across environments.

These activities collectively ensure a technically sound and scalable solution while validating the project's feasibility.

9.4 Resources/Technology Enablers

The development of this applications relies on a combination of several technologies and frameworks that enable efficient content extraction, tagging, and recommendation. Below are main technologies used to develop the application.

9.4.1 Frontend: Angular

The frontend is built with Angular, a robust framework for creating dynamic and scalable web applications. Key features include:

- **Components:** Modular UI building blocks for sections like dashboards and exercise viewers.
- **Services:** Handle backend communication and business logic.
- **Routing:** Enables seamless navigation across the application.
- **HTTP Client:** Facilitates data exchange with the backend.

For state management, the project uses NgRx, a reactive state management library for Angular. As for styling, it's implemented using **Tailwind CSS**, a utility-first framework that ensures responsive and consistent design with minimal effort. Combined, Angular and Tailwind CSS provide a reactive, scalable, and user-friendly frontend solution. More details are given in the detailed implantation section.

9.4.2 Backend: Django

The backend is developed using Django, a high-level Python web framework that emphasizes simplicity and scalability. It provides a clean architecture for managing data and APIs. Key aspects include:

- **Django REST Framework (DRF):** Used to create robust APIs for seamless communication with the Angular frontend. DRF simplifies tasks such as serialization, routing, and request handling.

- **Django ORM:** Django's built-in **Object-Relational Mapping (ORM)** streamlines database interactions by allowing developers to work with database records as Python objects. This abstraction enables:
 - Simplified querying and data manipulation.
 - Automatic database migrations for schema changes.
 - Integration with various databases without additional configuration.
- **JWT Authentication:** The backend implements **JSON Web Tokens (JWT)** for secure, stateless user authentication. JWTs allow users to log in and access resources efficiently while minimizing server load.

More details are as well given in the implantation details.

9.4.3 Database: PostgreSQL

The project uses **PostgreSQL** as the primary database for its reliability, scalability, and advanced features. Key highlights include:

- **PostgreSQL:** A robust relational database system used to store and manage user data, tagged content, and system interactions. It integrates seamlessly with Django's ORM for efficient querying and data manipulation.
- **pgAdmin:** A web-based database management tool used for visualizing and managing the PostgreSQL database. It simplifies tasks such as monitoring queries, viewing schemas, and analyzing data.

This combination ensures a powerful, efficient, and user-friendly database solution for the project.

9.4.4 AI Models

The project leverages advanced AI models to enable intelligent content extraction and tagging:

OpenAI's GPT-4o: A generative multimodal model capable of understanding and processing both text and visual inputs. It is used to extract key concepts, exercises, and examples from educational documents, ensuring comprehensive content analysis.

NLP Model: The NLP task I'm working on is **Natural Language inferencing (NLI)**. In NLI, given a premise and a hypothesis, the model predicts:

- **Entailment:** The hypothesis logically follows from the premise.
- **Contradiction:** The hypothesis contradicts the premise.

- **Neutral:** The hypothesis neither follows nor contradicts the premise.

The model I'll be using is XLNet-Large is already finetuned on multiple NLI datasets. The below table gives brief explanation of each dataset it was finetuned on.

Full Model Name: ynie/xlnet-large-cased-snli_mnli_fever_anli_R1_R2_R3-nli

Table 1: Finetuned Dataset Descriptions

Dataset	Description
SNLI (Stanford Natural Language Inference)	<ul style="list-style-type: none"> • Includes sentence pairs with labels: entailment, contradiction, and neutral. • Used for foundational training in NLI.
MNLI (Multi-Genre Natural Language Inference):	<ul style="list-style-type: none"> • Expands on SNLI with sentence pairs from diverse genres. • Evaluates the model's performance across genres.
FEVER (Fact Extraction and Verification)	<ul style="list-style-type: none"> • Focuses on fact-checking claims against evidence. • Adds robustness for real-world applications like misinformation detection.
ANLI (Adversarial NLI)	<ul style="list-style-type: none"> • Contains three rounds (R1, R2, R3) of adversarial examples that are particularly difficult for models to classify correctly. • Improves the model's ability to handle tricky cases.

9.4.5 Docker

The project uses Docker to containerize the entire application, ensuring consistent environments and simplified deployment. Key benefits include:

- **Isolation:** Each component (frontend, backend, and database) runs in its own container, preventing dependency conflicts.
- **Portability:** The application can be easily deployed across different environments without configuration issues.
- **Simplified Management:** Docker Compose is used to manage multi-container setups, streamlining the orchestration of the frontend, backend, and database.

9.5 Outputs of the Project

The project yielded several key outputs, demonstrating both the validity of the concept and the technical feasibility of the implementation. Moreover, there were two main components to the project. Firstly, was to evaluate a pipeline that automates the extraction of key concepts from educational PDF documents, maps them to relevant Common Core Standards using a NLI model, and evaluates the results against ground truth annotations as mentioned in section 9.3. The second big component is the application. In this section, I'll elaborate more in the evaluation of the pipeline and in section 10.3, there is more details regarding the application implementation.

9.5.1 AI Pipeline

The pipeline I used automates the extraction of key concepts from educational documents (for the time being, either images or PDFs), maps them to relevant Common Core Standards using a Natural Language Inference (NLI) model, and evaluates the results against ground truth annotations.

Tools Used For evaluation:

- Jupyter Notebook: interactive computing environment that allows you to create and share documents containing live code, visualizations, and narrative text. The Jupyter Notebook for this evaluation could be found the GitHub repository of the project named “Capstone_Notebook”
- Pytorch: A popular open-source machine learning library that provides support for deep learning and neural networks. In this pipeline, it's valuable for implementing and running the NLI model that maps educational concepts to Common Core Standards.
- OpenAI's Client Library: A software development kit that enables integration with OpenAI's APIs. This library is beneficial for this project on concept extraction from documents.
- LangChain and LangSmith: Langchain is an open-source framework designed to develop applications powered by language models. For this project, I'm utilizing it for its prompt management and prompt engineering features. As for langSmith, it's developer platform for debugging, testing, and monitoring LangChain applications. For this project, it's valuable for tracking the workflow of the pipeline and debugging any issues.

TRACE

Collapse Stats Filter

Show All ▾

ChatOpe... gpt-4o-... 0.46s 551,854

Run Feedback Metadata

Run ID Trace ID

11/27/2024, 11:48:48 AM

END TIME

11/27/2024, 11:48:49 AM

TIME TO FIRST TOKEN

N/A

STATUS

Error

TOTAL TOKENS

551,854 tokens / \$0.0827781

LATENCY

0.46s

TYPE

LLM

```
AuthenticationError("Error code: 401 -
{'error': {'message': 'Incorrect API key
provided: sk-YqNTE*****q_EA.
You can find your API key at
https://platform.openai.com/account/api-
keys.', 'type': 'invalid_request_error',
'param': None, 'code':
'invalid_api_key'}}")Traceback (most
recent call last):

File "/usr/local/lib/python3.10/dist-
packages/langchain_core/language_mod
els/chat_models.py", line 633, in generate
self._generate_with_cache(

File "/usr/local/lib/python3.10/dist-
packages/langchain_core/language_mod
els/chat_models.py", line 851, in
_generate_with_cache
result = self._generate(
```

Figure 3: LangSmith Debugging Features

- **Hugging Face:** The Hugging Face environment provided NLI model that was used for mapping educational concepts to Common Core Standards task. It also offers the Transformers library, which makes it easy to load those models and their dependencies, including the tokenizer.

