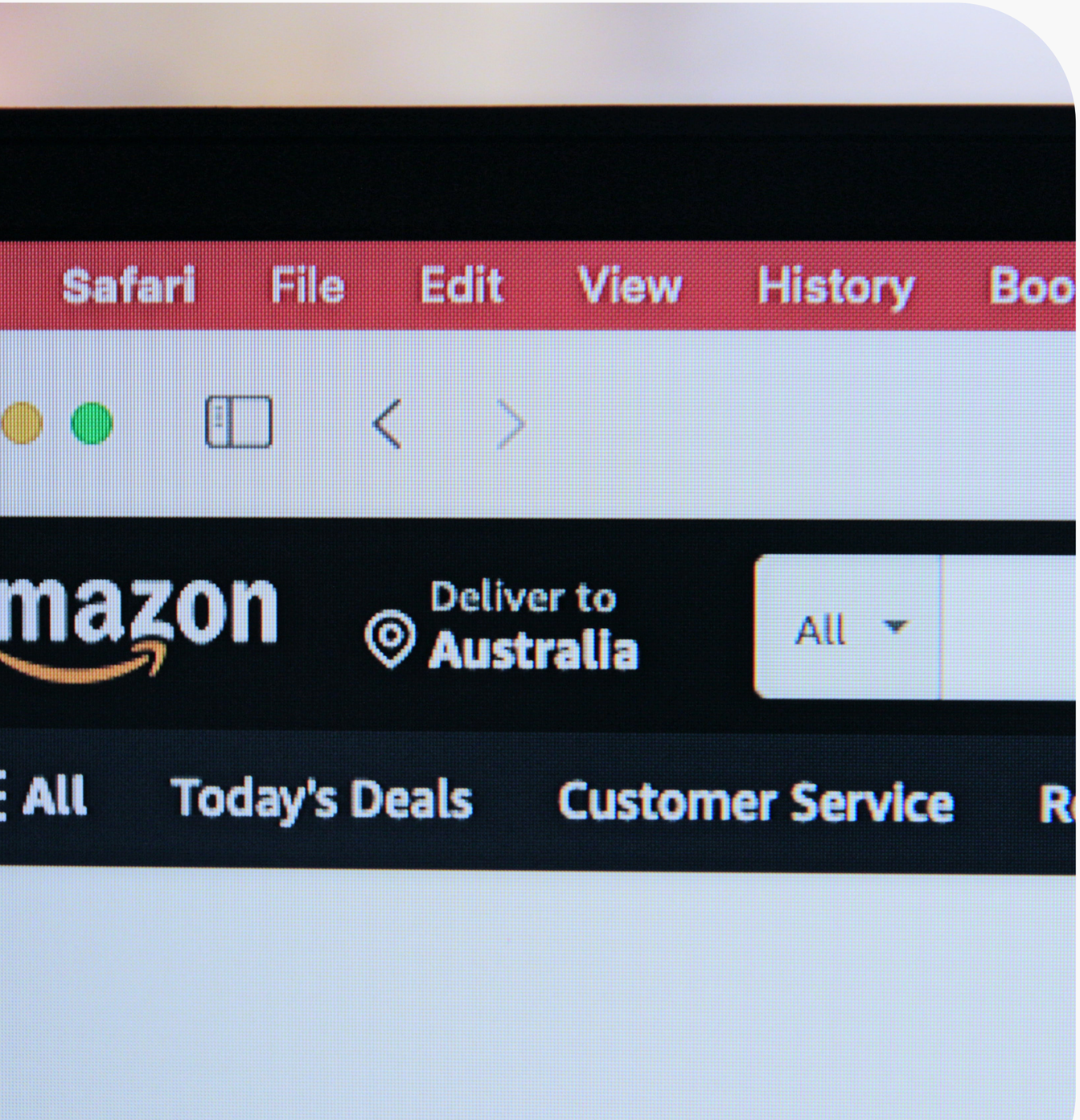
BDA PROJECT PHASE 2

Maaz Khaled

Eman Tahir

Yahya Qureshsi



INTRODUCTION

- This phase is divided into three major parts: model training, Flask setup and connection to frontend and backend, and coding of Kafka producer and consumer to stream real-time recommendations based on user preferences.

