



# UNIVERSITÀ DEGLI STUDI DI MESSINA

## DIPARTIMENT:

Scienze matematiche e informatiche, scienze fisiche e scienze  
della terra

**DEGREE COURSE:**  
**INFORMATICA**

---

## Software Development Project Report



## Movie Recommendation System By Bot

By: Tien M. Nguyen

Guided by: Professor Distefano Salvatore

---

Academic year 2017-2018

## Contents

I.	Overview .....	6
1.	Introduction .....	6
II.	Software process.....	6
1.	Chosen process: Scrum.....	6
2.	Why Scrum?.....	6
3.	Three phases in Scrum.....	7
4.	Three main roles in Scrum .....	7
5.	Sprint cycle.....	7
III.	Initial phase.....	8
1.	Project members.....	8
2.	Project ideal .....	8
3.	Technologies used.....	8
4.	Requirements Analysis.....	9
4.1.	Functional requirements.....	10
4.2.	Non-functional requirements .....	11
4.3.	Use case diagram .....	11
5.	Backlog refinement.....	12
6.	Project estimation.....	12
IV.	Sprint cycles .....	13
1.	Sprint 1.....	13
1.1.	Planning.....	13
1.2.	Implementation .....	13
1.3.	Review.....	14
1.4.	Retrospective .....	14
2.	Sprint 2.....	15
2.1.	Planning.....	15
2.2.	Implementation .....	15
2.3.	Review.....	16
2.4.	Retrospective .....	16
3.	Sprint 3.....	17
3.1.	Planning.....	17
3.2.	Implementation .....	17

3.3.	Review.....	18
3.4.	Retrospective .....	18
4.	Sprint 4.....	19
4.1.	Planning.....	19
4.2.	Implementation .....	19
4.3.	Review.....	19
4.4.	Retrospective .....	20
V.	Project termination.....	20
1.	Reason why project was terminated .....	20
2.	Product Owner mistakes.....	20
3.	Scrum Master mistakes.....	20
4.	Development Team mistakes .....	20
VI.	Initial phase (2 <sup>nd</sup> ).....	20
1.	Project members.....	20
2.	Project Ideal .....	21
3.	User Story.....	21
4.	Reuse from terminated Scrum.....	21
5.	Technologies used.....	21
6.	Requirements Analysis.....	22
6.1.	User requirements .....	22
6.2.	Use case diagram .....	23
6.3.	Functional requirements.....	23
6.4.	Non-functional requirements .....	25
6.5.	Domain requirements .....	25
6.6.	Data Preparation.....	25
7.	Architectural Design.....	26
7.1.	Components Diagram .....	26
7.2.	Database Design.....	27
7.3.	Backend Design .....	28
7.4.	Frontend Design.....	30
7.5.	Sequence Diagram .....	32
8.	Backlog refinement.....	34
8.1.	Epic.....	34

8.2.	Feature .....	34
8.3.	Product Backlog .....	35
8.4.	Priority assignment .....	37
9.	Project estimation (time and cost) .....	38
VII.	Sprint cycles (2 <sup>nd</sup> ) .....	39
1.	Sprint 1.....	39
1.1.	Planning.....	39
1.2.	Implementation .....	39
1.3.	Review.....	39
1.4.	Retrospective .....	40
2.	Sprint 2.....	41
2.1.	Planning.....	41
2.2.	Implementation .....	42
2.3.	Review.....	42
2.4.	Retrospective .....	44
3.	Sprint 3.....	44
3.1.	Planning.....	44
3.2.	Implementation .....	45
3.3.	Review.....	45
3.4.	Retrospective .....	47
4.	Sprint 4.....	47
4.1.	Planning.....	47
4.2.	Implementation .....	49
4.3.	Review.....	50
4.4.	Retrospective .....	51
VIII.	Project closure .....	52
1.	Release Burn Up Chart.....	52
2.	Verification test.....	52
3.	Validation test.....	55
4.	Operation plan and Deployment order .....	56
4.	Database Deployment .....	56
5.	Backend Deployment.....	60
6.	Frontend Deployment.....	66

7.	Scale solutions .....	75
IX.	Recommendation System Algorithm Design .....	76
1.	Foundation.....	76
2.	Loss Function .....	77
3.	Optimizing the loss function .....	77
4.	Development in Python .....	79
5.	Model input explanation .....	81
6.	Model explanation.....	84
7.	Functions that affect the model .....	84
8.	Model application.....	85
X.	Production Screenshots .....	87

## I. Overview

### 1. Introduction

This is a project report for Software Engineering at the University of Messina taught by Professor Distefano Salvatore. The purpose of the project is to apply a process to create a complete system software product.

The idea of the system can be chosen freely with the consent of the professor and will be presented below.

The product development process I use is Scrum, one of the most popular frameworks for Agile implementation. I work on this project alone and I will play all three main roles in a Scrum project. Taking on all three roles in Scrum is not recommended because Scrum in particular and software development processes in general were born to solve problems related to independent interpersonal cooperation.

### 2. Project Idea

The idea of the project came from the idea of researching and building recommendation systems. From there, we create a system that can be applied the recommendation system.

## II. Software process

### 1. Chosen process: Scrum

Scrum is one of the most popular frameworks for Agile implementation. Scrum focuses on managing iterative development rather than specific Agile practices.

### 2. Why Scrum?

As I say in the first section, Scrum is an Agile framework that is particularly suited for complex projects with rapidly changing requirements, which is often the case in software development. In this project, requirements change based on recommendation systems research is very likely to happen, so Scrum or Agile in general is a reasonable choice for this project. Additionally, Scrum allows for frequent iterations and feedback loops, which can help ensure that the final product meets the needs of stakeholders.

Mention of stakeholders, because this project only has the participation of one person, traditional Agile will not be suitable because all the members in the team will share all roles, all responsibilities. With Scrum, there is a clear division of roles in the three positions of Product Owner, Scrum Master and Development Team which makes it easier to simulate a team using this methodology. I can still understand the difficulty of each position in the project through the work that role needs to do.

Furthermore, Scrum can help ensure that the project remains on track and that potential issues are identified and addressed early on in the development process. Scrum's emphasis on frequent communication and collaboration can help ensure that all members of the team are aware of any potential issues and are working together to address them.

Overall, choosing Scrum can help ensure that the final product meets the needs of stakeholders, is delivered on time, and is of high quality.

### 3. Three phases in Scrum

There are three phases in Scrum:

- Initial phase: in this phase we establish the general objective for the project and design the software architecture. This phase includes the processes related to initiation of a project: Create Project Vision, Identify Stakeholder(s), Scrum Master and Product Owner, Form Scrum Team, Create Prioritized Product Backlog, and Conduct Release Planning.
- Sprint cycle phase: this is a series of sprint cycles, where each cycle develops an increment of the system. This phase includes the processes related to implementation of a project: Create Deliverables, Conduct Daily Standup, Groom Prioritized Product Backlog, and Deliver Deliverables.
- Project closure phase: wraps up the project, completes required documentation such as system help frames and user manuals and assesses the lessons learned from the project. This phase includes the processes related to releasing a project: Ship Deliverables and Retrospect Project.

### 4. Three main roles in Scrum

There are three main roles in Scrum:

- Product Owner: The Product Owner is responsible for maximizing the value of the product resulting from the work of the Development Team. They are also responsible for managing the Product Backlog. They do analysis, are business-oriented, do external interactions-interface. They will keep the team Build the right thing.
- Scrum Master: The scrum master is responsible for ensuring that Scrum is understood and enacted by all team members and stakeholders. They are servant leader, advisor, coach, mentor, facilitator. They will keep the team Build it fast.
- Development Team: The development team is responsible for delivering a potentially releasable increment of “Done” product at the end of each Sprint. They are self-management or self-organizing. They will keep the team Build things right.

### 5. Sprint cycle

Sprint are fixed length, normally 2-4 weeks. In this project one sprint is 5 working days.

Usually, each sprint will have 4 phases:

- Planning: before each sprint, the team selects what it will commit to delivering by the end of the sprint, starting at the top of the product Backlog. During the sprint, what the team committed to deliver does not change, and the end-date of the sprint does not change. If something major comes up, the Product Owner can direct the team to terminate the Sprint prematurely and start a new one.

- Implementation – Daily Scrum Meeting: each day, the team has a short meeting to update each other on progress and surface blocks. They stand up, to keep it fast. To keep the meeting to less than 15 minutes, everyone reports just 3 things: done since yesterday, done by tomorrow, and blocks. Burndown Chart are necessary for the team to successfully self-manage.
- Review: at the end of the sprint, the Product Owner, Team, Scrum Master, and Stakeholder(s) come together and see a demo of what the team has produced. The Product Owner gathers feedback from everyone on ways to improve what has been built. Review of the timeline, budget, potential capabilities, and marketplace for the next anticipated releases of functionality and capabilities.
- Retrospective: Team, Product Owner, and Scrum Master meet at the end of each Sprint to review their way of working, to improve effectiveness. Usually, it will take 3 hours for one month Sprint. This is the opportunity for the Scrum Team to improve and all members should be in attendance, discussing about: what went well, what could be improved, and what we will commit to improving in the next Sprint.

### III. Initial phase

#### 1. Project members

- Product Owner: Nguyen Minh Tien
- SCRUM Master: Nguyen Minh Tien
- Member in Team: Nguyen Minh Tien

Taking on all three roles in SCRUM is not recommended because SCRUM in particular and software development processes in general were born to solve problems related to independent interpersonal cooperation. I must do this project alone because I could not find the right partner at the right time. However, I will try to play each role independently so that I can understand in a project what the roles will contribute.

#### 2. Project ideal

The Ideal of Project is a website where users can get suggested which movies they might like. They will also be able to rate on a 5-point scale the movies they have watched. The system will use a machine learning algorithm to determine which movies users will like based on data given. The more movies they rated, the more accurate the suggested movies would be.

No User Story given.

#### 3. Technologies used

- Git for Source control  
Git is a popular version control system for tracking changes in software code, supporting collaboration, branching, and merging. It is widely used in software development to manage and organize code changes efficiently.
- Draw.io for Designing

Draw.io is a free, online diagramming tool for making flowcharts, UML diagrams, network diagrams, and more. It offers a simple interface, pre-made templates and shapes, collaboration, and integration with other tools. All diagrams in this project are drawn with Draw.io.

- Postman for Testing

Postman has a GUI and can send and view HTTP requests and responses.

- Azure DevOps for Scrum management

Azure DevOps is a suite of development tools, services, and features offered by Microsoft for software development teams to plan, develop, test, and deliver software. It includes services for Agile project management, continuous integration and delivery, testing, and more.

- PostgreSQL for Database Management

PostgreSQL is an open-source object-relational database system that uses and extends the SQL language.

- Django for Backend

Django is a high-level Python web framework used for developing dynamic and scalable web applications. It provides a clean and efficient way to build and maintain applications, with features such as URL routing, template engine, object-relational mapping, and security features.

- ReactJS for Frontend

React is a JavaScript library for building user interfaces, commonly used for developing single-page applications and mobile applications. It uses a component-based approach, allowing developers to build complex UIs by composing simple and reusable components. React also uses a virtual DOM for efficient updates and rendering, making it fast and efficient for dynamic applications.

#### 4. Requirements Analysis

In the initial iteration, I prioritized rapid development of the ideal solution and analyzed the data to identify the suitable machine learning models for the data.

The software will allow the user to perform the various functions:

- Create Account
- Login
- Search for movie
- Get suggested movie
- See a list of movies
- See a specific movie
- Rate a movie
- Logout

The “Create Account” operation allows a new user or guest who has just had access to the software to be able to register and access the software functions. Registration is required since we need user rating movie data.

The “Login” operation allows a user who already has access to the application to be able to reconnect to the system by entering the same data that he entered during the previous registration period.

The “Search for Movie” operation will allow a logged in user to find the exact movie they want to rate.

The “Get Suggested Movie” operation will allow a logged in user to get their might-like movies by the machine learning algorithm.

The “See a list of movies” operation will allow a logged in user to get a list after the operations “Search for Movie” or “Get Suggested Movie”.

The “Rate a Movie” operation will follow the “Search for Movie” which will allow user to rate a movie and save the rate in the database, maybe a function “Remove a Rate” will appear in the future.

The “Get Suggested Movie” operation is the main operation which will allow a user to get their suggested movies that they might like to depend on the machine learning model we develop.

Finally, the “Logout” operation will allow a logged in user to logout from system. Users will need to login again if they need to do any other action on the next connection.

#### 4.1. Functional requirements

Functional requirements refer to the product features or functions that must be implemented by the development team to enable users to carry out their tasks. It is important to make these requirements clear for both the development team and stakeholders. Typically, functional requirements describe the system's behavior in specific circumstances. Below are functional requirements for software:

FR.1: Authenticate Management:

- FR.1.1: Registration User
- FR.1.2: Verify User

FR.2: Movie Management:

- FR.2.1: Movie Searching
- FR.2.2: Movie Suggestion

FR.3: Rating Management:

- FR.3.1: Create Rating

#### 4.2. Non-functional requirements

Nonfunctional requirements, not related to the system functionality, rather define how the system should perform. Below are non-functional requirements for software:

NFR.1: Friendly Graphic User Interface.

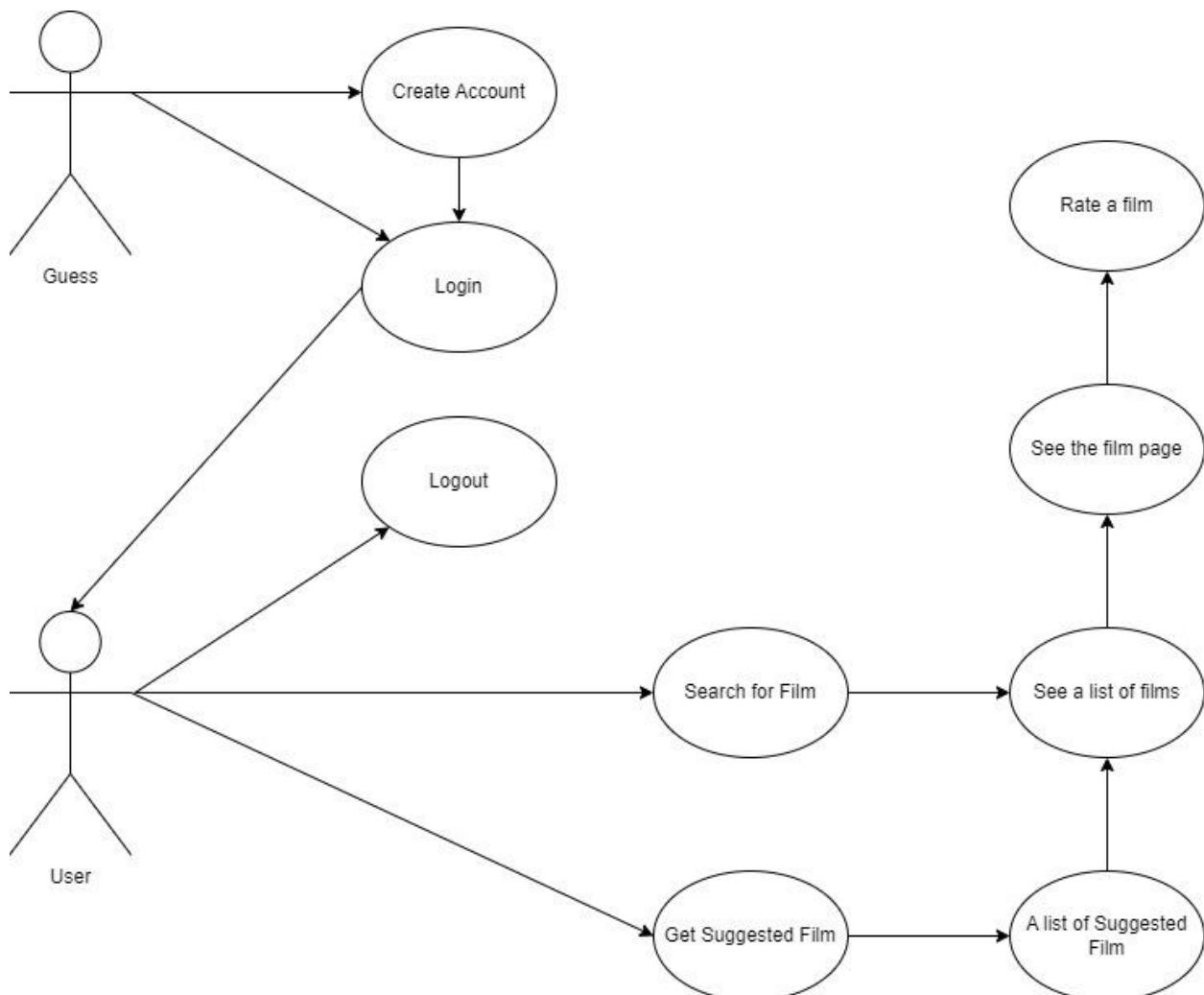
NFR.2: Multiplatform compatible.

NFR.3: Fast response to user.

NFR.4: Movie cover should be available.

#### 4.3. Use case diagram

The specifications for the first software prototype involve a thorough analysis of the customer requirements, which is important and requires careful attention as it determines whether the customer's needs goods are met or not. A "use case diagram" will be used to describe the relationship clearly and accurately between users and the actions they can take while using the software.



## 5. Backlog refinement

From the Requirements Analysis, as a Product Owner, I have refined the backlog as follows (priority order is from top to bottom).

Order	Title	State	Assigned To
1	> Research the Movie Recommendation System	... ● To Do	Tien Nguyen
2	> Authentication: User Registration (Signup)	● To Do	Tien Nguyen
3	> Authentication: User Verification (Login)	● To Do	Tien Nguyen
4	> Authenticate Management: Logout	● To Do	Tien Nguyen
5	> Data preparation	● To Do	Tien Nguyen
6	> Movie Management: Suggestion	● To Do	Tien Nguyen
7	> Movie Management: Searching	● To Do	Tien Nguyen
8	> Movie Management: Get specific movie detail	● To Do	Tien Nguyen
9	> Rating Management: Create Rating	● To Do	Tien Nguyen

## 6. Project estimation

There will be three approaches to building a Recommendation System so the First Issue will take 3 Sprints to complete.

At the same time, we can complete the authentication system in 2 Sprints and Data preparation in 1 Sprint.

There will be one last Sprint for combining components into a complete system.

All the designs will be done in Sprint where the feature or system is in “Doing” status.

Testing will take place right when a feature is in “Done” status as unit tests. The final test phase will take place when all the system is done as a system test.

So, there will be 4 Sprints:

1. Sprint 1 runs from 2<sup>nd</sup> Jan to 6<sup>th</sup> Jan (5 working days). Issues: 1,2.
2. Sprint 2 runs from 9<sup>th</sup> Jan to 13<sup>th</sup> Jan (5 working days). Issues: 1,3,4.
3. Sprint 3 runs from 16<sup>th</sup> Jan to 20<sup>th</sup> Jan (5 working days). Issues: 1,5.
4. Sprint 4 runs from 23<sup>rd</sup> Jan to 27<sup>th</sup> Jan (5 working days). Issues: 6,7,8,9.

## IV. Sprint cycles

### 1. Sprint 1

#### 1.1. Planning

As we discussed in section III.5 Project estimation, in this Sprint, Issues 1 and 2 will be taken care of. All the tasks for these two issues are as below:

Task	Assignee	Status
2 Research the Movie Recommendation System	Tien Nguyen	To Do
17 Research on Neighborhood-based Collaborative Filtering	Tien Nguyen	To Do
18 Research on Matrix Factorization Collaborative Filtering	Tien Nguyen	To Do
19 Conclude and determine the model to apply in the project	Tien Nguyen	To Do
14 Analyze MovieLens ml-100k data set	Tien Nguyen	Doing
15 Research on Recommendation System in general	Tien Nguyen	Doing
16 Research on Content-based Recommendation System	Tien Nguyen	Doing
20 Backend for Registration feature	Unassigned	Doing
21 Frontend for Registration feature	Unassigned	Doing

#### 1.2. Implementation

The result of Analyze MovieLens ml-100k data set appears in the document: "Recommendation\_System".

The result of Research on Recommendation System in general appears in the document: "Recommendation\_System".

The result of Research on Content-based Recommendation System appears in the document: "Recommendation\_System".

For Backend for Registration feature, I use Django for rapid development since its authentication system can be developed with a few configurations.

For Frontend for Registration feature, I use ReactJS, a popular JavaScript library for building user interfaces.

### 1.3. Review

The screenshot shows a Taskboard view with three columns: To Do, Doing, and Done. The To Do column contains three tasks: 17 Research on Neighborhood-based Collaborative Filtering, 18 Research on Matrix Factorization Collaborative Filtering, and 19 Conclude and determine the model to apply in the project. The Doing column contains two tasks: 21 Frontend for Registration feature and 20 Backend for Registration feature. The Done column contains three tasks: 14 Analyze MovieLens ml-100k data set, 15 Research on Recommendation System in general, and 16 Research on Content-based Recommendation System. Each task card includes a summary, assignee (Tien Nguyen), state (e.g., Doing, Unassigned, Done), and a green plus sign for adding more details.

