Web Application for PDF Summaries and Quizzes

Introduction:

The project we will be building is a web application that allows students/teachers to login and upload custom PDFs. They can then understand the content in a digestible way by chatting with a chat-bot fed with information from the PDF. They will then be able to generate quizzes for appropriate content in the PDF to test their understanding. It will provide information based on the given data provided rather than sourcing through the web making it more accurate and helpful for students that need to absorb lots of information about their specific course.

Aims and Objectives:

Goal: The goal is to create a user-friendly web application that smooths the process of understanding PDF documents through interactive chat-bots and tailored quizzes.

Objectives:

1. Develop an interactive user interface that allows the user to easily upload and manage PDF files.

2. Implement an algorithm that parses the document and then feeds it into an LLM to generate responses to queries.

3. Allow users to chat with the bot in an intuitive way.

4. Create a quiz feature which generates questions based on the PDF content.

5. Provide feedback based on quiz performance, this can be a score out of 10.

6. Ensure that the conversations and quizzes are saved for future reference,and that they can be organized by the user to fit their needs.

Measuring Criteria:

- User satisfaction ratings.
- Accuracy of the generated summary.
- Number of completed quizzes.

Design decisions:

- The design of our application will focus on simplicity and user-friendliness.
- Student focused design that will allow users to organize their chats and quizzes as needed. E.g. folders for math, chem and etc.

Features of the web application:

1. **PDF Upload:** Users can upload PDF files.
2. **Chat bot:** Uses uploaded PDF embedding as content to generate answers to user queries.
3. **Interactive Quizzes:** Create a quiz with the provided information and be able to take a quiz. Quiz will be in MCQ format to aid memorisation.
4. **Feedback Feature:** Feedback on quiz performance and what they need to go over.
5. **Organization**. Users can organize their environment to suit their needs.

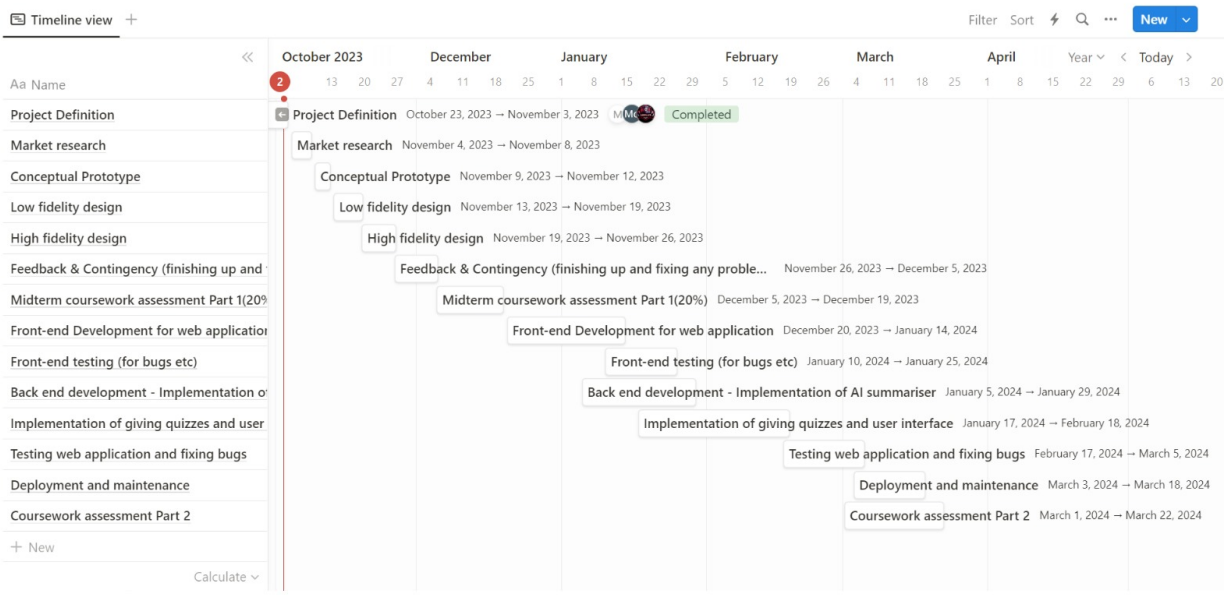
Context of use:

For students/teachers that need an efficient way to understand and memorize exhaustive documents.

Deliverable components of the software:

Project Timeline:

Timeline



Timeline was made using Notion and everyone in the team has access and the ability to see who needs to do which task and what the deadlines are etc.

How we plan to develop:

Tools that will be used to create the software are listed below:

- ❖ Front End: HTML, CSS, and JavaScript
- ❖ Back End: Python
- ❖ Cloud services: Amazon Web Services
- ❖ System Design: UML diagrams

Things we need to learn:

- Python framework.
- Get a better understanding of Amazon Cognito for account management.
- Langchain to create PDF embeddings
- Processes and algorithms used to analyze given PDF content and generate queries. E.g. questions, quiz generation.
- Efficiently develop responses from our NLP.

Teamwork:
There are 3 of us in the group, and we use WhatsApp to communicate. Meetings are done through either Discord/Zoom for when there are important deadlines or one of the members decides that we need one in order to keep everyone up to date on the current trend etc. We are all familiar with HTML/CSS and javascript so the front-end for the web application can be done by all of us easily. For the back end we are going to use Python which is a relatively easy language when compared to Java which we are currently learning so there shouldn't be any problems then. Each milestone will be updated to the Notion timeline so that we can keep track of completed assignments as well as things that need to be done before the deadline.