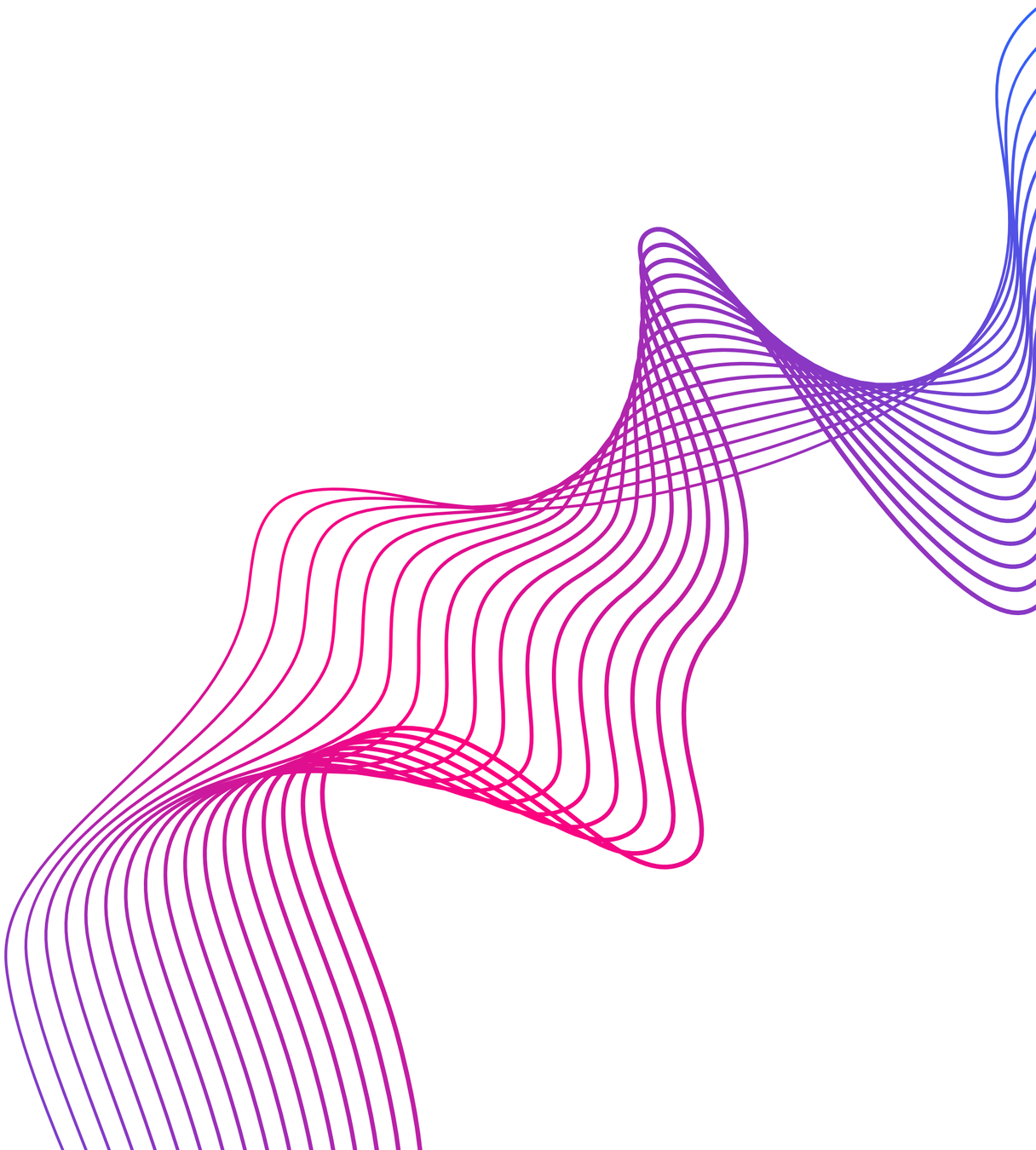
Part 1

Selecting ML Algorithm

The first part of the project involved model training. This required selecting an appropriate machine learning algorithm and fine-tuning it using a dataset that was cleaned and pre-processed. The goal was to create a model that could accurately predict user preferences based on their past behavior.