To Do	Doing	Done
<p><input checked="" type="checkbox"/> 17 Research on Neighborhood-based Collaborative Filtering TN Tien Nguyen State Doing</p>		<p><input checked="" type="checkbox"/> 14 Analyze MovieLens ml-100k data set TN Tien Nguyen State Done</p>
<p><input checked="" type="checkbox"/> 18 Research on Matrix Factorization Collaborative Filtering TN Tien Nguyen State To Do</p>		<p><input checked="" type="checkbox"/> 15 Research on Recommendation System in general TN Tien Nguyen State Done</p>
<p><input checked="" type="checkbox"/> 19 Conclude and determine the model to apply in the project TN Tien Nguyen State To Do</p>		<p><input checked="" type="checkbox"/> 16 Research on Content-based Recommendation System TN Tien Nguyen State Done</p>
<p>+ [green]</p>		
<p><input checked="" type="checkbox"/> 6 Authentication: User Registration (Signup) TN Tien Nguyen State Doing</p>	<p><input checked="" type="checkbox"/> 21 Frontend for Registration feature Unassigned State Doing</p>	<p><input checked="" type="checkbox"/> 20 Backend for Registration feature Unassigned State Done</p>
<p>+ [green]</p>		

After the first Sprint, most of the tasks set out have been completed. However, because I spent too much time learning about the recommendation system, I could not complete the authentication function.

### 1.4. Retrospective

Certainly, in the next Sprint the time spent on the recommendation system will be reduced because the background knowledge has been learned in this Sprint. I'll have to try to build the authentication function in the next Sprint.

## 2. Sprint 2

### 2.1. Planning

As we discussed in section III.5 Project estimation, in this Sprint, Issues 1, 3 and 4 will be taken care of. All the tasks for these two issues are as below:

	To Do	Doing	Done
1	<p>2 Research the Movie Recommendation System TN Tien Nguyen State Doing</p> <p>18 Research on Matrix Factorization Collaborative Filtering TN Tien Nguyen State To Do</p>		
2	<p>6 Authentication: User Registration (Signup) TN Tien Nguyen State Doing</p>	<p>19 Conclude and determine the model to apply in the project TN Tien Nguyen State To Do</p> <p>21 Frontend for Registration feature Unassigned State Doing</p>	
3	<p>7 Authentication: User Verification (Login) TN Tien Nguyen State Doing</p>	<p>22 Backend for Verification feature Unassigned State Doing</p> <p>23 Frontend for Verification feature Unassigned State Doing</p>	
4			

### 2.2. Implementation

The result of Research on Neighborhood-based Collaborative Filtering appears in the document: "Recommendation\_System".

For Backend for Verification feature, I use Django for rapid development since its authentication system can be developed with a few configurations.

For Frontend for Verification feature and Frontend for Registration feature, I made it with ReactJS.

### 2.3. Review

So, what I planned for this week is done. I have finished tuning Django as well as creating a GUI with the help of the ReactJS framework.

To Do	Doing	Done			
<p><input checked="" type="checkbox"/> 2 Research the Movie Recommendation System TN Tien Nguyen State Doing</p>	<p><input checked="" type="checkbox"/> 18 Research on Matrix Factorization Collaborative Filtering TN Tien Nguyen State To Do</p>	<p><input checked="" type="checkbox"/> 17 Research on Neighborhood-based Collaborative Filtering TN Tien Nguyen State Done</p>			
<p><input checked="" type="checkbox"/> 19 Conclude and determine the model to apply in the project TN Tien Nguyen State To Do</p>					
<p>+ 6 Authentication: User Registration (Signup) TN Tien Nguyen State Done</p>		<p><input checked="" type="checkbox"/> 21 Frontend for Registration feature Unassigned State Done</p>			
<p>+ 7 Authentication: User Verification (Login) TN Tien Nguyen State Done</p>		<p><input checked="" type="checkbox"/> 22 Backend for Verification feature Unassigned State Done</p>			
<p>+ 23 Frontend for Verification feature Unassigned State Done</p>					

### 2.4. Retrospective

In the next week, I will research another algorithm for the Recommendation System from which to conclude and choose the best option for this project.

Along with that is to apply that recommendation system model to actual data and hope to be able to complete the application soon.

### 3. Sprint 3

#### 3.1. Planning

As we discussed in section III.5 Project estimation, in this Sprint, Issues 1 and 5 will be taken care of. All the tasks for these two issues are as below:

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options
<p>ollapse all</p>					
<p>2 Research the Movie Recommendation System</p> <p>TN Tien Nguyen</p> <p>State Doing</p>	To Do			Doing	
				<p>18 Research on Matrix Factorization Collaborative Filtering</p> <p>TN Tien Nguyen</p> <p>State Doing</p>	<p>19 Conclude and determine the model to apply in the project</p> <p>TN Tien Nguyen</p> <p>State Doing</p>
<p>12 Authenticate Management: Logout</p> <p>TN Tien Nguyen</p> <p>State Doing</p>				<p>24 Frontend for Logout feature</p> <p>TN Tien Nguyen</p> <p>State Doing</p>	
<p>13 Data preparation</p> <p>TN Tien Nguyen</p> <p>State Doing</p>				<p>33 ml-1m rating data export</p> <p>TN Tien Nguyen</p> <p>State Doing</p>	<p>34 ml-1m movie data export</p> <p>TN Tien Nguyen</p> <p>State Doing</p>
				<p>35 Recommendation System Model export</p> <p>TN Tien Nguyen</p> <p>State Doing</p>	

#### 3.2. Implementation

The result of Research on Matrix Factorization Collaborative Filtering appears in the document: “Recommendation\_System”.

The result of Research on Conclude and determine the model to apply in the project appears in the document: “Recommendation\_System”.

For Frontend for Logout feature, in client side I remove session so that user need to login again.

For Frontend for Verification feature and Frontend for Registration feature, done.

The result of ml-1m rating data export appears in the document: “Data\_Preparation”.

The result of ml-1m movie data export appears in the document: “Data\_Preparation”.

The result of Recommendation System Model export appears in the document: “Data\_Preparation”.

### 3.3. Review

So, what I planned for this week is done. I have finished all the tasks taken.

### 3.4. Retrospective

Perhaps if there are more requests from customers in the future, the Backend will have to change the framework to be able to customize it more deeply as well as master the logic thereby limiting security issues even though Django is supported by the community quite well. good but maybe one day there will be a zero-day vulnerability.

## 4. Sprint 4

### 4.1. Planning

As we discussed in section III.5 Project estimation, in this Sprint, Issues 6,7,8 and 9 will be taken care of. All the tasks for these two issues are as below:

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options
▲ Collapse all	To Do			Doing	
9 Movie Management: Suggestion TN Tien Nguyen State Doing				25 Backend for Suggestion feature Unassigned State Doing	26 Frontend for Suggestion feature Unassigned State Doing
8 Movie Management: Searching TN Tien Nguyen State Doing				31 Backend for Searching feature Unassigned State Doing	32 Frontend for Searching feature Unassigned State Doing
11 Movie Management: Get specific movie detail TN Tien Nguyen State Doing				27 Backend for Get movie detail feature Unassigned State Doing	28 Frontend for Get movie detail feature Unassigned State Doing
10 Rating Management: Create Rating TN Tien Nguyen State Doing				29 Backend for Create rating feature Unassigned State Doing	30 Frontend for Create rating feature Unassigned State Doing

### 4.2. Implementation

All the tasks done by development.

### 4.3. Review

All the tasks done by development.

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 4	Person: All	Filter	Reset	Print
▲ Collapse all	To Do			Doing		Done				
9 Movie Management: Suggestion TN Tien Nguyen State Done				25 Backend for Suggestion feature Unassigned State Done	26 Frontend for Suggestion feature Unassigned State Done					
8 Movie Management: Searching TN Tien Nguyen State Done				31 Backend for Searching feature Unassigned State Done	32 Frontend for Searching feature Unassigned State Done					
11 Movie Management: Get specific movie detail TN Tien Nguyen State Done				27 Backend for Get movie detail feature Unassigned State Done	28 Frontend for Get movie detail feature Unassigned State Done					
10 Rating Management: Create Rating TN Tien Nguyen State Done				29 Backend for Create rating feature Unassigned State Done	30 Frontend for Create rating feature Unassigned State Done					

#### 4.4. Retrospective

Perhaps if there are more requests from customers in the future, the Backend will have to change the framework to be able to customize it more deeply as well as master the logic thereby limiting security issues even though Django is supported by the community quite well. good but maybe one day there will be a zero-day vulnerability.

### V. Project termination

#### 1. Reason why project was terminated

After presenting the project progress to the Stakeholders, there were some points they were not satisfied with, and the Product Owner decided to immediately stop the project and start over.

Stakeholders: Professor Distefano Salvatore, Nguyen Minh Tien.

#### 2. Product Owner mistakes

The Sprints are not properly described: the product and sprint backlog are incomplete; description is missing as well as priority assignment and task identification.

There is no role such as Admin who can manage movies and users.

#### 3. Scrum Master mistakes

In the retrospective there are no burndown charts.

Django is good for rapid development but it's hard to deeply control the whole system.

And because of using Django, we are forced to use a RDBMS which is not good for machine learning model.

#### 4. Development Team mistakes

Final design is missing, there are no architectural diagrams such as component/deployment/class diagrams explain the architectural, its components and the overall software system organization.

There is no algorithm design, which can be represented by activity, sequence/collab or state chart diagram.

The testing phase is missing.

There is quite a long description of the recommendation system, but it should be contextualized with the requirements and associated with specific tasks identified in the sprint backlog.

### VI. Initial phase (2<sup>nd</sup>)

#### 1. Project members

- Product Owner: Nguyen Minh Tien
- Scrum Master: Nguyen Minh Tien
- Development Team member: Nguyen Minh Tien.

Taking on all three roles in SCRUM is not recommended because SCRUM and software development processes in general were born to solve problems related to independent interpersonal cooperation.

## 2. Project Ideal

The Ideal of Project is a website where users can get suggested which movies they might like. They will also be able to rate on a 5-point scale the movies they have watched. The system will use a machine learning algorithm to determine which movies users will like based on data given. The more movies they rated, the more accurate the suggested movies would be.

## 3. User Story

The system should be divided into three separate components: Database, Backend, Frontend.

Backend will connect to Database through a driver library, Backend and Frontend will communicate with each other via HTML Request.

Database is optional and does not have to be noticed. The only note is to ensure the security of the data.

The Backend must be developed as can release designs in the end explaining to the Stakeholders the algorithms and mechanisms therein.

The Frontend will need the following features for users: Create Account, Login, Logout, etc. for Authentication; After login successfully, user can access to these features:

- Search Movie  
There will be an input form and users can get a list of movies matched to what they type.
- Suggest Movie  
Recommendation system will give users top movies they might like.

After clicking on any movie on the movies list, users will turn to a page where they can see more details about the movie, here users can create rating or update rating and delete rating.'

## 4. Reuse from terminated Scrum

- Data extraction
- Frontend with ReactJS Library
- Movie Recommendation System Research

## 5. Technologies used

- Git for Source control.
- Draw.io for Designing.
- Postman for Unit test.
- Azure Dev for Scrum management and CICD.
- Microsoft Azure for Deployment.
- MongoDB instead of PostgreSQL for Database.

- Flask Framework instead of ~~Django Framework~~ for Backend.
- ReactJS library for Frontend.

## 6. Requirements Analysis

### 6.1. User requirements

The software will allow the user to perform the various functions:

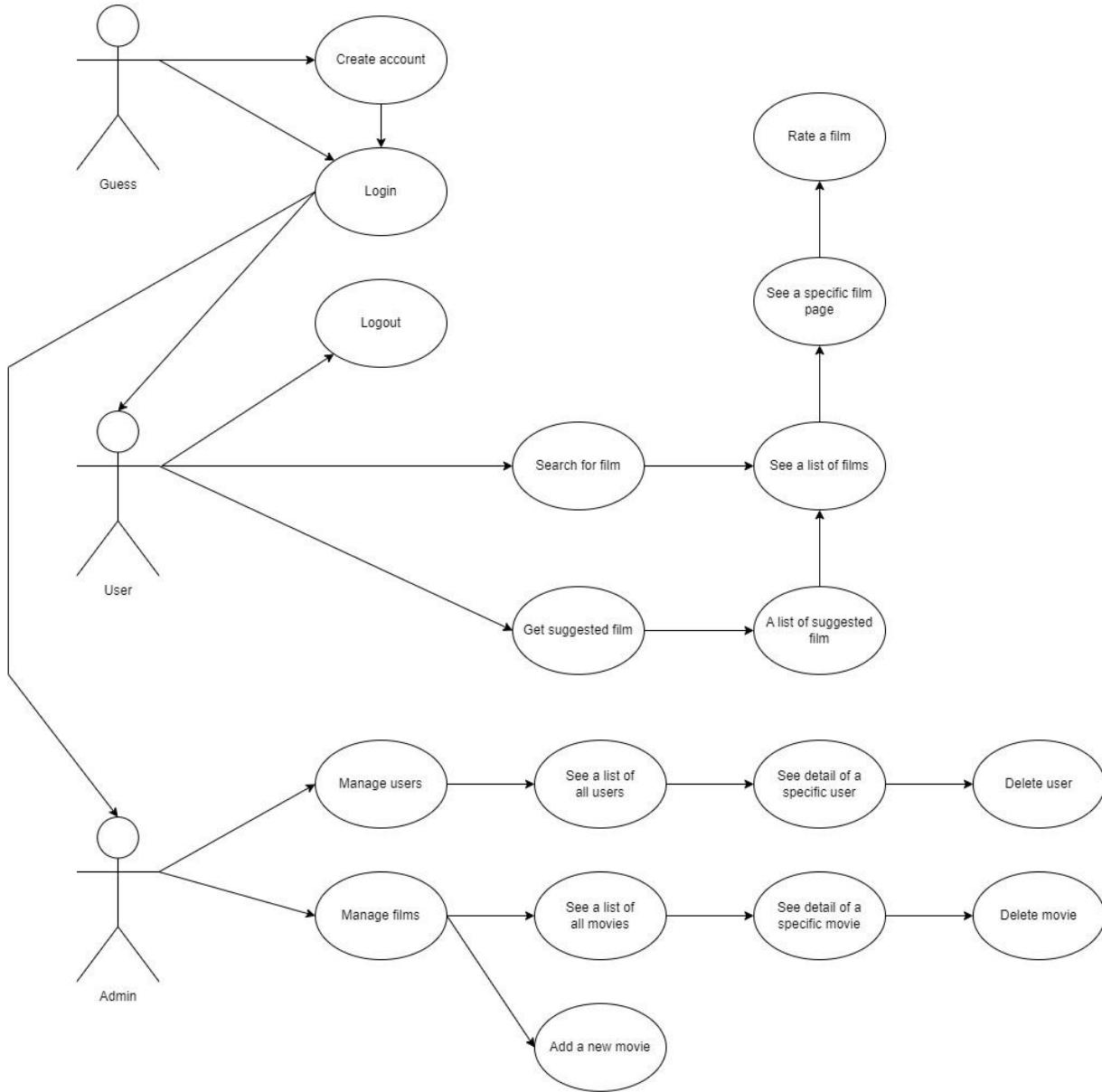
- Create Account
- Login
- Logout
- Search for movie
- Get suggested movie
- Get a specific movie detail
- Create new rating
- Read existed rating
- Update existed rating
- Delete existed rating

If the logged in user is an Admin, they will be able to perform the various functions:

- Login
- Logout
- Get a list of all users
- Get a specific user detail
- Delete a specific user
- Get a list of all movies
- Get a specific movie detail
- Delete a specific movie
- Add a new movie

## 6.2. Use case diagram

From the Functional Requirements analyzed in the previous section, I have come up with a use case diagram as follows:



## 6.3. Functional requirements

Functional requirements refer to the product features or functions that must be implemented by the development team to enable users to carry out their tasks. It is important to make these requirements clear for both the development team and stakeholders. Typically, functional requirements describe the system's behavior in specific circumstances. Below are functional requirements for software:

**FR.1: User Authenticate Management:**

- FR.1.1: User Registration
- FR.1.2: User Login
- FR.1.3: User Verification
- FR.1.4: User Logout
- FR.1.5: User Load

**FR.2: Movie Management:**

- FR.2.1: User Search Movie
- FR.2.2: User Get Movie Details

**FR.3: Rating Management:**

- FR.3.1: User Create Rating
- FR.3.2: User Read Rating
- FR.3.3: User Update Rating
- FR.3.4: User Delete Rating

**FR.4: Recommendation System:**

- FR.4.1: User Get Suggested Movies
- FR.4.2: User Get Prediction For A Movie

**FR.5: Admin Authenticate Management:**

- FR.5.1: Admin Login
- FR.5.2: Admin Logout
- FR.5.3: Admin Load

**FR.6: Admin Users Management:**

- FR.6.1: Admin Get All Users
- FR.6.2: Admin Get User Details
- FR.6.3: Admin Delete User

**FR.7: Admin Movies management:**

- FR.7.1: Admin Get All Movies
- FR.7.2: Admin Get Movie Details
- FR.7.3: Admin Add New Movie
- FR.7.4: Admin Delete Movie

#### 6.4. Non-functional requirements

Nonfunctional requirements, not related to the system functionality, rather define how the system should perform. Below are non-functional requirements for software:

NFR.1: Friendly Graphic User Interface.

NFR.2: Multiplatform compatible.

NFR.3: Fast response to user.

NFR.4: Movie cover should be available

NFR.5: User passwords should be encrypted.

#### 6.5. Domain requirements

Since the production will be a web application, domain requirements can include expectations related to performance, scalability and usability. In this project, the domain requirements are:

DR.1: Must have internet connection.

DR.2: Access through a Browser.

#### 6.6. Data Preparation

Since the data set for model training is MovieLens ml-1m: [MovieLens 1M Dataset | GroupLens](#)

The data is not so clean, so I need to prepare 3 things:

- Ratings with Timestamp

Last time, when I worked with PostgreSQL database, I needed to turn epoch timestamp to date data type to match SQL data type, but I no longer need it since I now use MongoDB.

- Movies with Poster URLs

This is needed for NFR.4: Movie covers should be available. BeautifulSoup4 helped me.

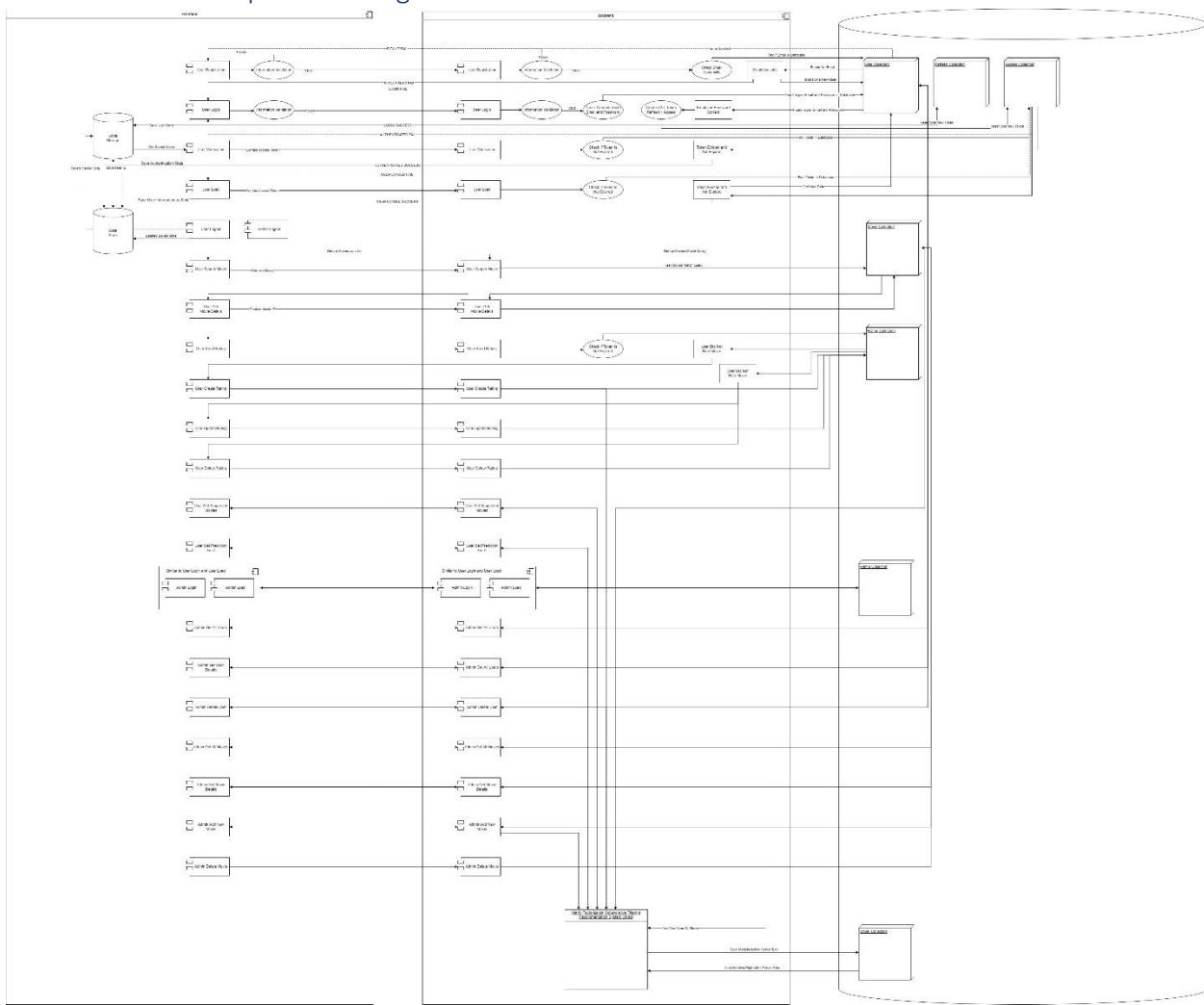
- Trained  $X, W, b$  and  $d$  to save starting time of Backend

All 4 elements can be exported as a csv file.