Components of the Pipeline:

- **PDFProcessor Class:** This class Converts PDF pages into optimized, Base64-encoded images suitable for AI processing. Moreover, GPT-4o requires images to be Base64-encoded.
- **ConceptAnalyzer Class:** Uses an GPT-4o model to extract key concepts from images and provided metadata from given mathematical documents. As you could see in figure 4, JSON format is return so that it gets further processed by the NLI model. Each chapter of the textbook is fed to GPT-4o and as output we get JSON format of the concepts the chapter covers.

```

{
  "concepts": [
    {
      "name": "Algebraic Expressions",
      "description": "Understanding and manipulating variable expressions, including evaluating them for given variable values."
    },
    {
      "name": "Substitution",
      "description": "Replacing variables in an expression or equation with specific numerical values to simplify or solve."
    },
    {
      "name": "Order of Operations",
      "description": "A standard sequence (PEMDAS: Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) to evaluate expressions consistently."
    },
    {
      "name": "Exponents",
      "description": "Expressions involving repeated multiplication, such as powers, and their evaluation."
    },
    {
      "name": "Equations and Inequalities",
      "description": "Solving mathematical statements involving equality (=) or inequality (<, >, ≤, ≥)."
    },
    {
      "name": "Functions",
      "description": "Relationships between inputs and outputs, described using equations, tables, or graphs, and understanding domains and ranges."
    },
    {
      "name": "Graphing Functions",
      "description": "Representing functions visually on a coordinate plane and analyzing their behavior."
    }
  ]
}

```

Figure 4: Chapter 1 Processed Output by GPT-4o

```

Qodo Gen: Options | Test this method
def _process_concepts(self, base64_images: List[str], metadata: Dict) -> Dict:
    """Process document for concepts analysis."""
    content = [
        {
            "type": "text",
            "text": f"""
You are an advanced AI assistant tasked with extracting key math concepts from a document. The document
uploaded with the following metadata:
1. **Grade level:** {metadata['grade']}
2. **Domain/Module:** {metadata['domain']}
3. **File name:** {metadata['file_name']}

Your goals are:
1. Identify the **concepts** covered in the document (e.g., algebra, calculus, geometry).
2. For each concept, extract its name and provide a concise description based on the content of the docu
3. Optionally infer the significance of the grade level, domain/module, or file name if it helps clarify
concepts or content.


### Output Schema:
{{
  "concepts": [
    {{
      "name": "string (name of the concept)",
      "description": "string (description of the concept)"
    }}
  ]
}}

Note, make sure you return valid json format with no escape problems"""
        ]
    ]

```

Figure 5: ConceptAnalyzer Method with Concept Extraction Prompt

- **NLIProcessor Class:** The extracted concepts are processed using an NLI model (ynie/xlnet-large-cased-snli_mnli_fever_anli_R1_R2_R3-nli). The NLI model evaluates the semantic relationship between the concepts and the standards to determine the likelihood of a match. It calculates probabilities for entailment, neutrality, or contradiction between the concept descriptions and standard descriptions.
- **DatabaseConnection Class:** The goal here is to Retrieve information about educational standards, domains, and clusters from a database.
- **Evaluation:** Last step is to evaluate the NLI tags against the ground truth that was already tagged by each chapter. Below are the results:



```

metrics = standards_comparator.compare_to_ground_truth(all_standard_matches, ground_truth_standards)

print("Evaluation Metrics:")
print(f"Precision: {metrics['precision']:.2f}")
print(f"Recall: {metrics['recall']:.2f}")
print(f"F1 Score: {metrics['f1']:.2f}")

```

✓ 0.0s Python

```

Evaluation Metrics:
Recall: 0.89
Precision: 0.73
F1 Score: 0.80

```

Figure 6: The Pipeline Evaluation Results

Again, for thorough details, refer to the notebook on the GitHub of the project.

9.6 Outcomes

The project successfully demonstrated the feasibility of an intelligent learning system for K-12 mathematics education. The pipeline effectively automated the extraction of key concepts from educational materials and achieved relatively good results when mapping them to relevant educational standards. This validation confirmed the pipeline's ability to align content with learning objectives.

The pipeline was then integrated into the application. The system now provides content base recommendations based on extracted and tagged materials, showcasing its potential to enhance learning experiences. These outcomes highlight the project's success in combining with advanced AI models.

9.7 Challenges

The project faced several challenges, primarily stemming from time constraints. A lack of time limited the depth of investigation into certain aspects of the pipeline, such as fine-tuning NLI model for better results and conducting comprehensive evaluations. For instance, determining thresholds for semantic matching in the NLI model was based on heuristic methods rather than rigorous experimentation, which may leave room for optimization.

Additionally, while the pipeline demonstrated good initial results, further exploration into alternative models or configurations could enhance its performance. Balancing the development of the application and evaluating the pipeline also posed a challenge, requiring compromises to meet the project timeline. Despite these limitations, the project successfully demonstrated the feasibility of the system and set a solid foundation for future improvements.

10 METHODOLOGY & CAPSTONE DESIGN

10.1 Architecture

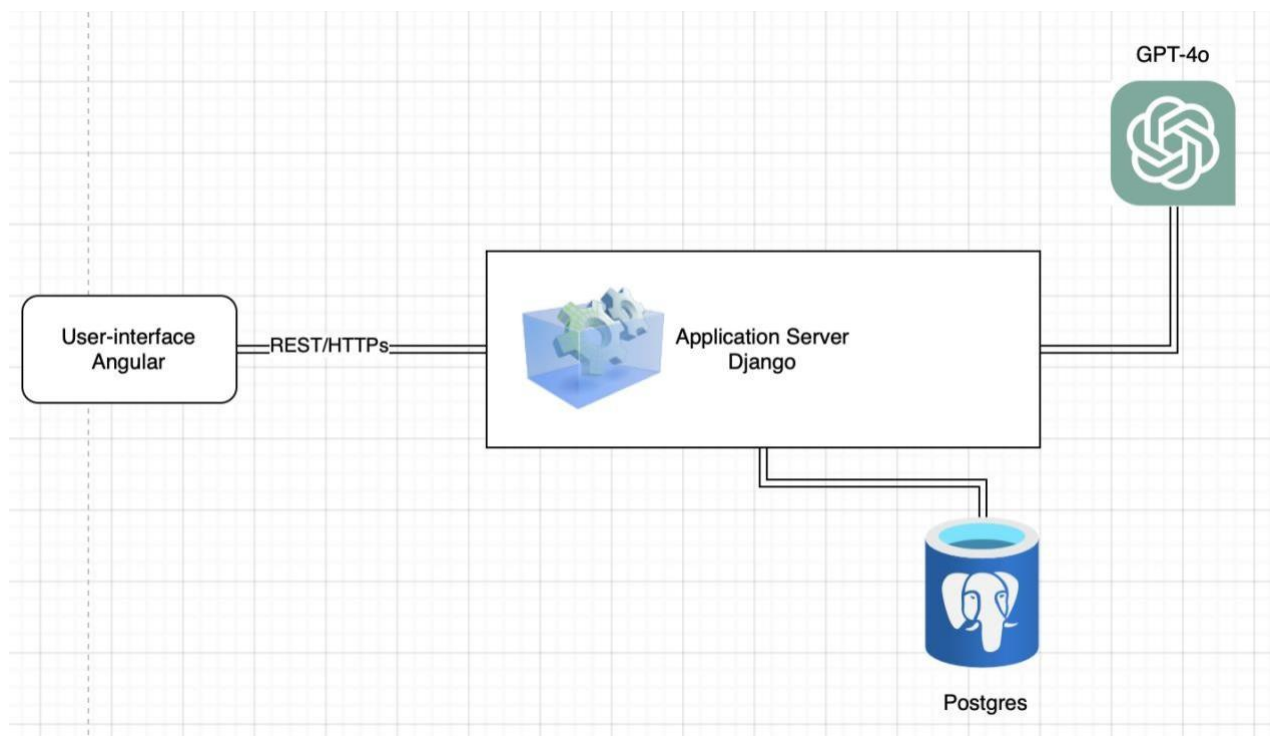


Figure 7: Application's Higher-Level Architecture

Based on the technology stack, The system follows a Model-View-Controller (MVC) architecture to ensure scalability, modularity, and ease of maintenance.

