Sprint 1 Planning

Team 10: Essay Grading System

Atul Aneja, Jalaleldeen Aref, Shivank Tibrewal, Sidhant Chitkara, Tarang Khanna

Sprint Overview: During this sprint we hope to understand and establish a basic idea of how our essay grading web application will work. The front end team will aim to create a simple user experience, with a responsive web interface that can take user input to the server for the back end. The database team will coordinate so that user input can be stored in the database and moved back and forth. Lastly, the grading component will definitely be the largest portion. We aim to have basic grading established, such as grammar and language checking, and see some tangible progress from the deep learning grading component also.

SCRUM Master: Tarang Khanna

Risks and Challenges: The biggest challenge our project faces is that it is quite dependant on one, extremely large component, which is the essay grading algorithm. This makes measuring progress, as well as splitting the task across sprints pretty difficult. Another challenge is implementing our front end in Django, which will make connecting to the back end easier in Python but has a learning curve as some team members have not worked with Django before. Lastly, one big risk of our project is comparing our essay grader's performance to human graders as manual grading has a very vast array of possible outcomes that we must account for.

Current Sprint Detail (5.0 points)

User story 4:
As a user, I can receive a letter grade for my essay:
The tasks are:

1) Visualize the data set/explore the data.
2) Choose ML model based on data set attributes
3) Augment data if needed
4) Train multiple models on data
5) Choose the best model based

After this I expect the model to take essays as input and output a predicted score, although the score might be low, it will be improved in further sprints.
Note, this user story needs a lot of tasks.

As a user, I can receive a letter grade for my essay:
Description of the tasks are:

Task Number	Task Description	Team Member	Workload
1)	Visualize the data set/explore the data. By utilizing histogram, qq plot, scatter plot on the data, heat map.	Tarang Khanna	10
2)	Choose ML model based on data set attributes such as mean, variance, linearity.	Tarang Khanna	10
3)	Augment data if needed, for example removing outliers, removing null values, etc	Tarang Khanna	5
4)	Train multiple models on data to do prediction of grades. Try training K-nearest neighbors, SVM, logistic regression, and	Tarang Khanna	10

	other algorithms using scikit learn. Also train a neural network using Keras with Tensorflow backend. Training can be done by giving X input and Y output, where X is the essay broken down into features, and Y is the predicted grade.		
5)	Test models and choose the best one based on cross validation accuracy. Testing will be done on the data that was not used for training. By picking a model we trained in step 4), we will give it essays and check what it predicts, and we will compare its prediction to the grade given by a human (which is included in the Kaggle dataset we are using).	Tarang Khanna	4
6)	Get prediction from the algorithm to the frontend by making the API endpoint for it	Atul Aneja	5

Acceptance criteria for user story 4:

Given I am testing the machine learning model separately, when I give the algorithm an essay, it should output a letter grade. This will be used for training and testing. (Developers perspective)

Given I am on the upload page and the user story for transferring the essay to the backend is working, when I upload my essay for grading, then I expect to receive a letter grade for my essay, shown to me on the frontend. (Users perspective)

Given I am testing the machine learning model separately, when I run cross validation tests on the model I expect to get an accuracy of greater than 60%. (Developers perspective)

User story 1:
As a user, I can sign up for the web application:

The tasks are:
1) Set up user database using FireBase
2) Set up initial framework in javascript and Python
3) Implement front-end based on UI mockups using Django
4) Implement application server to communicate user data with back end
5) Connect front-end to backend for data transfer using API

Description of the tasks are:

Task Number	Description	Team Member	Workload
1	Implement database using Firebase. This would require populating our table with the required fields.	Jalaleldeen Aref and Sidhant Chitkara	15
2	Set up our Django development framework in the repository to create an HTML and JS front end that can be tied to a Python controller	Jalaleldeen Aref and Sidhant Chitkara	10
3	The front end sign up screen must be appropriately styled and designed, and responsive to enable user input	Jalaleldeen Aref and Sidhant Chitkara	5
4	The application server needs to communicate the new user objects to the database	Jalaleldeen Aref and Sidhant Chitkara	5
5	Connect front-end to backend for data transfer using API (setup an api)	Atul Aneja	15

Acceptance Criteria for user story 1:
Given I am on the sign-up page, I should be able to sign-up with valid user details.

Given that a user enters sign up data, the validated data should be verified for uniqueness and inserted as a new user (developers perspective)

User story 2:
As a user, I can log in to my profile in the web application.
This user story will have a lot of tasks in common with user story 1.

