Anonymous Publication Submission and Review Platform Using ML

CSCI 409: Senior Project II

Final Project Report

CSCI 409 Group: 35

Team Members: Shyngys Satkan, Dastan Bekmukhanbetov, Nurmukhammed Aitymbetov, Meiirlan Suiirkhanov

Project Advisor: Assistant Professor Dimitrios Zormpas

1. Executive Summary

The goal of this project is to automate the process of matching authors and reviewers preserving anonymity by developing a Publication Submission and Review website. The current matching process is not efficient due to multiple reasons: The occurrence of bias due to reviewers' personal preferences or knowledge of authors' identities; Traditional review process is time-consuming and could be the major reason for delays in publication of the papers; Mismatch between authors and reviewers in many cases where reviewers may not have relevant expertise in the field. The proposed platform utilizes machine learning techniques to anonymize submissions, classify papers into relevant fields, offer newly submitted papers into relevant reviewers, and allow them to provide decisions along with valuable feedback. The website also serves as a monitoring tool for authors to view their submitted papers and similarly for reviewers to view the assigned papers.

2. Introduction

Our motivation derives from the desire to improve fairness in the academic publication process by ensuring that contributions are assessed only on the quality of material, rather than the authors' names or other relevant criteria. We hope to increase the openness of publication reviews, therefore encouraging a more inclusive academic community.

We created an anonymous publication submission and review platform that uses an effective machine learning model to protect the authors' privacy and the integrity of the review process. The platform is designed with Django and the Django REST Framework (DRF) for backend operations, React for the frontend interface, and PyTorch for training machine learning model. Our technology automates the process of connecting papers with the most appropriate reviewers based on their expertise, reducing human interference and any bias. Furthermore, it makes it easier to provide feedback to assigned publications using an intuitive interface. We believe that our website would significantly contribute to the academic publication review process.

3. Background and Related Work

With the increasing number of scholarly articles, there is a new trend in using machine learning algorithms for the paper review process. Many of these methods rely on semi-automated

procedures that use abstract screening. This method is based on an algorithm which takes an abstract of the paper and converts it into a numerical value that incorporates contextual links and text semantics.

There is such a system presented by Leyton-Brown et al. [1] which has been successfully deployed at recent AI conferences. The program processes input data, including reviewer names and bids, to calculate reviewer-paper ratings. Next, the appropriate reviewer-paper matchings are identified by optimization algorithm. One thing to note is that the procedure was fully algorithmic, i.e. without use of machine learning techniques.

Furthermore, Anjum et al. [2] provide an optimization problem for the same topic. The authors suggest a paper-reviewer matching strategy that uses shared subject area to enhance the process. The approach, which uses abstract topic vectors to describe submissions and reviewers, beats previous techniques, according to the authors' results. The suggested technique is effective, as evidenced by experimental assessments on benchmark and freshly obtained datasets. Additionally, 88% of reviewers reported familiarity with the assigned topic.

Recent approaches use NLP and deep learning techniques to evaluate abstracts and choose the most relevant papers for review [3]. One system [4] sends queries to academic databases and selects publications from authors with comparable study topics while also gathering information. A strongly connected subgraph method groups researchers according to their networks, further optimizing the process. As a result, it addresses two issues at once: finding relevant reviewers and reducing potential biases in the review process.

Many prior research suggest strategies for automatically identifying appropriate paper-reviewer pairings in academia. Nonetheless, unlike previous achievements, our current study aims to achieve a more comprehensive and ambitious goal. We create and deploy an all-in-one system that addresses all aspects of the peer-review process, including enlisting reviewers and identifying conflicts of interest. The suggested system is completely anonymous, with no requirement for a list of potential reviewers to be entered into the system, and it examines several elements of article reviewing without requiring human participation.

4. Project Approach

In this section, we will provide a full overview of our solution. First, we will demonstrate how responsibilities and duties were assigned among team members. Then we'll discuss the details of the machine learning part, which includes an overview of the system, implementation of classifier and matcher models, and evaluation dataset. After that, we will go over the workflow of features and solutions, including database model diagram, use case diagram, and features of authors and reviewers. In addition, architecture diagrams will be presented and discussed. Finally, we will discuss the software development approaches employed throughout the project.

4.1 Task responsibilities

1) Nurmukhammed: was responsible for the machine learning part of the project. He designed the architecture of the ML models, trained them and deployed them using the FastAPI microservice.
2) Shyngys: was responsible for the frontend of authorization page, authors and reviewers dashboard, paper submission page, feedback submission form, and preview of PDF articles in browser.
3) Meiirlan: was responsible for the backend, particularly for paper submission API, author and reviewer dashboards, parsing articles from arXiv website, and comprehensive testing of the API methods.
4) Dastan: was also responsible for the backend for the user authorization API using access token, feedback form for reviewers, database model construction, design of the whole flow of the request, and interaction between backend and machine learning services.

4.2 Machine Learning

4.2.1 Overview of the system

The core of our platform is a two-part ML algorithm that analyzes submitted paper abstracts to find the best matching reviewers who have the required expertise in the topic. It is important to note that all the components within this system focus only on the papers' content and do not take into account any identifying information related to the author or potential reviewer. This ensures a robust, fair, and efficient reviewer selection process, minimizing biases and maximizing the relevancy of matches. Figure 1 depicts the overall architecture of the ML system.

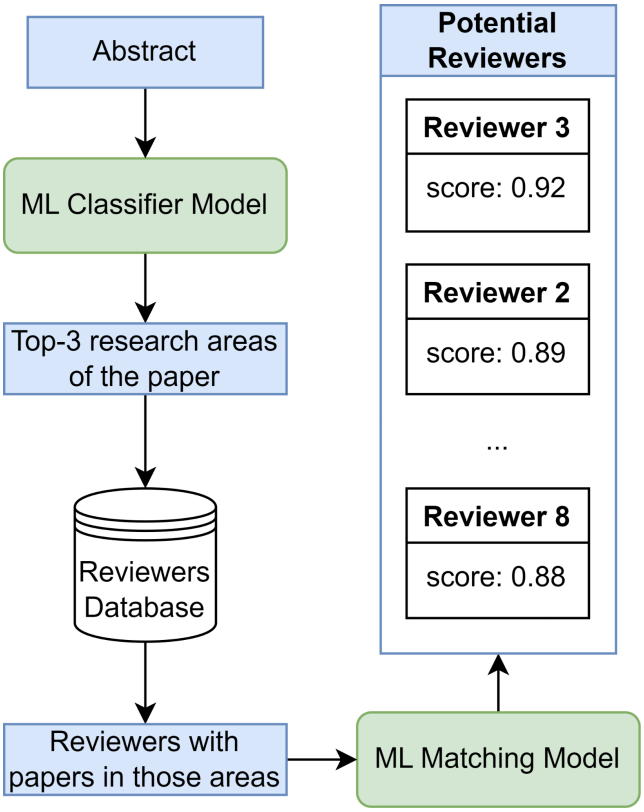

Figure 1: Overall architecture of the ML system

First component of the algorithm is a classification system that classifies the abstract of the paper into one of the 150+ predefined topics across different subjects and their subfields. After the top-3 topics of the incoming paper are identified, we search the database for the potential reviewers who have written a certain amount of papers on at least one of these topics.