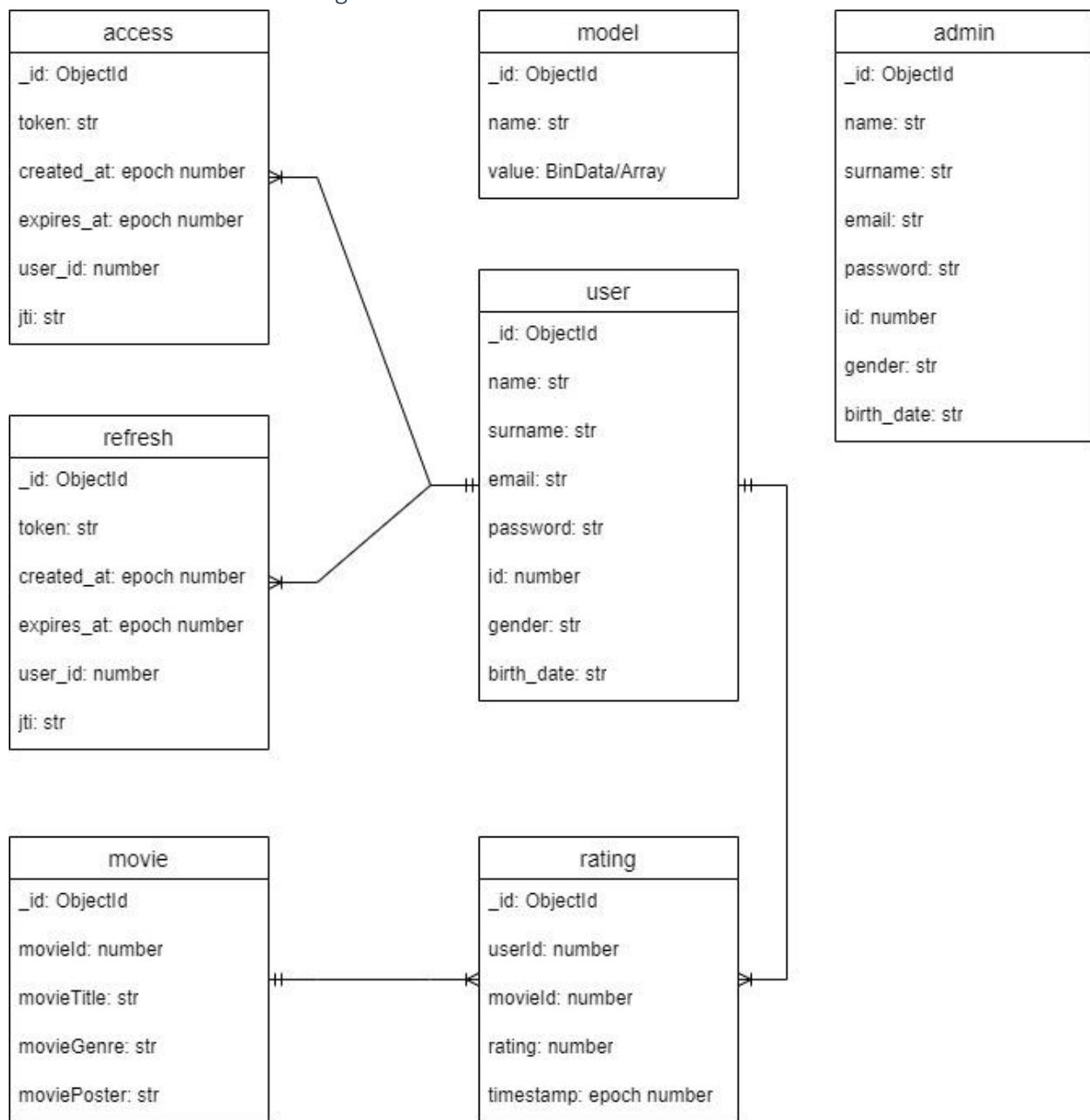
All the code for Data Preparation can be found in data\_preparation.ipynb.

## 7. Architectural Design

### 7.1. Components Diagram

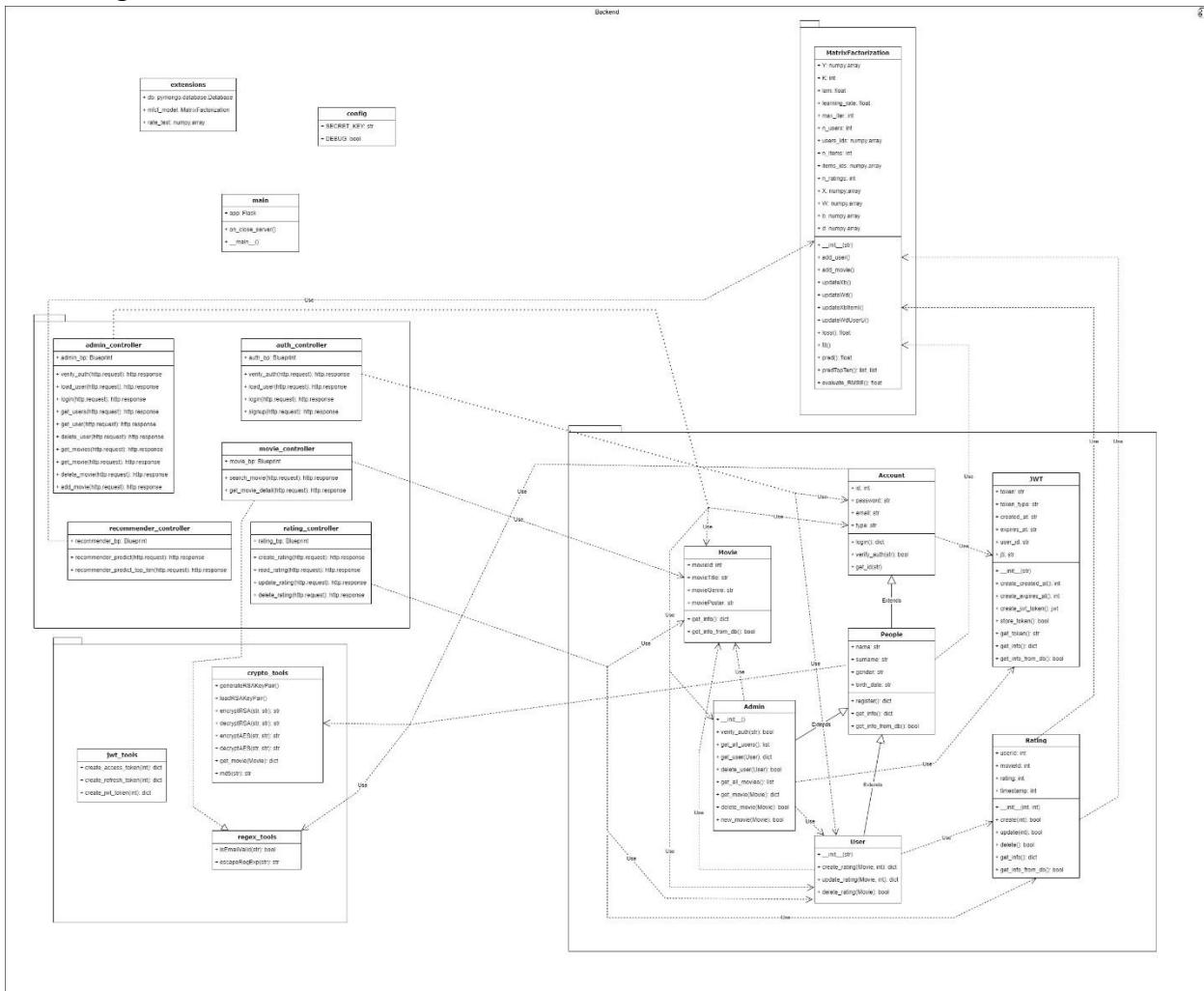


## 7.2. Database Design



### 7.3. Backend Design

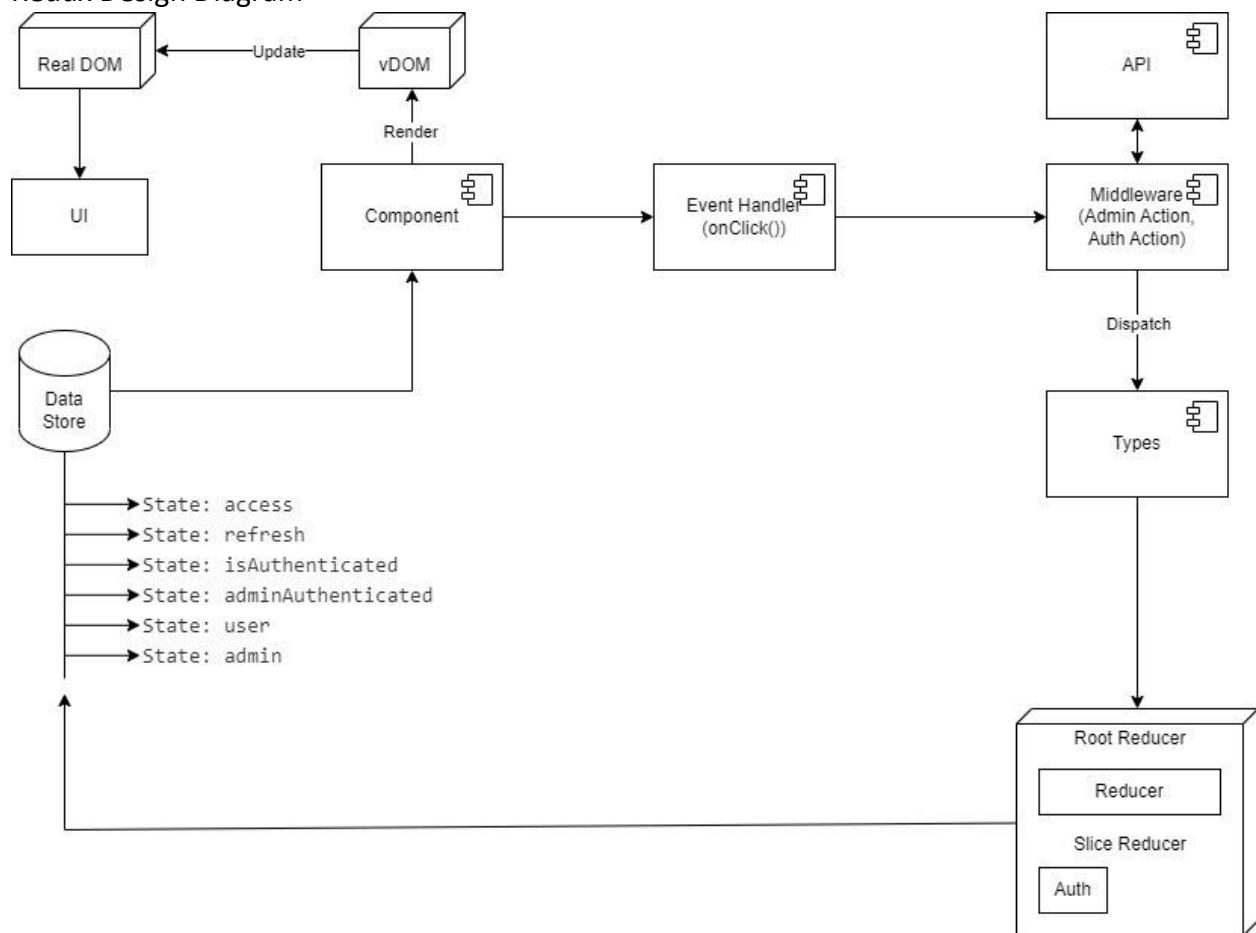
# Class Diagram



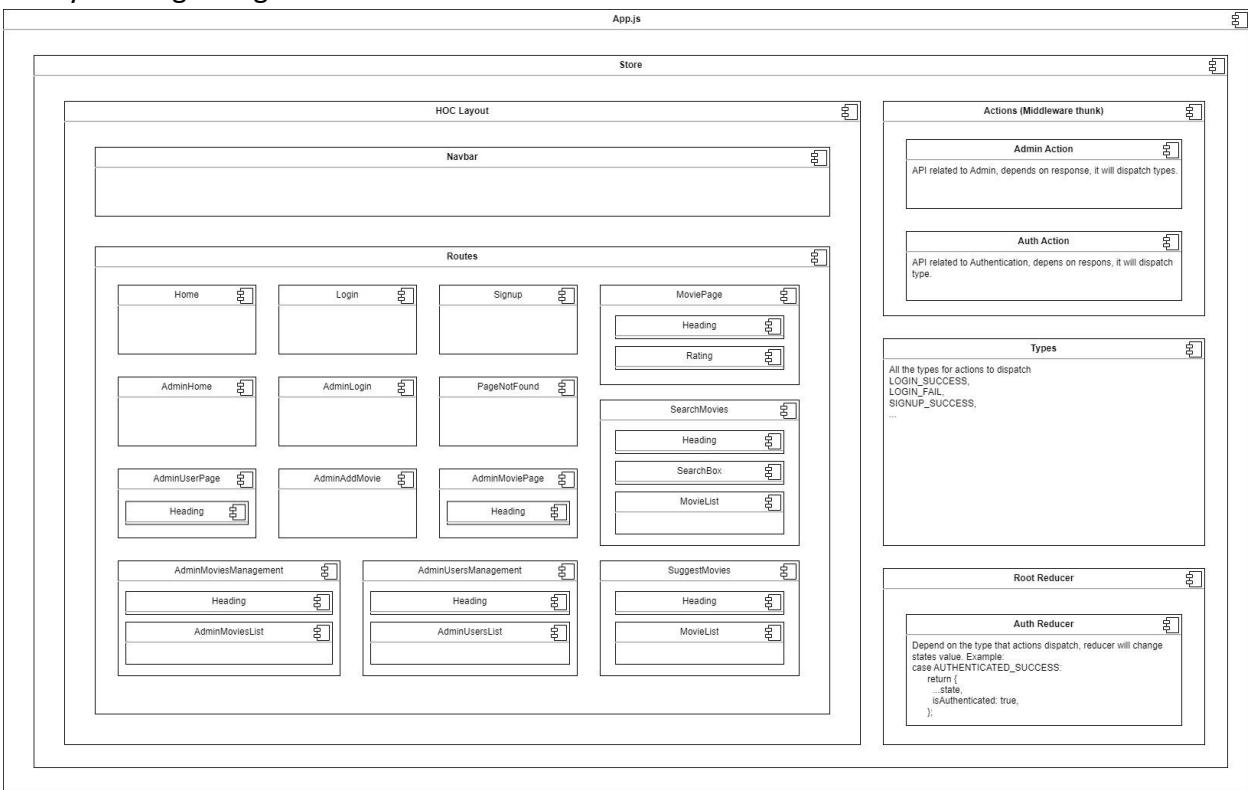
```
/MRSBB/Backend/
└── mrsbb-be/
    ├── main.py
    ├── config.py
    ├── Dockerfile
    ├── requirement
    └── controller/
        ├── admin.py
        ├── auth.py
        ├── movie.py
        ├── rating.py
        ├── recommender.py
        └── auth.py
    └── mlmodel/
        └── recommender.py
    └── model/
        ├── auth.py
        ├── auth.py
        ├── auth.py
        ├── auth.py
        ├── auth.py
        └── auth.py
    └── security/
        ├── auth.py
        ├── auth.py
        └── auth.py
    └── keys/
        ├── create.html
        ├── index.html
        └── update.html
└── mrsbb-venv/
```

## 7.4. Frontend Design

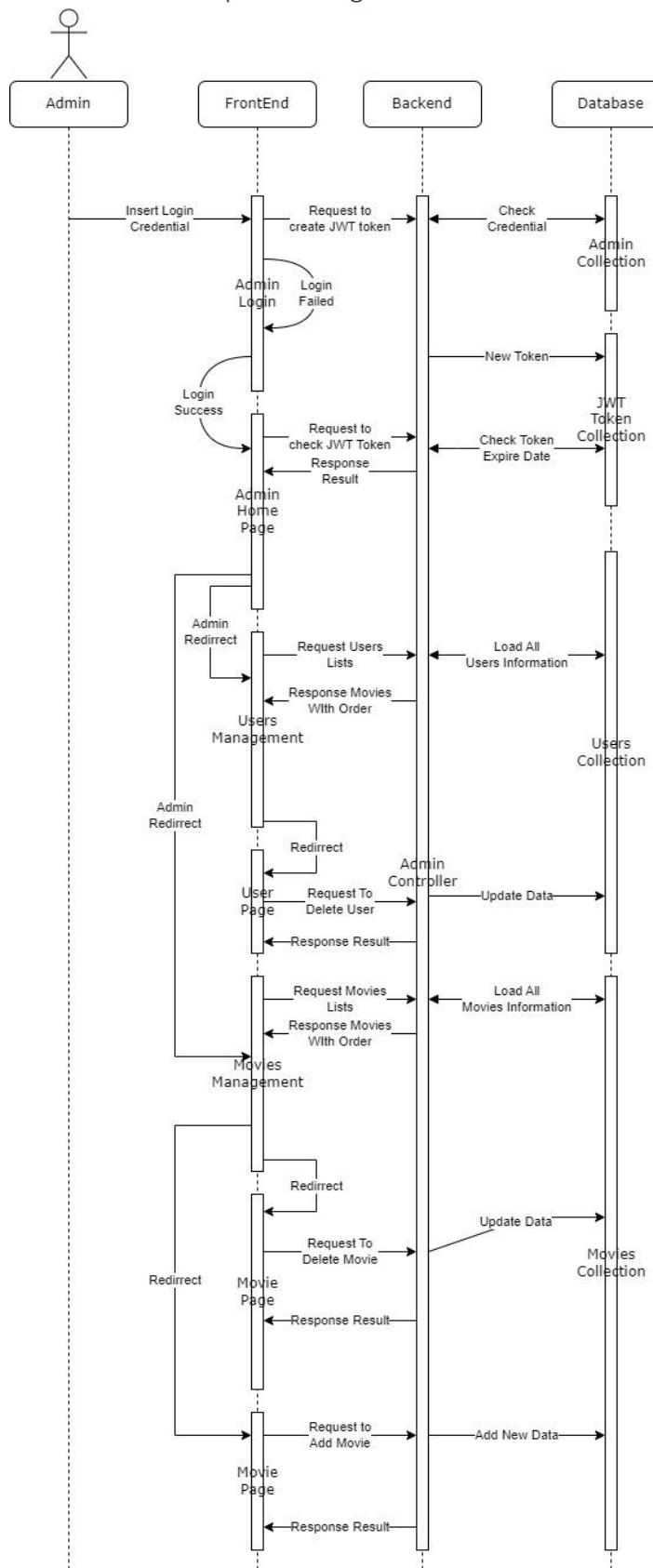
Redux Design Diagram

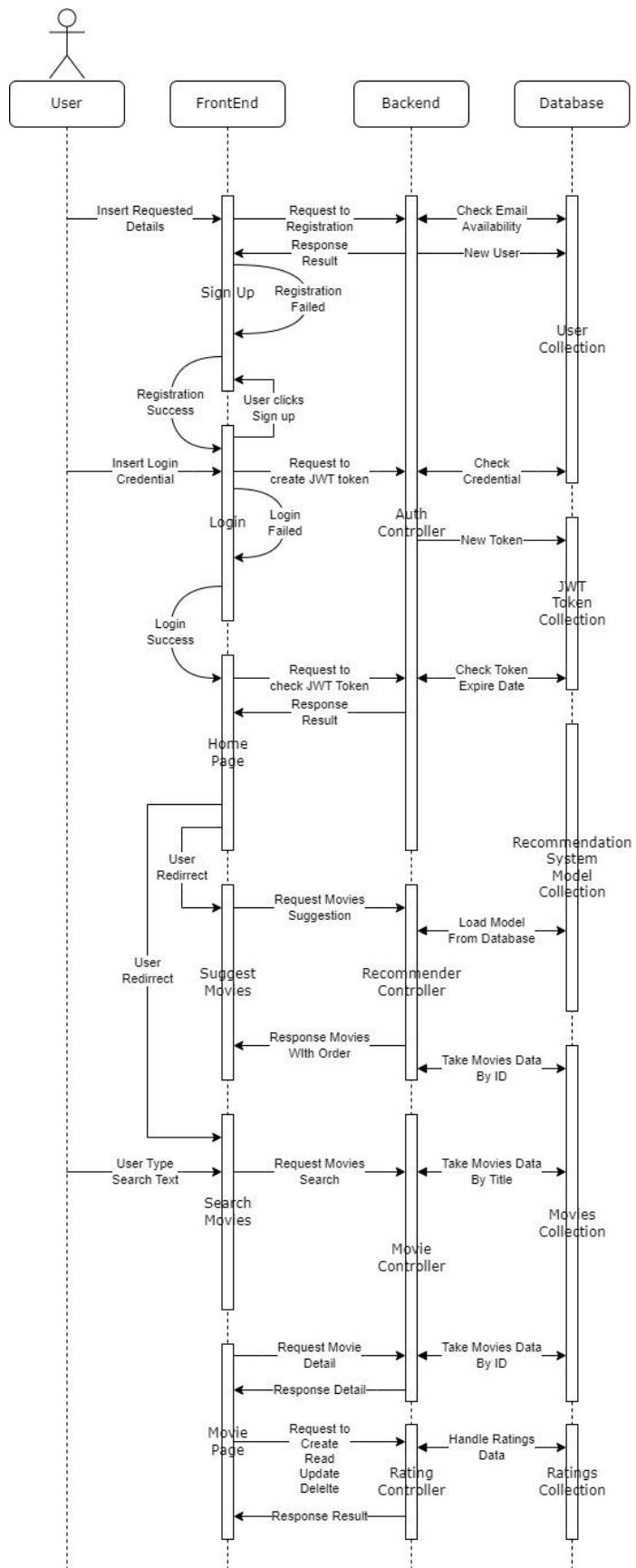


## UI Layer Design Diagram



## 7.5. Sequence Diagram





## 8. Backlog refinement

From the Requirements Analysis, as a Product Owner, I have refined the backlog as follows.