10.2 Design

After establishing the high-level architecture, I proceed to create a detailed design for the application. Initially, I create a class diagram to capture the various components of the system and their relationships. Next, I identify the database tables that correspond to the identified classes.

10.2.1 Class Diagram

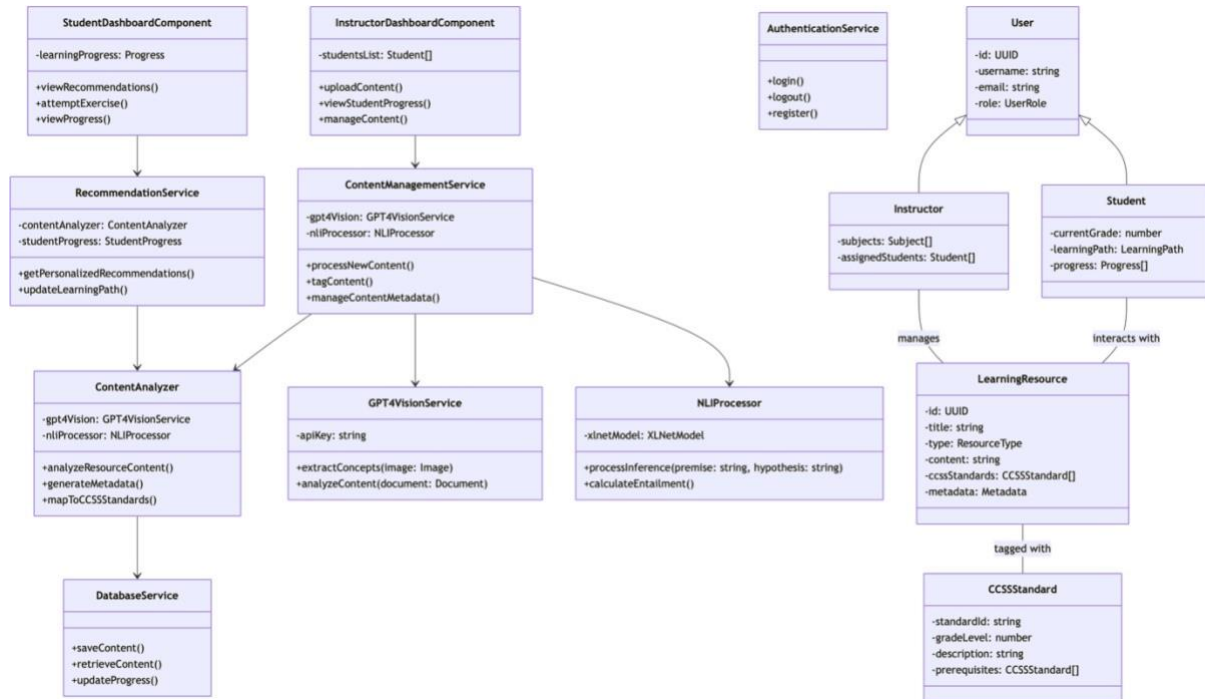


Figure 8: Class Diagram for the Application

10.2.2 Entity-Relationship Diagram (ERD)

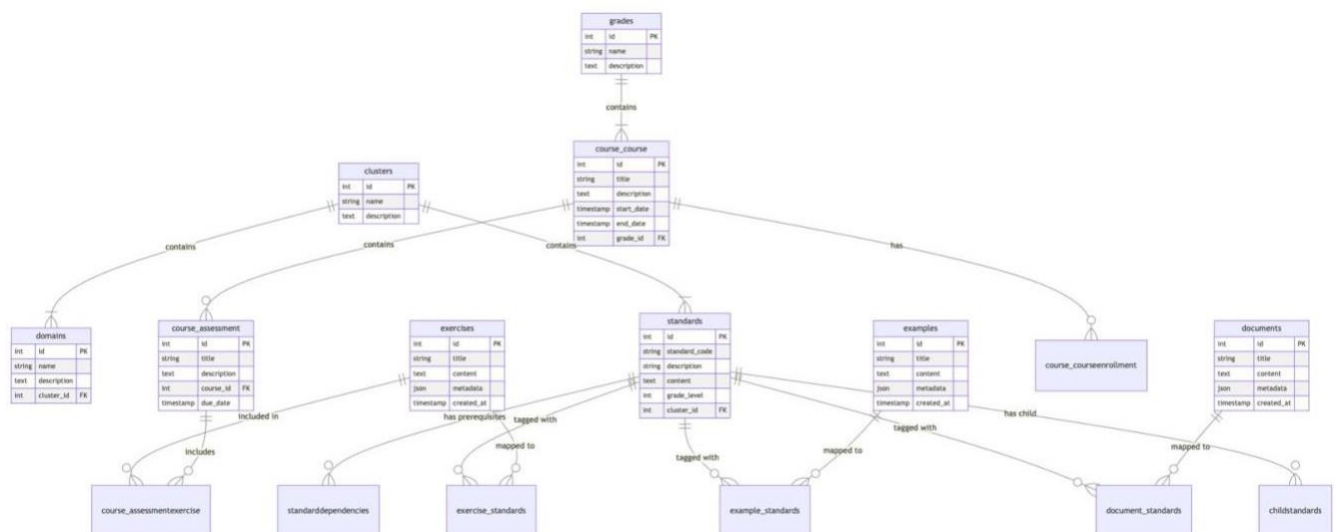


Figure 9: ERD Diagram for the Application

10.3 Implementation Details

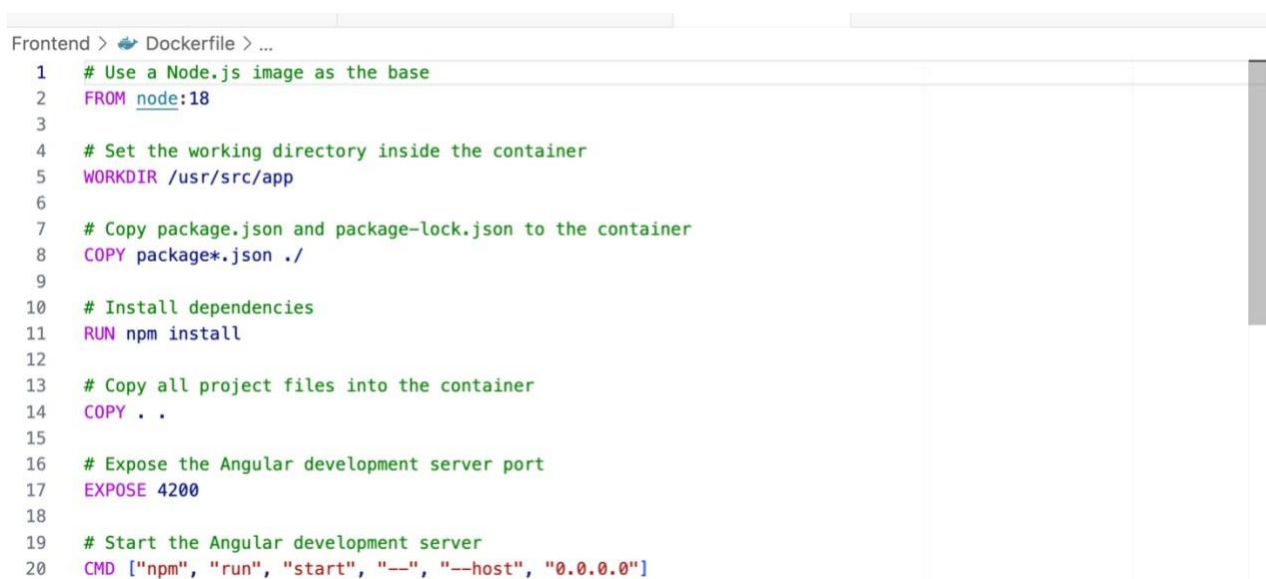
10.3.1 Environment Setup

Firstly, I had to setup the development environment of the application. I took a step-by-step approach to setting up each component individually before introducing Docker for containerization.

10.3.1.1 Frontend

To set up the Angular frontend, I began by ensuring I had Node.js installed on my local machine. Node.js provides the necessary runtime for building and running Angular applications. Once that was confirmed, I installed the Angular CLI globally, as it simplifies the process of creating and managing Angular projects. I used the following command to do so, `npm install -g @angular/cli`. After installation, I created project using `ng new Frontend`. This command automatically generated a well-structured project directory with all the necessary files and configurations. It also installed essential dependencies defined in the package.json file. The Angular CLI was incredibly useful throughout the project, as it allowed me to quickly generate components, services, and other modules. For instance: `ng generate component <component-name>` command. This command not only created the component files but also registered the component in the `app.module.ts` file or their respective module automatically.

After ensuring the application was functional, I prepared it for Docker integration. I created a Dockerfile to containerize the Angular project,



```
Frontend > Dockerfile > ...
1  # Use a Node.js image as the base
2  FROM node:18
3
4  # Set the working directory inside the container
5  WORKDIR /usr/src/app
6
7  # Copy package.json and package-lock.json to the container
8  COPY package*.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy all project files into the container
14 COPY . .
15
16 # Expose the Angular development server port
17 EXPOSE 4200
18
19 # Start the Angular development server
20 CMD ["npm", "run", "start", "--", "--host", "0.0.0.0"]
```

Figure 10: The Angular Dockerfile

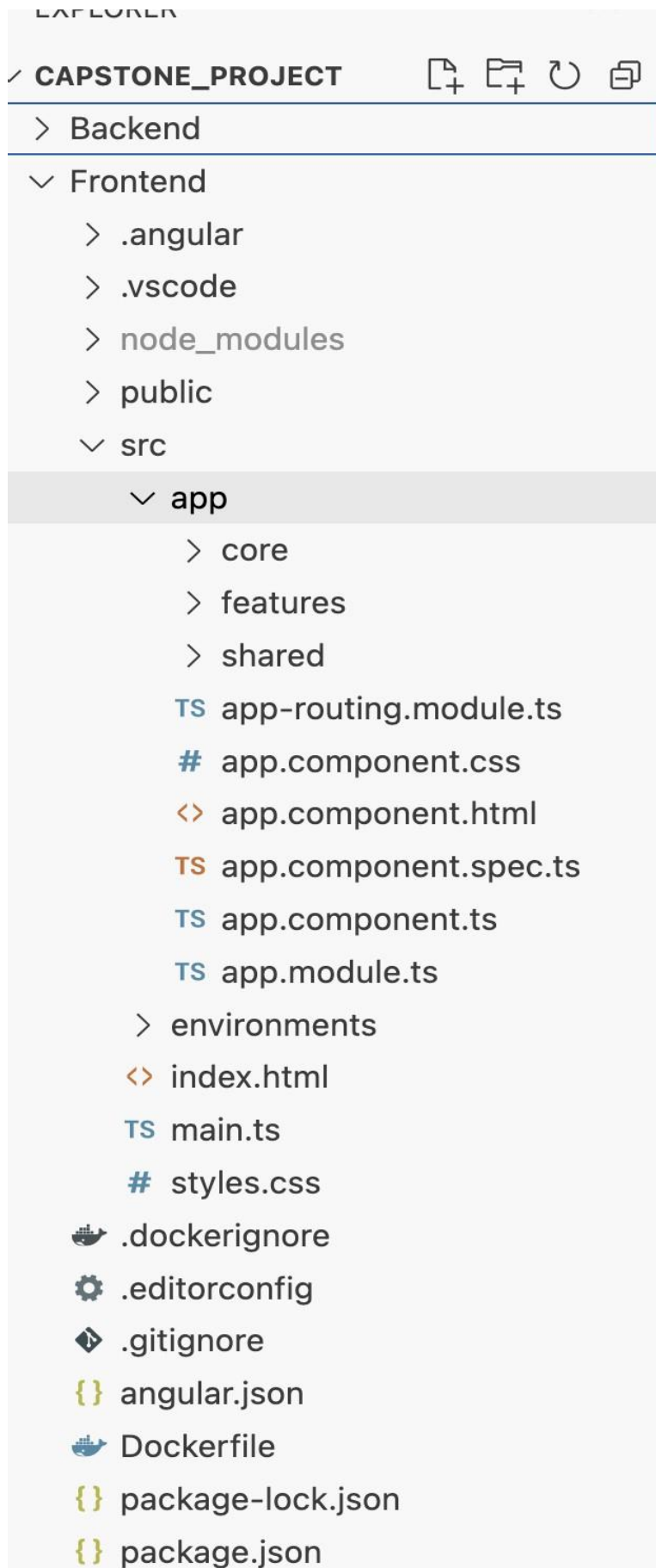


Figure 11: Frontend Structure

10.3.1.2 Backend:

For the backend, I have done similar steps as the frontend setting up the development environment. I began by ensuring that Python was installed on my machine. Next, I installed Django using Python's package manager, pip. Then I created Django project using *django-admin startproject Backend*. This command initialized the backend's structure, including a *manage.py* file for managing the application and a project directory containing the necessary configuration files. After setting up the project, I created the first Django app within the project using *python manage.py startapp authentication*. I registered this app in the *INSTALLED_APPS* section of the *settings.py* file to ensure it was recognized by Django. To manage the application data, I configured Django to use a PostgreSQL database. In the *settings.py* file, I updated the database configuration as the default configuration uses SQLite and provided database credentials through environment variables.

