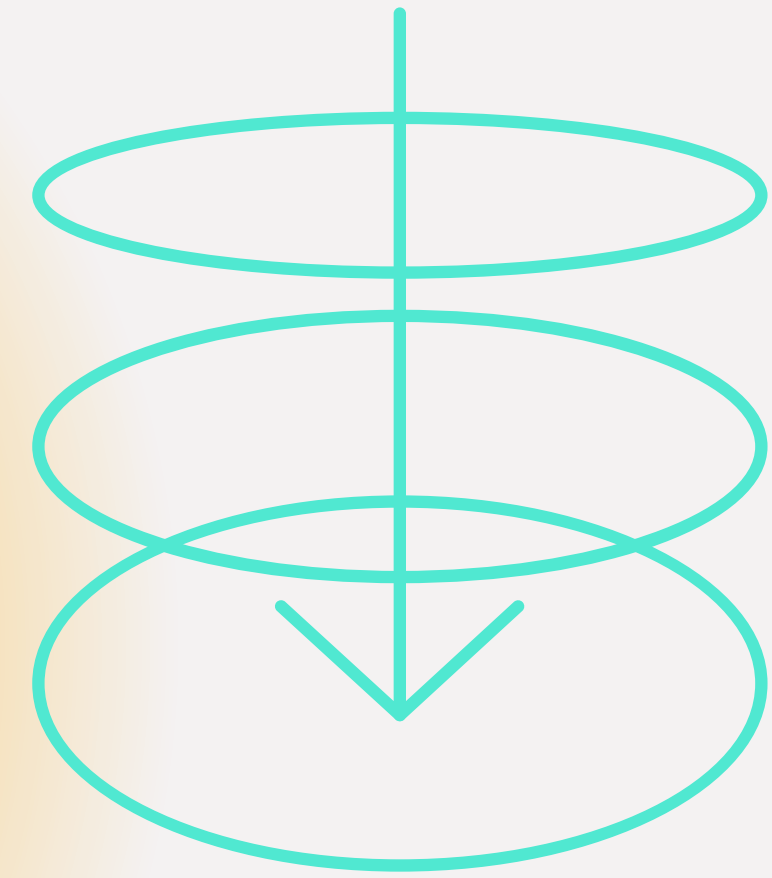


WID2002

COMPUTING MATHEMATICS 2

Student's Productivity
Apps Recommendation
System

and its impact on today's
students.



PRESENTED BY GROUP APPLE PI

Group Members

NAME	MATRIC NO
Samiha Tasnim Dristy	S2112287
Peter Siow Wei Chun	U2102775
Muhammad Showherda Ad-Din	S2117118
Mohammed Al Aubasani	S2102848
Isaiah Disimond	U2000487

1. Problem Statement
2. Analysis
3. Statistics
4. Conclusions

It is important to
understand the utilities
of computing maths

Recommendation

PROBLEM STATEMENT

Students struggle to leverage educational technology effectively due to distractions and a vast array of options. They face difficulty in finding suitable digital tools that align with their educational needs and goals.

Goal

Create a personalized student productivity recommendation system based on app ratings to optimize their learning experience, enhance academic productivity, and improve engagement and learning outcomes.

Collaborative filtering technique

Artificial

Differentiation

Trello

Matrix

Matrix factorization is a collaborative filtering technique that decomposes a user-app rating matrix into latent factors to uncover patterns and make personalized recommendations based on user preferences and app characteristics. By applying matrix factorization, the system can effectively analyze app ratings and suggest relevant and useful productivity apps tailored to each student's needs and interests.

UserID	AppID	Rating
1	1	1
1	5	5
1	6	1
1	10	5
1	15	4
1	17	4
1	18	5
1	20	5
2	11	1
2	13	5

Data Description

1. Data collected from Google Play using the Google Play Scraper library, including reviews, ratings, and user names for productivity apps.
2. Cleaned and preprocessed data saved as "df_reviews.csv" with user names, app packages, and ratings.
3. Rearranged data into "df_rearranged.csv" with columns: UserID, AppID, and Rating.

AppID	AppName
1	Microsoft OneNote
2	Trello
3	Wunderlist
4	Todoist
5	Google Keep
6	Any.do
7	Workflowy
8	Dropbox
9	Google Docs
10	Google Sheets
11	Google Slides
12	Google Drive
13	Google Classroom
14	Mendeley
15	GitHub
16	TED
17	Udemy
18	Duolingo
19	Quizlet
20	Memrise
21	Anki
22	RescueTime
23	Boosted Productivity
24	StudyBlue
25	Remind
26	Grammarly Keyboard

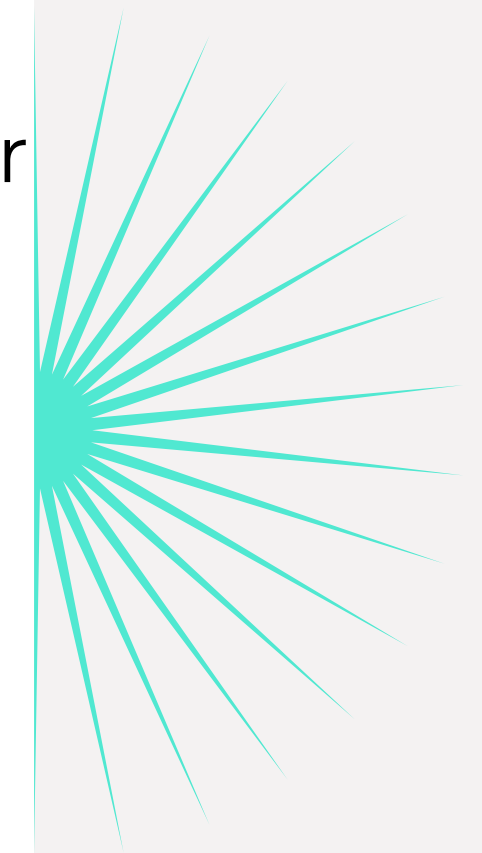
Data Description

We also mapped the App names to AppIDs in "df_app_names.csv" for easier interpretation of recommendations.

Let's learn to use these new tools

Data Analysis

1. Dataset collected from Google Play using the Google Play Scraper library.
2. Dataset includes user-generated reviews, ratings, and app details.
3. Data cleaned to remove duplicates and handle missing values.
4. Structured dataset with users as rows and apps as columns.
5. Analysis focuses on uncovering trends and patterns between users and apps based on ratings.



Data Analysis

```
1]: import pandas as pd

# Read the CSV file into a DataFrame
df_reviews = pd.read_csv('df_reviews_updated.csv', index_col=0)

# Create a dictionary to map usernames to sequential IDs
username_to_id = {name: i+1 for i, name in enumerate(df_reviews.index)}

# Create a dictionary to map app packages to sequential IDs
app_package_to_id = {package: i+1 for i, package in enumerate(df_reviews.columns)}

# Create lists to store the rearranged data
user_ids = []
app_ids = []
ratings = []

# Iterate over each row in the DataFrame
for index, row in df_reviews.iterrows():
    # Get the username and corresponding ID
    username = index
    user_id = username_to_id[username]

    # Iterate over each column in the row
    for column, rating in row.iteritems():
        # Skip columns where the rating is 0
        if rating == 0:
            continue

        # Get the app package and corresponding ID
        app_package = column
        app_id = app_package_to_id[app_package]

        # Append the data to the lists
        user_ids.append(user_id)
        app_ids.append(app_id)
        ratings.append(rating)

# Create a new DataFrame with the rearranged data
df_rearranged = pd.DataFrame({'UserID': user_ids, 'AppID': app_ids, 'Rating': ratings})

# Print the rearranged DataFrame
print(df_rearranged)










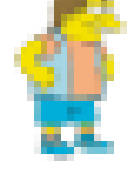

# Save the rearranged DataFrame to a CSV file
df_rearranged.to_csv('df_rearranged.csv', index=False)
```

The analysis involved arranging the data into a table with columns representing UserID, AppID, and Rating. This table was then used to identify trends and preferences among users. Frequency analysis was performed to count the occurrences of each rating value, providing insights into users' inclinations towards different applications.

User Ratings Distribution












The user ratings distribution was presented in a rearranged table, consisting of UserID, AppID, and Rating columns. This arrangement allowed for a thorough examination of user preferences and behavioral patterns. A comprehensive frequency analysis was conducted to tally the occurrences of each rating value, providing valuable insights into the most prevalent ratings and user inclinations towards different applications.

Solution Used:

		Items					
							
Users		10	-1	8	10	9	4
		8	9	10	-1	-1	8
		10	5	4	9	-1	-1
		9	10	-1	-1	-1	3
		6	-1	-1	-1	8	10

➔ User-item Interaction matrix

Workings of Matrix Factorization

						
	10	-1	8	10	9	4
	8	9	10	-1	-1	8
	10	5	4	9	-1	-1
	9	10	-1	-1	-1	3
	6	-1	-1	-1	8	10

**User-item Interaction Matrix
(R)**

\approx

**User Matrix
(Q)**

\times

**Item Matrix
(P)**

Our dataset

	Microsoft One	Trello	Wunderlist	Todoist	Google Keep	Any.do	Workflowy	Dropbox	Google Docs	Google Sheets	Google Slides	Google Drive	Google Classroom
Aadit	1	0	0	0	5	1	0	0	0	5	0	0	0
Aahan	0	0	0	0	0	0	0	0	0	0	1	0	5
Aaliyah	0	0	0	0	0	0	0	0	1	1	4	0	1
Aarav	0	0	0	0	4	4	0	0	5	3	0	5	1
Aarush	0	0	0	5	5	0	0	0	0	0	1	0	5
Abbas	5	5	0	1	3	4	0	5	5	5	5	5	1
Abbott	2	0	0	5	5	0	0	0	0	0	5	5	1
Abby	1	0	0	5	3	1	0	5	1	0	3	0	2
Abdel	5	0	5	1	5	5	0	5	5	5	4	4	1
Abdelrahman	0	0	0	1	5	5	0	0	0	0	4	0	0

We then choose a value for K, let's say 26 (as we have 26 apps in our data).

$$R^{(user \times app)} \approx X^{(user \times K)} \times Y^{(K \times app)} = \hat{R}^{(user \times app)}$$

After substituting the values;

$$R^{(2632 \times 26)} \approx X^{(2632 \times 26)} \times Y^{(26 \times 26)} = \hat{R}^{(2632 \times 26)}$$

How do we factorize the rating matrix R ?

$$\widehat{r_{ij}} = x_i y_j = \sum_{k=1}^{k=K} x_{ik} y_{kj}$$

Where,

$$\widehat{r_{ij}} \in \hat{R}, \quad x_{ik} \in X, \quad \text{and} \quad y_{kj} \in Y$$

and

$$x_i = i^{th} \text{ row in } X,$$

$$y_j = j^{th} \text{ column in } Y$$

Now we calculate the difference between R and \hat{R} for values in the matrix that are not 0.

$$e_{ij} = (r_{ij} - \widehat{r_{ij}}) \quad \text{where } r_{ij} \neq 0$$

$$e_{ij} = r_{ij} - \sum_{k=1}^{k=K} x_{ik} y_{kj}$$

How do we factorize the rating matrix R?

Using this we can calculate the squared error.

$$e_{ij}^2 = \left(r_{ij} - \sum_{k=1}^K x_{ik} y_{kj} \right)^2$$

When it comes to a single instance of k the equation for error becomes.

$$e_{ij}^2 = (r_{ij} - x_{ik} y_{kj})^2$$

At this instance we try to minimize the error therefore we take partial derivatives with respect to x_{ik} and y_{kj} . So we get,

$$\frac{\partial e_{ij}^2}{\partial x_{ik}} = -2 (r_{ij} - \widehat{r_{ij}}) y_{kj}$$

and

$$\frac{\partial e_{ij}^2}{\partial y_{kj}} = -2 (r_{ij} - \widehat{r_{ij}}) x_{ik}$$

How do we factorize the rating matrix R?

Using this we can calculate the squared error.

$$e_{ij}^2 = \left(r_{ij} - \sum_{k=1}^K x_{ik} y_{kj} \right)^2$$

When it comes to a single instance of k the equation for error becomes.

$$e_{ij}^2 = (r_{ij} - x_{ik} y_{kj})^2$$

At this instance we try to minimize the error therefore we take partial derivatives with respect to x_{ik} and y_{kj} . So we get,

$$\frac{\partial e_{ij}^2}{\partial x_{ik}} = -2 (r_{ij} - \widehat{r_{ij}}) y_{kj}$$

and

$$\frac{\partial e_{ij}^2}{\partial y_{kj}} = -2 (r_{ij} - \widehat{r_{ij}}) x_{ik}$$

How do we factorize the rating matrix R?

Now we can use these partial derivatives to update the values for x_{ik} and y_{kj} . We are basically doing gradient descent here.

$$x'_{ik} = x_{ik} + 2 \alpha e_{ij} y_{kj}$$

and

$$y'_{kj} = y_{kj} + 2 \alpha e_{ij} x_{ik}$$

After we have done all the updates we can calculate the total error by summing the error e_{ij} over all the values where $r_{ij} \neq 0$.

$$Total\ Error = \sum_{(r_{ij} \in R, r_{ij} \neq 0)} e_{ij}^2$$

If we expand it, we get,

$$Total\ Error = \sum_{(r_{ij} \in R, r_{ij} \neq 0)} \left(r_{ij} - \sum_{k=1}^{K} x_{ik} y_{kj} \right)^2$$

How do we factorize the rating matrix R?

This is the bare bones of matrix factorization, actual implementations of it have some regularization parameters like β to avoid overfitting.

After regularization, the equation for squared error becomes,

$$e_{ij}^2 = \left(r_{ij} - \sum_{k=1}^{k=K} x_{ik} y_{kj} \right)^2 + \frac{\beta}{2} \sum_{k=1}^{k=K} (||X||^2 + ||Y||^2)$$

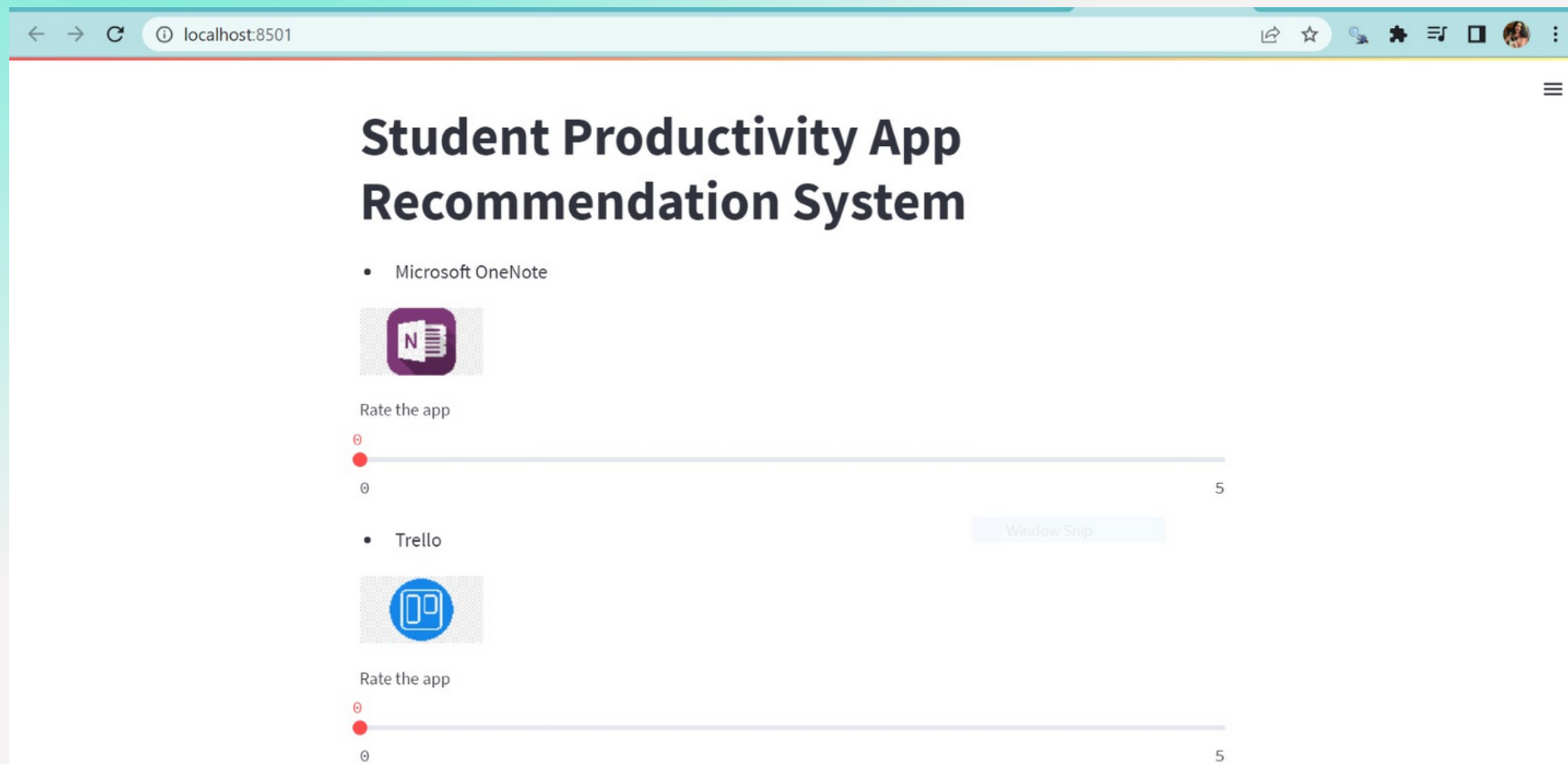


And the equation for the updating becomes,

$$x'_{ik} = x_{ik} + 2 \alpha (e_{ij} y_{kj} - \beta x_{ik})$$

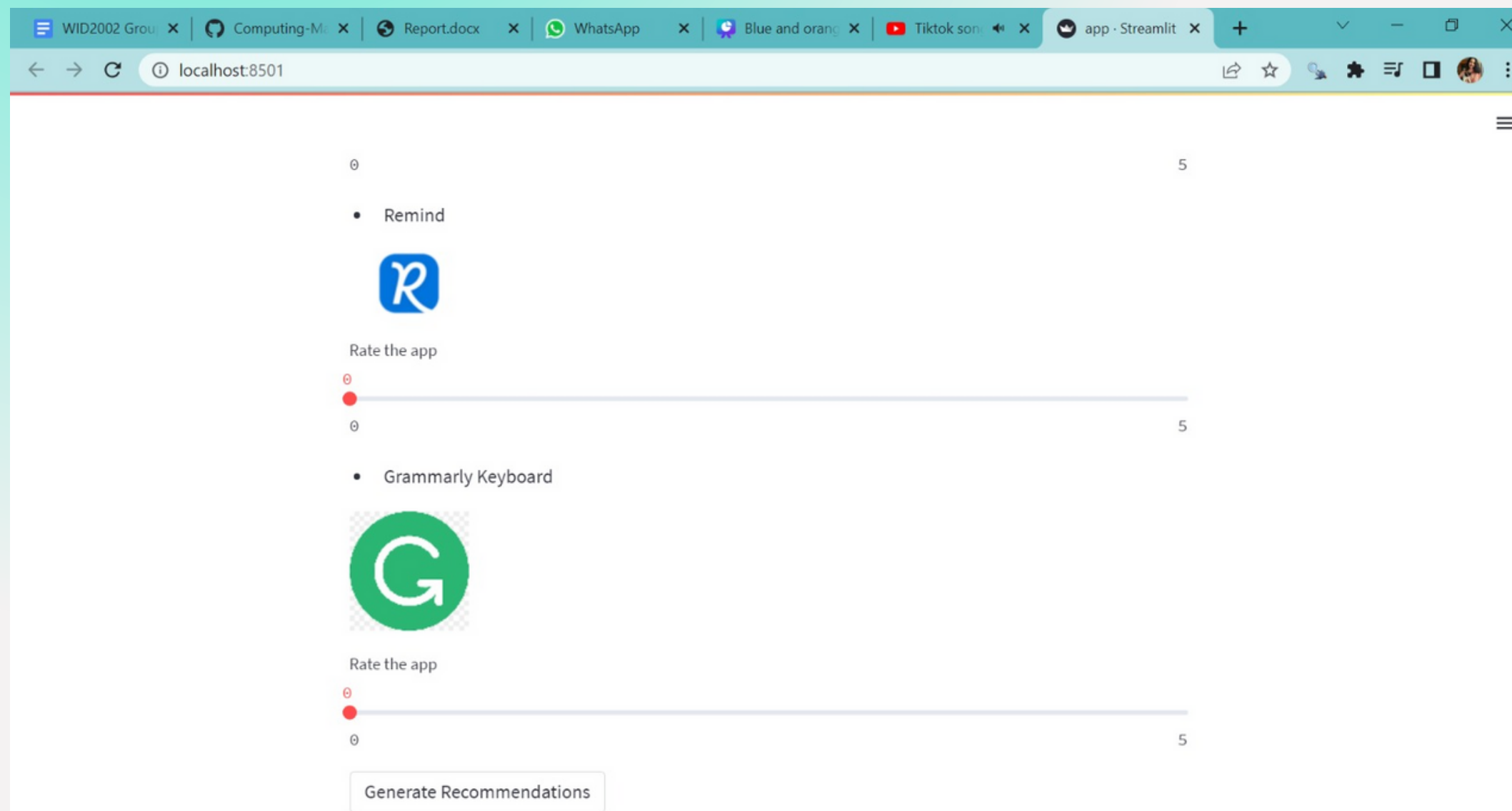
$$y'_{kj} = y_{kj} + 2 \alpha (e_{ij} x_{ik} - \beta y_{kj})$$

Website



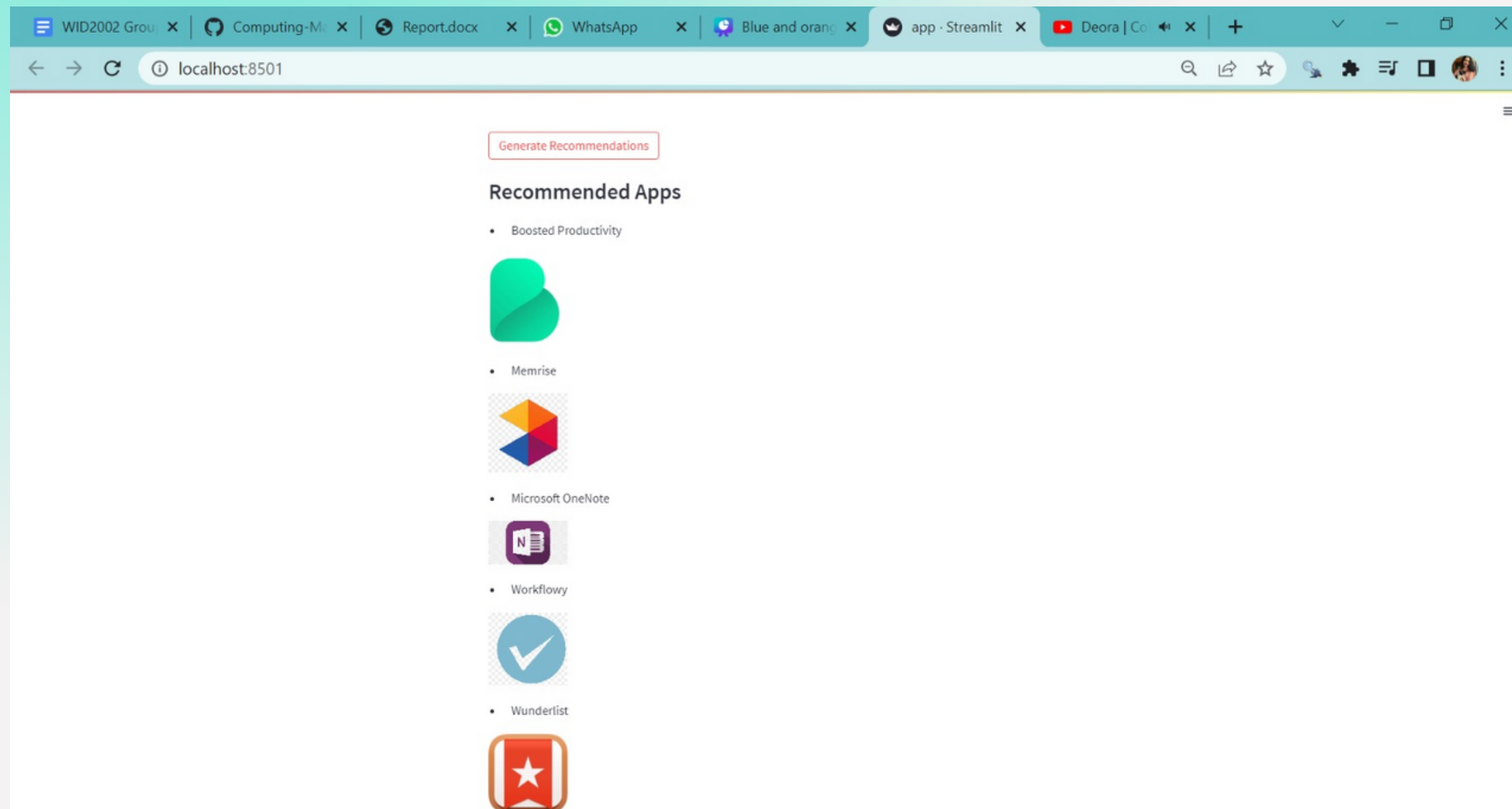
Display of the Top 26 Productive Apps: Users to our website are given a choice of 26 productive apps when they initially enter. The apps come with their names and logos.

Website



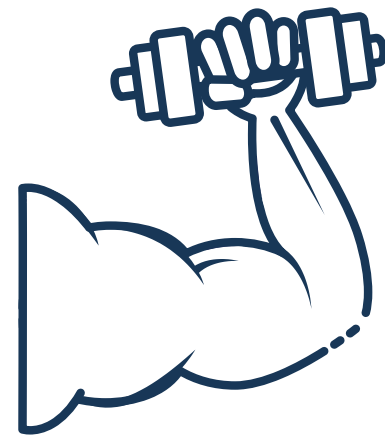
User Rating Input: A slider is provided underneath each app's image to allow users to input their ratings for the apps they are most familiar with. Each app can be rated on a scale of 0-5 by users based on how beneficial or relevant it seems to be for their education.

Website



- Generate Response: The user can go on by clicking the "Generate Recommendation" button when they have done rating the apps they are familiar with.
- Recommended Apps Display: After a user clicks the generate recommendation button, our website evaluates their ratings and presents them with five potential app choices.

Strengths



- Simplicity of the recommendation system
- Easy learning curve.
- High effectiveness of the underlying mathematical model.
- Easy to use & simple user interface design.

Limitations



- Data source limitation
- Sparse data challenges
- Cold start problem not addressed.
- Evaluation metrics not specified.
- Lack of validation and real-world testing.

Future Works

1. Enhanced Data Collection: Improve data gathering techniques for a more comprehensive dataset.
2. Handling Sparse Data: Address missing values in the user-item interaction matrix.
3. Dynamic Recommendation: Adapt and update recommendations in real-time.
4. Addressing the Cold Start Problem: Provide recommendations for new users or items with limited data.
5. Evaluation Metrics: Define metrics to assess recommendation system performance.

Conclusions

This project demonstrates the effectiveness of matrix factorization for a productivity app recommendation system.

It utilizes Google Play data and suggests future improvements such as advanced techniques, contextual information, and evaluation.

User feedback, real-time updates, and addressing sparse data are essential considerations. Enhancements in UI design and visualization enhance user engagement. Overall, it lays the foundation for a robust and user-centric recommendation system.

Thank you