The additional tasks are:
1) Implement front-end based on UI mockups using Django
2) Use application server to communicate user input to back end
3) Verify user login information
4) Develop an api endpoint to support login.

Description of the tasks are:

Task Number	Description	Team Member	Workload
1	The front end of the login screen must be appropriately styled and designed, and responsive to enable user input	Jalaleldeen Aref and Sidhant Chitkara	5
2	The application server needs to communicate the new user objects to the database	Jalaleldeen Aref and Sidhant Chitkara	5
3	Verify user login information by comparing login info with tokens stored in the database.	Jalaleldeen Aref and Sidhant Chitkara	10
4	Develop an api endpoint to support login.	Atul Aneja	5

Acceptance Criteria for user story 2:

Given that I am on the login screen, I should be prompted to re-enter my information if it could not be verified

Given that a user enters login information, I should receive and efficiently cross reference the user object on the database's end (developers perspective)

Given that a user enters any null information, I should account for missing data to count as invalid (developer's perspective)

User story 3:
As a user, I can upload an essay question and answer in the form of a text file

The tasks for this user story are:
1) Implement front-end essay submission screen based on UI mockups
2) Accept text file into web application, verify format
3) Develop an api endpoint to transfer essay files

Task Number	Description	Team Member	Workload
1	Implement front-end essay submission screen based on UI mockups	Jalaleldeen Aref and Sidhant Chitkara	10
2	Accept text file into web application, verify format	Jalaleldeen Aref	7
3	Develop an api endpoint to transfer essay files to and from the web application and between the database and grading components	Atul Aneja and Sidhant Chitkara	10

Acceptance Criteria for user story 3:

Given that I am using the grading service, the web application should take in my essay in the form of any rtf (Rich Text Format) file. (User's perspective)

Given that user enters a text file for grading, the web application should ensure that the text entered is readable and not corrupt text. (Developer's perspective)

Given that user enters a text file for grading, the web application must ensure that it is of an appropriate size before file transfer to the back-end and database (developers perspective)

User story 8:
As a user, I can receive adequate feedback on the validity of the data in my essay.

The additional tasks are:
1) Find relevant docs online, and check its credibility.
2) Extract information from the essay (like keywords).
3) Compare doc online with keywords
4) Create an api endpoint to help communicate the needed information (for the fact checking).

Description of the tasks are:

Task Number	Description	Team Member	Workload
1	Find relevant docs online using google API, and check its credibility with a database of credibility URLS.	Shivank	10
2	Extract facts from the essay to run fact checking on (setup NLP).	Shivank and Atul	19 (8-Atul, Shivank -11)
3	Compare doc online's facts with keywords using tfidf and other techniques to check similarity between doc and essay facts.	Shivank	18
4	Create an api endpoint to help communicate the needed information (for the fact checking).	Atul Aneja	3

Acceptance Criteria for user story 8:

Given that I upload an essay to be graded, the fact checking component should isolate data that needs verification, and reference only the required data (user's perspective)

Given I am testing the fact checking component separately, when I give the algorithm an essay, it should extract the facts from the essay and print it out. (Developers perspective)

Given I am testing the fact checking component separately, when I give the algorithm an essay, it look up relevant docs to the extracted facts. We can measure relevance by manual inspection. (Developers perspective)

Given I am testing the fact checking component separately, when I give the algorithm an essay, it should extract the facts from the essay and after checking it online, it should tell me the reliability of the facts. To test this we need to give some fake facts and some real facts and test them with the algorithm. By seeing if they are clearly classified we can see how good the component is. (Developers perspective)

Remaining user stories:

Backlog ID	**Functional Requirements**
5	As a user, I can receive a breakdown with points of how this grade was derived
6	As a user, I can receive adequate feedback on the quality of language in my essay
7	As a user, I can receive adequate feedback on the theme and relevance of my essay
9	As a user, I can save my grades and feedback to be viewed later
10	As a user, I can upload a folder of files at once for batch grading (if time permits)
11	As a user, I can get confidence scores of how certain the algorithm is of the grade it has given to my essay(s) (if time permits)

Hours per sprint per Developer:

Developer	**Hours per sprint**
Atul Aneja	41
Jalaleldeen Aref	40
Shivank Tibrewal	39
Sidhant Chitkara	38
Tarang Khanna	39