Just like in the frontend, I proceed dockerizing the Django Application.

```
1  # backend/Dockerfile
2
3  FROM python:3.12-slim
4
5  RUN apt-get update && apt-get install -y \
6     poppler-utils
7
8  ENV PYTHONUNBUFFERED=1
9
10 WORKDIR /app
11
12 COPY requirements.txt ./
13 # Upgrade pip to the latest version
14 RUN pip install --upgrade pip && pip install -r requirements.txt
15
16 COPY Backend .
17
18 EXPOSE 8000
19
20 CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
```

Figure 12: Django Dockerfile

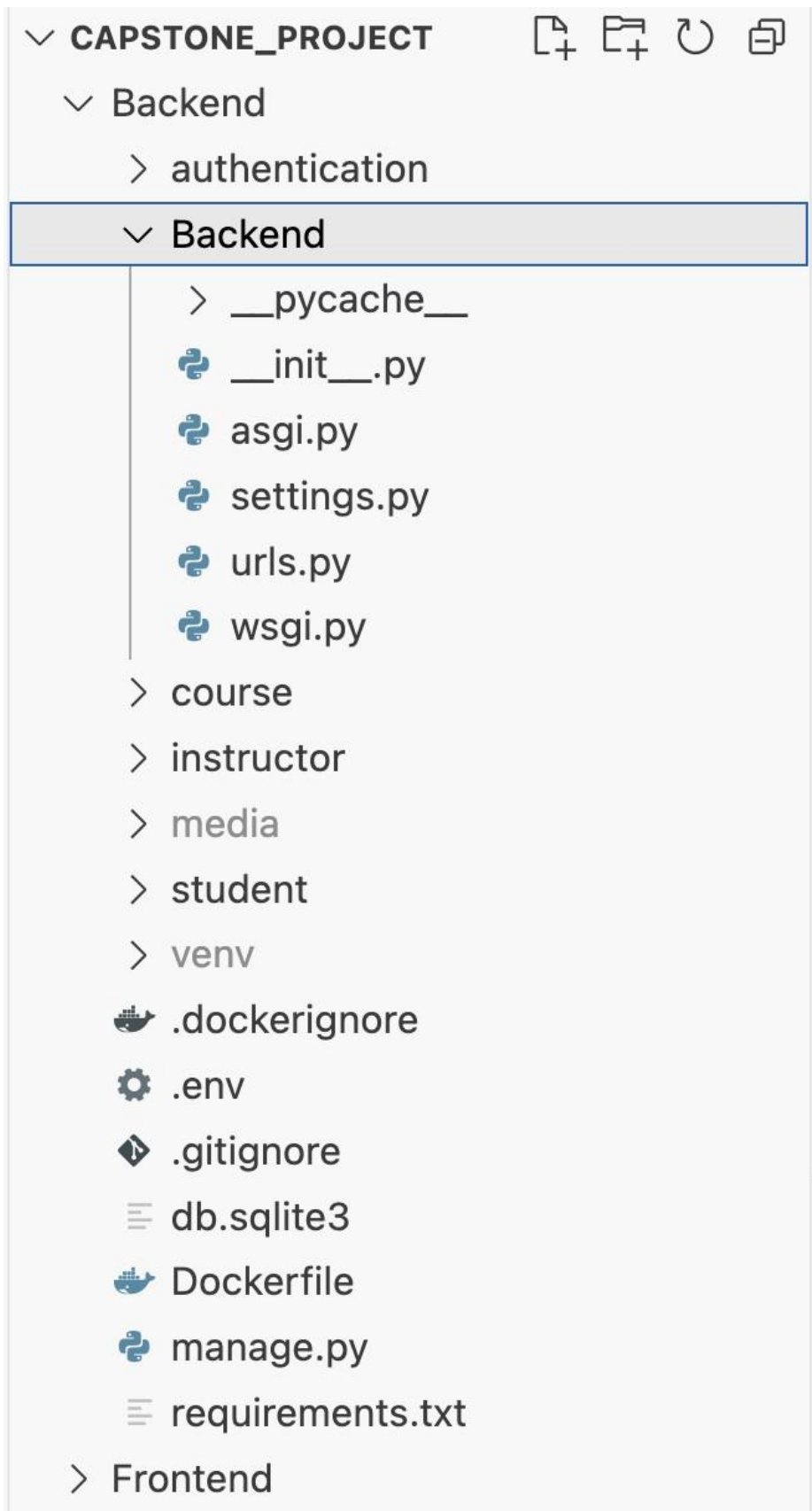


Figure 13: Backend Structure

10.3.1.3 Setting up the Docker Environment

After setting up the Angular frontend, and Django backend, I used **Docker Compose** to integrate these components along with the Database (Postgres) into a cohesive system. Docker Compose allowed me to orchestrate the services, ensuring they worked together seamlessly environments.

```
# docker-compose.yml
version: '3.8'

services:
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
    ports:
      - "4200:4200"
    volumes:
      - ./Frontend:/usr/src/app
      - /usr/src/app/node_modules
    depends_on:
      - backend
    # command: ["npm", "run", "start", "--", "--host", "0.0.0.0"]

  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "8000:8000"
    volumes:
      - ./Backend:/app
      - /app/venv
    env_file:
      - ./backend/.env
    depends_on:
      - db
```

Figure 14: Application's Docker Compose File

```
db:
  image: postgres:latest
  ports: # Move the ports key here, not inside the environment block
    - "5432:5432"
  volumes:
    - db_data:/var/lib/postgresql/data/
    - ./dump.sql:/docker-entrypoint-initdb.d/dump.sql # Mount the dump file
  env_file: # Add this to correctly load your environment variables
    - ./backend/.env

volumes:
  static_volume:
  db_data:
```

Figure 15: Application's Docker Compose File

In the docker compose file, I made some notable configurations for the application and development to be smooth.

- To enhance development efficiency, I mounted the backend and frontend source code directory into the Django and Angular containers respectively. This setup enables real-time hot-reloading of changes made to the codebase without needing to rebuild the Docker image after modifications, streamlining the development process.
- I managed sensitive configuration details, such as database credentials and Open-AI API key, using an `.env` file. This file was loaded into both the backend and database containers using the `env_file` directive. By externalizing these variables, I ensured the security of sensitive data while maintaining flexibility for different environments.
- The application is dependent on some data (particularly the Common Core Standards data). Therefore, it's important initialize the database tables schema and preload essential data during the first startup. Hence, I mounted a `dump.sql` file into the PostgreSQL container. This file contains tables for the Common Core Standards data, which as I mentioned the application is dependent on.
- Additionally, I defined a persistent volume, `db_data`, to ensure that all database data is preserved across container restarts. This approach allows the system to retain critical information, such as user progress and metadata, even if the container is stopped or recreated.

10.3.2 Adding Features

10.3.2.1 *Developing Backend APIs*

- The first major component I implemented in the system was user management. To streamline the process and leverage existing, well-tested functionality, I utilized Django Allauth, a powerful library for handling user authentication and registration. This allowed me to focus on customizing user roles while relying on a proven solution for core user authentication features. For user roles, Django also provides out of the box mechanisms to handle and give permissions to different type users. On the Django admin Interface, I was able create different user groups and assign users to those different groups.