Model Training

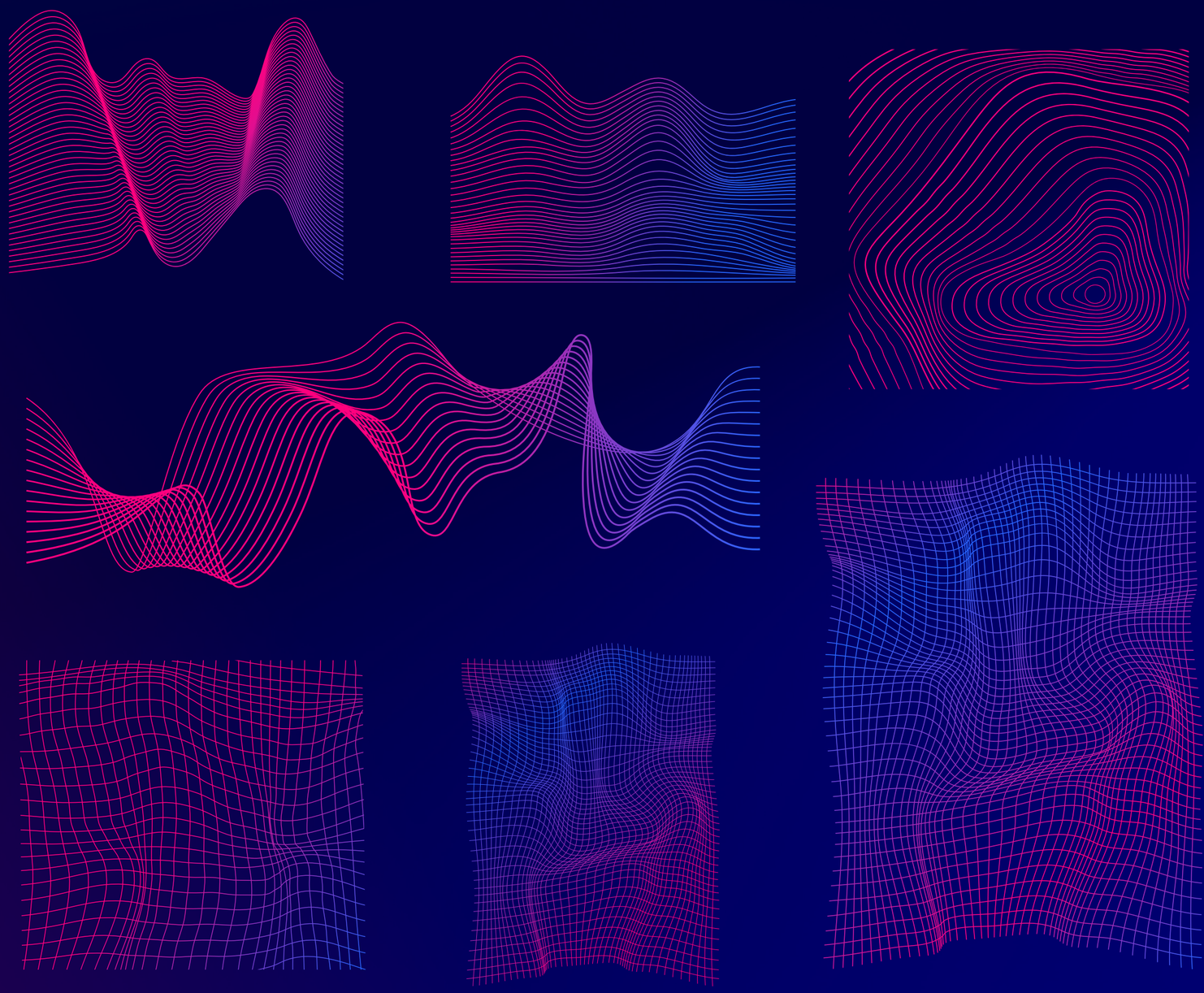
The model was trained on a large dataset of user interactions with the application, such as clicking on items, adding them to a cart, or purchasing them. The algorithm used for the model training was chosen based on its ability to handle large datasets and its performance in predicting user preferences accurately.



Part 2

The second part of the project involved setting up Flask, a web application framework, and connecting it to both the frontend and backend. The frontend was designed using modern web technologies such as HTML and CSS.

The backend connection included kafka producer and consumer. The services were developed using Python and its libraries. The connection between the frontend and backend was established using RESTful API endpoints that could handle requests and responses.



Part 3

1	The third part of the project involved coding the Kafka producer and consumer to stream real-time recommendations based on user preferences
2	The goal was to provide personalized recommendations to users in real-time based on their past interactions with the application
3	The producer was responsible for collecting user behavior data and pushing it to a Kafka topic.
4	The consumer was responsible for processing the data from the Kafka topic and generating personalized recommendations for the user
5	The recommendations were then sent back to the frontend via Flask.

Member Contributions

Yahya

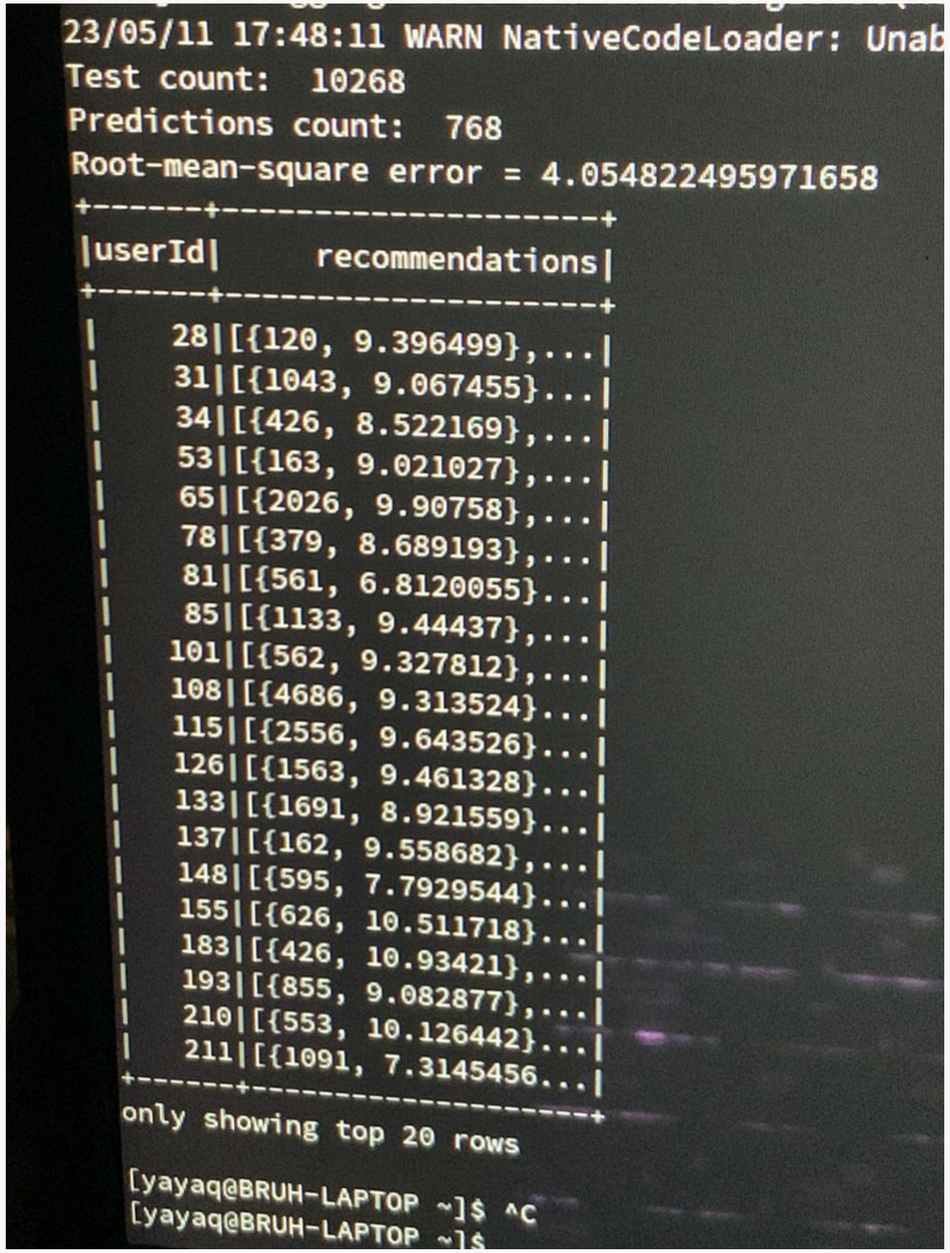
responsible for training the machine learning model that was used to generate recommendations for users..

Maaz

set up a Flask application, connecting the server to both the front-end interface and the Kafka stream

Eman

developed the code to set up a Kafka producer and consumer that allowed real-time recommendations to be streamed

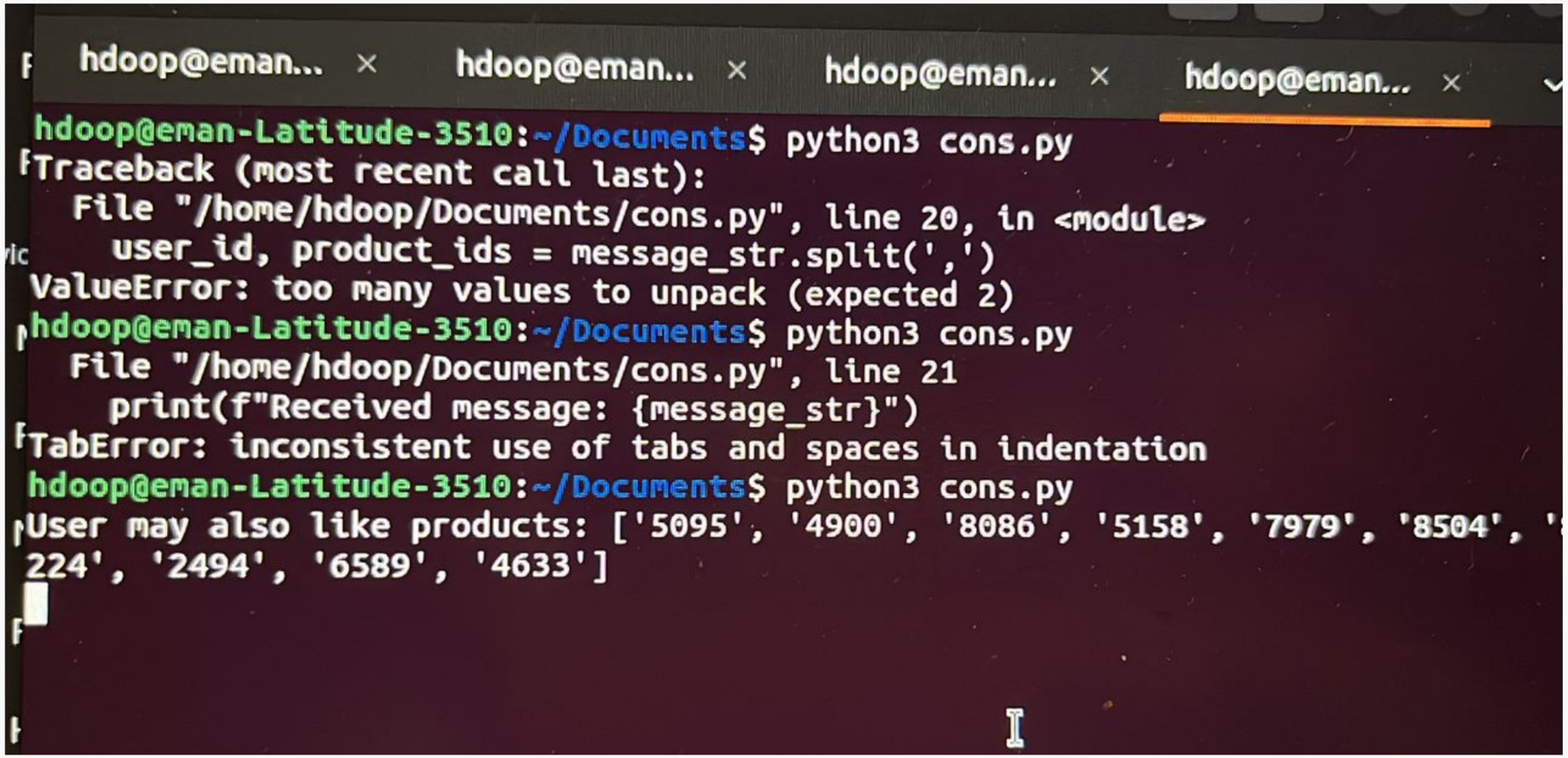


Screenshots

Model Training

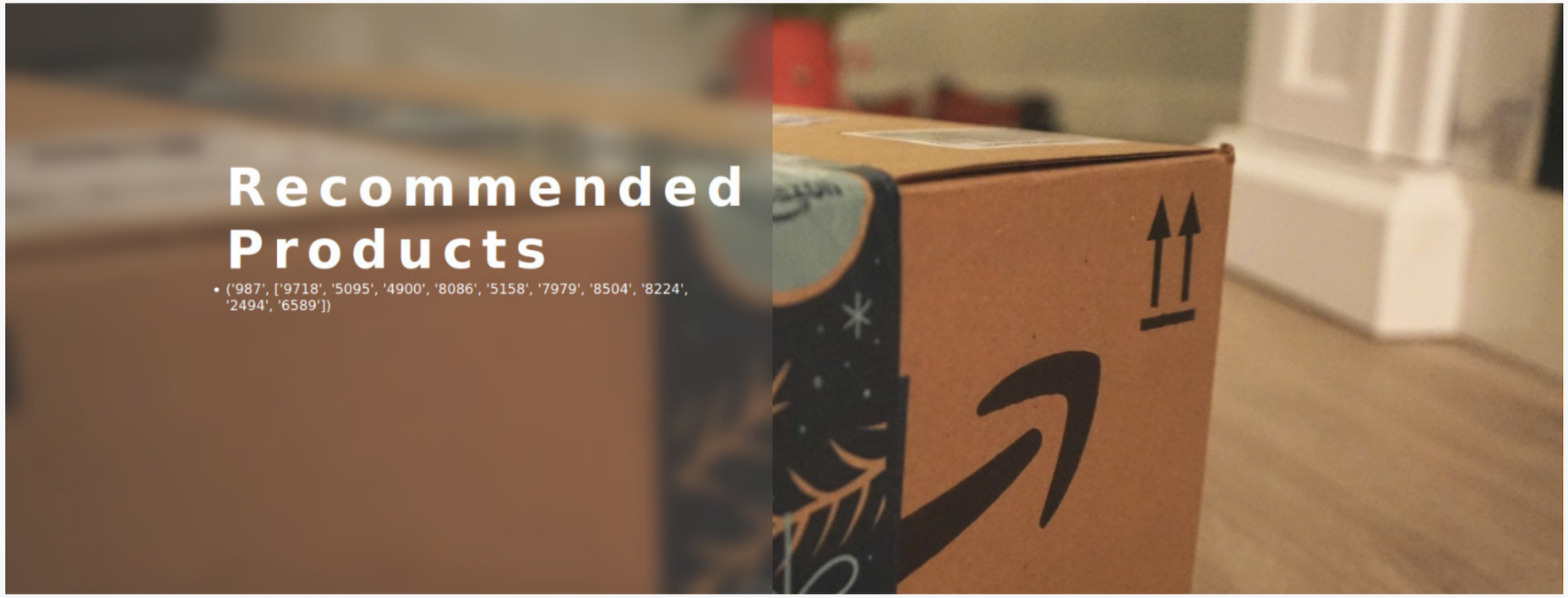
Screenshots

Consumer



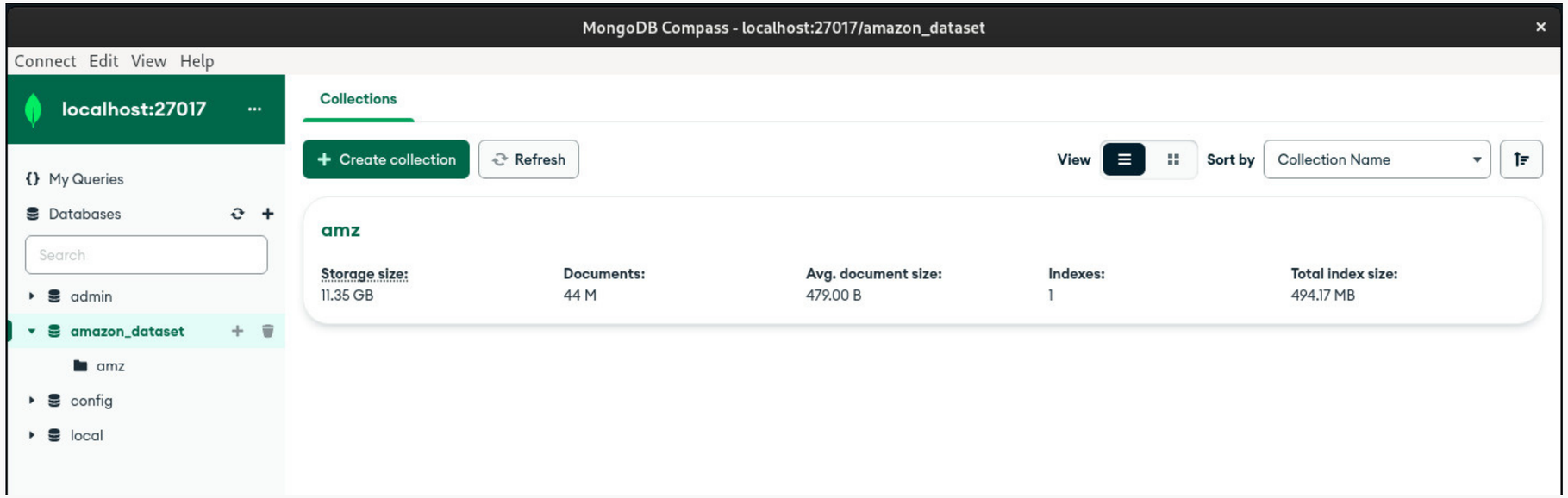
Screenshots

Front-end recommendation



Appendix A

Data upload to mongoDB



The idea was to enhance our recommendation system by leveraging the wealth of data available through the Amazon API. By accessing detailed product information, we could present users with more tailored suggestions based on specific product characteristics. However, gaining access to the Amazon API proved to be a significant challenge. The API requires a seller account, which we were unable to obtain during the project's timeframe. As a result, we had to put this part of our project on hold.

Appendix A: Continued

Data Upload to MongoDB:

After the initial phase of the project, we decided to upload our data to MongoDB, a document database, for better data management and accessibility. This task was fraught with complications due to the complex nature of our data and the intricacies of MongoDB's data upload process.

Despite the challenges, we successfully uploaded the data to MongoDB. However, due to the complications we faced, this task was completed after the deadline of Phase 1. This delay underscored the technical difficulties associated with manipulating large, complex datasets and the importance of allocating sufficient time for data management tasks.

Conclusion:

While we faced some hurdles in our quest to build an enhanced recommendation system, we learned valuable lessons about the complexities of implementing advanced features and dealing with real-world data. Despite not being able to leverage the Amazon API as initially planned, we still managed to construct a robust recommendation system that combines the strengths of ALS and Content-Based Filtering. Our experience with MongoDB, while challenging, was a testament to our team's resilience and problem-solving skills. As we move forward, we will use these experiences to guide our future work and continue to improve our recommendation system