### 8.1. Epic

There are 3 epics corresponding to 3 components separated as stakeholders require in User Story

Backlog	Analytics	+ New Work Item	View as Board	...	Backlog items
Order	Priority	Work Item Type	Title		State
1	Epic	> 🏛 Database			New
1	Epic	> 🏛 Backend			New
1	Epic	▽ 🏛 Frontend			New

### 8.2. Feature

From 7 Functional Requirements we analyzed before, I created 14 corresponding features. Each Functional Requirement will need 2 features for Backend and Frontend. They are:

Backlog	Analytics	+ New Work Item	View as Board	Column Options	...
Order	Priority	Work Item Type	Title		State
1	Epic	▽ 🏛 Database			New
1	Product Backl...	> 📊 Database Design			New
1	Epic	▽ 🏛 Backend			New
1	Feature	> 🏆 User Authentication Management Backend			New
1	Feature	> 🏆 Movie Management Backend			New
1	Feature	> 🏆 Rating Management Backend			New
1	Feature	> 🏆 Recommendation System Backend			New
3	Feature	> 🏆 Admin Authenticate Management Backend			New
3	Feature	> 🏆 Admin Users Management Backend			New
3	Feature	> 🏆 Admin Movies Management Backend			New
1	Product Backl...	📊 Backend Architecture Design			New
1	Epic	▽ 🏛 Frontend			New
1	Feature	> 🏆 User Authenticate Management Frontend			New
1	Feature	> 🏆 Movie Management Frontend			New
1	Feature	> 🏆 Rating Management Frontend			New
1	Feature	> 🏆 Recommendation System Frontend			New
3	Feature	> 🏆 Admin Authenticate Management Frontend			New
3	Feature	> 🏆 Admin Users Management Frontend			New
3	Feature	> 🏆 Admin Movies Management Frontend			New
+	1	Product Backl...	📊 Frontend Architecture Design	...	New

### 8.3. Product Backlog

There are 48 product backlogs to finish feature and complete the system:

MRSBB System Team					Column Options
Order	Priority	Work Item Type	Title		State
1	Epic		Database		
1	Product Backl...		Database Design		
1	Epic		Backend		
1	Feature		User Authentication Management Backend		
1	Product Backl...		User Registration Backend		
1	Product Backl...		User Login Backend		
1	Product Backl...		User Verification Backend		
2	Product Backl...		User Load Backend		
1	Feature		Movie Management Backend		
1	Product Backl...		User Search Movie Backend		
1	Product Backl...		User Get Movie Details		
1	Feature		Rating Management Backend		
1	Product Backl...		User Create Rating Backend		
1	Product Backl...		User Read Rating Backend		
3	Product Backl...		User Update Rating Backend		
3	Product Backl...		User Delete Rating Backend		
1	Feature		Recommendation System Backend		
1	Product Backl...		Release Algorithm Design		
1	Product Backl...		User Get Suggested Movies Backend		
4	Product Backl...		User Get Prediction For A Movie		
3	Feature		Admin Authenticate Management Backend		
3	Product Backl...		Admin Login Backend		
2	Product Backl...		Admin Load Backend		

Order	Priority	Work Item Type	Title	State
3	Feature	Admin Users Management Backend	Admin Get All Users Backend	New
3	Product Backl...	Admin Get User Details Backend	Admin Delete User Backend	New
4	Product Backl...	Admin Movies Management Backend	Admin Get All Movies Backend	New
3	Product Backl...	Admin Get Movie Details Backend	Admin Add New Movie Backend	New
4	Product Backl...	Backend Architecture Design	Admin Delete Movie Backend	New
1	Product Backl...	Frontend	User Authenticate Management Frontend	New
1	Epic	User Registration Frontend	User Login Frontend	New
1	Feature	User Verification Frontend	User Logout Frontend	New
2	Product Backl...	User Load Frontend	Movie Management Frontend	New
1	Feature	User Search Movie Frontend	User Get Movie Details Frontend	New
1	Product Backl...	Rating Management Frontend	User Create Rating Frontend	New
1	Product Backl...	User Read Rating Frontend	User Update Rating Frontend	New
3	Product Backl...	User Delete Rating Frontend		New

Order	Priority	Work Item Type	Title	State
1	Feature	 Recommendation System Frontend		
1	Product Backl...	 User Get Suggested Movies Frontend		
4	Product Backl...	 User Get Prediction For A Movie Frontend		
3	Feature	 Admin Authenticate Management Frontend		
3	Product Backl...	 Admin Login Frontend		
3	Product Backl...	 Admin Logout Frontend		
2	Product Backl...	 Admin Load Frontend		
3	Feature	 Admin Users Management Frontend		
3	Product Backl...	 Admin Get All Users Frontend		
3	Product Backl...	 Admin Get User Details Details		
4	Product Backl...	 Admin Delete User Frontend		
3	Feature	 Admin Movies Management Frontend		
3	Product Backl...	 Admin Get All Movies Frontend		
3	Product Backl...	 Admin Get Movie Details Frontend		
3	Product Backl...	 Admin Add New Movie Frontend		
4	Product Backl...	 Admin Delete Movie Frontend		
1	Product Backl...	 Frontend Architecture Design		

#### 8.4. Priority assignment

There are 4 priority levels in this project.

1. Must do
2. High Priority
3. Second-level Priority
4. Low Priority

First, all epics will have priority 1 since they must have for system deployment.

To be able to do Priority Assignment work, let's go into the essence of the system:

The purpose of the project is to study the Recommendation System and apply it in practice. So, FR.4: Recommendation System will need priority 1. However, FR.4.2 will have priority 4 because in real life situations, the user usually will not need an exact prediction rating for a specific movie.

There are 3 must have components for Recommendation System, they are:

- Users

- Movies
- Ratings

The more Users have Ratings on Movies, the more Recommendation System increase accuracy, so FR.1, FR.2, FR.3 will have priority 1. However, FR.3.3 and FR.3.4 will have priority 3.

All FR related to Admin will have priority 3. However, FR.6.3, FR.7.4 will have priority 4.

Finally, all product backlogs related to Design will have priority 1.

All Priority appears in section 7.3 in the column Priority.

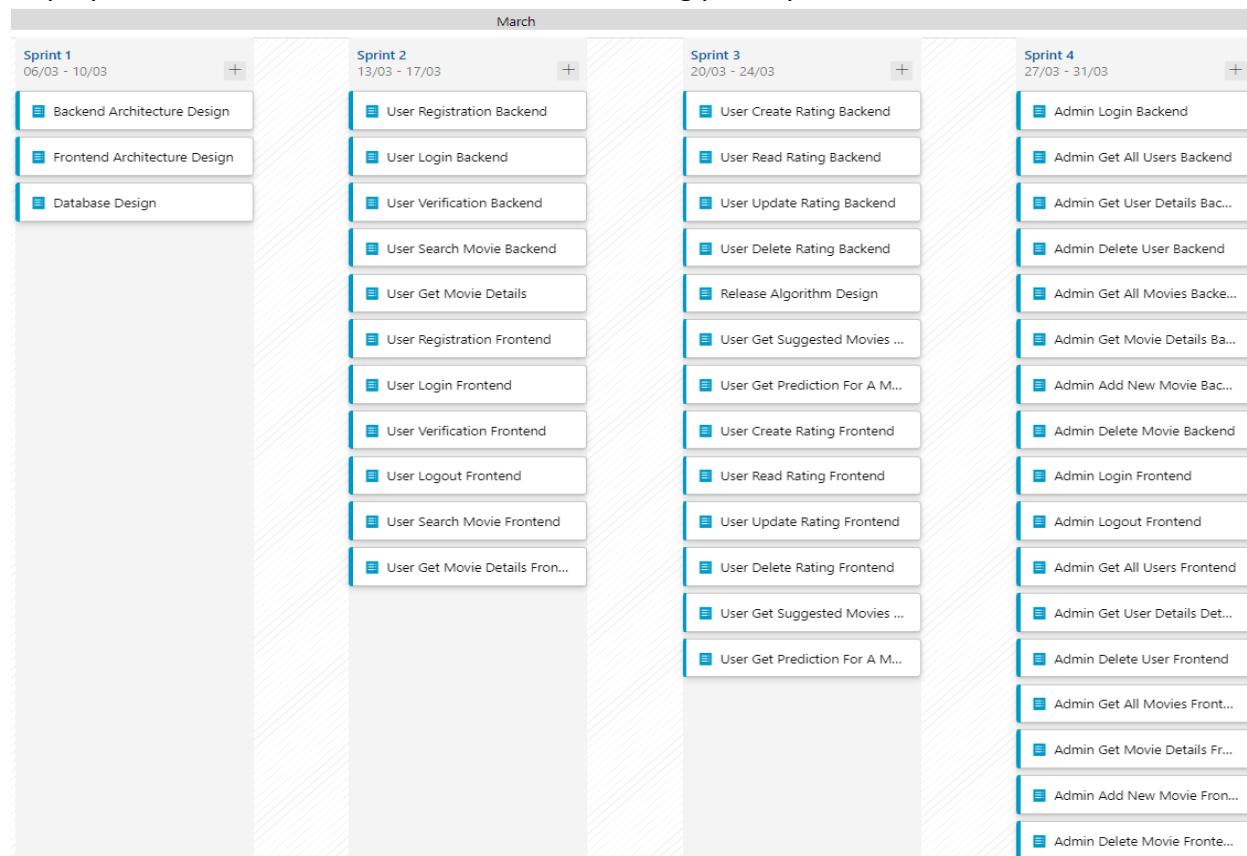
## 9. Project estimation (time and cost)

In this project, 1 Sprint is 5 working days long.

There are 4 Sprints:

1. Sprint 1 runs from 06/Mar/2023 to 10/Mar/2023 (5 working days).
2. Sprint 2 runs from 13/Mar/2023 to 17/Mar/2023 (5 working days).
3. Sprint 3 runs from 20/Mar/2023 to 24/Mar/2023 (5 working days).
4. Sprint 4 runs from 27/Mar/2023 to 31/Mar/2023 (5 working days).

Deployment cost: Azure Free Plan with 60mins using per day.



## VII. Sprint cycles (2<sup>nd</sup>)

### 1. Sprint 1

#### 1.1. Planning

In Scrum Planning, we will do task identification. The team identifies the tasks that need to be complete to deliver the product backlog items that are selected for the sprint.

In Sprint 1, tasks will be as below:

To Do		In Progress	Done
<div>+ Collapse all</div> <div><div><span>63 Backend Architecture Design</span>  Tien Nguyen State: New</div><div><span>109 Release Class Diagram</span>  Tien Nguyen State: To Do</div><div><span>112 Release Component Architecture Design Diagram</span>  Tien Nguyen State: To Do</div></div> <div>[+]</div>			
<div><div><span>64 Frontend Architecture Design</span>  Tien Nguyen State: New</div><div><span>110 Release Redux Design Diagram</span>  Tien Nguyen State: To Do</div><div><span>111 Release UI Layer Design Diagram</span>  Tien Nguyen State: To Do</div></div> <div>[+]</div>			
<div><div><span>65 Database Design</span>  Tien Nguyen State: New</div><div><span>113 Release MongoDB schema diagrams</span>  Tien Nguyen State: To Do</div></div> <div>[+]</div>			

#### 1.2. Implementation

All the designs are located in section IV. Architectural Design.

To Do		In Progress	Done
<div>+ Collapse all</div> <div><div><span>63 Backend Architecture Design</span>  Tien Nguyen State: New</div><div><span>109 Release Class Diagram</span>  Tien Nguyen State: In Progress</div><div><span>112 Release Component Architecture Design Diagram</span>  Tien Nguyen State: In Progress</div></div> <div>[+]</div>			
<div><div><span>64 Frontend Architecture Design</span>  Tien Nguyen State: New</div><div><span>110 Release Redux Design Diagram</span>  Tien Nguyen State: In Progress</div><div><span>111 Release UI Layer Design Diagram</span>  Tien Nguyen State: In Progress</div></div> <div>[+]</div>			
<div><div><span>65 Database Design</span>  Tien Nguyen State: New</div><div><span>113 Release MongoDB schema diagrams</span>  Tien Nguyen State: In Progress</div></div> <div>[+]</div>			

#### 1.3. Review

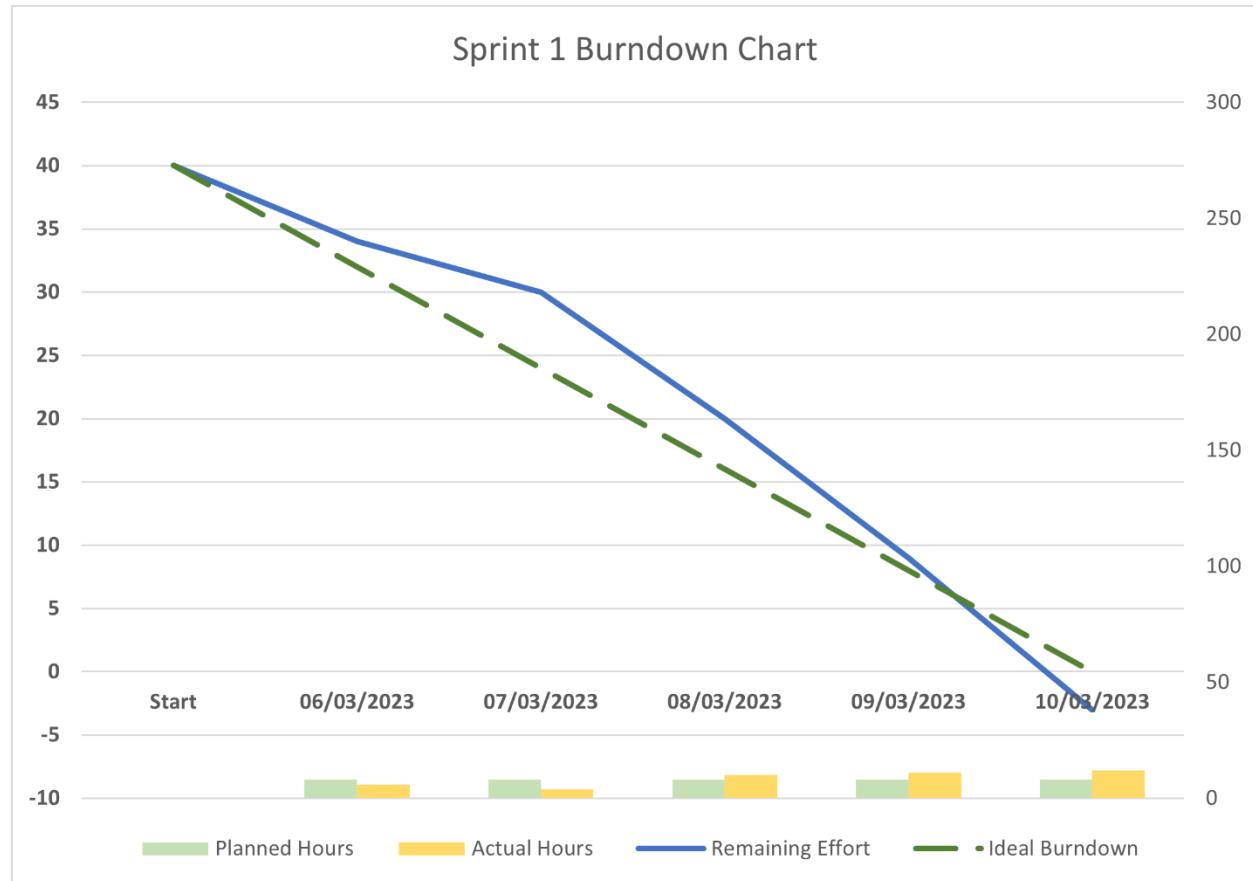
The most difficult task of a process is to have a good system design. With a good enough design after the Requirement Analysis process, the Development Team can accelerate quickly and

complete the Functional Requirements on time and perform as expected.

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 1	Person: All	Filter	Settings
Collapse all									
	To Do			In Progress		Done			
63 Backend Architecture Design Tien Nguyen State: Done					109 Release Class Diagram Tien Nguyen State: Done	112 Release Component Architecture Design Diagram Tien Nguyen State: Done			
64 Frontend Architecture Design Tien Nguyen State: Done					110 Release Redux Design Diagram Tien Nguyen State: Done	111 Release UI Layer Design Diagram Tien Nguyen State: Done			
65 Database Design Tien Nguyen State: Done					113 Release MongoDB schema diagrams Tien Nguyen State: Done				

#### 1.4. Retrospective

##### Release Burndown Chart for Sprint 1



Design takes more time than how long the team expected, at Day 1 and Day 2, the team couldn't find the right process to release design diagram. After Day 2, All the remaining days of the week, the team has to work overtime to release diagrams on time.

For more information, please refer to the file "Sprint\_1\_burndown\_chart.xlsx".

## 2. Sprint 2

### 2.1. Planning

In the Sprint 1, because the team did not manage time good enough, so they have to work overtime at some point while there are days that they did not take all 8 hours for works.

In Sprint 2, the team should take more attention for time management.

Task identified for Sprint 2:

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 2	Person: All	Filter	Settings
Collapse all									
				To Do	In Progress				Done
				<div><p>66 User Registration Backend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>123 Build API Route</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>124 Connect Function to Database</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
				<div><p>67 User Login Backend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>125 Build API Route</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>130 Connect Function to Database</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
				<div><p>68 User Verification Backend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>126 Build API Route</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>131 Connect Function to Database</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
				<div><p>69 User Search Movie Backend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>127 Build API Route</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>132 Connect Function to Database</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
Collapse all									
				To Do	In Progress				Done
				<div><p>70 User Get Movie Details Backend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>128 Build API Route</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>133 Connect Function to Database</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
				<div><p>86 User Registration Frontend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>135 UI Implementation</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>140 Build API Call</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
				<div><p>87 User Login Frontend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>136 UI Implementation</p><p>Tien Nguyen</p><p>State: To Do</p></div>	<div><p>141 Build API Call</p><p>Tien Nguyen</p><p>State: To Do</p></div>			
				<div><p>88 User Verification Frontend</p><p>Tien Nguyen</p><p>State: New</p></div>	<div><p>142 Build API Call</p><p>Tien Nguyen</p><p>State: To Do</p></div>				

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 2	Person: All	Filter	Settings
To Do									
<div> <span>89 User Logout Frontend</span>  <span>Tien Nguyen</span>   <span>State: ● New</span> </div> <div> <span>137 UI Implementation</span>  <span>143 Build API Call</span>   <span>State: ● To Do</span> </div> <div> <span>138 UI Implementation</span>  <span>144 Build API Call</span>   <span>State: ● To Do</span> </div>									
<div> <span>90 User Search Movie Frontend</span>  <span>Tien Nguyen</span>   <span>State: ● New</span> </div> <div> <span>139 UI Implementation</span>  <span>145 Build API Call</span>   <span>State: ● To Do</span> </div> <div> <span>140 UI Implementation</span>  <span>146 Build API Call</span>   <span>State: ● To Do</span> </div>									
<div> <span>118 User Load Backend</span>  <span>Tien Nguyen</span>   <span>State: ● New</span> </div> <div> <span>129 Build API Route</span>  <span>134 Connect Function to Database</span>   <span>State: ● To Do</span> </div> <div> <span>141 Build API Call</span>  <span>147 Build API Call</span>   <span>State: ● To Do</span> </div>									
<div> <span>121 User Load Frontend</span>  <span>Tien Nguyen</span>   <span>State: ● New</span> </div> <div> <span>148 Build API Call</span>  <span>149 Build API Call</span>   <span>State: ● To Do</span> </div>									

## 2.2. Implementation

During the implementation, the team also did Unit Test using Postman after completing a feature but due to time rush, they were not able to complete the document for that. We will demonstrate testing in the following system testing section.

## 2.3. Review

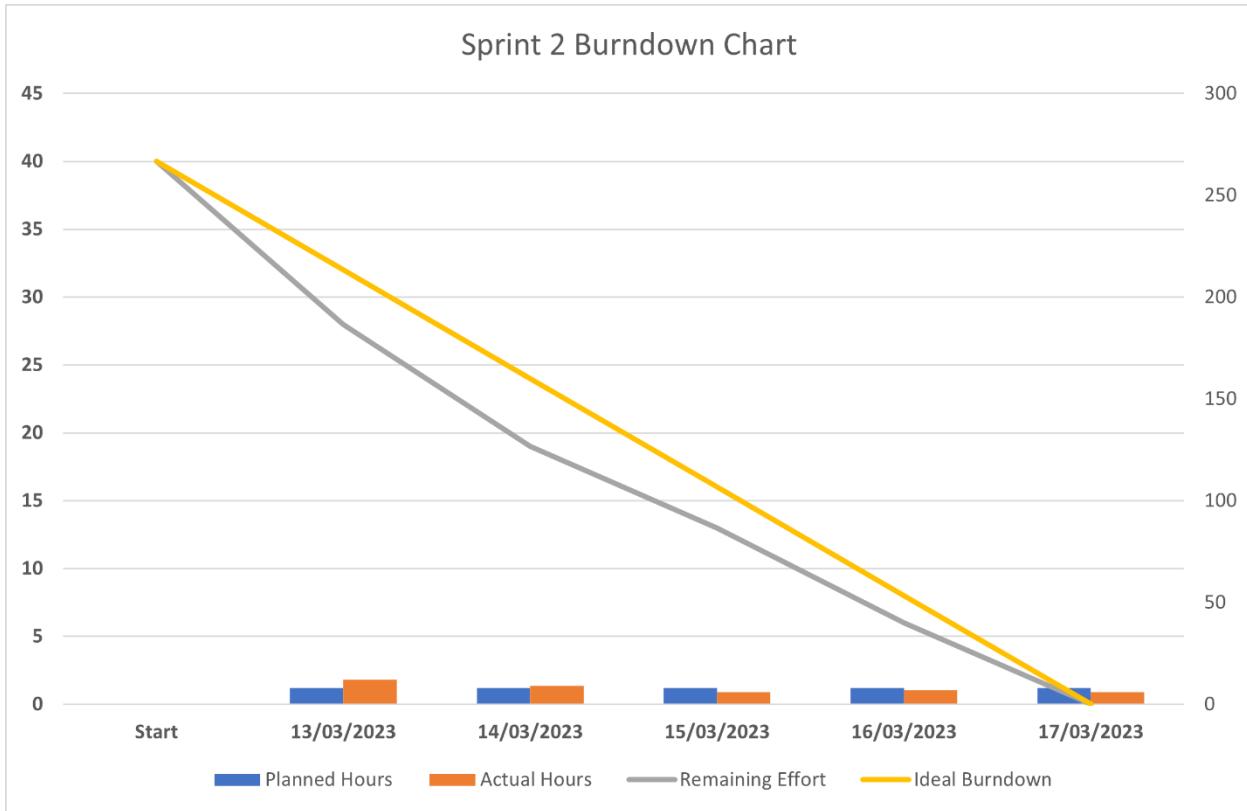
Thanks to the design of the system in Sprint 1, the implementation in Sprint 2 can run more smoothly, there are no backlogs remaining.