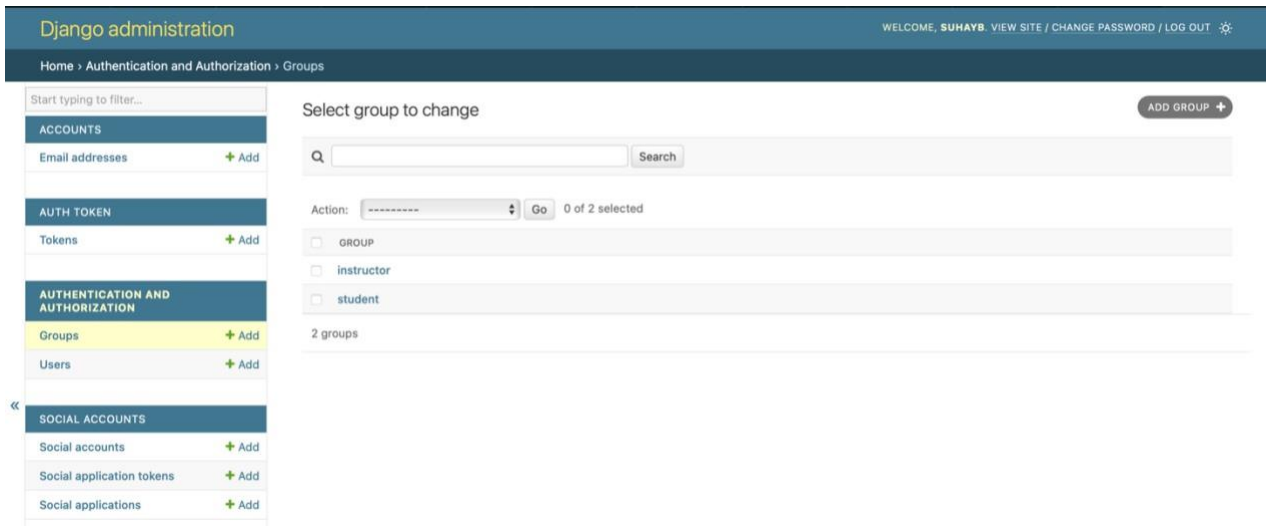


Figure 16: Django Admin Panel

- To build the student, instructor, and course Django apps, I relied heavily on Django's Object-Relational Mapping (ORM) to create and manage database tables without writing raw SQL. Using Django's ORM, I could define models that represented database tables, and Django automatically handled schema generation and migrations when applied. Additionally, I leveraged serializers for data transformation and ModelViewSet to simplify the creation of API endpoints.
 - **Django ORM** allowed me to define Python classes that represent database tables. The fields in the model correspond to the columns in the table. For example:

```

3
Qodo Gen: Options | Test this class
4 class Assessment(models.Model):
5     title = models.CharField(max_length=255)
6     course = models.ForeignKey('Course', on_delete=models.CASCADE, related_name='assessments')
7     created_by = models.ForeignKey(User, on_delete=models.CASCADE, related_name='created_assessments')
8     is_generated = models.BooleanField(default=False)
9     created_at = models.DateTimeField(auto_now_add=True)
10    updated_at = models.DateTimeField(auto_now=True)
1
2    def __str__(self):
3        return f"{self.title} - {self.course.name}"
4
5
Qodo Gen: Options | Test this class
6 class AssessmentExercise(models.Model):
7     assessment = models.ForeignKey(Assessment, on_delete=models.CASCADE, related_name='exercises')
8     exercise = models.ForeignKey(Exercise, on_delete=models.CASCADE)
9     order = models.PositiveIntegerField()
10
11
Qodo Gen: Options | Test this class
12 class Meta:
13     ordering = ['order']
14     unique_together = [
15         ['assessment', 'exercise'],
16         ['assessment', 'order'],
17     ]
18
19

```

Figure 17: Example of Django Model

- Each field in the Assessment model (e.g., title, course, created_by) maps to a column in the database.
- **Relationships:**
 - ForeignKey: Creates a one-to-many relationship. For example, an assessment belongs to a course.
 - related_name: Allows reverse queries, such as accessing all assessments of a course via course.assessments. The exact SQL query that is executed for this is handled by Django's ORM.
- **Timestamps:** The created_at and updated_at fields automatically track when a record is created or updated.
- **Methods:** Custom methods like __str__ improve readability and usability in admin and debugging contexts.
- After defining the model, I used Django's migration system to apply these changes to the database: *python manage.py makemigrations and python manage.py migrate*
- **Serializers in Django REST Framework (DRF)** handle the conversion of complex data types, such as Django models, into native Python datatypes. They also validate and transform incoming data into a format suitable for saving to the database. For example:

```

132
133
134
Qodo Gen: Options | Test this class
135 class AssessmentExerciseSerializer(serializers.ModelSerializer):
136     exercise = ExerciseSerializer()
137
Qodo Gen: Options | Test this class
138 class Meta:
139     model = AssessmentExercise
140     fields = ['id', 'assessment', 'exercise', 'order']
141
Qodo Gen: Options | Test this class
142 class AssessmentSerializerSecond(serializers.ModelSerializer):

```

Figure 18: Django Rest Framework Serializer

- The AssessmentExerciseSerializer is used to serialize (convert to JSON in my case) and deserialize (convert from JSON) AssessmentExercise objects.
- By including the ExerciseSerializer as a nested serializer, I can retrieve detailed information about each exercise linked to an assessment.
- I used **DRF's ModelViewSet** to handle common API operations like list, create, update, and delete. ModelViewSet provides these actions by default, reducing boilerplate code. For example:

```

16 Qodo Gen: Options | Test this class
17 class AssessmentViewSet(viewsets.ModelViewSet):
18     serializer_class = AssessmentSerializer
19     permission_classes = [IsAuthenticated]
20
21     def get_queryset(self):
22         return Assessment.objects.filter(created_by=self.request.user)
23
24 Qodo Gen: Options | Test this method
25 def perform_create(self, serializer):
26     print("DEBUG: Perform create called")
27     serializer.save(created_by=self.request.user)

```

Figure 19: ModelViewSet Class

- For customization, we could override the default methods. As shown in the figure, the `get_queryset` method ensures that users can only access assessments they created, hence overrides the default `get_queryset`. Moreover, the `perform_create` method automatically associates the logged-in user (`self.request.user`) with the newly created assessment. ModelViewSet allows us to set custom permissions and serializer. As shown in figure 19, I provided my own serializer as well built in permission class, however, we could provide our own permissions.

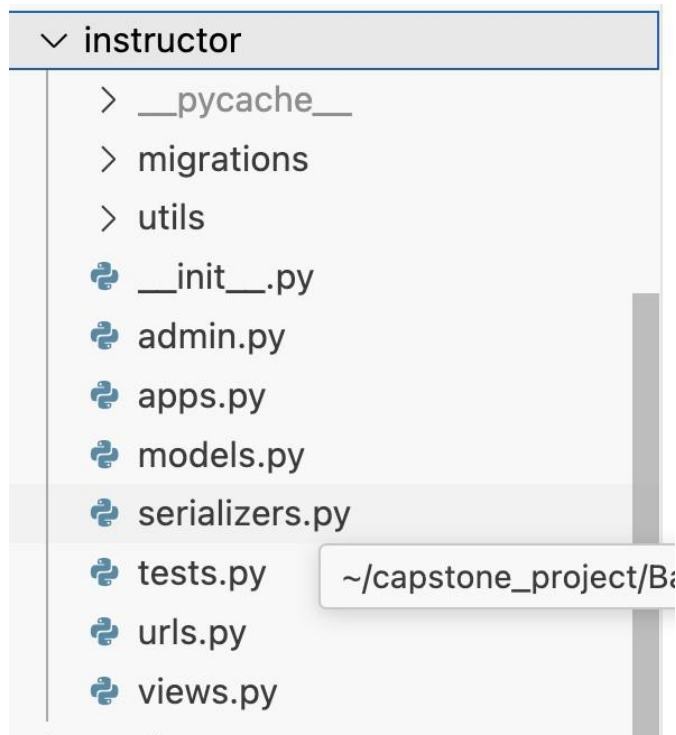


Figure 20: The Structure of the instructor App in Django Backend

10.3.2.2 *Developing Frontend Features:*

Angular's modular architecture is one of its key strengths, allowing the application to be divided into cohesive, reusable, and independently manageable pieces. As evident from the provided project structure in figure 12, the application is organized into three primary modules: **Core**, **Features**, and **Shared**, alongside the root AppModule. Each module has a well-defined responsibility:

1. **Core Module:**

- This module encapsulates singleton services, guards, interceptors, and utilities that are shared across the entire application. For example, the services directory houses services like authentication.service.ts that manage authentication logic, while guards contain route guards to secure pages. The Core Module ensures that these components are instantiated only once and are available globally.

2. **Features Module:**

- The Features Module is divided into submodules (instructor and student), representing distinct areas of functionality. Each submodule contains components, services, and other logic specific to that feature, ensuring a clear separation of concerns and making the application easy to extend or modify.

3. **Shared Module:**

- The Shared Module contains reusable components, layouts, and other elements shared across features. For instance, generic UI components or templates (layouts, headers, footers and so on) can be reused without duplication, reducing redundancy and maintaining consistency.

4. **Root Module:**

- The AppModule acts as the entry point of the application, importing and orchestrating the Core, Features, and Shared modules while setting up global configurations like routing.

This modular setup provides multiple benefits:

- **Scalability:** Features can be developed and maintained independently, making it easier to scale the application as requirements grow.

- **Reusability:** Shared components and services reduce duplication and promote consistency across the application.
- **Maintainability:** Clear separation of concerns enables better code organization and simplifies debugging and testing.
- **Lazy Loading:** Features can be loaded on demand, improving initial load times and application performance.

```
const routes: Routes = [  
  {  
    path: '',  
    component: MainLayoutComponent,  
    canActivate: [AuthGuard],  
    children: [  
      {  
        path: 'student',  
        loadChildren: () =>  
          import('./features/student/student.module').then((m) => m.StudentModule),  
        canActivate: [AuthGuard, RoleGuard],  
        data: { role: 'student' }  
      },  
      {  
        path: 'instructor',  
        loadChildren: () =>  
          import('./features/instructor/instructor.module').then((m) => m.InstructorModule),  
        canActivate: [AuthGuard, RoleGuard],  
        data: { role: 'instructor' }  
      }  
    ]  
  },  
  {  
    path: 'login', component: LoginComponent,  
    canActivate: [LoginGuard]  
  }  
];
```

Figure 21: Main Routing - Modules are Lazily loaded

State Management

To manage the application's state effectively, I used **NgRx**, a reactive state management library for Angular. NgRx provides a predictable state container, enabling a unidirectional data flow and making the application more predictable and maintainable. NgRx centralizes the application's state, making it easier to manage and debug.

```

1  import { createAction, props } from '@ngrx/store';
2  import { TokenResponse } from '../../models/user.model';
3
4
5
6  export const initializeAuth = createAction('[Auth] Initialize')
7
8  export const initializeAuthSuccess = createAction(
9      '[Auth] Initialize Success',
10     props<{ token: TokenResponse | null; roles: string[] | null }>()
11 );
12
13
14
15 export const login = createAction(
16     '[Auth] Login',
17     props<{ username: string; password: string }>()
18 );
19
20 export const loginSuccess = createAction(
21     '[Auth] Login Success',
22     props<{ token: TokenResponse }>()
23 );
24
25 export const loginFailure = createAction(
26     '[Auth] Login Failure',
27     props<{ error: string }>()
28 );
29
30 export const getRoles = createAction('[Auth] Get Roles');
31
32 export const getRolesSuccess = createAction(
33     '[Auth] Get Roles Success',
34     props<{ roles: string[] }>()
35 );

```

Figure 22: Application's Authentication Actions

10.3.3 Security

Security is a critical aspect of any web application, and I implemented several measures to ensure data integrity, secure user authentication, and controlled access to resources. Below, I describe how **JWT**

(JSON Web Tokens) and Angular Guards were utilized to provide a robust security at different layers for both the backend and frontend.

10.3.3.1 Backend Security with JWT

For user authentication and secure data transmission, I used **JWT (JSON Web Tokens)**. JWT is a compact, self-contained token format that securely transmits information between parties.

1. Authentication Workflow:

- When a user logs in, the backend verifies their credentials and generates a JWT containing essential information including user ID and user role.
- The token is then returned to the frontend, where it is stored in the local storage

2. Token Validation:

- For every subsequent API request, the frontend includes the JWT in the Authorization header:

3. Secure Data Transmission:

- Since JWTs are tamper-proof (due to their cryptographic signature), they ensure that the transmitted data has not been altered.
- The payload of the token includes minimal, non-sensitive information to prevent data leakage in case of interception.

```
frontend > src > app > core > interceptors > TS auth.interceptor.ts > [E] authIntercepto
1  // core/interceptors/auth.interceptor.ts
2  import { HttpInterceptorFn } from '@angular/common/http';
3
4  export const authInterceptor: HttpInterceptorFn = (req, next) => {
5      const token = localStorage.getItem('access_token');
6
7      if (token) {
8          const clonedRequest = req.clone({
9              setHeaders: {
10                  Authorization: `Bearer ${token}`
11              }
12          });
13          return next(clonedRequest);
14      }
15
16      return next(req);
17  };
```

Figure 23: Interceptor that Includes Every Request the JWT Stored in the Local Storage

10.3.3.2 Frontend Security Layer

On the frontend, I used Angular Guards to control access to routes and protect sensitive components. Angular Guards act as a security layer by intercepting navigation attempts and allowing or denying access based on custom logic.

```
5 import { Observable, filter, switchMap } from 'rxjs';
6 import { map, withLatestFrom } from 'rxjs/operators';
7 import { AuthState } from '../models/user.model';
8 import { selectIsAuthenticated, selectUserRoles, selectIsInitialized } from '../store/auth/auth.selectors'
9
10
11 Qodo Gen: Options | Test this class
12 @Injectable({
13   providedIn: 'root'
14 })
15 export class LoginGuard implements CanActivate {
16   constructor(
17     private store: Store<{ auth: AuthState }>,
18     private router: Router
19   ) {}
20
21 Qodo Gen: Options | Test this method
22 canActivate(): Observable<boolean | UrlTree> {
23   return this.store.select(selectIsInitialized).pipe(
24     filter(loading => !loading), // Wait for initialization
25     switchMap(() => this.store.select(selectIsAuthenticated).pipe(
26       withLatestFrom(this.store.select(selectUserRoles)),
27       map(([isAuthenticated, roles]) => {
28         if (isAuthenticated) {
29           if (roles && roles.length > 0) {
30             return this.router.createUrlTree(['/${roles[0].toLowerCase()}']);
31           }
32           return this.router.createUrlTree(['/']);
33         }
34         return true;
35       })
36     ))
37   );
38 }
```

Figure 24: Login Guard that prevents Authenticated Users from the Login View

11 RESULTS/ UI OF THE APPLICATION

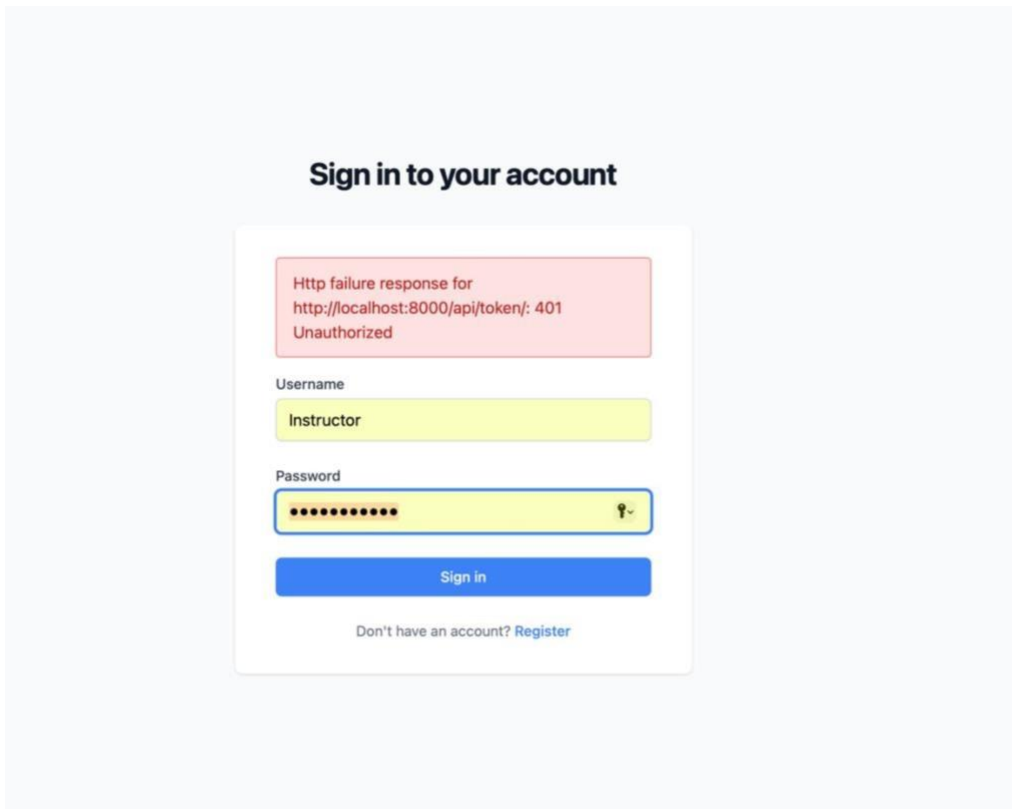


Figure 25: Failed Login

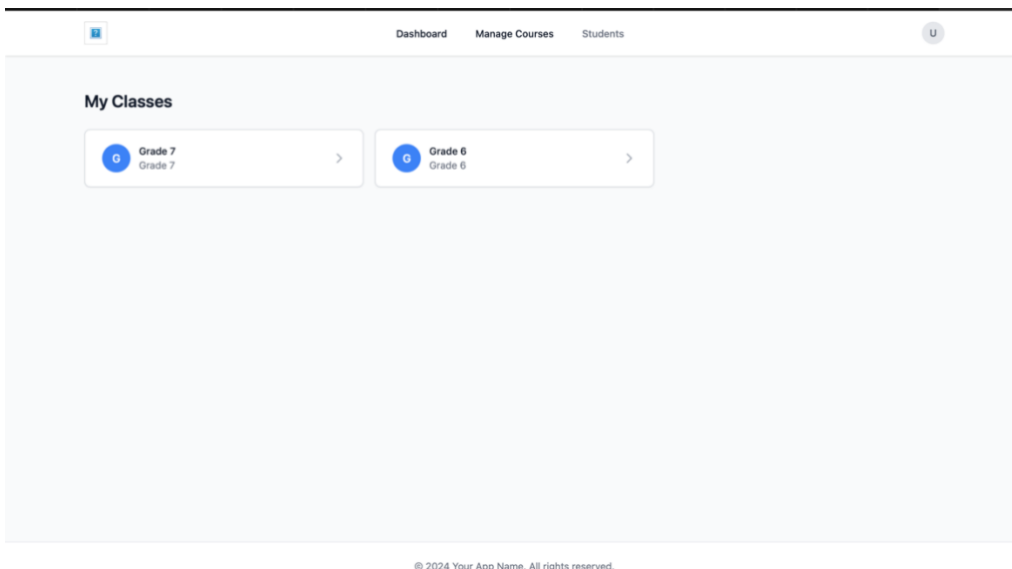


Figure 26: Instructor's Class View

Dashboard
Manage Courses
Students

U

Manage module content

Back to Modules

Documents
Exercises

Generate Exercises

Assessment Title
Homework 1

Number of Exercises
10

Select Cluster
Apply And Extend Previous Understandings Of Arithmetic To Algebraic Expressions.

Select Standard
Select a standard

Generate Exercises

© 2024 Your App Name. All rights reserved.

Figure 27: Instructor's Generate Exercise View

Dashboard
Courses
Assignments

U

Course Materials
Math | Grade 6

Back to Modules

Documents
Exercises

drftest5

proper test2

Uploaded Nov 30, 2024, 4:02:04 PM

⬇

math-g8-m1-topic-a-lesson-4-student

Proper test document for exponentials Angular

Uploaded Nov 30, 2024, 3:35:03 PM

⬇

drftest2

proper test2

Uploaded Nov 30, 2024, 3:32:26 PM

⬇

© 2024 Your App Name. All rights reserved.

Figure 28: Student's Course Material View

Course Materials
Math | Grade 6

Documents Exercises

now1 Back to Assessments

Exercise 0

$$5^{94} \times 5^{78} =$$

Your Answer

Enter your answer here...

Exercise 1

$$(-3)^9 \times (-3)^5 =$$

Your Answer

Figure 29: Student's Exercise View

Exercise 0

$$14^{23} \times 14^{18} =$$

Your Answer

4

Recommendations

Similar Exercises Similar Examples Related Documents

$$(-72)^{10} \times (-72)^{13} =$$

$$5^{94} \times 5^{78} =$$

Figure 30: Student's Recommendation View

12 LEARNING STRATEGIES

To achieve the goals of this capstone project, I employed a combination of learning strategies. I utilized both theoretical and practical knowledge gained from my coursework to guide the development process. Additionally, I extensively reviewed documentation for the technologies used, enabling me to deepen my understanding and apply them effectively. Throughout the project, I leveraged my ability to adapt and relearn, ensuring I could overcome challenges and implement innovative solutions. This dynamic approach to learning was integral to accomplishing the project's objectives.

13 FUTURE WORK

The next phase of this project will focus on enhancing the AI pipeline to improve efficiency and adaptability for broader applications, particularly in Learning Management Systems (LMS). Key areas for improvement include:

1. **Fine-Tuning the NLI Model:** Refining the Natural Language Inference (NLI) model with domain-specific datasets to increase accuracy in mapping educational content to standards and improve the quality of recommendations.
2. **Incorporating Layout Analysis Models:** Introducing advanced layout analysis models to effectively process document structures, isolating and forwarding only relevant sections to the AI pipeline. This approach will reduce computational overhead and operational costs.
3. **LMS Integration:** Leveraging open-source LMS platforms as a foundation, the system will be extended with the developed intelligent features. This integration will provide seamless content recommendation and personalized learning paths within established educational frameworks.

By addressing these areas, the project aims to deliver a cost-effective, scalable solution that enhances the educational experience for students and instructors alike while minimizing system complexity and operational expenses.

14 CONCLUSION

This project represents a significant step forward in enhancing K-12 mathematics education through an intelligent, personalized learning platform. By integrating robust backend systems, a scalable frontend architecture, and advanced technologies like AI and machine learning, the application delivers tailored content to students, addressing their unique learning needs while empowering instructors to track and guide progress effectively. Adhering to international standards ensures the system is reliable, secure, and ethically designed, fostering trust and credibility among users.

REFERENCES:

“Angular”, <https://angular.dev>

Common Core State Standards Initiative. (2010). *Common core state standards for mathematics*.
https://corestandards.org/wp-content/uploads/2023/09/Math_Standards1.pdf

Ennouamani, S., & Mahani, Z. (2017, December 1). *An overview of adaptive e-learning systems*. IEEE
Xplore. <https://doi.org/10.1109/IN%E2%84%A1CIS.2017.8260060>

“Djanog”, <https://www.django-rest-framework.org>

Kabudi, Tumaini, et al. “AI-Enabled Adaptive Learning Systems: A Systematic Mapping of the
Literature.” *Computers and Education: Artificial Intelligence*, vol. 2, Mar. 2021, p. 100017,
<https://doi.org/10.1016/j.caeai.2021.100017>

APPENDIX A: CODE

<https://github.com/SuhWali/Capstone>