Classification and initial filtering steps are followed by the Matching Model. It computes the latent representations of the incoming paper and reviewers' research profiles, and assigns ranking scores based on their matching degree.

These two components involve close communication with each other and the backend server. Figure 2 depicts the way submission of the paper is handled, which shows the exact way in which the different components of the web application interact with each other.

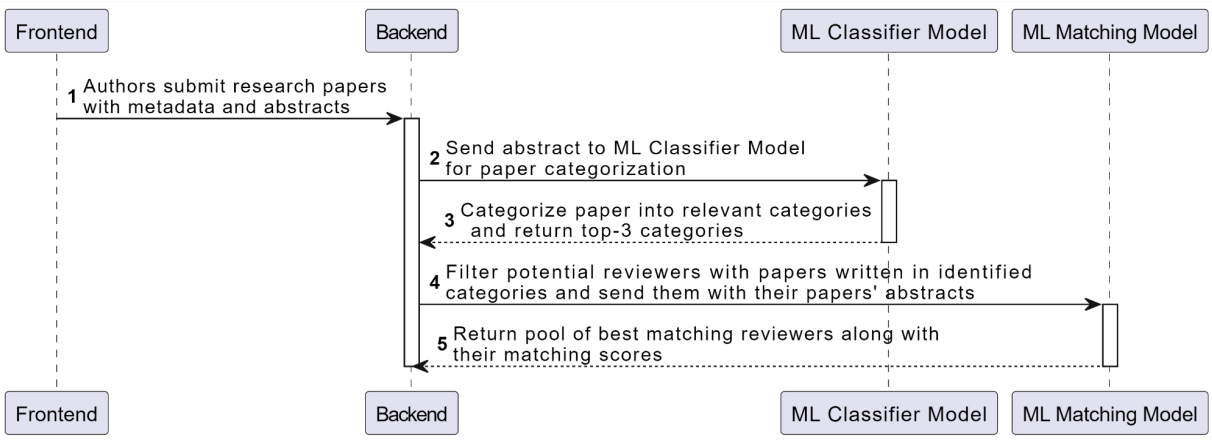

Figure 2: Sequence diagram of the ML interacting with other components of the system

4.2.2 Implementation of the Classifier Model

The Classifier Model is used to identify the areas of research covered by the incoming paper. Specifically, the model outputs the top-3 research fields associated with the paper's abstract. The reason for taking three categories is that the papers usually combine several areas and the number of such areas is three on average for the papers found within the arXiv dataset [6]. These categories are used to obtain the initial set of potential reviewers by selecting only those who have written a certain amount of papers on the respective categories. The threshold number of required past papers for selecting the reviewer is a tunable parameter, which can be adapted to the specific needs of the system's user (e.g. conferences).

For the Classifier Model we used Gated Recurrent Unit (GRU) [7] to process the abstract of the incoming paper. Specifically, you can observe the architecture of the Classifier Model in Figure 3.

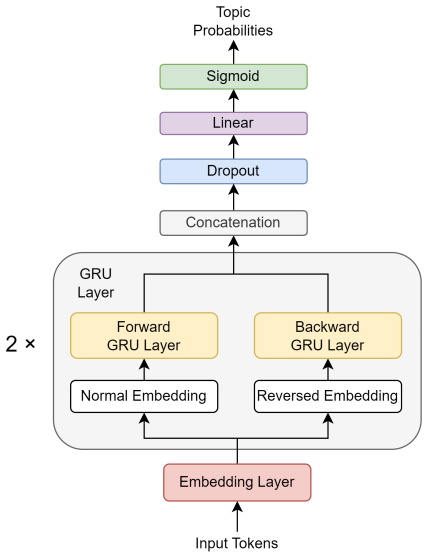

Figure 3: The architecture of the Classifier Model

Embedding layer transforms the input tokens into vectors that represent the inputs in the feature-rich latent space in order to capture the semantic meanings of the words. Extracted embeddings, then, go through two layers of bidirectional GRUs, meaning that these embeddings will be processed in both ways: from left to right and vice versa. After the bidirectional results are concatenated into one vector, Dropout layer is applied to regularize the model, as it can easily overfit the training data and converge poorly on testing data. Then, we apply a fully-connected linear layer with sigmoid output to extract the probabilities of individual topic categories. The three highest probabilities are selected to represent the research area associated with the paper.

4.2.3 Implementation of the Matcher Model

After getting the initial set of potential reviewers, we employ a second component of the system: Matching Model. It assigns scores to each potential reviewer, which represents the matching degree between reviewer's expertise and the research areas that the paper covers. Then, using these scores, reviewers are ranked, with higher scores meaning better match.

In particular, we employ a scoring algorithm based on the TF-IDF representation [8]. TF-IDF vectorization essentially analyzes the text and assigns different weights to different words based on their significance in the broader context: research profiles. So, our matching algorithm

computes TF-IDF representations of the incoming paper and reviewers' research profiles, and assigns matching scores based on cosine similarity of two representations.

4.2.4 Training Dataset

It has to be noted that the Matching Model does not need explicit training, as it is a statistical method that learns the TF-IDF representation of the data during the actual usage. Thus, the only component that required training was the Classifier Model.

For training the Classifier Model we used publicly available arXiv dataset. The original dataset consists of 2.5 million data entries. However, the distribution of the topical categories were imbalanced, meaning that the categories like 'Artificial Intelligence' and 'Computer Science' dominated the dataset. It might affect the performance of the model in the negative way as the model will be more biased towards selecting the dominant category. To overcome this problem, we preprocessed the dataset by balancing the number of papers corresponding to the given category. Specifically, we set the hard threshold of 15,000 entries per category. This reduced the size of the dataset from 2.5M to 1.4M data entries.

Furthermore, we split the dataset into a training set and testing set with a testing set's ratio of 0.15. Then, we applied stratification across the main topical category of the papers to ensure the same distribution of categories between the training set and the testing set.

To limit the memory usage of the model, we set the hard threshold of 250,000 unique tokens for the generated vocabulary size too.

The Classifier Model was trained on P100 GPUs for 20 epochs (~5 hours). Beyond 20 epochs, performance gains were negligible.

4.3 Workflow of features and solutions

1. Database relations and use case diagram: According to Figure 5, the system has 2 types of users: authors and reviewers. The users can use the "Login" action to gain access to their submitted papers and assigned papers for review.

Authors can submit a paper, view the dashboard listing all submissions, monitor status of the review, and view the feedback given by the reviewers. When the author fixes the issues specified

in feedback, they can resubmit the paper for the next iteration of the review process. Reviewers can see the list of papers recommended for review, view the dashboard listing all papers under review, provide feedback using the form with comments, and monitor the updates on the paper from the author.

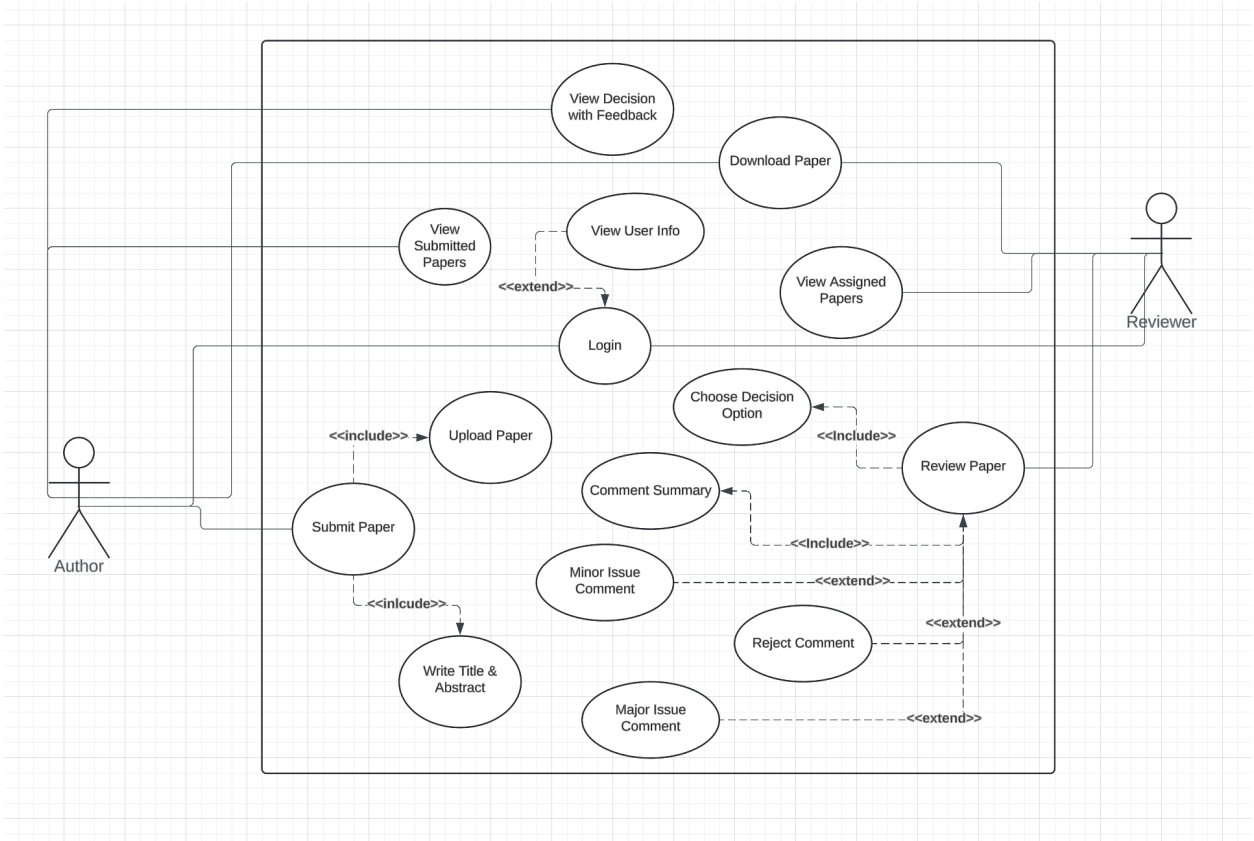

Figure 5. Use case diagram

To carry out the tasks outlined above, we constructed a database to store all of the essential information. The database tables and relations are illustrated on Figure 6. In the Users table, there are boolean fields is_author and is_reviewer which defines whether the current user is author or reviewer. The main table of the database is Publications, which has many to many relations with Users (author and reviewer), and Feedback tables. The file of the publication is stored in a field as a pointer to the actual path of the file. The Feedback table also has many to many relations with Users and has field review_status which indicates whether the publication was accepted or rejected.

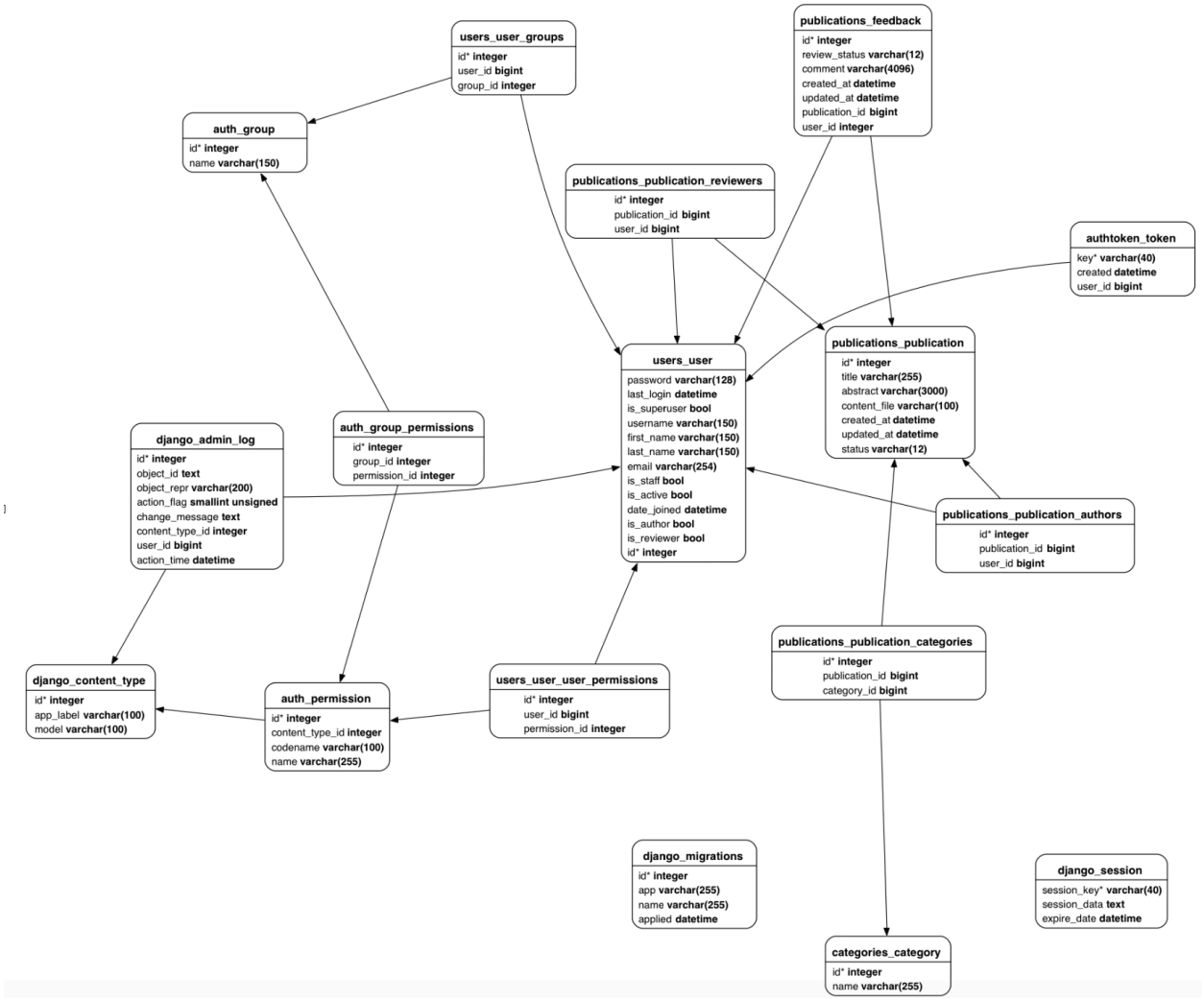

Figure 6. Database schema

2. Design of features:

All users regardless of authorization status can view the login page shown in Figure 7.

a) Author/Publisher features:

Authors can login using their credentials and navigate to the home page shown in Figure 8. You can see a navigation bar with pages such as Home, Review, Submission and Profile. Logged in users can navigate across these pages details of which will be discussed further in this section. Author can click on “submit paper” button and a modal view will appear prompting the Title, Abstract and PDF file of the publication to be reviewed shown in Figure 9.

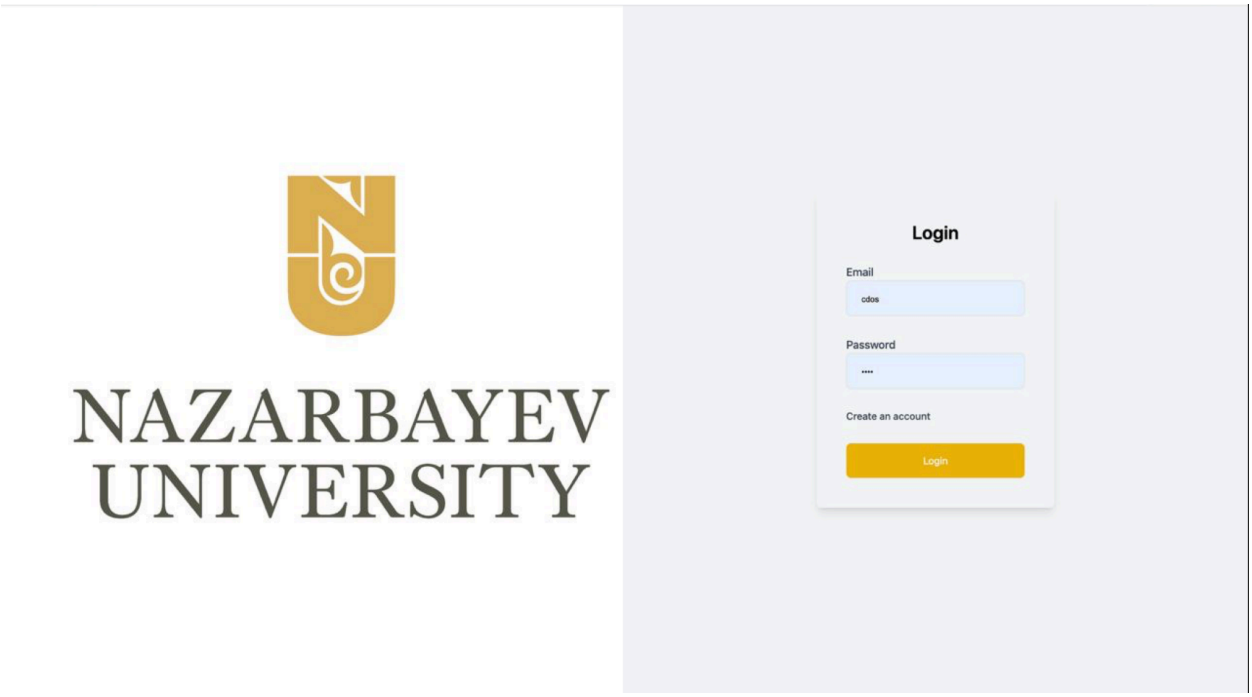

Figure 7. Login Page

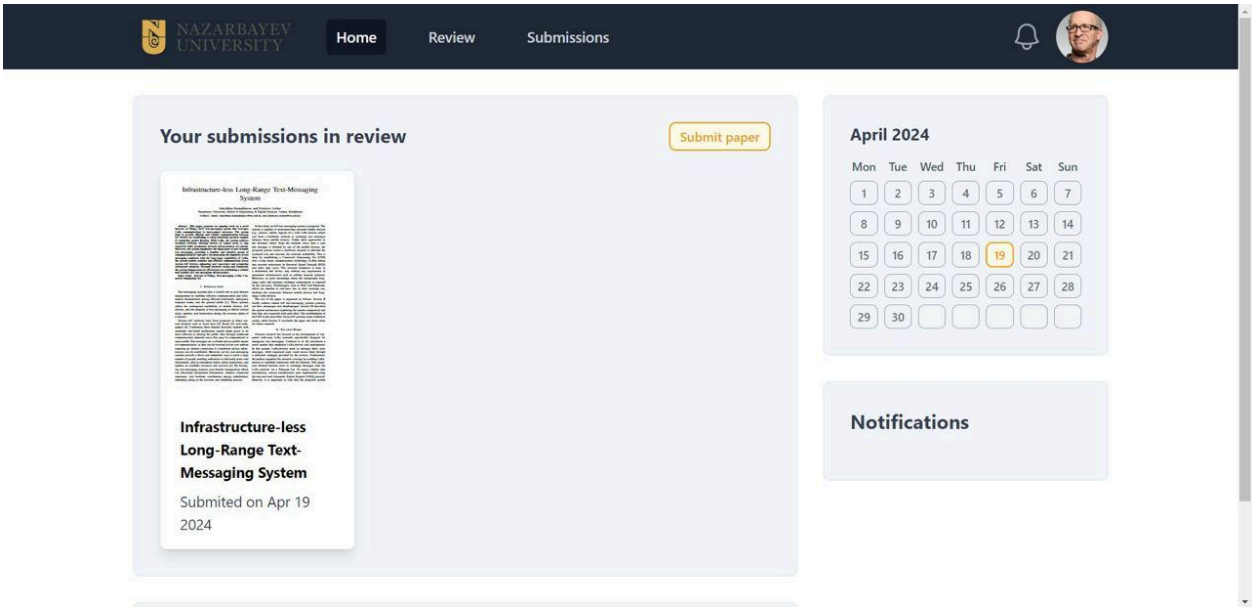

Figure 8. Home Page

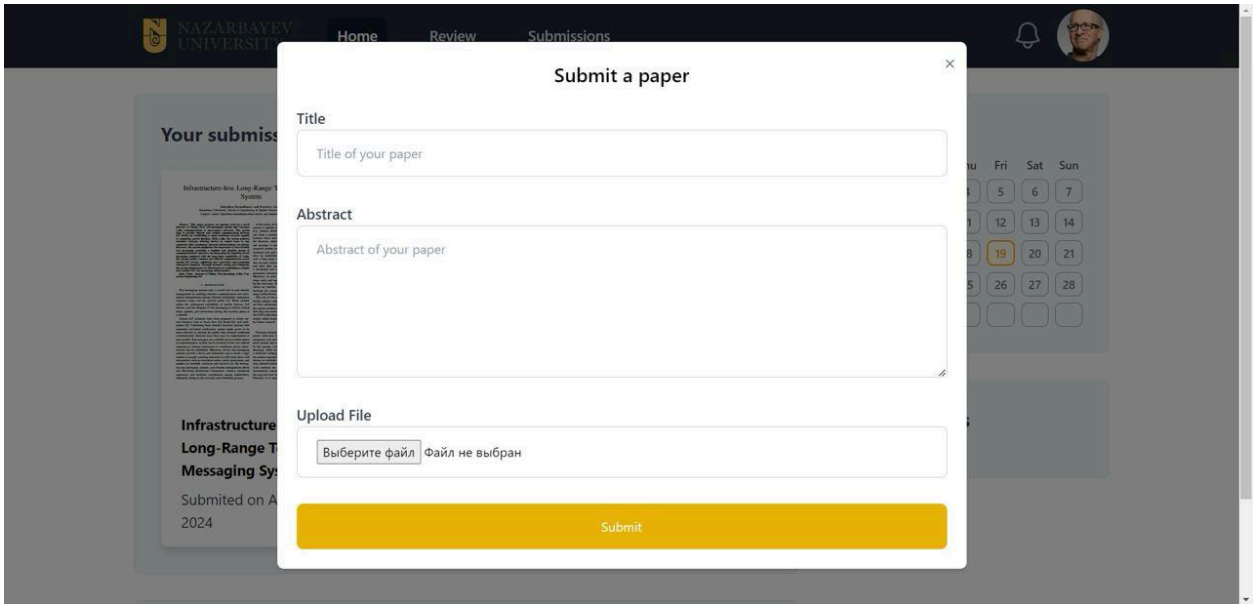

Figure 9. Paper Submission Modal View

a) Reviewer features:

After logging in as a reviewer you can navigate to the "Review" page shown in Figure 10 and see assigned papers for review. (matched using a machine learning model considering a particular reviewer's expertise).

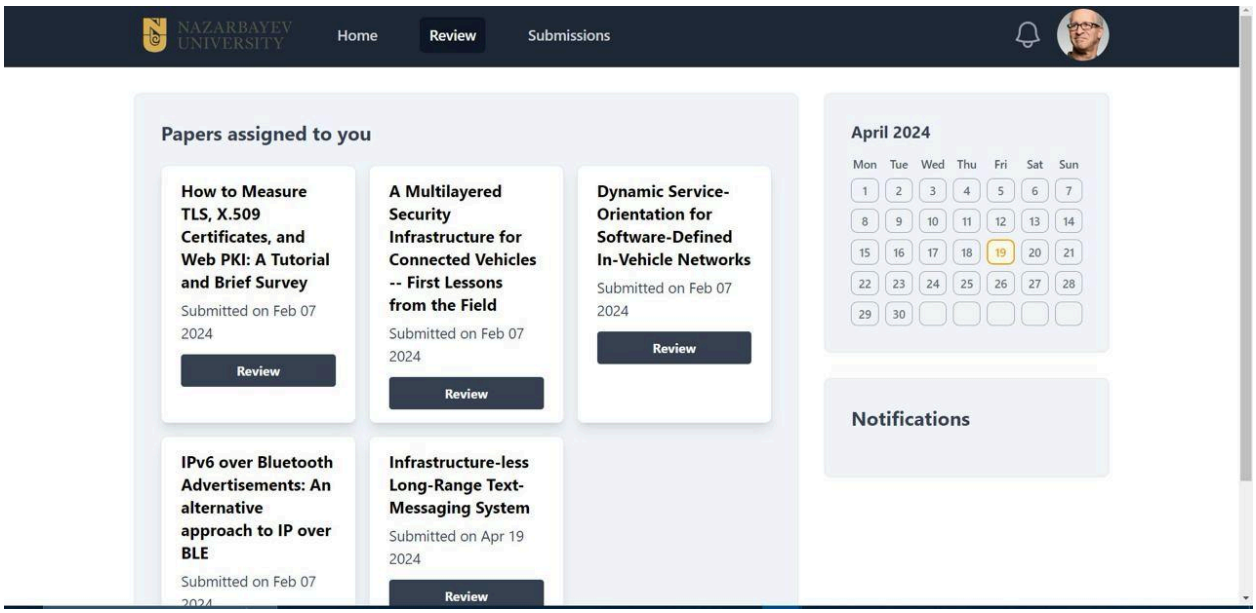

Figure 10. Review Page

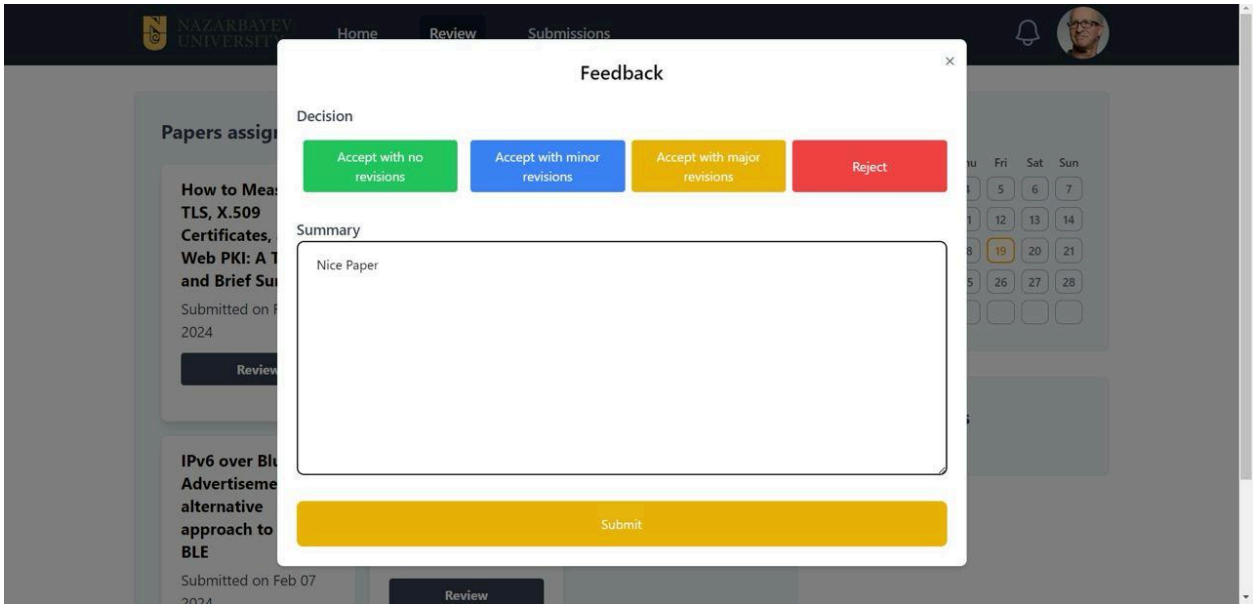

Figure 11. Feedback Modal View

After reading and getting acquainted with the content of the paper, the reviewer can click on the "Review" button and the feedback modal view will appear as shown in Figure 11. Here, the reviewer can choose their decisions and write corresponding comments based on the decision.

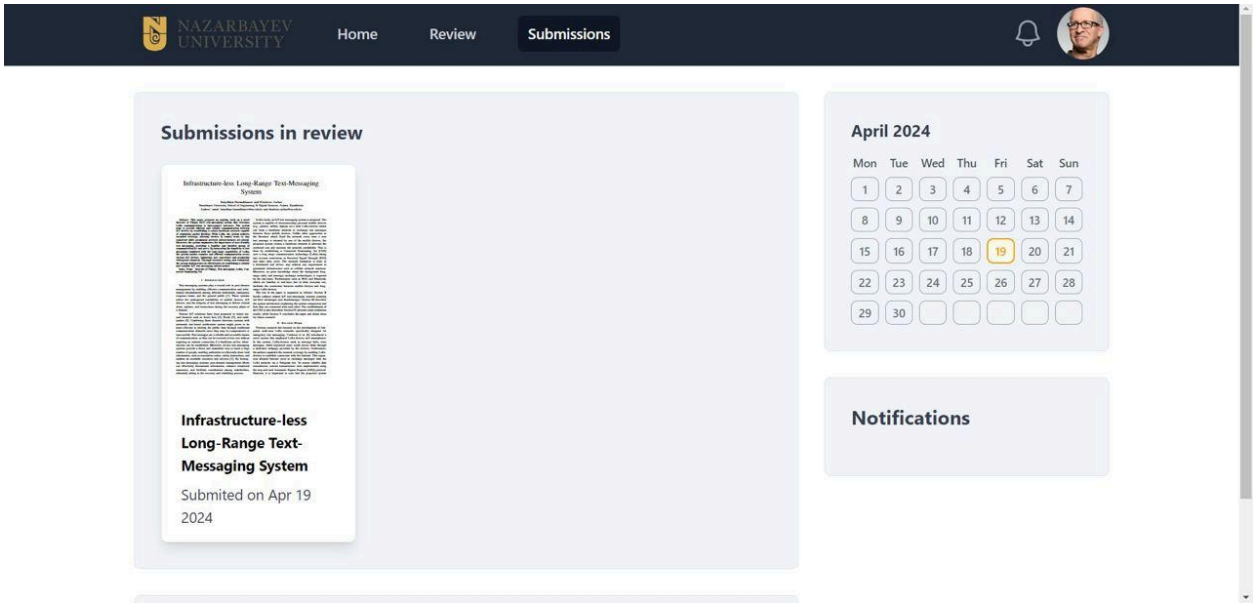

Figure 12. Submissions Page

After receiving review from all assigned reviewers, publishers will receive Notification that their paper has been reviewed. They can navigate to “Submissions” page (figure 12) to check the results of review process as shown in Figure 13.

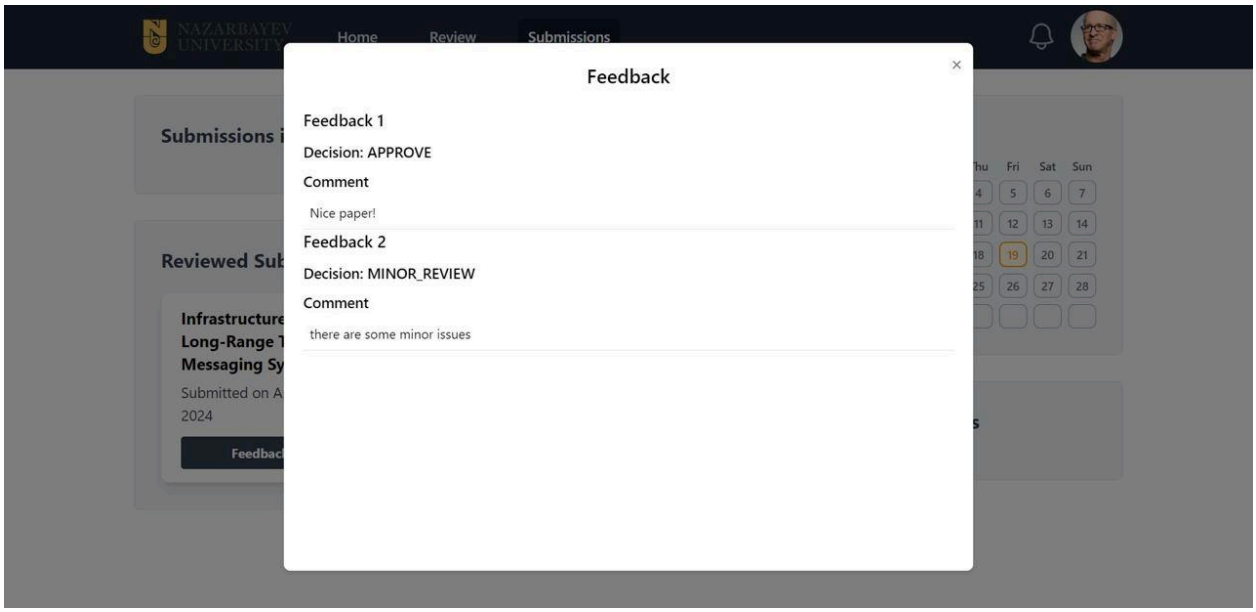

Figure 13. Feedback Modal View

4.4 Third-party tools

During the development of the web application, we relied on third-party technologies to assist us integrate complex functionality.

1. *AWS Cloud Storage:* AWS is a web-based interface implemented by Amazon to manage the storage of static content. AWS is a cloud-based storage system that allows users to safely store and retrieve user-generated material. We utilize it to keep each authors’'s PDF papers in their corresponding folder.

2. React PDF Viewer: It is an extension of the React framework designed for displaying PDF files as if they were images in the browser. We used it to allow reviewers to preview the submitted papers for convenience without downloading them into the device.

4.5 Architecture diagram

We utilized React JS for the frontend and Python Django for the backend. We designed our API to perform CRUD operations and send requests to the machine learning microservice, retrieving the necessary information from the model. The deployment process into Heroku consisted of

independently deploying machine learning microservice, backend microservice, frontend service, and PostgreSQL database. Figure 14 shows the architecture diagram.

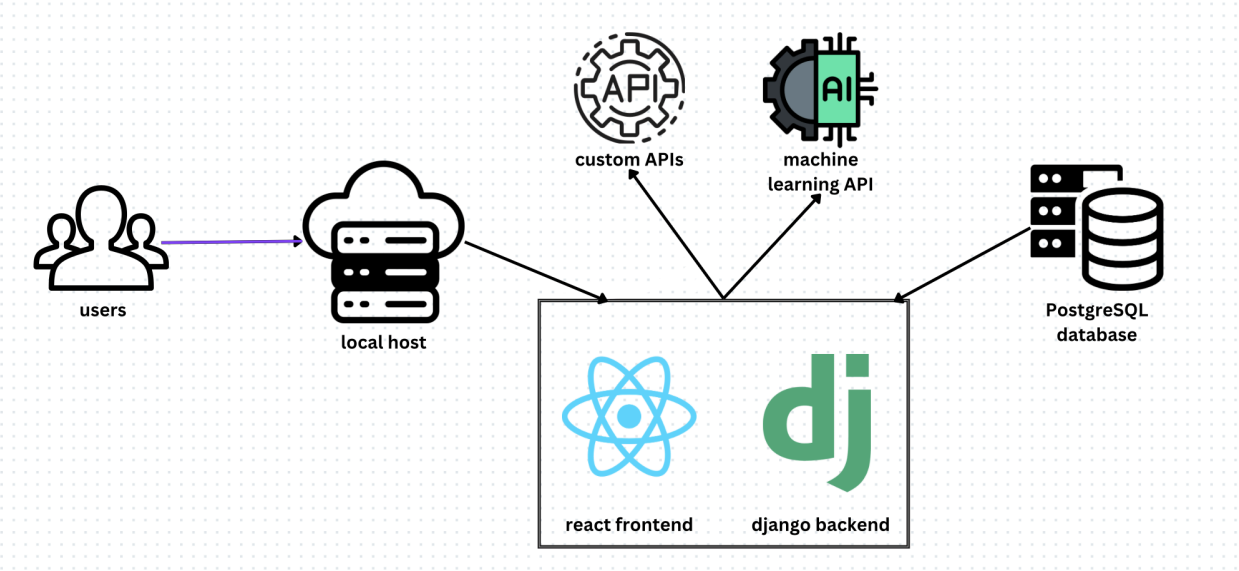

Figure 14. Architecture diagram

4.6 Planning and management

For scheduling and planning we created a Gantt chart dividing the deadlines of tasks implementation into months of the academic year. For the code review and storage, we used GitHub repository.

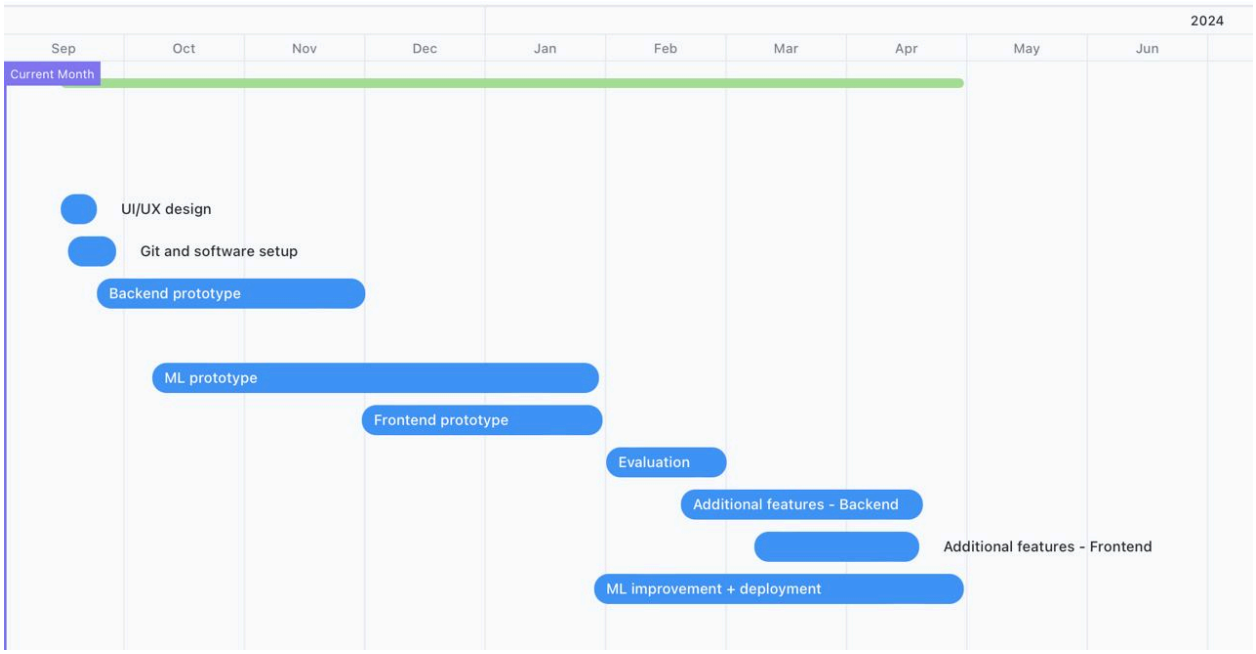

Figure 15. Gantt chart

In the previous two semesters, we did our best to stick to our timetables. In the first semester, we agreed that we will meet once a week to work on the project. However, we noticed that it was not sufficient for us to meet our deadlines, so we decided to meet 2-3 times a week to collaborate on different parts of the project. After the collaboration, we have discussed what we did and what is expected to be done till the next meeting, writing the key notes in the Telegram group.

5. Project Execution

5.1 Design decisions

Regarding the backend database design we decided it would be optimal to implement the main 3 models which are User, Publication and Category. There is a many-to-many relationship between User and Publication and many-to-many relationship between Publication and Category. On the frontend side we adhered common UI/UX guidelines such as consistency and minimalistic layouts in order to achieve user friendly design.

5.2 Changes in project

Over the course of the Spring semester we investigated several best practices in the peer review process and we have come to the conclusion that feedback should contain four main parts which are Summary, Major Issues, Minor Issues and Decision with following options: accept with no revisions, accept with minor revisions, accept with major revisions and reject [5]. This structure of the feedback required us to implement a separate model named Feedback with a one-to-many relationship towards the Publication model, because it would be cumbersome to encompass it all in one model.

Regarding the ML architecture, initially we wanted to implement only the scoring mechanism which would be used to rank the reviewers by their relevance. In fact, this is a common approach in the modern conferences, where only the similarity scores are computed. However, the conferences are usually dedicated to only a few subject areas like Artificial Intelligence or Computer Networks and have a manageable number of submissions, thus it is feasible to run the scoring algorithm on all of them. Our project, on the other hand, encompasses 150+ subject areas, thus it is pointless to run similarity scores between Computer Networks expert and Physics related paper. To save the computational resources, we decided to introduce the initial filtering step achieved by classifying the paper into a few topical categories and cutting out the obvious dissimilarities.

5.3 Problems

There are several problems we encountered over the course of 2 semester period:

1. We initially tried to utilize SQLite instead of PostgreSQL due to its ease of use. Unfortunately, Heroku, in which we deployed our project, does not support SQL.
2. Initially, we all “pushed” our contributions to the main branch of the project in Github and sometimes forgot to “pull” the recent code to stay up-to-date with the main code. As a consequence, there were cases when some commits were lost. We solved this problem by utilizing Pull Requests so that all team members would be notified and creating separate branches for different features to be implemented.
3. We tried to construct a custom component that enables users to view the PDF version of the submitted paper in detail. However, building our own tool required handling all relevant issues manually, which increased the complexity and development time of your

application significantly. So we decided to search for a reliable and well-supported tool such as `react-pdf-viewer/core` library and associated plugins like `defaultLayoutPlugin` that are designed specifically to handle PDF rendering in a web environment. These tools manage many complexities such as rendering, zooming, and navigating pages efficiently.

4. We encountered several problems with Django database migrations due to constant adjustments to the structure of our models. Sometimes, we were unable to apply migrations to the database because of conflicts between migration files. We resolved this issue by adhering strictly to Django's migration requirements and by first visualizing the model structure using tools like Lucidchart.
5. The main problem that occurred during the training of the ML models was the lack of computational resources required to train and load big models. We mainly used Kaggle's free weekly credits for the P100 GPU. However, it was not always sufficient, sometimes forcing us to wait until the credits were renewed. Even then, as the complexity of our Classifier Model grew, we could not rely solely on Kaggle's computational resources. As a solution, we started to rent out the Google Colab GPUs for training. Apart from the training process, we were required to balance the model's complexity so that it could be loaded into the microservice that would be running it for our web platform.

6. Evaluation

Since the ML system is the core of our project, we defined the performance of the ML model to be the main success criteria for our project. Thus, to evaluate the quality of the whole project, we decided to evaluate the quality of the paper-reviewer matches of the ML system.

First of all, we needed to evaluate the performance of the Classifier Model in order to proceed further, as the poor performance of the classifier would result in a bottleneck: if the system searches for the reviewer in the wrong subject area, then the system would never suggest good matches no matter how good the Matching Model is. We performed the accuracy evaluation of the model on the testing set of the arXiv dataset. The final version of the model resulted in ***92.67%*** accuracy (Figure 16).

```
[34]: test_model(model, test_loader)

Test Accuracy: 92.67%
```

Figure 16. Classifier Model testing output

To evaluate the Matching Model and, thereby, the whole ML system, we conducted manual testing, as there is no publicly available dataset with the paper-reviewer assignment evaluations to test our system on. For this manual testing, we constructed 30 reviewer profiles with different areas of research (they were in the same high-level area but had different subtopic focuses) and collected corresponding papers for each of these research areas. Then, we set the number of reviewers that can be assigned to a particular paper to be 2. Then, we tested if our platform would assign the papers to the correct reviewers. We considered the assignment to be correct if at least one of the assigned reviewers were in the same research area covered by the paper. In ***26 out of 30 cases***, the assignments were correct. Although we realize the very limited nature of the test, we believe that it provides some insights into the potential of the system.

7. Conclusion and possible future work

Overall, nearly all of our initial objectives were met. We did not strive to satisfy certain unreasonable expectations that we had established at the outset. In other words, after finishing our work on the last day of the Fall semester, we discovered several inconsistencies and redundancies, which we chose to remove. As a consequence, we established new goals for the spring semester and accomplished them all.

Our aim was to make a convenient and user-friendly platform for authors and reviewers. In our opinion, we succeeded in achieving this goal. The creation of the Publication Submission and Review website is a big step forward in automating and enhancing the fairness of the academic peer review process. Using machine learning approaches, we effectively addressed various inefficiencies inherent in traditional systems, such as prejudice and time delays, by anonymizing submissions and connecting articles with relevant reviewers based on expertise.

The list of possible future work for the project:

1. Design mobile application for accessing the platform

2. Increase the robustness of the whole system
3. Adding multi-language support for the website
4. Performance metrics to track ML models and user experience
5. Integration with existing academic databases for direct submission to journals or conferences.

The progress of this project demonstrates the dynamic nature of technical innovation in academia. Moving forward, ongoing review and modification will be critical to ensuring the system's relevance and success, eventually leading to a more equal and strong academic community.

References

[1] K. Leyton-Brown, Y. Nandwani, H. Zarkoob, C. Cameron, N. Newman, D. Raghu, et al., "Matching papers and reviewers at large conferences," arXiv preprint arXiv:2202.12273, 2022.

[2] O. Anjum, H. Gong, S. Bhat, W-M. Hwu, and J. Xiong, “Pare: A paper-reviewer matching approach using a common topic space,” arXiv preprint arXiv:1909.11258, 2019.

[3] K. E. Chai, R. L. Lines, D. F. Gucciardi, and L. Ng, "Research screener: a machine learning tool to semi-automate abstract screening for systematic reviews," Systematic reviews, vol. 10, pp. 1-13, 2021.

[4] Y. Im, G. Song, and M. Cho, "Perceiving conflict of interest experts recommendation system based on a machine learning approach," Applied Sciences, vol. 13, no. 4, p. 2214, 2023.

[5] MIT Communication Lab, "Peer Review," MIT CommKit, [Online]. Available: https://mitcommlab.mit.edu/broad/commkit/peer-review/

[6] Kaggle. "arXiv Dataset." [Online]. Available: https://www.kaggle.com/datasets/Cornell-University/arxiv

[7] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," EMNLP 2014, Jun. 3, 2014, https://doi.org/10.48550/arXiv.1406.1078

[8] F. Karabiber, "TF-IDF — Term Frequency-Inverse Document Frequency," Learn Data Science, [Online]. Available: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency [Accessed: April 18, 2024].