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 2	Person: All	Filter	Settings
To Do									
<div> <span>66 User Registration Backend</span>  <span>Tien Nguyen</span>   <span>State: ● Done</span> </div> <div> <span>67 User Login Backend</span>  <span>Tien Nguyen</span>   <span>State: ● Done</span> </div> <div> <span>68 User Verification Backend</span>  <span>Tien Nguyen</span>   <span>State: ● Done</span> </div> <div> <span>69 User Search Movie Backend</span>  <span>Tien Nguyen</span>   <span>State: ● New</span> </div> <div> <span>70 User Get Movie Details Backend</span>  <span>Tien Nguyen</span>   <span>State: ● Done</span> </div>									
<div> <span>123 Build API Route</span>  <span>124 Connect Function to Database</span>   <span>State: ● Done</span> </div> <div> <span>125 Build API Route</span>  <span>130 Connect Function to Database</span>   <span>State: ● Done</span> </div> <div> <span>126 Build API Route</span>  <span>131 Connect Function to Database</span>   <span>State: ● Done</span> </div> <div> <span>127 Build API Route</span>  <span>132 Connect Function to Database</span>   <span>State: ● Done</span> </div> <div> <span>128 Build API Route</span>  <span>133 Connect Function to Database</span>   <span>State: ● Done</span> </div>									

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 2	Person: All	Filter
						Sprint 2 - Iteration 1		
		Planning		Development		Testing		Deployment
<span>86 User Registration Frontend</span>  Tien Nguyen State: Done						<span>135 UI Implementation</span>  Done		<span>140 Build API Call</span>  Done
<span>87 User Login Frontend</span>  Tien Nguyen State: Done						<span>136 UI Implementation</span>  Done		<span>141 Build API Call</span>  Done
<span>88 User Verification Frontend</span>  Tien Nguyen State: Done						<span>142 Build API Call</span>  Done		
<span>89 User Logout Frontend</span>  Tien Nguyen State: Done						<span>137 UI Implementation</span>  Done		<span>143 Build API Call</span>  Done
<span>90 User Search Movie Frontend</span>  Tien Nguyen State: Done						<span>138 UI Implementation</span>  Done		<span>144 Build API Call</span>  Done

## 2.4. Retrospective

### Release Burndown Chart:



## 3. Sprint 3

### 3.1. Planning

#### Task identification for Sprint 3:

The Taskboard interface shows the following backlog items for Sprint 3:

Category	Item ID	Description	State	Owner
To Do	71	User Create Rating Backend	New	Tien Nguyen
	147	Build API Route	To Do	
	154	Connect Function to Database	To Do	
	+ [Add]			
To Do	72	User Read Rating Backend	New	Tien Nguyen
	148	Build API Route	To Do	
	155	Connect Function to Database	To Do	
	+ [Add]			
To Do	73	User Update Rating Backend	New	Tien Nguyen
	149	Build API Route	To Do	
	156	Connect Function to Database	To Do	
	+ [Add]			
To Do	74	User Delete Rating Backend	New	Tien Nguyen
	150	Build API Route	To Do	
	157	Connect Function to Database	To Do	
	+ [Add]			

The image consists of two vertically stacked screenshots of a Microsoft Azure DevOps Taskboard. Both screenshots show a board with four columns: To Do, In Progress, and Done, with an additional column header 'In Progress' appearing between the first two.

**Top Screenshot Data:**

Column	Work Item ID	Description	Status
To Do	151	Release Document About Recommendation System	To Do
To Do	152	Build API Route	To Do
To Do	153	Build API Route	To Do
To Do	160	UI Implementation	To Do
To Do	166	Build API Call	To Do

**Bottom Screenshot Data:**

Column	Work Item ID	Description	Status
To Do	161	UI Implementation	To Do
To Do	167	Build API Call	To Do
To Do	162	UI Implementation	To Do
To Do	168	Build API Call	To Do
To Do	163	UI Implementation	To Do
To Do	169	Build API Call	To Do
To Do	164	UI Implementation	To Do
To Do	170	Build API Call	To Do
To Do	165	UI Implementation	To Do
To Do	171	Build API Call	To Do

### 3.2. Implementation

I reuse most of the parts that have been programmed from Scrum before, of course, it will have to be slightly modified due to changes in the Database and API structure in the Backend.

### 3.3. Review

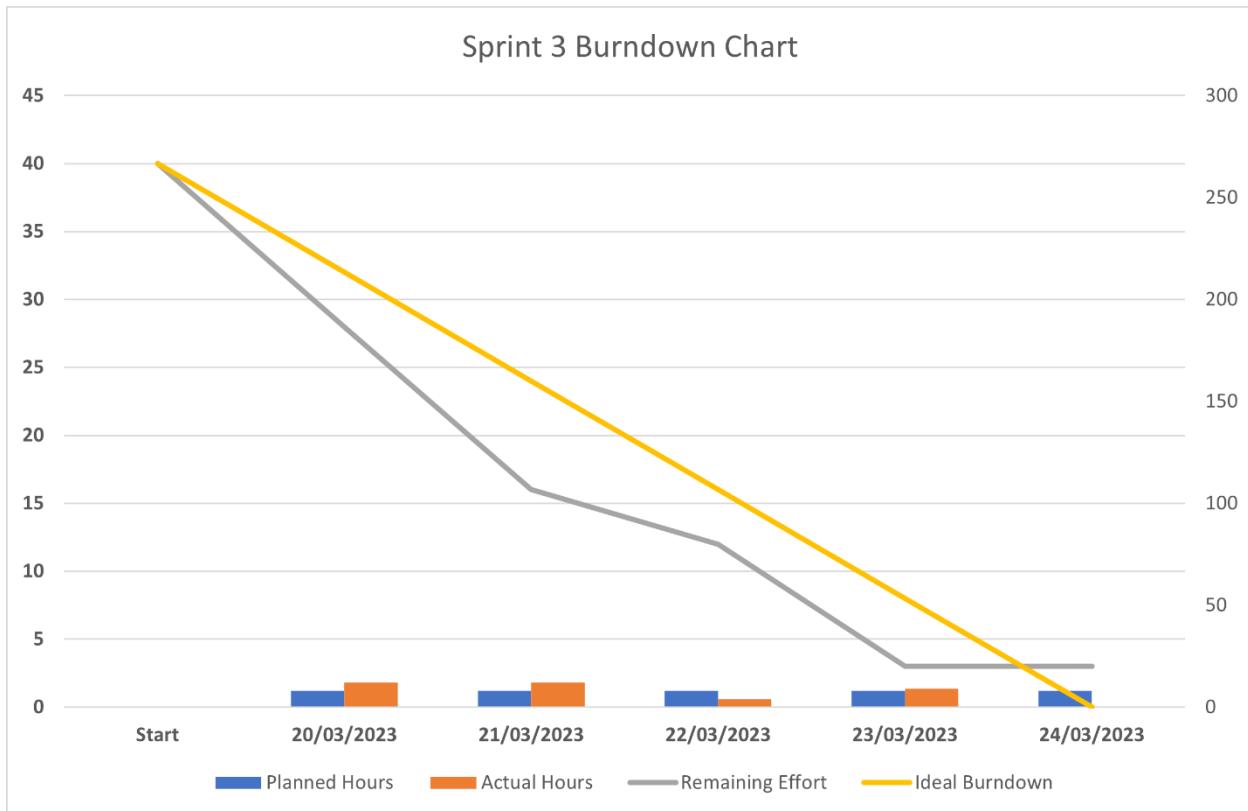
Implementation in Sprint 3 is quite smooth because most of the backlogs in Sprint 3 can be reused from the previous Scrum. However, there is still an unfinished backlog.

Taskboard		Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 3	Person: All	Filter
		To Do		In Progress		Done			
<span data-bbox="204 261 221 270">+</span> <span data-bbox="228 261 328 278">71 User Create Rating Backend</span> <span data-bbox="228 278 328 297">Tien Nguyen</span> <span data-bbox="228 297 328 310">State</span> <span data-bbox="228 310 328 323">Done</span>	<span data-bbox="376 261 396 270">+</span>				<span data-bbox="1122 261 1140 270">+</span> <span data-bbox="1147 261 1222 278">147 Build API Route</span> <span data-bbox="1147 278 1222 297"></span> <span data-bbox="1147 297 1222 310">State</span> <span data-bbox="1147 310 1222 323">Done</span>	<span data-bbox="1321 261 1339 270">+</span> <span data-bbox="1346 261 1421 278">154 Connect Function to Database</span> <span data-bbox="1346 278 1421 297"></span> <span data-bbox="1346 297 1421 310">State</span> <span data-bbox="1346 310 1421 323">Done</span>			
<span data-bbox="204 335 221 344">+</span> <span data-bbox="228 335 328 354">72 User Read Rating Backend</span> <span data-bbox="228 354 328 373">Tien Nguyen</span> <span data-bbox="228 373 328 386">State</span> <span data-bbox="228 386 328 399">Done</span>	<span data-bbox="376 335 396 344">+</span>				<span data-bbox="1122 335 1140 344">+</span> <span data-bbox="1147 335 1222 354">148 Build API Route</span> <span data-bbox="1147 354 1222 373"></span> <span data-bbox="1147 373 1222 386">State</span> <span data-bbox="1147 386 1222 399">Done</span>	<span data-bbox="1321 335 1339 344">+</span> <span data-bbox="1346 335 1421 354">155 Connect Function to Database</span> <span data-bbox="1346 354 1421 373"></span> <span data-bbox="1346 373 1421 386">State</span> <span data-bbox="1346 386 1421 399">Done</span>			
<span data-bbox="204 411 221 420">+</span> <span data-bbox="228 411 328 428">73 User Update Rating Backend</span> <span data-bbox="228 428 328 447">Tien Nguyen</span> <span data-bbox="228 447 328 460">State</span> <span data-bbox="228 460 328 473">Done</span>	<span data-bbox="376 411 396 420">+</span>				<span data-bbox="1122 411 1140 420">+</span> <span data-bbox="1147 411 1222 428">149 Build API Route</span> <span data-bbox="1147 428 1222 447"></span> <span data-bbox="1147 447 1222 460">State</span> <span data-bbox="1147 460 1222 473">Done</span>	<span data-bbox="1321 411 1339 420">+</span> <span data-bbox="1346 411 1421 428">156 Connect Function to Database</span> <span data-bbox="1346 428 1421 447"></span> <span data-bbox="1346 447 1421 460">State</span> <span data-bbox="1346 460 1421 473">Done</span>			
<span data-bbox="204 485 221 494">+</span> <span data-bbox="228 485 328 504">74 User Delete Rating Backend</span> <span data-bbox="228 504 328 523">Tien Nguyen</span> <span data-bbox="228 523 328 536">State</span> <span data-bbox="228 536 328 549">Done</span>	<span data-bbox="376 485 396 494">+</span>				<span data-bbox="1122 485 1140 494">+</span> <span data-bbox="1147 485 1222 504">150 Build API Route</span> <span data-bbox="1147 504 1222 523"></span> <span data-bbox="1147 523 1222 536">State</span> <span data-bbox="1147 536 1222 549">Done</span>	<span data-bbox="1321 485 1339 494">+</span> <span data-bbox="1346 485 1421 504">157 Connect Function to Database</span> <span data-bbox="1346 504 1421 523"></span> <span data-bbox="1346 523 1421 536">State</span> <span data-bbox="1346 536 1421 549">Done</span>			
<span data-bbox="204 561 221 570">+</span> <span data-bbox="228 561 328 578">75 Release Algorithm Design</span> <span data-bbox="228 578 328 597">Tien Nguyen</span> <span data-bbox="228 597 328 610">State</span> <span data-bbox="228 610 328 623">Done</span>	<span data-bbox="376 561 396 570">+</span>				<span data-bbox="1122 561 1140 570">+</span> <span data-bbox="1147 561 1222 578">151 Release Document About Recommendation System</span> <span data-bbox="1147 578 1222 597"></span> <span data-bbox="1147 597 1222 610">State</span> <span data-bbox="1147 610 1222 623">Done</span>				

Taskboard	Backlog	Capacity	Analytics	+ New Work Item	Column Options	Sprint 3	Person: All	Y
A Collapse all	To Do	In Progress	Done					
76 User Get Suggested Movies Backend Tien Nguyen State: Done			152 Build API Route 158 Connect Function to Recommendation System State: Done					
77 User Get Prediction For A Movie Backend Tien Nguyen State: Done			153 Build API Route 159 Connect Function to Recommendation System State: Done					
92 User Create Rating Frontend Tien Nguyen State: Done			160 UI Implementation 166 Build API Call State: Done					
93 User Read Rating Frontend Tien Nguyen State: Done			161 UI Implementation 167 Build API Call State: Done					
94 User Update Rating Frontend Tien Nguyen State: Done			162 UI Implementation 168 Build API Call State: Done					
95 User Delete Rating Frontend Tien Nguyen State: Done			163 UI Implementation 169 Build API Call State: Done					
96 User Get Suggested Movies Frontend Tien Nguyen State: Done			164 UI Implementation 170 Build API Call State: Done					
97 User Get Prediction For A Movie Frontend Tien Nguyen State: New	165 UI Implementation State: To Do	171 Build API Call State: To Do						

### 3.4. Retrospective

Release Burndown Chart:



Looking at the Burndown Chart, we can see that the work was completed earlier than planned quite early in the beginning of the Sprint but by the end did not complete 100% of the Backlogs. Is there a problem with the team? This will be discussed in the Planning section of Sprint 4.

## 4. Sprint 4

### 4.1. Planning

Sprint 4 is the place with the most Backlogs in all 4 Sprints, but we still have to add Backlog not completed in Sprint 3. This backlog has Priority 4 so we'll still do it last if Sprint 4 doesn't have enough time.

Task identified for Sprint 4 as below.

Taskboard Backlog Capacity Analytics + New Work Item Column Options

Sprint 4 Person: All

To Do		In Progress	Done
<a href="#">#78 Admin Login Backend</a> Tien Nguyen State: New	<a href="#">#172 Build API Route</a> <a href="#">#181 Connect Function to Database</a> State: To Do		
<a href="#">#79 Admin Get All Users Backend</a> Tien Nguyen State: New	<a href="#">#173 Build API Route</a> <a href="#">#182 Connect Function to Database</a> State: To Do		
<a href="#">#80 Admin Get User Details Backend</a> Tien Nguyen State: New	<a href="#">#174 Build API Route</a> <a href="#">#183 Connect Function to Database</a> State: To Do		
<a href="#">#81 Admin Delete User Backend</a> Tien Nguyen State: New	<a href="#">#175 Build API Route</a> <a href="#">#184 Connect Function to Database</a> State: To Do		
<a href="#">#82 Admin Get All Movies Backend</a> Tien Nguyen State: New	<a href="#">#176 Build API Route</a> <a href="#">#185 Connect Function to Database</a> State: To Do		

Taskboard Backlog Capacity Analytics + New Work Item Column Options

Sprint 4 Person: All

To Do		In Progress	Done
<a href="#">#83 Admin Get Movie Details Backend</a> Tien Nguyen State: New	<a href="#">#177 Build API Route</a> <a href="#">#186 Connect Function to Database</a> State: To Do		
<a href="#">#84 Admin Add New Movie Backend</a> Tien Nguyen State: New	<a href="#">#178 Build API Route</a> <a href="#">#187 Connect Function to Recommendation System</a> State: To Do		
<a href="#">#85 Admin Delete Movie Backend</a> Tien Nguyen State: New	<a href="#">#179 Build API Route</a> <a href="#">#188 Connect Function to Database</a> State: To Do		
<a href="#">#98 Admin Login Frontend</a> Tien Nguyen State: New	<a href="#">#190 UI Implementation</a> <a href="#">#199 Build API Call</a> State: To Do		
<a href="#">#99 Admin Logout Frontend</a> Tien Nguyen State: New	<a href="#">#191 UI Implementation</a> State: To Do		

Taskboard Backlog Capacity Analytics | New Work Item Column Options

Sprint 4 | Person: All | Filter | Export | Refresh

To Do		In Progress	Done
<a href="#">100 Admin Get All Users Frontend</a> Tien Nguyen State: New	<a href="#">192 UI Implementation</a> <a href="#">200 Build API Call</a> State: To Do		
<a href="#">101 Admin Get User Details Details Frontend</a> Tien Nguyen State: New	<a href="#">193 UI Implementation</a> <a href="#">201 Build API Call</a> State: To Do		
<a href="#">102 Admin Delete User Frontend</a> Tien Nguyen State: New	<a href="#">194 UI Implementation</a> <a href="#">202 Build API Call</a> State: To Do		
<a href="#">103 Admin Get All Movies Frontend</a> Tien Nguyen State: New	<a href="#">195 UI Implementation</a> <a href="#">203 Build API Call</a> State: To Do		
<a href="#">104 Admin Get Movie Details Frontend</a> Tien Nguyen State: New	<a href="#">196 UI Implementation</a> <a href="#">204 Build API Call</a> State: To Do		
<a href="#">105 Admin Add New Movie Frontend</a> Tien Nguyen State: New	<a href="#">197 UI Implementation</a> <a href="#">205 Build API Call</a> State: To Do		
<a href="#">106 Admin Delete Movie Frontend</a> Tien Nguyen State: New	<a href="#">198 UI Implementation</a> <a href="#">206 Build API Call</a> State: To Do		
<a href="#">119 Admin Load Backend</a> Tien Nguyen State: New	<a href="#">180 Build API Route</a> <a href="#">189 Connect Function to Database</a> State: To Do		
<a href="#">120 Admin Load Frontend</a> Tien Nguyen State: New	<a href="#">207 Build API Call</a> State: To Do		

## 4.2. Implementation

In Sprint 4, there are 3 Functional Requirements implemented are:

FR.5: Admin Authenticate Management

FR.6: Admin Users Management

FR.7: Admin Movies management

FR.5 is pretty similar to FR.1: User Authenticate Management so it will not take long time to implement.

The other two FRs (6 and 7) also don't take too much time because there's no need to focus too much on UX when the UI just needs to be usable, not too beautiful.

### 4.3. Review

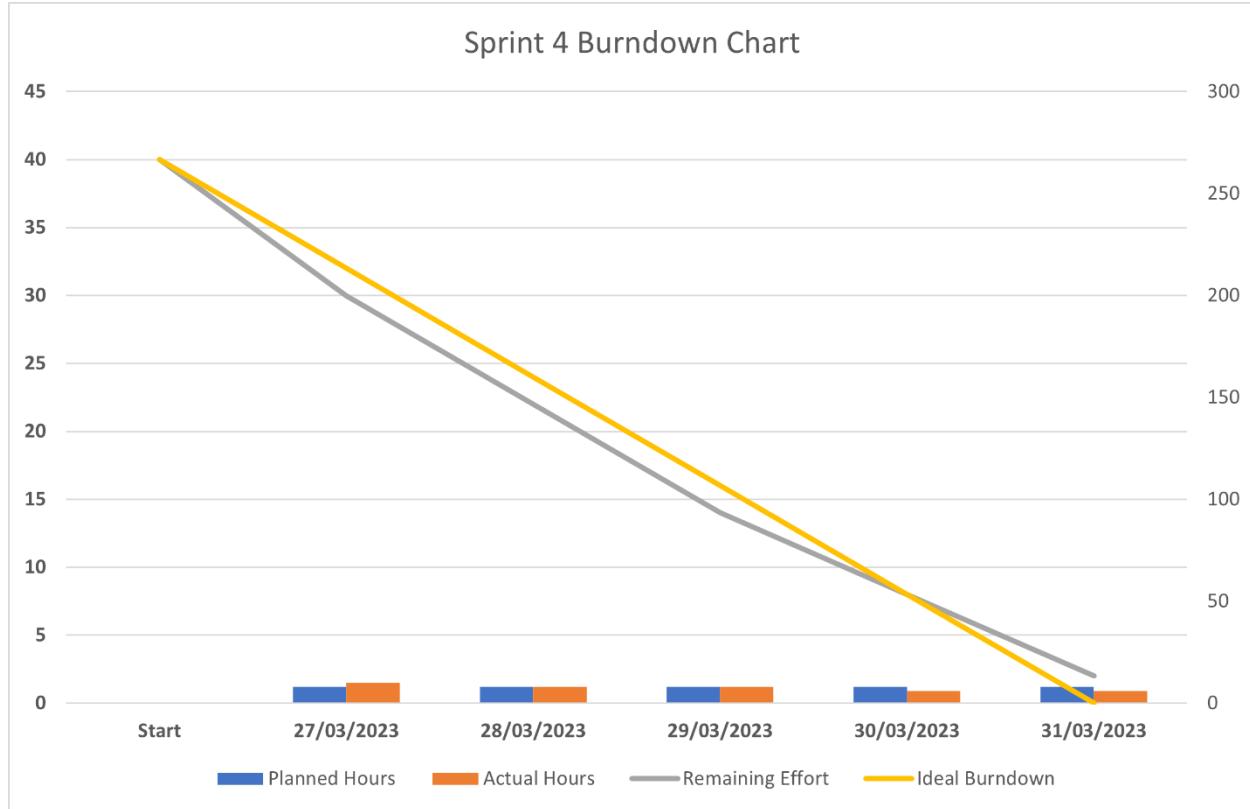
While all the Backlogs planned for Sprint 4 have been completed, the Backlogs from Sprint 3 have not yet been completed. However, the product can still be released because this Backlog is only in Priority 4, it's okay to have it or not.

