Test Genie – Final Project Proposal

By Sean(seanyl2), Sanhorn (sanhorn2), Kelly (kellyy3) and Jasmine (jchan59)

Pitch

Whether professors fail to provide them or students find them ineffective, many students struggle to find the proper resources to study for exams. Test Genie analyzes the relevant textbooks, assignments, and past exams provided to create a practice exam for the student to utilize to study.

Functionality

1. Users can upload files of past exams, assignments, and lecture notes
2. Users can get practice exams, lecture summaries, and other study materials
3. Users can create accounts to access previously generated materials and other personalized content

Components

- Frontend: The frontend is responsible for showing the user interface, and letting users upload files and create resources. To implement the frontend, we will use the React framework and the JavaScript programming language. We chose React because of our previous experience with it, and therefore find it easier to use than many other frameworks. Similarly, a few of us have used JavaScript in the past and find that the learning curve will be lower for those who have not used it. JavaScript also does not require additional configuration or tooling as TypeScript does, making it easier to use. We will test using the React Testing Library, which allows us to interact with Test Genie from the perspective of a user.
- Backend: As for our backend, we will use Python's Django framework. Python is the most popular language for machine learning and data science, as it provides sufficient libraries and documentation specifically for backend and api development, which is an important part of our project. Moreover, to integrate with the LLM model, which is based on Pytorch, it would be a lot more convenient for us to use Python along with it. The

backend would be receiving api requests from the frontend, sending the data to the model to be trained, and later returning the model's response back to the user.

- LLM: The Llama LLM provided by Meta is a widely known open-source model, it has been trained on trillions of tokens and showed exceptional performance. We plan to integrate our project with this model as it is built under Pytorch in Python, which means it would be more compatible with our backend. Specifically, we plan to use the Llama HugginFace Transformer because it is easier to import and fine-tune, so that it can be utilized for our specific task.
- Database: SQLite

Our application is separated into a frontend and backend, as it allows us to focus on each component separately. The frontend focuses on user interactions, while the backend processes data. This allows the backend to handle tasks without impacting the user interface. Using React for the frontend and Django for the backend allows us to have efficient data handling while providing an efficient user interface. Additionally, we can work on both the frontend and backend simultaneously, making it time-efficient.

Weekly Planning

Week # (Date)	**Tasks**
Week 1 (9/20-9/27)	1.Initialize the project and install all necessary dependencies 2.Setup the Github repository for project structure
Week 2 (9/27-10/4)	1. Test on different LLM Models / Find our benchmark model 2. Create the design for the frontend (colors, fonts, etc.) 3. Collect our dataset -> open source exams / exams of courses that we've previously taken
Week 3 (10/4-10/11)	1. Develop the homepage UI for the website 2. Create API endpoints and manage file storage with Django

Week 4 (10/11-10/18)	1. Focus on file upload and connect with backend 2. Setup SQLite and connect with backend using Django
Week 5 (10/18-10/25)	1. Implement state management to handle uploaded files 2. Ensure API endpoints are secure and allow frontend and backend to communicate well
Week 6 (10/25-11/1)	1. Start designing the user dashboard to display uploaded files and generated content 2. Set up the LLM Model / Connect with our backend to load the LLM model 3. Develop API endpoints to process files using the LLM Model
Week 7 (11/1-11/8)	1. Fine-tuning the LLM model with our collected dataset (past exams, lecture notes...etc) 2. Look for newer publications to incorporate models that are better at text recognition with our model 3. Finish developing frontend user dashboard UI
Week 8 (11/8-11/15)	1. Conduct Evaluations and compare with other SOTA LLM Models / Continue with Fine-Tuning 2. Develop the UI for requesting and displaying generated practice exams.
Week 9 (11/15-11/22)	1. Develop UI for accessing previously uploaded or generated content 2. Develop APIs to fetch and manage user-specific data for the dashboard.
Week 10 (11/22-11/29)	1. Conduct comprehensive testing to ensure flawless user experience 2. Cleanup the code to maintain readability

Potential Risks

1. **Frontend and Backend Integration Risk**: Since our team isn't familiar with calling backend methods from the frontend. This could lead to delays (debugging may take several days), requiring guidance from a mentor.

2. **LLM Model Integration Risk**: Poorly fine-tuned language models could generate low-quality content or incorrect predictions, which might reduce the overall effectiveness of the application. Text Recognition capabilities in current models may also be important, as our model requires high recognition accuracies.
3. **Uploaded Content Risk**: There is a risk that user-uploaded materials (like exams, notes, and assignments) may not be properly read or stored by the website, leading to usability issues.
4. **LLM Model Ethical Concerns**: Since the LLM is trained on a large dataset, there is a risk that it could inadvertently generate biased or inappropriate responses, depending on the nature of the queries or input it receives. This could negatively impact user trust in the application and lead to reputational damage.

Teamwork

We will use GitHub for its collaboration and version control tools, allowing us to efficiently work together on this project. We will have continuous communication with one another before pushing new versions into the repository.

We will divide into 2 sub-teams of 2 people each. Sean and Sanhorn will work on the backend, and Kelly and Jasmine will work on the frontend. This is because Sean and Sanhorn have experience with AI and Django, which we are using in the backend. Kelly and Jasmine have experience with UX/UI and have worked with JavaScript. We will communicate through text messages and Slack to determine what needs to be done for the week, and each team will collaborate to get their part done.

Continuous Integration

To maintain code quality and ensure a smooth development process, we will implement a Continuous Integration (CI) strategy. Our CI approach will include the following elements:

1. **Test Library**: For the backend, we will be using the PyTest library to run our unit tests. For the frontend, we will use the React Testing Library to verify that our user interface behaves correctly.
2. **Style Guide**: Our team prefers the snack_case style more. We will use the automated tools: Flake8 for Python & ESLint for JavaScript. These two tools will check for our overall code qualities and styles, ensuring a consistent structure of our codebase.
3. **Pull Request Workflow**: Each sub-team will work on its own feature branches and create pull requests when it finishes its part and is ready to merge. Every PR will be reviewed by at least one person from another sub-team to make sure the code does not contain any problems. To avoid merge conflict, we will meet up every week to discuss the current part we are working on and assign reviewers for each week's work.

By incorporating these integration plans, we hope to maintain high code quality and ensure that the project remains stable throughout the development process. This strategy will make our code easier to read, easier to identify issues, and easier to collaborate within our team.