BACHELOR PROJECT

AI-Enhanced Comprehensive Learning Platform

Contents

v.1: Bachelor project

Exploration of AI-Powered Tutoring Systems and Personalized Learning Solutions

Objective

The initial phase of our project is dedicated to a thorough examination of the current landscape of AI-powered tutoring systems and personalized learning solutions. Our goal is to investigate existing models, dissect their strengths and weaknesses, compare their methodologies, and delve into their architecture and implementation strategies. This exploration is crucial for laying the foundational knowledge required to develop an innovative approach to AI tutoring. By addressing identified gaps and incorporating cutting-edge technologies, we aim to enhance the efficiency, inclusivity, and adaptability of educational environments.

Research Methodology

To achieve our objectives, we will employ a multi-faceted research methodology comprising the following components:

Literature Review

- **Objective**: To conduct an exhaustive literature review aimed at identifying critical peer-reviewed articles, case studies, and white papers that detail the latest advancements and challenges in AI tutoring and personalized learning systems.
- **Approach**: Utilize academic databases, scholarly journals, and technology publications to gather relevant literature. This process will help in understanding the theoretical frameworks, empirical findings, and practical applications of current AI tutoring systems.

Technology Assessment

- **Objective**: To evaluate the underlying technologies and algorithms of existing models, with a particular focus on their adaptability to diverse learning styles, scalability across various educational contexts, and effectiveness in engaging users.
- **Approach**: Analyze the technical documentation, developer guides, and research papers related to prominent AI tutoring platforms. Special attention will be given to AI methodologies such as machine learning, natural language processing, and adaptive learning technologies that facilitate personalized education.

Expected Outcomes

The culmination of this phase will result in a comprehensive understanding of the state-of-the-art in AI-powered tutoring and personalized learning. By synthesizing insights from our literature review, technology assessment, and market analysis, we will identify key gaps and opportunities in the

current ecosystem. This knowledge will serve as the cornerstone for the next phase of our project, where we will design and propose an innovative AI tutoring model that aims to surpass the limitations of existing solutions and significantly enhance the learning experience.

State of the art:

State of the Art: Integrating AI in Educational Tools for Enhanced Learning

This section outlines the state-of-the-art framework for the proposed project, which aims to integrate various AI-driven tools to improve educational delivery, assessment, and administration. The project leverages these tools to create a comprehensive, personalized, and efficient learning environment for students.

1. Content Creation and Personalization

GitHub Copilot

- **Application**: Automating the creation of structured lesson plans and personalized study materials.
- **Methodology**: Analyzing curriculum standards and student data to generate relevant content through AI.

ChatGPT

- **Application**: Generating lesson plan rubrics and creative prompts for students.
- **Methodology**: Acting as a smart assistant for teachers, providing suggestions and information retrieval for lesson improvement.

Smart Sparrow and Knewton (A Wiley Brand)

- **Application**: Designing personalized and differentiated learning paths.
- **Methodology**: Leveraging AI to adapt learning experiences to each student's pace and understanding.

2. Interactive Learning and Assessment

ClassPoint

- **Application**: Engaging students with AI-driven quizzes and interactive slides.
- **Methodology**: Using interactive elements to gather instant feedback and adjust teaching strategies.

DreamBox Learning

- **Application**: Specializing in math and reading for differentiated learning.
- **Methodology**: Tailoring content to the learner's level through intelligent adaptivity.

3. Exam Preparation and Analytics

ExamSoft

- **Application**: Creating personalized exams and analytics.
- **Methodology**: Analyzing student performance data to tailor exams and identify learning outcomes improvements.

Implementation Methodology

- **Integration and Customization**: Ensure the complementary integration of tools to address specific educational needs.
- **Data-Driven Decision Making**: Use data analytics for monitoring tool effectiveness and adapting learning paths.
- **Continuous Improvement and Adaptation**: Stay updated with AI and educational technology advancements to enhance learning outcomes.

This framework emphasizes the importance of a cohesive, adaptive, and data-informed approach in using AI tools to revolutionize education, aiming to achieve a personalized, engaging, and inclusive learning environment.

Scientific question:

Study the Impact of Adaptive Learning on education: does it ensure student interaction, and provide optimal learning outcomes?