We will also meet in person every week or so to discuss any problems and update everyone on the progress although this would depend on the problem itself and if it can be solved through text rather than wasting everyone's time and forcing them to meet up at a spot.

Dependencies:

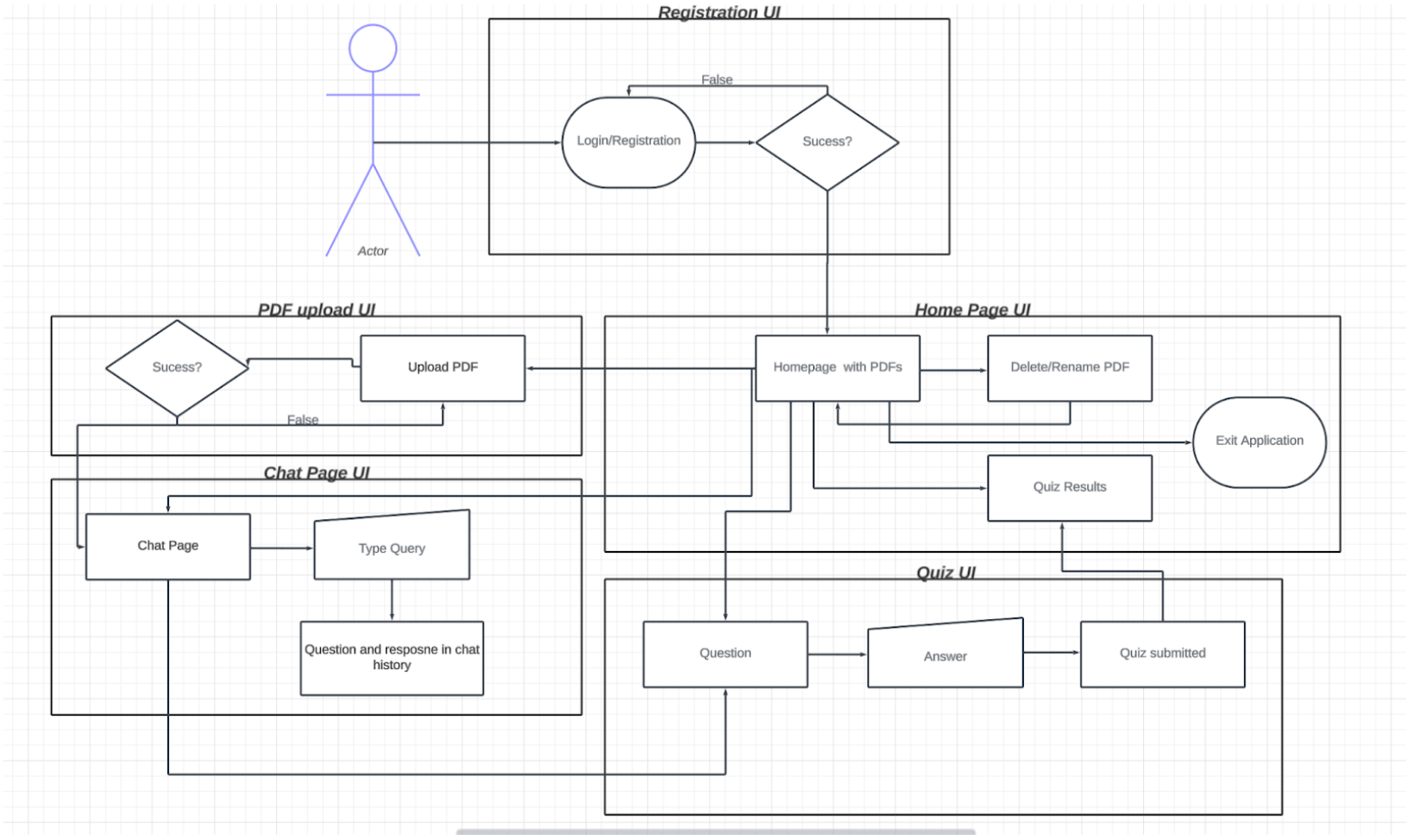
Amazon Cognito to help user account management. (possibly other AWS services too)
Langchain for parsing PDFs.
Huggingface Transformers API for NLP.

Planning and Requirements Gathering:

The requirements for the software will be gathered through student interviews and surveys which will allow us to understand the user needs and preferences.
We will also take a look at similar products/applications and find their shortcomings.
Since our target audience are students, we are researching popular tools used by students to ensure we have features that suit their needs. Our design choices will generally focus on enhancing the viewing experience to avoid user eye strain.

Software Specification:

To describe our software we used a high level UML diagram, this will provide a clear understanding of our system's functionality, and the basics of its design. These were created using Lucid.



This flowchart starts with user logging in or registering. Once they sign in, they are redirected to our homepage. From here, they can choose to upload a PDF, start a chat or a quiz.

If they select "Upload PDF," they are directed to the upload screen. Once the PDF upload is successful, they are redirected to the newly created parsed PDF's chat.

If the user selects a specific PDF, they can choose to be redirected to the chat or quiz screen respective to the selected PDF.

From the homepage screen , the user can also manage their uploaded PDFs by editing or deleting them. This will take place on the homepage screen and will not involve redirecting.

Flowchart ends when the user decides to exit the app.

Below are UML use case diagrams for each individual screen: