Software Requirement Specification

Introduction

Purpose

This software will be a web application to be used by students who wish to test their readiness to study new machine learning concepts.

Project Scope:

The system will allow users to input a concept they wish to learn in Machine Learning. It will then fetch the prerequisites required to fully understand the concept and generate an assessment to test the users understanding of the prerequisite concepts. The quiz generation, and its evaluation will be automated. The prerequisites graph needs to be added manually and can be automated in the future.

Environmental Characteristics

On the client side, the web app can be accessed using any web browser with no installation on either the laptop or the phone, with UI features being primarily designed to be compatible with laptops. The front end is hosted on Netlify and the backend is hosted on Render. External software dependencies are React, Vite, Node.js, Express and Mongoose. It will also use third party API keys to access two machine learning models for quiz generation and the chatbot. This party API keys will also be used for password reset.

Overall Description

Product Perspective

EduAssess allows students to ensure they possess sufficient prerequisite knowledge before learning a new concept allowing them to systematically progress through their course work. It is a new standalone platform that is not supposed to be a replacement for any existing software.

Product Features

The project aims to take a machine learning concept as input from the user and generate a list of questions which can be used to accurately understand the student's foundation and assess whether they can proceed to that concept. The learner should be able to get prerequisites for the concepts they wish to learn, and a quiz to test these prerequisites so that they can proceed to the new concept.

User Classes

The primary user of the concept is students interested in machine learning. They will create accounts and use the quiz generation tool to self-assess their progress and determine their readiness for the new concept they wish to learn. In the future it can be scaled to include teachers who can use to software for their classrooms to track their students progress and identify the gaps in their understanding.

Operating Environment

The web-based system will run on standard web browsers and are recommended to be used only on laptops and desktops over hand-held devices. The server side can be deployed on a Linux based cloud hosted virtual server.

Design and Implementation Constraints

The design needs to be scalable so that the increasing number of users and verification requests shall not degrade the performance of the product. It must also follow the tenets laid out in the DPDP act to protect learner data. Deployment on free hosting sites like render may lead to cold starts and increased latency for users. The backend database needs to be scalable to handle increased demand.

User Documentation

After the web app is deployed it will be handed over with the Software Requirement Specification, Design documentation, Testing Documentation, and User Manual.

Functional Requirements

1. Fetching prerequisites

Description: The system accepts a machine learning concept provided as input through a dropdown and displays the prerequisite concepts.

Input: User selects the required concept

Processing: A list of prerequisites is fetched from a hardcoded concept graph.

Output: Prerequisites are displayed to the user.

2. Generating Assessment

Description: The system uses the prerequisites to create a quiz for the user to test their knowledge.

Input: The concept is selected from the dropdown list and a button labelled generate assessment is selected.

Processing: The prerequisites are fetched and sent to the backend LLM to generate the assessment.

Output: An assessment comprising of a mix of numerical, one word and multiple-choice questions is generated.

3. Generating report

Description: The user's test results are analyzed to see if they can proceed to the new concept or if they need to review something, and if so, what needs to be reviewed.

Input: The user's answers to the generated quiz.

Processing: Use the user's answers and compare it to the answer key in the backend to check for any wrong answers and generate a comprehensive report.

Output: The students assessment results and an analysis of whether they should proceed with the new topic is generated.

4. Chatbot

Description: Basic queries of the user regarding navigation of the website and prerequisites for the concepts can be answered using the chatbot.

Input: The user's basic doubts.

Processing: The question is sent to an open source LLM using and API key which generates a response using our knowledge base.

Output: Response is displayed to the user.

5. Login

Description: Only registered users can access the quiz generation dashboard. They need to login using a unique mail ID and a secure password.

Input: The user's email ID and password.

Processing: The email and password are compared to the email and password stored in the user table in the backend.

Output: If they match the user is taken to the dashboard, or else they are shown an error message.

6. Register

Description: To create accounts for new users.

Input: The user's mail ID and password.

Processing: The mail ID is compared to the others in the database to ensure it is unique. Then the mail ID and the hashed password are stored in the database so they can be retrieved upon logging in.

Output: If the mail ID is already in the database an error is generated, otherwise a confirmation message is displayed to the user and they are redirected to the login page.

7. Logout

Description: To allow user to logout of their account.

Input: The user clicks the logout button.

Processing: The token generated while logging in is removed from local storage.

Output: The user is redirected to the login page.

8. Forgot Password and Change Password

Description: Sending a mail to the user with an OTP in case they forget their password to allow them to reset it.

8.1 Sending the OTP

Input: The user enters the mail ID after being prompted to.

Processing: The mail ID is sent to the database and compared with the existing records to see if it is present. If it is present a 6-digit OTP is generated.

Output: If the mail ID is not in the database an error message is generated. If it is there, a 6-digit OTP is generated and sent to the registered mail ID.

8.2 Entering the OTP

Input: The user enters the OTP they were sent.

Processing: The OTP entered is compared to the one generated.

Output: If the OTP matches the user is allowed to reset their password.

8.3 Change password

Input: The user enters the new password they wish to use.

Processing: The password is hashed and stored with the previously used mail.

Output: A confirmation message and redirection to the login page.

9. Change password

Description: The user can change their password if they wish to.

Input: The user's old password and the new one they wish to set it to.

Processing: The new password is hashed and stored in the database.

Output: Confirmation message is shown to the user.

External Interface Requirements

User Interface

Every page will have a theme toggle to switch between light mode and dark mode. Additionally, after logging in, all pages will have the chatbot symbol in the lower right corner and a navbar on the left side of the page.

Software Interfaces

The software uses a MERN stack and its associated dependencies. External third party LLMs are being used for the chatbot and quiz generation. Email integration will also be done using nodemailer.

Non-Functional Requirements

Performance Requirements

Latency Requirements

1. Prerequisites should be fetched in under 5 seconds.
2. Quiz should be generated in under 1 minute.
3. Assessment after quiz is generated should be under 30 seconds.

Reliability Requirements

1. If a concept is not found in our list of prerequisites the chatbot is still able to give a suitable answer when asked about the prerequisites.
2. All interactions with the third-party API are logged as a precaution.

Security Requirements

1. Ensure the cors configuration only allows the actual deployed front end and the locally deployed versions to access the backend.
2. The password is hashed before being stored in the database.
3. The API keys are securely stored as environment variables and cannot be accessed by unauthorized personnel.