Scientific paper: https://www.mdpi.com/2071-1050/15/4/3115

The study offers a detailed bibliometric analysis on adaptive learning from 2000 to 2022, highlighting rapid development within this research area. Key findings include:

A significant contribution from authors like Qiao J. F., Han H. G., and Song Q., with China leading in publications.

Core journals identified, notably IEEE Transactions on Neural Networks and Learning Systems, emphasize the field's growth.

Major research topics emerged: deep learning applications in education, development of adaptive learning models, intelligent tutoring systems, and advanced modeling technologies for learning.

Technological advancements are driving the field's evolution, focusing on feature extraction, adaptation models, and computational modeling as current research frontiers.

This analysis underscores adaptive learning's interdisciplinary nature, bridging computer science and educational technology, and highlights the crucial role of emerging technologies in education.

Scientific paper: https://www.mdpi.com/2073-431X/10/2/16

The document "A Review of Agent-Based Programming for Multi-Agent Systems" presents a systematic examination of agent-based programming (ABP) languages, their applications, extensions, and comparative analyses. It emphasizes the role of intelligent, autonomous agents in symbolic artificial intelligence, where agents reactively or proactively decide on actions based on available world information. The paper highlights the diverse range of agent programming languages (APLs), including both established and emerging languages, and discusses their applications across various domains.

Key insights include:

- **Agent Programming Languages (APLs):** The review identifies several APLs, focusing on both well-maintained veteran languages and newer entrants. The Belief-Desire-Intention (BDI) model emerges as a popular framework among the discussed APLs, which are often implemented in Java for cross-platform compatibility.
- **Extensions and Comparisons:** The paper discusses significant extensions to existing APLs that enhance their functionality or apply them to new, specific scenarios. It also addresses the challenge of qualitatively and quantitatively comparing these languages, given their diverse underlying models of agency.
- **Applications:** A wide range of applications is covered, illustrating the utility of APLs in domains like electricity markets, robotics, and the Internet of Things (IoT). The Multi-Agent Programming Contest (MAPC) is highlighted as a platform for testing and comparing APLs in complex scenarios.

The paper underscores the importance of bridging the gap between theoretical agent concepts and practical programming paradigms to encourage wider adoption of APLs. It suggests that future research should focus on addressing usability issues, including improving documentation, providing more real-world examples, and enhancing the development of benchmarks for APL evaluation.

This document serves as a comprehensive resource for understanding the current state of agent-based programming, identifying future research directions, and appreciating the practical applications of agent technologies in solving complex, distributed problems.

Setting Up a Self-Improving GPT Model in a Django Project

Introduction

This section outlines the process of integrating a self-improving Generative Pre-trained Transformer (GPT) model into a Django-based project. The aim is to enable the GPT model to autonomously update its knowledge base with new information, thereby providing increasingly relevant and interesting outputs for the project's requirements.

Requirements

- Python (3.8 or newer)
- Django (3.2 or newer)
- Transformers library by Hugging Face
- A database setup for Django (e.g., SQLite for development, PostgreSQL for production)
- Access to a GPT model (e.g., GPT-3 via OpenAI API or a suitable alternative)

Step-by-Step Guide

Step 1: Environment Setup

1. Create a virtual environment for your Django project:

   bashCopy code

   python -m venv venv

2. Activate the virtual environment:
   - On Windows: **venv\Scripts\activate**
   - On Unix or MacOS: **source venv/bin/activate**
3. Install Django and the Transformers library within the virtual environment:

   bashCopy code

   pip install django transformers

Step 2: Django Project and Application Setup

1. Start a new Django project if you haven't already:

   bashCopy code

   django-admin startproject yourprojectname

2. Navigate to your project directory and create a new Django app:

bashCopy code

python manage.py startapp ai_module

Step 3: Integrate GPT Model

1. Obtain API keys if you're using a hosted GPT model like GPT-3 from OpenAI.
2. In your Django app (**ai_module**), create a new Python script (e.g., **ai_integration.py**) to handle interactions with the GPT model.
3. Use the Transformers library to load your GPT model. For hosted models, configure the API request to include your API key and desired parameters (temperature, max tokens, etc.).

Example:

pythonCopy code

from transformers import pipeline def generate_response(prompt): generator = pipeline('text-generation', model='gpt-2') # Example with GPT-2 response = generator(prompt, max_length=50, num_return_sequences=1) return response[0]['generated_text']

Step 4: Dynamic Learning and Updating

1. Implement a mechanism to collect new information from user inputs or other data sources within your Django app.
2. Design a function to update the model's knowledge. For self-hosted models, this could involve fine-tuning the model on new data. For hosted services, consider periodically sending aggregated insights to improve future queries.
3. Ensure compliance with data privacy and model usage policies when collecting and using data for updates.

Step 5: Integrating the AI Module with Django Views

1. In your **ai_module**, create or update views to handle requests that require AI-generated content.
2. Use the **generate_response** function within these views to get the GPT model's output.
3. Return the model's response as part of your HTTP response, ensuring it's appropriately formatted for your front end.

Step 6: Testing and Iteration

1. Test the integration thoroughly with different prompts to ensure the model provides relevant and accurate responses.
2. Collect feedback on the AI-generated content and refine your data update mechanisms and model configurations accordingly.

Conclusion

By following these steps, you can set up a GPT model within your Django project that not only enhances the project's capabilities with AI-generated content but also improves over time as it receives new information. This approach ensures that your application remains at the cutting edge of AI-driven innovations, providing value through dynamic learning and adaptation.

Building agents (AI tutor and personalized learning) with Flowise

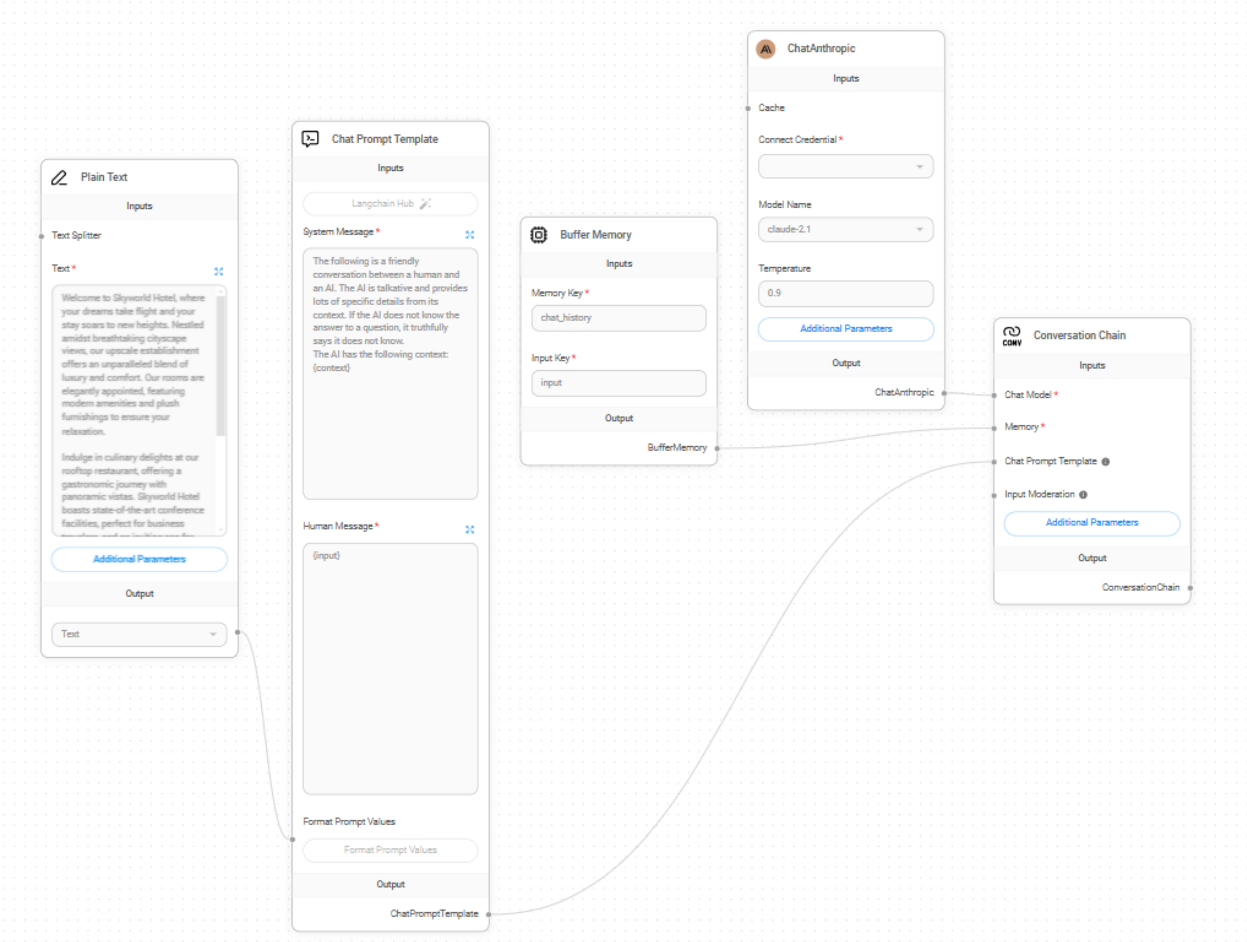
Flowise is an advanced platform that empowers developers to create custom Large Language Model (LLM) applications through an intuitive drag-and-drop interface. It facilitates seamless integration into existing applications via APIs, SDKs, and an embedded chat widget. This platform supports a variety of open-source LLMs, including HuggingFace and Orca, and enables self-hosting on major cloud services like AWS, Azure, and GCP, making it a versatile tool for developing AI-driven solutions.

In the context of building AI assistants or tutors for a bachelor project, Flowise offers a unique approach to enhance chatbot capabilities beyond what standard LLMs provide. For instance, while LLM-based chatbots are powerful, they have limitations, such as the inability to perform web searches or complex mathematical reasoning effectively. Flowise addresses these limitations by allowing the integration of specialized nodes into the chatbot's workflow, such as a ConversationalAgent node for orchestration, a Buffer Memory node for context retention, and a Chat OpenAI node equipped with GPT models for advanced conversational abilities. Additionally, tool nodes like Calculator for arithmetic tasks and SERP API for internet queries can be integrated, significantly expanding the chatbot's functionality.

This system enables the creation of sophisticated AI assistants capable of conducting quizzes, facilitating educational content, and providing personalized tutoring experiences. By leveraging the modularity of Flowise, developers can construct AI agents that not only respond to queries but also remember conversation histories, access the internet for real-time information retrieval, and perform specific tasks requested by users. This approach allows for the development of a highly interactive and responsive educational tool, tailor-made to the requirements of a bachelor project.

For a detailed guide on setting up your first AI agent with Flowise, including step-by-step instructions on configuring nodes and integrating API keys, the Just Ship AI tutorial offers a practical walkthrough. This resource can serve as a valuable reference for incorporating advanced AI functionalities into your project, ensuring a comprehensive and engaging user experience.

Github: https://github.com/FlowiseAI/Flowise



Personalized learning inside the webapp

Integrating a machine learning model to personalize learning experiences in web applications offers a tailored approach to education, accommodating individual learning speeds, styles, and levels. This document outlines a structured method for implementing such a model, exemplified by a model found on Hugging Face or similar platforms, focusing on categorizing users into different learning levels: beginner, intermediate, and advanced.

1. Model Comprehension

Firstly, it's imperative to fully understand the chosen model's capabilities, inputs, and outputs. Key aspects include:

- **Inputs:** Identify the data the model requires, which might encompass user responses, interaction behaviors, or stated preferences.
- **Outputs:** Comprehend how the model's outputs correlate with user learning preferences and abilities, facilitating their alignment with predefined learning levels.

2. Data Collection

Gather initial data through:

- **Onboarding Quizzes/Surveys:** Craft these tools to get the user's knowledge base and learning preferences.
- **Activity Analysis:** Monitor and analyze how users engage with the platform, focusing on preferred content types and interaction patterns.

3. Model and Web App Integration

- **API Integration:** Leverage the Hugging Face API (or equivalent) to connect your web application with the model, ensuring seamless data exchange.
- **Data Processing:** Prepare your application to interpret the model's output, aligning it with your educational categorizations (beginner, intermediate, advanced).

4. Learning Experience Personalization

- **Adaptive Content:** Modify content difficulty and format based on the model's analysis, ensuring it meets the learner's current level and preferences.
- **Feedback Mechanisms:** Utilize quizzes and exercises to validate the model's level placement, adjusting as necessary to optimize learning.

5. Continuous Adaptation and Improvement

- **User Feedback Collection:** Implement feedback channels for users to express their satisfaction with the content's difficulty and relevance.

6. Privacy and Transparency

- **Privacy Policy Transparency:** Clearly communicate how user data is utilized for personalization purposes, adhering to all relevant data protection laws.
- **Informed Consent:** Explicitly acquire user consent for data collection and analysis.

7. Evaluation and Iterative Development

- **A/B Testing:** Conduct comparative studies between personalized and static learning paths to assess the effectiveness of personalization.
- **Iterative Refinement:** Use outcomes and feedback to continuously refine the integration, potentially retraining the model with specific user data to enhance its predictive accuracy.

8. Documentation

- **Guidance Materials:** Provide comprehensive guides explaining the personalized learning system's operation and maximizing its benefits.

9. Models on Huggingface:

Examples: https://huggingface.co/Minej/bert-base-personality
https://huggingface.co/Nasserelsaman/microsoft-finetuned-personality

This are models, that already have personalized learning, for my case I could change the input of the different angles of the models and use it as use case for my project. There are a lot of different models and even we can put different models together so that they need to work together.

Developing a GUI for AI Interactions in Django

Overview

The GUI serves as the bridge between your project's AI capabilities and its users, enabling intuitive interactions and display of the AI-generated content. By following best practices in Django template creation and seeking inspiration from existing projects, you can create a compelling user experience that showcases the innovative features of your AI-driven application.

Design Considerations

- **User Experience (UX)**: Prioritize simplicity and clarity in your design to facilitate ease of use.
- **Responsive Design**: Ensure your GUI is accessible across various devices and screen sizes.
- **Interactivity**: Incorporate interactive elements that engage users and encourage them to explore AI-generated content.

Implementation Steps

Step 1: Define the Layout

1. Sketch the layout of your GUI, identifying key components such as input fields for user prompts, areas to display AI-generated responses, and navigation elements.
2. Consider using Bootstrap or another CSS framework to expedite development and ensure a responsive design.

Step 2: Explore Existing Templates and Work

1. Look for Django template repositories or open-source projects for inspiration. Websites like GitHub, GitLab, or Django Packages can be valuable resources.
2. Evaluate templates and projects that have integrated AI or chat functionalities to understand how they structure their GUI and manage user interactions.

Step 3: Customize Your Django Templates

1. Based on your layout and inspiration from existing work, create or modify templates in your Django app. These templates should include:

- **Base Template**: Defines the overall HTML structure (header, footer, etc.) and includes links to CSS and JavaScript files.
- **Home Template**: The main interface for user interaction with the GPT model, including form fields for input and a section to display responses.

2. Utilize Django's template inheritance feature to maintain consistency across your application and reduce redundancy.

Step 4: Implement AJAX for Dynamic Content Loading

1. To enhance user experience, use AJAX to submit user prompts and display AI-generated responses without reloading the page.
2. Update your Django views to handle AJAX requests and return JSON responses that include the AI-generated content.

Step 5: Testing and Refinement

1. Test the GUI across different devices and browsers to ensure compatibility and responsiveness.
2. Gather user feedback on the GUI's usability and make necessary adjustments to improve the interface and interaction design.

Conclusion

By carefully designing and implementing a GUI for your Django project, you can significantly enhance the way users interact with your AI-driven features. Drawing inspiration from existing templates and projects, combined with thoughtful customization and testing, will lead to a user-friendly interface that effectively showcases the capabilities of your self-improving GPT model.

Architecture:

Architecture for an AI-Enhanced Comprehensive Learning Platform, designed to revolutionize the educational experience by integrating advanced AI to offer personalized learning pathways, real-time tutoring, and a suite of interactive tools aimed at enhancing student engagement and learning outcomes.

User Registration and Login: The platform provides a seamless user registration and login process. New users are prompted to register by entering their email, password, full name, and phone number, ensuring a secure and personalized experience from the outset. Upon registration, users undergo an initial skill assessment, answering questions related to their proficiency in various computer languages (e.g., Python, Java) and overall coding skills (ranging from Beginner to Engineer). This critical step allows the AI to tailor the learning pathway to each student's unique needs and skill levels.

Skill Assessment: Before accessing the main features of the web application, new users complete a skill assessment questionnaire. Questions gauge the user's proficiency in several programming

languages and their coding experience level. This information is pivotal for customizing the learning experience, ensuring that content is appropriately challenging and relevant.

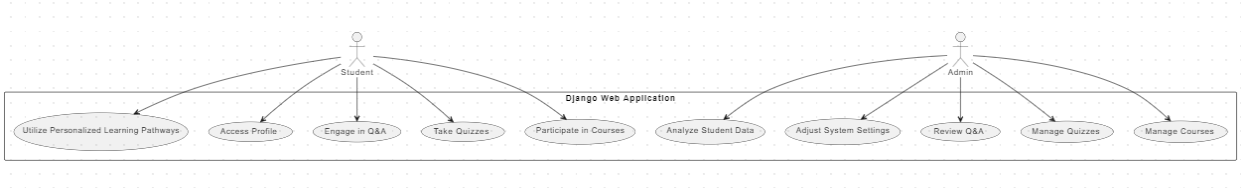
Dashboard and Features: Post-login, the dashboard serves as the central hub for accessing the platform's diverse features:

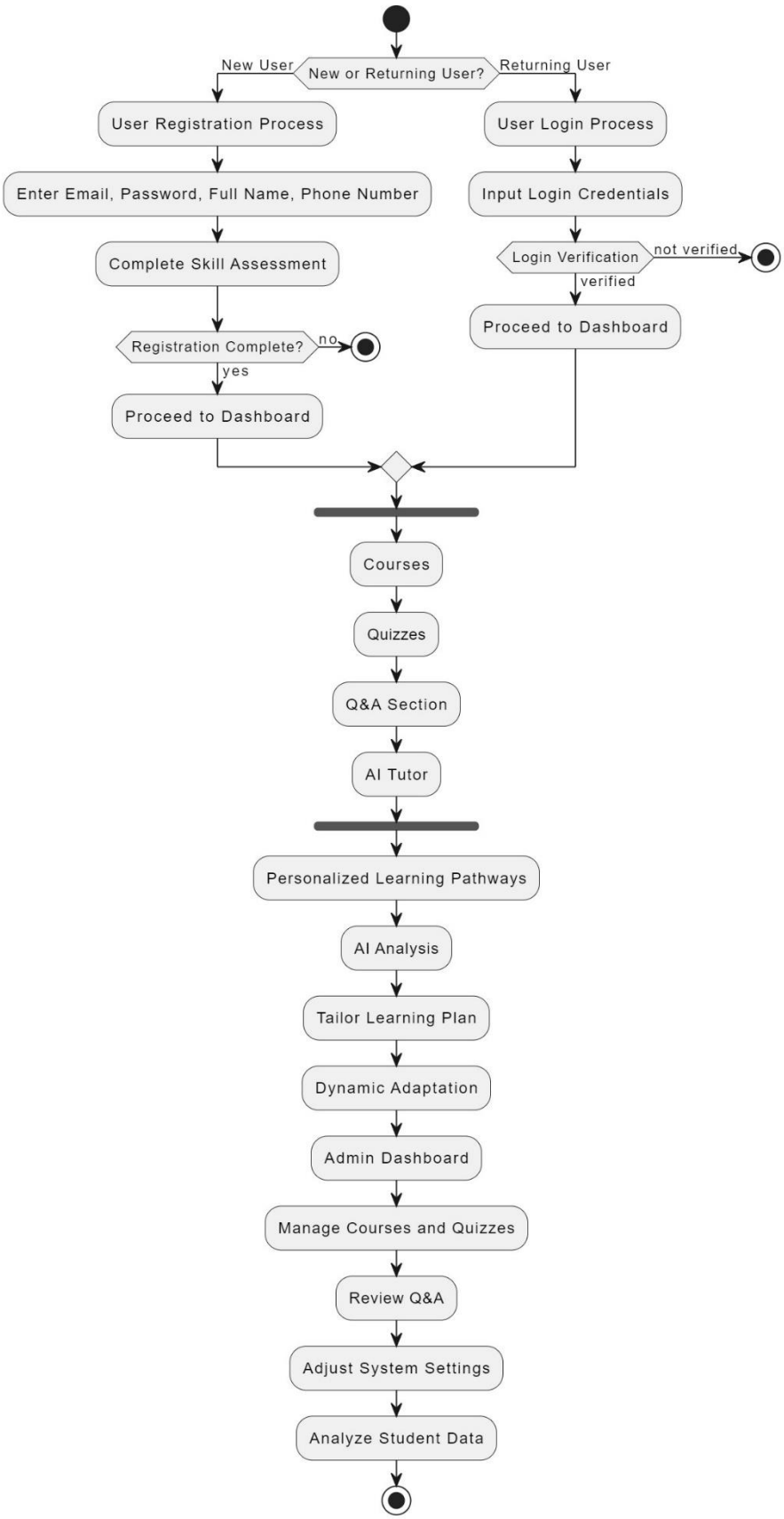
- **Courses**: Students can enroll in and participate in various courses tailored to their learning paths.
- **Quizzes**: Interactive quizzes allow students to assess their understanding and retention of course material.
- **Q&A Section**: A collaborative space for students to ask questions and receive answers, fostering a community of learning.
- **AI Tutor**: The intelligent tutoring interface offers real-time assistance, homework help, and simulates test environments for practice and assessment, providing explanations and guidance at the student's level.
- **Personalized Learning Pathways**: Powered by AI, the platform analyzes each student's strengths, weaknesses, learning styles, and pace, curating a customized learning plan that adapts dynamically to progress and feedback.

Admin Dashboard: For administrators, the platform offers robust tools for managing courses, quizzes, and user engagement. Admins can adjust system settings, analyze student data, and refine the educational content based on feedback, ensuring the platform remains adaptive and efficient across diverse educational environments.

Core Objective and Features: At its core, the platform aims to address the adaptive learning challenge, questioning how AI-driven personalization affects student interaction and learning outcomes. By integrating personalized learning pathways with an intelligent tutoring interface, the platform aspires to scale adaptability and efficiency, making it an indispensable tool for after-school learning, homework assistance, and exam preparation.

Backend:




New or Returning User?
New User
Returning User
User Registration Process
Enter Email, Password, Full Name, Phone Number
Complete Skill Assessment
Registration Complete?
no
yes
Proceed to Dashboard
User Login Process
Input Login Credentials
Login Verification
not verified
verified
Proceed to Dashboard
Courses
Quizzes
Q&A Section
AI Tutor
Personalized Learning Pathways
AI Analysis
Tailor Learning Plan
Dynamic Adaptation
Admin Dashboard
Manage Courses and Quizzes
Review Q&A
Adjust System Settings
Analyze Student Data

PDF based AI agents that to the lessons:

In the architecture of the AI-Enhanced Comprehensive Learning Platform, a pivotal feature is the integration of an AI Assistant capable of transforming PDF content into structured lessons tailored to individual students. This AI Assistant employs advanced natural language processing (NLP) and machine learning algorithms to analyze educational materials provided in PDF format. It then intelligently segments this content into discrete, digestible lessons that align with the curriculum's scope and sequence. By evaluating the student's progress, preferences, and performance data, the AI dynamically adjusts the content complexity and topics to ensure personalized learning paths. This approach not only facilitates a more engaging and effective learning experience but also allows for the automated creation of a diverse and adaptive curriculum. The AI Assistant's capacity to convert static PDF documents into interactive, personalized learning experiences represents a significant leap forward in educational technology, offering scalable, personalized education solutions that meet the unique needs of each learner.

PDF for python lesson:
https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf

PDF for java lessons :
https://www.cs.cmu.edu/afs/cs.cmu.edu/user/gchen/www/download/java/LearnJava.pdf

PDF for JavaScript lessons:
https://www.tutorialspoint.com/javascript/javascript_tutorial.pdf

PDF for C lessons:
https://vardhaman.org/wp-content/uploads/2021/03/CP.pdf