To Do	In Progress	Done
78 Admin Login Backend Tien Nguyen State: Done		172 Build API Route 181 Connect Function to Database State: Done
79 Admin Get All Users Backend Tien Nguyen State: Done		173 Build API Route 182 Connect Function to Database State: Done
80 Admin Get User Details Backend Tien Nguyen State: Done		174 Build API Route 183 Connect Function to Database State: Done
81 Admin Delete User Backend Tien Nguyen State: Done		175 Build API Route 184 Connect Function to Database State: Done
82 Admin Get All Movies Backend Tien Nguyen State: Done		176 Build API Route 185 Connect Function to Database State: Done
83 Admin Get Movie Details Backend Tien Nguyen State: Done		177 Build API Route 186 Connect Function to Database State: Done

To Do	In Progress	Done	
84 Admin Add New Movie Backend Tien Nguyen State: Done		178 Build API Route 187 Connect Function to Recommendation System State: Done	
85 Admin Delete Movie Backend Tien Nguyen State: Done		179 Build API Route 188 Connect Function to Database State: Done	
97 User Get Prediction For A Movie Frontend Tien Nguyen State: New	165 UI Implementation State: To Do	171 Build API Call State: To Do	
98 Admin Login Frontend Tien Nguyen State: Done		190 UI Implementation 199 Build API Call State: Done	
99 Admin Logout Frontend Tien Nguyen State: Done		191 UI Implementation State: Done	
100 Admin Get All Users Frontend Tien Nguyen State: Done		192 UI Implementation 200 Build API Call State: Done	

#### 4.4. Retrospective

#### Release Burndown Chart

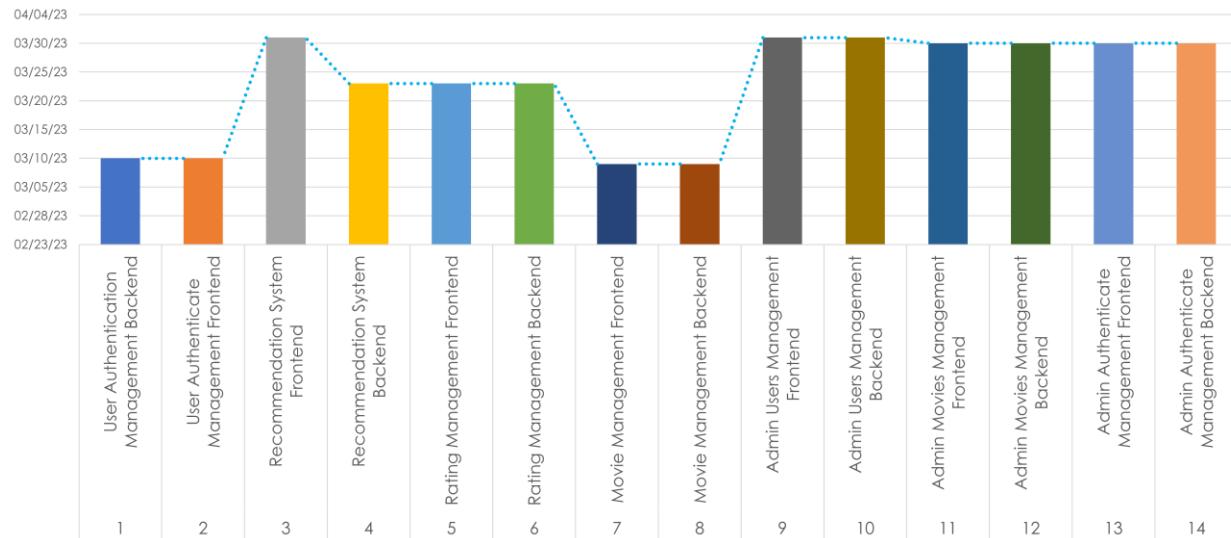


Looking at the Burndown Chart, we can see that the team has found the right pace and process for the project. The Backlogs are completed quite close to the time, not too late or too early like the previous times. Finishing too early is also not so good as finishing too late.

## VIII. Project closure

### 1. Release Burn Up Chart

More Details in file “Release-Burn-Up-Chart.xlsx”



From the Burn Up Chart, we can see that most of the product's features are completed in the last Sprint. There are 4 features completed in the early stage, 3 features completed in the middle stage, and 7 features completed in the final stage.

### 2. Verification test

Questions need answers in verification test:

- Are we building the product right?
- Does the software meet the specifications?

FR.1: User Authenticate Management:

FR.1.1: User Registration

Passed: Verify that all required fields are present, such as email, password, confirm password, ...

Passed: Verify that user can successfully register with valid information.

Passed: Checking if a user can register with valid data only in mandatory fields

Passed: A user get registered should be allowed to login to the application.

Passed: A user tries to register an existing email then an error message should get displayed.

Passed: Verify if the registration form has all the fields required by the requirement document.

Passed: Verify that a user can navigate to Login if they have an account.

FR.1.2: User Login

Passed: Verify that a user can log in with valid credentials.

Passed: Verify that a user cannot log in with invalid credentials.

Passed: Verify that a user cannot log in with invalid email and password.

Passed: Verify that Browser will save login session at local storage.

Passed: Verify that a user can navigate to Register if they don't have an account.

#### FR.1.3: User Verification

Passed: Verify that the user can successfully access the application with access token.

Passed: Verify that a logged in user doesn't need to login again if they refresh the web page.

Passed: Verify that a logged in user can navigate to another function of application.

#### FR.1.4: User Logout

Passed: Verify that at any function of the application, user can press logout.

Passed: Verify that after user press logout, all the information of user and token will be deleted.

#### FR.1.5: User Load

Passed: Verify that after user get verified, user information will be loaded.

Passed: Verify that user information will not be in local storage.

Passed: Verify that user information will be in Data Store as a state.

### FR.2: Movie Management:

#### FR.2.1: User Search Movie

Passed: User cannot search for movie with less than 3 characters.

Passed: User can search for movie with more than 3 characters.

Passed: User will get fast response when call multiple searches at a short time.

Passed: User will get a list of movies with covers for a clear view.

#### FR.2.2: User Get Movie Details

Passed: User can get all information about a movie such as: Title, Genre.

Passed: User can see movie cover.

### FR.3: Rating Management:

#### FR.3.1: User Create Rating

Passed: User will be allowed to create a new rating when they see a movie page.

Passed: After user rated a movie, their W, d in machine learning model get trained.

#### FR.3.2: User Read Rating

Passed: When a user sees a movie page, they can see their existing rating.

Passed: If user didn't rate that movie, it will display as blank star.

#### FR.3.3: User Update Rating

Passed: Verify if update rating function will only appear with existing rating.

Passed: Verify if update rating will change their W, d in machine learning model.

Passed: User should see updated rating value after server confirm.

#### FR.3.4: User Delete Rating

Passed: Verify if delete rating function will only appear with existing rating.

Passed: User should see blank rating star after server confirm rating is deleted.

### FR.4: Recommendation System:

#### FR.4.1: User Get Suggested Movies

Passed: Verify if user can get a list of suggested movies.

Passed: Verify there is no rated movie in the list.

Passed: Evaluate model with training set, testing set.

#### FR.4.2: User Get Prediction for A Movie

Passed: API call for this function work.

### FR.5: Admin Authenticate Management:

#### FR.5.1: Admin Login

Passed: Same as User Login

Passed: User credentials cannot be used to be login as admin.

Passed: Admin credentials cannot be used to be login as user.

#### FR.5.2: Admin Logout

Passed: When an admin press logout, there is no information left in the local storage.

Passed: When an admin press logout, there is no information left in the data store.

**FR.6: Admin Users Management:**

**FR.6.1: Admin Get All Users**

Passed: Admin can see all users in the database.

Passed: Click on the list will direct admin to see a specific user information.

**FR.6.2: Admin Get User Details**

Passed: Admin can see all information about user.

Passed: Admin cannot know user password nor see user encrypted password.

**FR.6.3: Admin Delete User**

Passed: Admin can only delete existing user.

Passed: Admin will access this feature in the specific user details.

**FR.7: Admin Movies management:**

**FR.7.1: Admin Get All Movies**

Passed: Admin can see all movies in the database.

Passed: Click on the list will direct admin to see a specific movie information.

**FR.7.2: Admin Get Movie Details**

Passed: Admin can see all information about movie.

**FR.7.3: Admin Add New Movie**

Passed: Admin can access this feature through FR.7.1.

Passed: After pressing a button, admin will turn into a page where they can fill the new movie.

**FR.7.4: Admin Delete Movie**

Passed: Admin can only delete existing movie.

Passed: Admin will access this feature in the specific movie details.

### **3. Validation test**

Questions need answers in validation test:

- Are we building the right product?
- Does the software meet the user requirements?

Can this application suggest movies to the user? Accepted

How does this application suggest movies to users? By a machine learning model. Accepted.

To get more accurate suggestions, what does a user need to do? Give more ratings to more movies, update their rating or delete existing ratings.

To rate a movie, what does a user need to do? Create an account, login and access a movie page.

How can a user access a movie page? Through Movie Search or Movie Suggestion.

Basically, all the requirements of the stakeholders for the application have been met and the validation test passed.

#### 4. Operation plan and Deployment order

Target cost: free plan.

There are 3 components in this project:

- Database
- Backend
- Frontend

Frontend will only work when it gets response from Backend.

Backend will only work when it gets data from database to process.

So, the deployment order will be as follows:

1. Database
2. Backend
3. Frontend

The cost plan will aim to free, details will be in the next sections.

#### 4. Database Deployment

With a logged in user at MongoDB Atlas, we can deploy a free database:

The cost plan is M0.

The provider is Azure.

The region is Netherlands (westeurope).

Name is MRSBBDB.

**Deploy your database**

Use a template below or set up [advanced configuration options](#). You can also edit these configuration options once the cluster is created.

M10	\$0.09/hour	
For production applications with sophisticated workload requirements.		
STORAGE 10 GB	RAM 2 GB	vCPU 2 vCPUs

SERVERLESS	\$0.12/1M reads	
For application development and testing, or workloads with variable traffic.		
STORAGE Up to 1 TB	RAM Auto-scale	vCPU Auto-scale

M0	FREE	
For learning and exploring MongoDB in a cloud environment.		
STORAGE 512 MB	RAM Shared	vCPU Shared

**Provider**

**Region** ★ Recommended region ⓘ

Netherlands (westeurope) ★

**Name**  
You cannot change the name once the cluster is created.

MRSBBDB

**FREE**

**Create**

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

I'll deploy my database later

[Access Advanced Configuration](#)

Next, we have to set up an authenticate method, in this project, I choose username and password

MINHTIEN'S ORG - 2019-08-11 > PROJECT 0

## Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. [Learn more about security setup](#)

### 1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Create a database user using a username and password. Users will be given the [read and write to any database privilege](#) by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

**Username**

iamtiennng

**Password** ⓘ

wf8tbc8Qcyeqb9ol

**Create User**

Success! iamtiennng has been deleted.

Next, we have to setup IP Access List, 0.0.0.0/0 is anywhere

✓ Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

**My Local Environment**



Use this to add network IP addresses to the IP Access List. This can be modified at any time.

**Cloud Environment**



Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

**ADVANCED**

Add entries to your IP Access List

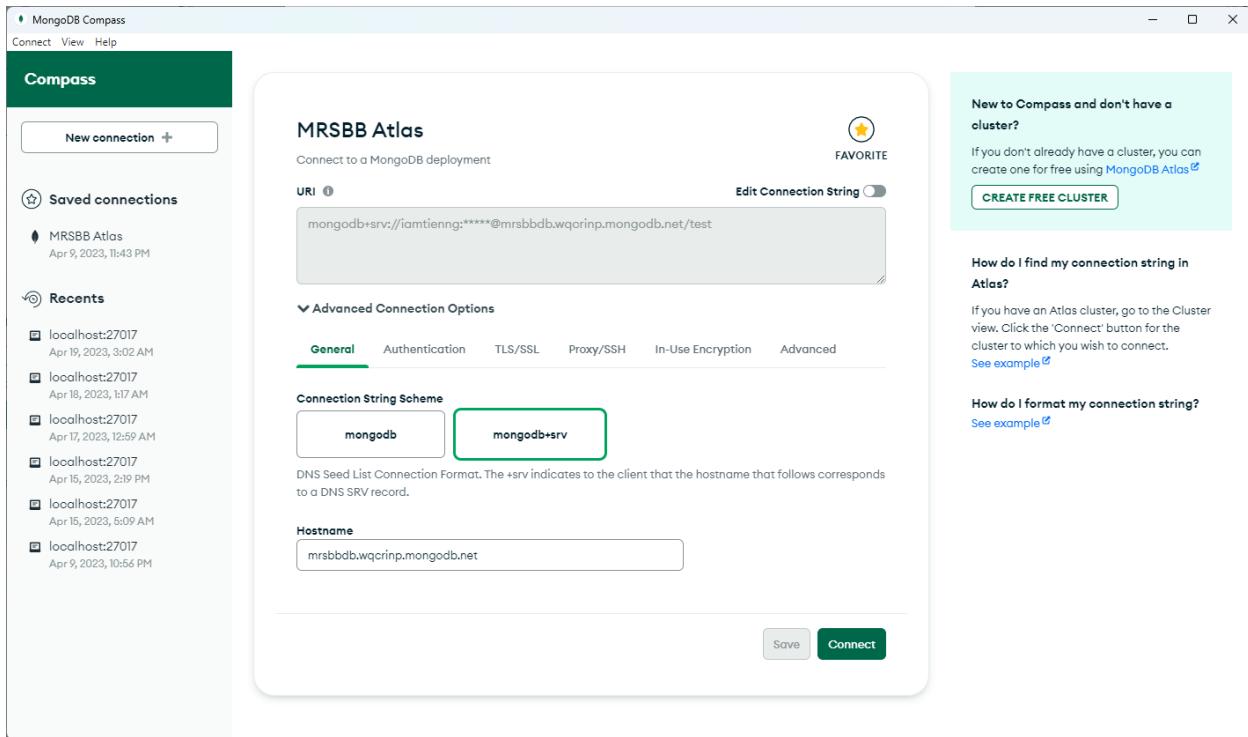
Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

IP Address	Description
<input type="text" value="Enter IP Address"/>	<input type="text" value="Enter description"/>
<input type="button" value="Add My Current IP Address"/>	
<input type="button" value="Add Entry"/>	

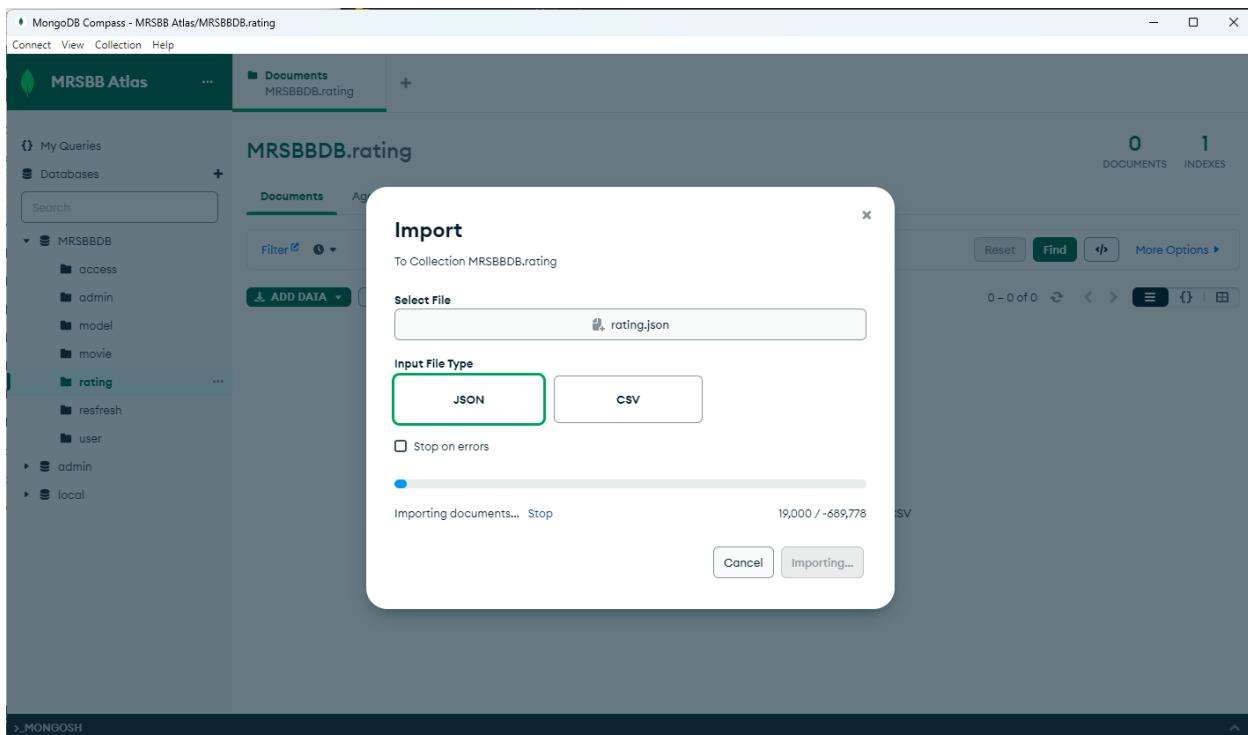
IP Access List	Description
192.167.111.197/32	 <a href="#">EDIT</a>  <a href="#">REMOVE</a>
192.167.111.158/32	 <a href="#">EDIT</a>  <a href="#">REMOVE</a>
80.181.68.206/32	My IP Address  <a href="#">EDIT</a>  <a href="#">REMOVE</a>
64.190.115.68/32	 <a href="#">EDIT</a>  <a href="#">REMOVE</a>
0.0.0.0/0	 <a href="#">EDIT</a>  <a href="#">REMOVE</a>

[See all 6 IP addresses on the Network Access Page](#)

Next, we connect the database with MongoDB Compass to setup initial data



Next, we create 7 collections and import all data in the json files in Database directory:



For Backend to access this deployed database, we have username, password and database url.

## 5. Backend Deployment

First, we need to setup requirement.txt file, this file will contain all the names of the packages and their respective versions that are needed for backend application. My requirement.txt file is:

```
Flask==2.2.2
Flask-Cors==3.0.10
wheel==0.40.0
rsa==4.7.2
pycryptodome==3.15.0
PyJWT==2.6.0
pandas==1.5.3
scikit-learn
pymongo==3.12.0
dnspython==2.3.0
```

With this requirement.txt file, a package-management system written in Python called Pip will install automatically when we set up docker container.

Next, we need to set up the Dockerfile, my Dockerfile will be as follows:

```
# author: @iamtiennng
# start by pulling the python image
FROM python:3.10-buster

# copy the requirements file into the image
COPY ./requirement.txt /app/requirement.txt

# switch working directory
WORKDIR /app

# install the dependencies and packages in the requirements file
RUN pip install -r requirement.txt

# copy every content from the local file to the image
COPY . /app

# configure the container to run in an executed manner
ENTRYPOINT [ "python" ]

CMD [ "main.py" ]
```

Before we move to next step, be sure change the localhost database to the database we deployed in the file extensions.py:

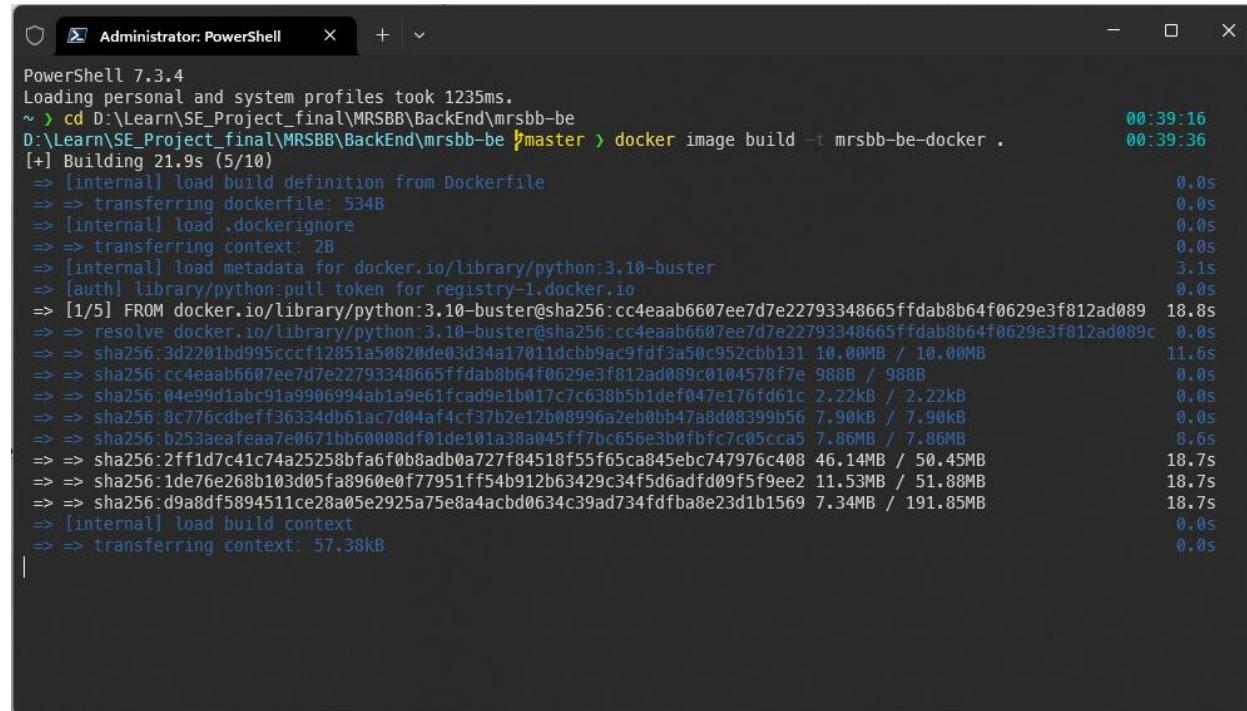
```
# connect to the database
# usernameDB = "iamtiennng"
# passwordDB = "d408CmCGCeCI8vzA"
client = MongoClient(
    'mongodb+srv://iamtiennng:d408CmCGCeCI8vzA@mrsbbdb.wqcrinp.mongodb.net/?retryWrites=true&w=majority')
# Localhost
# client = MongoClient("localhost", 27017)
db = client["MRSBBDB"]
```

Also, for deployment on azure, add this to Flask app main:

```
port = int(os.environ.get('PORT', 5000))
app.run(debug=False, host='0.0.0.0', port=port)
```

Run this command below in the terminal, we will build a docker image.

```
docker image build -t mrsbb-be-docker .
```



The screenshot shows a PowerShell window titled 'Administrator: PowerShell' running on Windows. The command 'docker image build -t mrsbb-be-docker .' is being executed. The output shows the progress of the build, including the creation of intermediate layers and the final image. The build took approximately 21.9 seconds. The Dockerfile path is D:\Learn\SE\_Project\_final\MRSBB\BackEnd\mrsbb-be\, and the Dockerfile name is Dockerfile. The command was run at 00:39:36 and completed at 00:39:56.

```
PowerShell 7.3.4
Loading personal and system profiles took 1235ms.
~> cd D:\Learn\SE_Project_final\MRSBB\BackEnd\mrsbb-be
D:\Learn\SE_Project_final\MRSBB\BackEnd\mrsbb-be\master> docker image build -t mrsbb-be-docker .
[+] Building 21.9s (5/10)
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 534B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.10-buster
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/5] FROM docker.io/library/python:3.10-buster@sha256:cc4eaab6607ee7d7e22793348665ffdab8b64f0629e3f812ad089 18.8s
=> => resolve docker.io/library/python:3.10-buster@sha256:cc4eaab6607ee7d7e22793348665ffdab8b64f0629e3f812ad089c 0.0s
=> => sha256:3d2201bd995cccf12851a50820de03d34a17011dcbb9ac9fdf3a50c952ccb131 10.00MB / 10.00MB 11.6s
=> => sha256:cc4eaab6607ee7d7e22793348665ffdab8b64f0629e3f812ad089c0104578f7e 988B / 988B 0.0s
=> => sha256:04e99d1abc91a9906994ab1a9e61fcad9e1b017c7c638b5b1def047e176fd61c 2.22kB / 2.22kB 0.0s
=> => sha256:8c776cdbeff36334db61ac7d04af4cf37b2e12b08996a2eb0bb47a08399b56 7.90kB / 7.90kB 0.0s
=> => sha256:b253aeefaa7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cc45 7.86MB / 7.86MB 8.6s
=> => sha256:2ff1d7c41c74a25258bfaf0b8adb0a727f84518f55f65ca845ebc747976c408 46.14MB / 50.45MB 18.7s
=> => sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09f5f9ee2 11.53MB / 51.88MB 18.7s
=> => sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 7.34MB / 191.85MB 18.7s
=> [internal] load build context
=> => transferring context: 57.38kB 0.0s
```

```

Administrator:PowerShell
>> => sha256:04e99d1abc91a9906994ab1a9e61fcad9e1b017c7c638b5b1def047e176fd61c 2.22kB / 2.22kB 0.0s
=> => sha256:8c776cdbeff36334db61ac7d04af4cf37b2e12b08996a2eb0bb47a8d08399b56 7.90kB / 7.90kB 0.05s
=> => sha256:b253aea7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5 7.86MB / 7.86MB 8.6s
=> => sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 50.45MB / 50.45MB 20.3s
=> => sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09ff5f9ee2 51.88MB / 51.88MB 42.8s
=> => sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 191.85MB / 191.85MB 76.8s
=> => sha256:267a744fc65d3314725b6ab93b395ef2401f31959c1ecf03bf8c2176b622d2e6 6.15MB / 6.15MB 24.1s
=> => extracting sha256:2ff1d7c41c74a25258bfa6f0b8adb0a727f84518f55f65ca845ebc747976c408 0.9s
=> => extracting sha256:b253aea7e0671bb60008df01de101a38a045ff7bc656e3b0fbfc7c05cca5 0.1s
=> => extracting sha256:3d2201bd995ccf12851a058020de03d34a17011dcbb9ac9fd3a50c952ccb131 0.1s
=> => sha256:63a71fb213c6ef6ed5fe84ed1f81afcdeeee9d68284472d9eb3db5b49c5443cf 19.02MB / 19.02MB 34.4s
=> => sha256:40b447977237e8de3b0935a421187a655dc14a64eced639f360d233ea6573b6 244B / 244B 35.0s
=> => sha256:07bd2bf441b082b03c7eaf625391869751f62e35c15eec1b89b687afdaa2fc2 3.08MB / 3.08MB 37.1s
=> => extracting sha256:1de76e268b103d05fa8960e0f77951ff54b912b63429c34f5d6adfd09ff5f9ee2 1.0s
=> => extracting sha256:d9a8df5894511ce28a05e2925a75e8a4acbd0634c39ad734fdfba8e23d1b1569 3.1s
=> => extracting sha256:267a744fc65d3314725b6ab93b395ef2401f31959c1ecf03bf8c2176b622d2e6 0.1s
=> => extracting sha256:63a71fb213c6ef6ed5fe84ed1f81afcdeeee9d68284472d9eb3db5b49c5443cf 0.3s
=> => extracting sha256:40b447977237e8de3b0935a421187a655dc14a64eced639f360d233ea6573b6 0.0s
=> => extracting sha256:07bd2bf441b082b03c7eaf625391869751f62e35c15eec1b89b687afdaa2fc2 0.1s
=> [internal] load build context 0.0s
=> => transferring context: 57.38kB 0.0s
=> [2/5] COPY ./requirement.txt /app/requirement.txt 0.3s
=> [3/5] WORKDIR /app 0.0s
=> [4/5] RUN pip install -r requirement.txt 228.2s
=> [5/5] COPY . /app 0.1s
=> exporting to image 1.8s
=> => exporting layers 1.7s
=> => writing image sha256:2170f64c74f4c1a31f841e6659ad24fec19b59b064f2c43a09f62970907d4da6 0.0s
=> => naming to docker.io/library/mrsbb-be-docker 0.0s
D:\Learn\SE_Project_final\MRSBB\BackEnd|mrsbb-be > | 00:45:47

```

After having a docker image, we need to add tag which will be our docker username so we can push the image to docker hub with this command:

```
docker tag mrsbb-be-docker iamtiennng/mrsbb-be-docker
```

Name	Tag	Status	Created	Size	Actions
iamtiennng/mrsbb-be-docker c1921f7bd5af	latest	Unused	15 minutes ago	1.31 GB	<a href="#">View packages and CVEs</a> <a href="#">Pull</a> <a href="#">Push to Hub</a>

Create a resource group on Azure portal:

## Create a resource group

Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

### Project details

Subscription \* ⓘ

Azure for Students

Resource group \* ⓘ

MRSBB-System-Resource

### Resource details

Region \* ⓘ

(US) East US

## Create Web App on Azure portal:

The pricing plan is Free F1.

Domain name will be mrsbb-system-be.azurewebsites.net

## Create Web App

Basics Docker Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more](#)

### Project Details

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ

Azure for Students

Resource Group \* ⓘ

MRSBB-System-Resource

[Create new](#)

### Instance Details

Need a database? [Try the new Web + Database experience.](#)

Name \*

mrsbb-system-be

.azurewebsites.net

Code  Docker Container  Static Web App

Operating System \*

Linux  Windows

Region \*

West Europe

ⓘ Not finding your App Service Plan? Try a different region or select your App Service Environment.

### Pricing plans

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app.  
[Learn more](#)

Linux Plan (West Europe) \* ⓘ

(New) ASP-MRSBBSysTemResource-a5bb

[Create new](#)

Pricing plan

Free F1 (Shared infrastructure)

[Explore pricing plans](#)

After that we need to identify the docker image and tag

Basics Docker Networking Monitoring Tags Review + create

Pull container images from Azure Container Registry, Docker Hub or a private Docker repository. App Service will deploy the containerized app with your preferred dependencies to production in seconds.

Options Single Container

Image Source Docker Hub

Docker hub options

Access Type \* Public

Image and tag \* iamtiennng/mrsbb-be-docker

Startup Command

## Create Web App ...

Basics Docker Networking Monitoring Tags Review + create

Summary



Free sku

Estimated price - Free

### Details

Subscription	2acfa2b0-6a34-4ab6-aec4-d284e91d6b00
Resource Group	MRSBB-System-Resource
Name	mrsbb-system-be
Publish	Docker Container
Image:Tag	iamtiennng/mrsbb-be-docker
Server URL	<a href="https://index.docker.io">https://index.docker.io</a>

### App Service Plan (New)

Name	ASP-MRSBBSYSTEMRESOURCE-96ba
Operating System	Linux
Region	West Europe
SKU	Free
ACU	Shared infrastructure
Memory	1 GB memory

### Monitoring (New)

Application Insights	Enabled
Name	mrsbb-system-be
Region	West Europe

### Deployment

Continuous deployment	Not enabled / Set up after app creation
-----------------------	---

[Create](#)

[< Previous](#)

[Next >](#)

[Download a template for automation](#)

After creating the webapp, we need to wait for azure to deploy.

The screenshot shows the Azure portal's 'Overview' page for a web app named 'Microsoft.Web-WebApp-Portal-ed9043ca-bca6'. The top navigation bar includes a search bar, 'Delete', 'Cancel', 'Redeploy', 'Download', and 'Refresh' buttons. On the left, a sidebar lists 'Overview', 'Inputs', 'Outputs', and 'Template'. The main content area displays a message: 'Deployment is in progress'. It provides deployment details: Deployment name: Microsoft.Web-WebApp-Portal-ed9043ca-bca6, Subscription: Azure for Students, Resource group: MRSBB-System-Resource. It also shows the start time: 4/21/2023, 1:22:10 AM and Correlation ID: 201055d0-b320-4c85-a2dd-fc25e48bca7b. Below this, a section titled 'Deployment details' shows a table with columns 'Resource', 'Type', and 'Status', which currently has 'No results.' listed. At the bottom, there are links to 'Give feedback' and 'Tell us about your experience with deployment'.

After that, our Backend is completely deployed

The screenshot shows the Azure portal's 'mrsbb-system-be' settings page for a Web App. The left sidebar includes sections for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment slots, Deployment Center, Configuration, Authentication, Application Insights, Identity, Backups, Custom domains, Certificates, Networking, Scale up (App Service plan), Scale out (App Service plan), Service Connector, and Locks. The main content area is divided into several sections: 'Essentials' (Resource group: MRSBB-System-Resource, Status: Running, Location: West Europe, Subscription: Azure for Students, Subscription ID: 2acfa2b6-6a34-4ab6-aec4-d284e91d6b00, Tags: Click here to add tags), 'Properties' (Web app: Name: mrsbb-system-be, Publishing model: Container, Container Image: iamtiennng/mrsbb-be-docker), 'Domains' (Default domain: mrsbb-system-be.azurewebsites.net, Custom domain: Add custom domain), 'Deployment Center' (Deployment logs: View logs), 'Application Insights' (Name: mrsbb-system-be, Region: West Europe), 'Networking' (Virtual IP address: 20.105.216.4, Outbound IP addresses: 20.86.239.120, 20.86.239.164, 20.86.238..., Additional Outbound IP addresses: 20.86.239.120, 20.86.239.164, 20.86.238..., Show More), and 'Hosting' (Plan Type: App Service plan, Name: ASP-MRSBBSYSTEMRESOURCE-92b6, Operating System: Linux, Instance Count: 0, SKU and size: Free (F1) Scale up).

## 6. Frontend Deployment

Again, we create another Web App on Azure portal for Frontend:

**Create Web App** ...

**Basics** Deployment Networking Monitoring Tags Review + create

App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. [Learn more ↗](#)

**Project Details**

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ Azure for Students

Resource Group \* ⓘ MRSBB-System-Resource

[Create new](#)

**Instance Details**

Need a database? [Try the new Web + Database experience. ↗](#)

Name \* mrsbb.azurewebsites.net

Publish \*  Code  Docker Container  Static Web App

Runtime stack \* Node 18 LTS

Operating System \*  Linux  Windows

Region \* West Europe

ⓘ Not finding your App Service Plan? Try a different region or select your App Service Environment.

**Pricing plans**

App Service plan pricing tier determines the location, features, cost and compute resources associated with your app. [Learn more ↗](#)

Linux Plan (West Europe) \* ⓘ ASP-MRSBBSYSTEMRESOURCE-92b6 (F1)

[Create new](#)

Dricing plan **Free F1** (Shared infrastructure)

[Review + create](#) [< Previous](#) [Next : Deployment >](#)

But this time, we will not configure on Azure portal, we will use Azure DevOps Pipeline instead.

Before we handle Pipeline, we need to change Backend API URL in Frontend by modifying the .env file:

```
REACT_APP_API_URL = "https://mrsbb-system-be.azurewebsites.net/"  
# Localhost  
# REACT_APP_API_URL = "http://localhost:5000"
```

After that, we need to use Git to push our source code to Repos in MRSBB System Project on Azure DevOps:

The screenshot shows the Azure DevOps interface for the 'MRSBB System' repository. On the left, there's a sidebar with navigation links: Overview, Boards, Repos (selected), Files, Commits, Pushes, and Branches. The main area displays a file tree with 'BackEnd', 'FrontEnd', '.gitignore', and 'README.md'. A table below lists these files with columns for Name, Last change, and Commits.

Name	Last change	Commits
BackEnd	4m ago	<a href="#">9117f340</a> Add Port to Flask app iamtiennng
FrontEnd	4h ago	<a href="#">bc86673c</a> First Commit iamtiennng
.gitignore	4h ago	<a href="#">bc86673c</a> First Commit iamtiennng
README.md	4h ago	<a href="#">bc86673c</a> First Commit iamtiennng

Select “Use the classic editor”.

The screenshot shows the 'New pipeline' configuration screen. The left sidebar includes 'Pipelines' (selected), Pipelines, Environments, Releases, Library, Task groups, Deployment groups, Test Plans, and Artifacts. The main area has tabs for Connect, Select (selected), Configure, and Review. Below the tabs, it says 'Where is your code?' and lists various options: Azure Repos Git (YAML), Bitbucket Cloud (YAML), GitHub (YAML), GitHub Enterprise Server (YAML), Other Git, and Subversion. A note at the bottom says 'Use the classic editor to create a pipeline without YAML.'

Select Azure Repos Git as repository for Pipeline

The screenshot shows the 'Select a source' step. It features a large right-pointing arrow icon and the text 'Select your repository'. Below it, a note says 'Tell us where your sources are. You can customize how to get these sources from the repository later.' To the right, there's a 'Select a source' section with icons for Azure Repos Git, GitHub, GitHub Enterprise Server, and Subversion. Below these are dropdown menus for 'Team project' (set to 'MRSBB System'), 'Repository' (set to 'MRSBB System'), and 'Default branch for manual and scheduled builds' (set to 'master'). A 'Continue' button is at the bottom.

Choose to start with an Empty job. We will set up 3 jobs, careful the path for npm run build and Publish Artifact: drop

... > MRSBB System-Cl

Tasks Variables Triggers Options History | Save & queue Discard Summary Queue ...

Pipeline Build pipeline

Get sources MRSBB System master

Agent job 1 Run on agent

+ npm install npm

npm run build npm

↑ Publish Artifact: drop Publish build artifacts

npm Task version 1.\*

Display name \* npm run build

Command \* custom

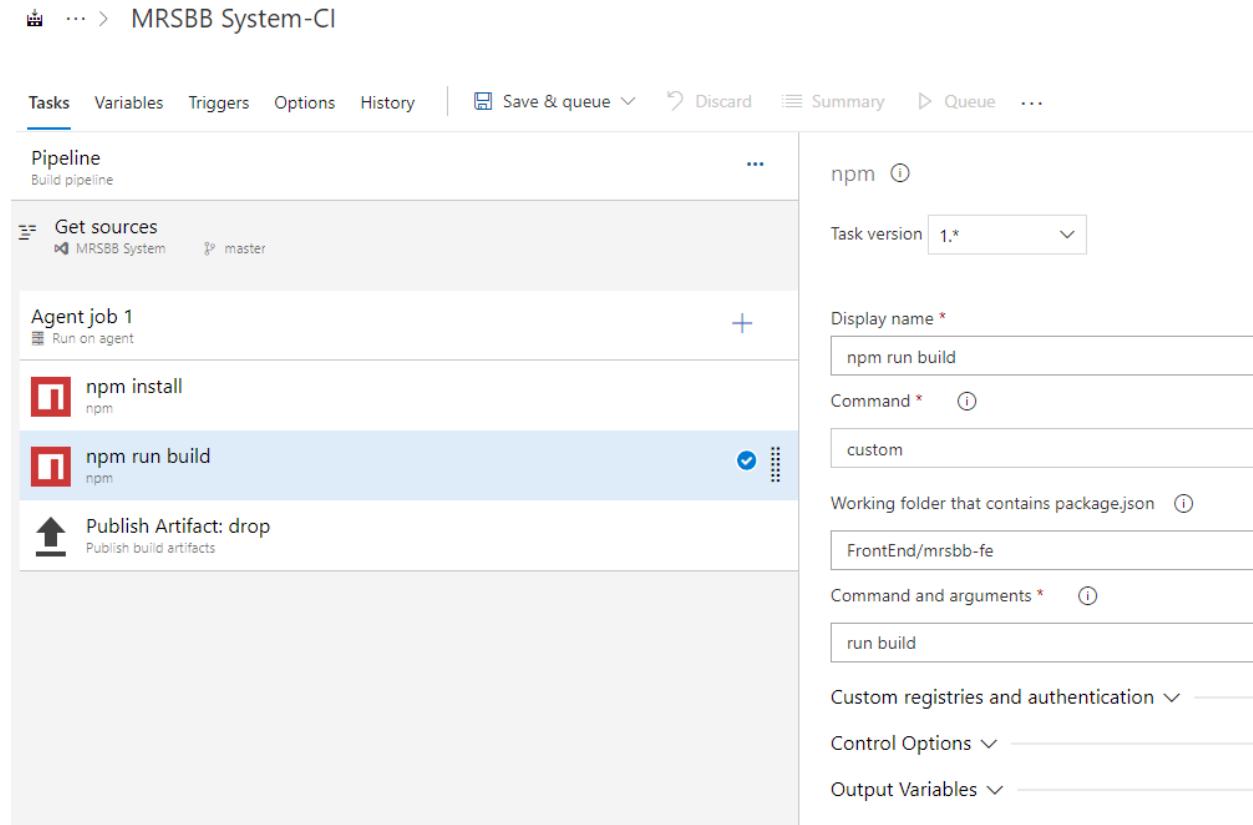
Working folder that contains package.json FrontEnd/mrsbb-fe

Command and arguments \* run build

Custom registries and authentication

Control Options

Output Variables



... > MRSBB System-Cl

Tasks Variables Triggers Options History | Save & queue Discard Summary Queue ...

Pipeline Build pipeline

Get sources MRSBB System master

Agent job 1 Run on agent

+ npm install npm

npm run build npm

↑ Publish Artifact: drop Publish build artifacts

Publish build artifacts Task version 1.\*

Display name \* Publish Artifact: drop

Path to publish \* FrontEnd/mrsbb-fe/build

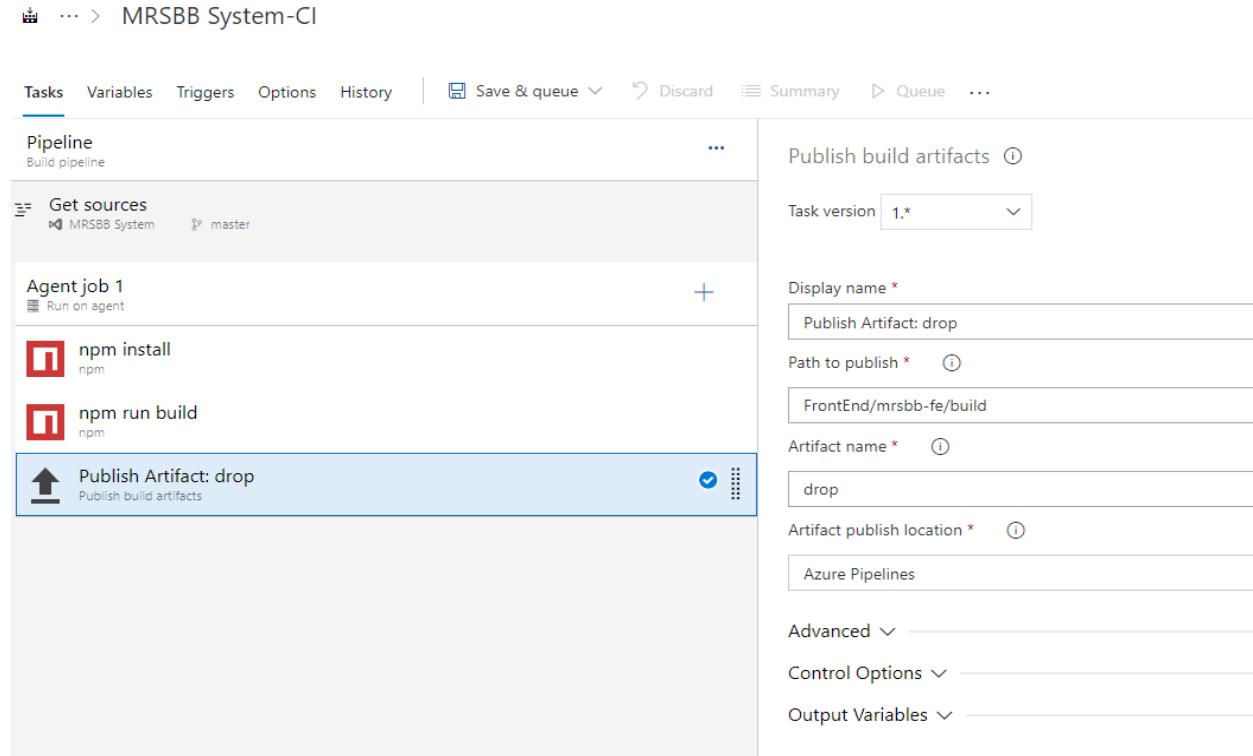
Artifact name \* drop

Artifact publish location \* Azure Pipelines

Advanced

Control Options

Output Variables



Go to the Triggers tab and select Enable continuous integration.

The screenshot shows the 'Triggers' tab in the Azure DevOps interface for the 'MRSBB System-CI' pipeline. The 'Continuous integration' section is selected. It displays two trigger types: 'Scheduled' and 'Build completion'. Under 'Scheduled', there is a note 'No builds scheduled' and a '+ Add' button. Under 'Build completion', there is a note 'Build when another build completes' and a '+ Add' button. On the right side, there are configuration options for the 'MRSBB System' trigger. These include a checked checkbox for 'Enable continuous integration' and an unchecked checkbox for 'Batch changes while a build is in progress'. Below these are 'Branch filters' and 'Path filters' sections, both with '+ Add' buttons. The 'Branch filters' section shows a dropdown set to 'Include' and a text input field containing 'master'. The 'Path filters' section has a similar '+ Add' button.

Click on Save & queue dropdown in the top menu and then click on Save & queue option. Run pipeline modal will open. Change Agent Specification to ubuntu latest and click on Save and run button.

## Run pipeline

X

Select parameters below and manually run the pipeline

Save comment

Agent pool

Azure Pipelines ▾

Agent Specification \*

ubuntu latest ▾

Branch/tag

master ▾

Select the branch, commit, or tag

Advanced options

Variables

1 variable defined



Demands

This pipeline has no defined demands



Enable system diagnostics

Cancel

Save and run

## After queue and run

The screenshot shows the Azure DevOps pipeline run details for a completed build. The summary section includes:

- Run #11 • Change API URL**
- MRSBB System-CI**
- Retention:** This run is being retained as one of 3 recent runs by master (Branch).
- Manually run by:** Tien Nguyen
- Repository and version:** MRSBB System, master, b6c94672
- Time started and elapsed:** Today at 04:33, 2m 16s
- Related:** 0 work items, 1 published; 1 consumed

The **Jobs** section lists:

Name	Status
Agent job 1	Success

## Steps to create a Release Pipeline

1. Create a new pipeline by selecting Pipelines -> Releases and clicking on New pipeline button.
2. Select template Azure App Service deployment and click on Apply button.

The screenshot shows the 'New release pipeline' creation interface. On the left, the pipeline structure is shown with 'Artifacts' and 'Stages'. The 'Stages' section has a 'Stage 1' card labeled 'Select a template'. On the right, the 'Select a template' pane is open, showing the 'Featured' section with the 'Azure App Service deployment' template selected. Other templates listed include:

- Deploy a Java app to Azure App Service
- Deploy a Node.js app to Azure App Service
- Deploy a PHP app to Azure App Service and Azure Database for MySQL
- Deploy a Python app to Azure App Service and Azure database for MySQL
- Deploy to a Kubernetes cluster
- IIS website and SQL database deployment
- Azure App Service deployment with...

3. Set Stage name to Release.

4. Click on Add an artifact and select the build pipeline that we created previously as Source. Click on Add button.

## Steps to configure Continuous Deployment

1. Click on the lightning icon in Artifacts and enable Continuous deployment trigger. This will create a new release every time a new build is available. Also add main branch as build branch filter and close the dialog.

2. Click on 1 job, 1 task link in Stages block to finish configuration of the Azure App Service deployment. Select our Azure subscription. We will be asked to authorize it by clicking on Authorize button. Also select App service name which we created previously.

All pipelines > New release pipeline

Pipeline Tasks Variables Retention Options History

**Release**  
Deployment process

Run on agent +

Deploy Azure App Service  
Azure App Service deploy

Stage name: Release

Parameters: Unlink all | Manage

Azure subscription: Azure for Students (2acfa2b0-6a34-4ab6-aec4-d284e91d6b00)

App type: Web App on Linux

App service name: mrsbb

Startup command:

3. Click on Deploy Azure App Service and select drop folder as folder which contains app service contents generated in build process.

Pipeline Tasks Variables Retention Options History

**Release**  
Deployment process

Run on agent +

Deploy Azure App Service  
Azure App Service deploy

Azure for Students (2acfa2b0-6a34-4ab6-aec4-d284e91d6b00)

App Service type: Web App on Linux

App Service name: mrsbb

Deploy to Slot or App Service Environment

Package or folder: \$(System.DefaultWorkingDirectory)/\_MRSBB System-CI/drop

Runtime Stack:

4. Click on Save in the top right corner and then click on OK without changing anything in the modal dialog which will appear.

5. Now click on Create release (next to the Save button) which will become enabled. Another modal dialog will appear, but we can leave everything as it is and click on the Create button.

6. Under Pipelines -> Releases we can see the newly created release pipeline running. It may take a few minutes until it succeeds.

↑ New release pipeline > Release-1 ▾

Pipeline Variables History + Deploy ▾ ⏺ Cancel ⏴ Refresh ⏪ Edit ▾ ...

**Release**

Manually triggered  
by  Tien Nguyen  
21/04/2023, 04:49

Artifacts

 \_MRSBB System-CI  
11  
master

**Stages**

**Release**  
In progress  
 Deploy Azure App Service  
🕒 00:01

Once release pipeline is successfully completed, we should be able to see your deployed application on <https://mrsbb.azurewebsites.net>

Finally, we need to setup Startup Command in Configuration of mrsbb on Azure Portal

The screenshot shows the Azure portal interface for a 'mrsbb | Configuration' page. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment slots, Deployment Center, and Configuration (which is selected). The main area has tabs for Application settings, General settings (selected), Path mappings, and Error pages (preview). Under 'Stack settings', it shows Stack (Node), Major version (Node 18), Minor version (Node 18 LTS), and Startup Command (npx serve -s). A note says 'Click here to upgrade to a higher SKU and enable additional features.'

From now on whenever we push new source code to the main branch, a new deployment for Frontend will be automatically run.

## 7. Scale solutions

A machine learning-based system in general and a recommendation system in particular will both be better as soon as the system becomes viral and has a large amount of data from a large number of users. At this point, we have to deal with the scale of the system.

There are two solutions to this problem that correspond to the two traditional scaling methods:

- Scale-up: Keeping only one object of the recommendation system when the Backend is initialized is great for Scale Up. But we are deploying the system as a micro-service (using docker to package the container and run it on a virtual machine of the azure server) and won't be able to take advantage of creating multiple replicas to serve more users. Updating the model in the database is extremely resource-intensive because the model needs to be updated with both a matrix as well as an array each time it needs to be updated.
- Scale-out: To be able to Scale Out the system. We can redesign the database so that each element in model W, X, d, b is a separate table. Every time there is a new user, W will be added a row and d will be added a number. Similarly, X and b will be added new values when Admin adds a new movie. Every time a user uses Rating Management, we only need to update a row of W and a value in list d to be able to use that value for another replica of the Backend. When the user needs to get the predicted rating for a movie, we only

need to load a line of W and X and a value of d and b to get the result, not the whole model as before.

## IX. Recommendation System Algorithm Design

The model for Recommendation System in this project is the 3<sup>rd</sup> model in the research: Matrix Factorization. For full details about the model, please refer to the file "Recommendation\_System". I will write a quick explanation follow.

### 1. Foundation

A key idea in recommendation systems, particularly in collaborative filtering, is the utility matrix. It is a matrix that records how a recommendation system's users interact with the items or simply rate them. The cells of the matrix contain the ratings or interactions between the users and the items, while the rows and columns of the matrix represent the users and the items, respectively.

	A	B	C	D	E	F
Bohemian Rhapsody	5	5	0	0	1	?
I Love Rock 'N Roll	5	?	?	0	?	?
Hotel California	?	4	1	?	?	1
Twinkle, Twinkle Little Star	1	1	4	4	4	?
Happy and You Know It	1	0	5	?	?	?

Figure 3.6: An example of utility matrix with song recommendation system. Songs (items) are rated by users on a scale from 0 to 5 stars. The "?" marks indicate that the data does not yet exist in the database. The recommendation system needs to manually fill in these values.

In content-based recommendation systems, each item is described by a vector  $x$  called the item profile. In that method, we need to find a coefficient vector  $w$  corresponding to each user such that the known rating that user gives the item is approximately equal to:

$$y \approx w^T x = x^T w$$

With this approach, the utility matrix  $Y$ , assuming it is filled, will approximate to:

$$Y \approx \begin{bmatrix} x_1^T w_1 & x_1^T w_2 & \dots & x_1^T w_N \\ x_2^T w_1 & x_2^T w_2 & \dots & x_2^T w_N \\ \dots & \dots & \dots & \dots \\ x_M^T w_1 & x_M^T w_2 & \dots & x_M^T w_N \end{bmatrix} = \begin{bmatrix} x_1^T \\ x_2^T \\ \dots \\ x_M^T \end{bmatrix} [w_1 \ w_2 \ \dots \ w_N] = X^T W$$

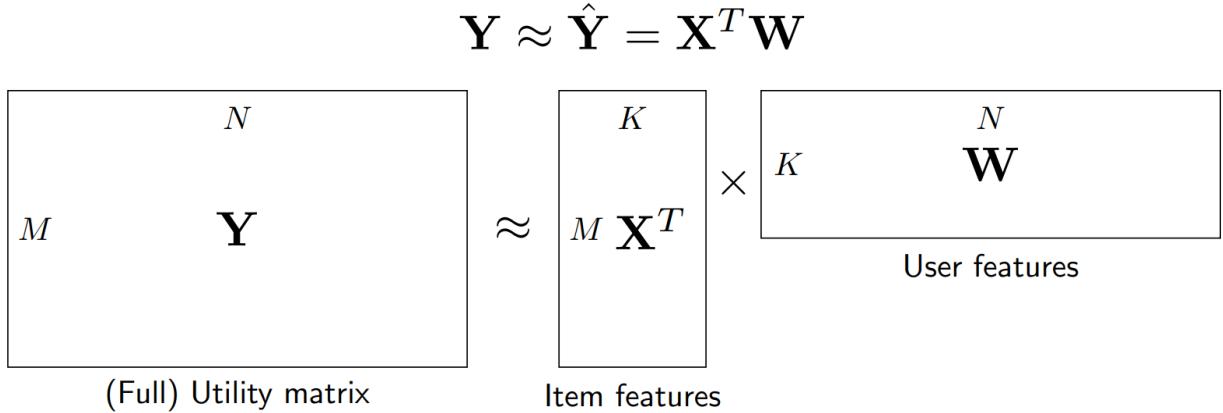
$M$ : number of items.

$N$ : number of users.

Now, suppose that we do not need to pre-construct the  $x$  profile items, but that the feature vector for each of these items can be trained concurrently with each user's model (here, a coefficient vector). This means, the variables in the optimization problem are both  $X$  and  $W$ ; where,  $X$  is the matrix of the entire profile item, each column corresponds to an item,  $W$  is the matrix of the entire user model, each column corresponds to a user.

With this approach, we are trying to approximate the utility matrix  $Y \in R^{MxN}$  by the product of two matrices  $X \in R^{MxK}$  and  $W \in R^{KxN}$ . Usually,  $K$  is chosen to be a much smaller number than

$M, N$ . Then, both the  $X$  and  $W$  matrices have a rank not exceeding  $K$ . Therefore, this method is also called low-rank matrix factorization.



## 2. Loss Function

The loss function for the Matrix Factorization Collaborative Filtering can be written as:

$$\mathcal{L}(X, W, b, d) = \frac{1}{2s} \sum_{n=1}^N \sum_{m:r_{mn}=1} (x_m^T w_n + b_m + d_n - y_{mn}) + \frac{\lambda}{2} (\|X\|_F^2 + \|W\|_F^2)$$

$r_{mn} = 1$  if m-th item has been rated by n-th user.

s: number of ratings in dataset.

$y_{mn}$ : rating of  $n - th$  user for  $m - th$  item.

$\|X\|_F$ : Frobenius norm of X.

$\frac{1}{2s} \sum_{n=1}^N \sum_{m:r_{mn}=1} (x_m^T w_n + b_m + d_n - y_{mn})$  is the loss data, is the mean error of the model.

$\frac{\lambda}{2} (\|X\|_F^2 + \|W\|_F^2)$  is  $l_2$  regularization loss, help avoid overfitting.

Simultaneous optimization of  $X, W, b, d$  is relatively complicated. Instead, the method used is to optimize one of the two pairs  $(X, b), (W, d)$  in turn, while the other pair is fixed. This process is repeated until the loss function converges.

## 3. Optimizing the loss function

When the pair  $(X, b)$  is fixed, the pair  $(W, d)$  optimization problem can be decomposed into  $N$  subproblems:

$$\mathcal{L}_1(w_n, d_n) = \frac{1}{2s} \sum_{m:r_{mn}=1} (x_m^T w_n + b_m + d_n - y_{mn})^2 + \frac{\lambda}{2} (\|w_n\|_2^2)$$

Each problem can be optimized using gradient descent. Our important job is to calculate the derivatives of each of these small loss functions with respect to  $w_n, d_n$ . Since the expression in the sign Sigma depends only on the items that have been rated by the user in question (corresponding to  $r_{mn} = 1$ ), we can simply by making  $\hat{X}_n$  a submatrix generated by the columns of  $X$  corresponds to the items that have been rated by user n,  $\hat{b}_n$  is the corresponding sub-bias vector, and  $\hat{y}_n$  is the corresponding rating. Then:

$$\mathcal{L}_1(w_n, d_n) = \frac{1}{2S} \|\hat{X}_n^T w_n + \hat{b}_n + d_n \mathbf{1} - \hat{y}_n\|^2 + \frac{\lambda}{2} (\|w_n\|_2^2)$$

With  $\mathbf{1}$  is the vector with all elements equal to 1 and the appropriate size. Its derivative is

$$\begin{aligned}\nabla_{w_n} \mathcal{L}_1 &= \frac{1}{S} \hat{X}_n (\hat{X}_n^T w_n + \hat{b}_n + d_n \mathbf{1} - \hat{y}_n) + \lambda w_n \\ \nabla_{d_n} \mathcal{L}_1 &= \frac{1}{S} \mathbf{1}^T (\hat{X}_n^T w_n + \hat{b}_n + d_n \mathbf{1} - \hat{y}_n)\end{aligned}$$

**Update equation for  $w_n, d_n$ :**

$$\begin{aligned}w_n &\leftarrow w_n - \eta \left( \frac{1}{S} \hat{X}_n (\hat{X}_n^T w_n + \hat{b}_n + d_n \mathbf{1} - \hat{y}_n) + \lambda w_n \right) \\ d_n &\leftarrow d_n - \eta \left( \frac{1}{S} \mathbf{1}^T (\hat{X}_n^T w_n + \hat{b}_n + d_n \mathbf{1} - \hat{y}_n) \right)\end{aligned}$$

$\eta$  is the learning rate.

Similarly, each column  $x_m$  of  $X$ , which is the feature vector for each item, and  $b_m$  can be found by optimizing the problem:

$$\mathcal{L}_2(x_m, b_m) = \frac{1}{2S} \sum_{n:r_{mn}=1} (w_n^T x_m + d_n + b_m - y_{mn})^2 + \frac{\lambda}{2} (\|x_m\|_2^2)$$

Let  $\hat{W}_m$  be the matrix created with the columns of  $W$  corresponding to the users who have rated item m,  $\hat{d}_m$  the corresponding bias sub-vector, and  $\hat{y}_m$  the corresponding rating vector. The problem becomes:

$$\mathcal{L}_2(x_m, b_m) = \frac{1}{2S} \|\hat{W}_m^T x_m + \hat{d}_m + b_m \mathbf{1} - \hat{y}_m\|^2 + \frac{\lambda}{2} (\|x_m\|_2^2)$$

**Update equation for  $x_m, b_m$ :**

$$\begin{aligned}x_m &\leftarrow x_m - \eta \left( \frac{1}{S} \hat{W}_m (\hat{W}_m^T x_m + \hat{d}_m + b_m \mathbf{1} - \hat{y}_m) + \lambda w_m \right) \\ b_m &\leftarrow d_m - \eta \left( \frac{1}{S} \mathbf{1}^T (\hat{W}_m^T x_m + \hat{d}_m + b_m \mathbf{1} - \hat{y}_m) \right)\end{aligned}$$

#### 4. Development in Python

First, we will write a Matrix Factorization class that performs the optimization of variables with a utility matrix given in the form of  $Y_{\text{data}}$  just like with NBCF. First, we declare some necessary libraries and initialize the Matrix Factorization class:

```
import json
import pandas as pd
import numpy as np

class MF(object):
    def __init__(self, Y, K, lam = 0.1, Xinit = None, Winit = None,
                 learning_rate = 0.5, max_iter = 10000, print_every = 100):
        self.Y = Y      # represents the utility matrix
        self.K = K      # number of features
        self.lam = lam  # regularization parameter
        self.learning_rate = learning_rate # for gradient descent
        self.max_iter = max_iter           # maximum number of iterations
        self.print_every = print_every     # print loss after each a few iters
        self.n_users = int(np.max(Y[:, 0])) + 1
        self.users_ids = np.unique(np.asarray(Y[:, 0].reshape(Y[:, 0].shape[0])))
        self.n_items = int(np.max(Y[:, 1])) + 1
        self.items_ids = np.unique(np.asarray(Y[:, 1].reshape(Y[:, 1].shape[0])))
        self.n_ratings = Y.shape[0]
        self.X = np.random.randn(self.n_items, K) if Xinit is None else Xinit
        self.W = np.random.randn(K, self.n_users) if Winit is None else Winit
        self.b = np.random.randn(self.n_items)  # item biases
        self.d = np.random.randn(self.n_users)  # user biases
```

Next, we write loss, updateXb, updateWd methods for class MF:

```
# return current loss value
def loss(self):
    L = 0
    for i in range(self.n_ratings):
        # user_id, item_id, rating
        n, m, rating = int(self.Y[i, 0]), int(self.Y[i, 1]), self.Y[i, 2]
        L += 0.5*(self.X[m].dot(self.W[:,n]) + self.b[m] + self.d[n] - rating)**2
    L /= self.n_ratings
    # regularization, don't ever forget this
    return L + 0.5*self.lam*(np.sum(self.X**2) + np.sum(self.W**2))
```

```

def updateWd(self):
    for n in self.users_ids:
        # get all items rated by user n, and the corresponding ratings
        ids = np.where(self.Y[:, 0] == n)[0]
        item_ids, ratings = self.Y[ids, 1].astype(np.int32), self.Y[ids, 2]
        Xn, bn = self.X[item_ids], self.b[item_ids]
        for i in range(30): # 30 iteration for each sub problem
            wn = self.W[:, n]
            error = Xn.dot(wn) + bn + self.d[n] - ratings
            grad_wn = Xn.T.dot(error)/self.n_ratings + self.lam*wn
            grad_dn = np.sum(error)/self.n_ratings
            # gradient descent
            self.W[:, n] -= np.array(self.learning_rate*grad_wn.reshape(-1))[0]
            self.d[n]     -= self.learning_rate*grad_dn

```

```

def updateXb(self):
    for m in self.items_ids:
        ids = np.where(self.Y[:, 1] == m)[0] # row indices of items m
        user_ids, ratings = self.Y[ids, 0].astype(np.int32), self.Y[ids, 2]
        Wm, dm = self.W[:, user_ids], self.d[user_ids]
        Wm = Wm.reshape(Wm.shape[0], Wm.shape[1])
        for i in range(30): # 30 iteration for each sub problem
            xm = self.X[m]
            error = Wm.T.dot(xm).reshape(-1,1) + self.b[m] + dm - ratings
            grad_xm = Wm.dot(error)/self.n_ratings + (self.lam*xm).reshape(-1,1)
            grad_bm = np.sum(error)/self.n_ratings
            # gradient descent
            self.X[m] -= np.array((self.learning_rate*grad_xm).T)[0]
            self.b[m] -= rs.learning_rate*grad_bm

```

The next part is the main optimization process of MF (fit), predicting the new rating (pred) and evaluating the model quality by root-mean-square error (evaluate\_RMSE).

```
def fit(self):
    for it in range(self.max_iter):
        self.updateXb()
        self.updateWd()
        if (it+1) % self.print_every == 0:
            rsme_train = self.evaluate_RMSE(self.Y)
            # print("iter = ",it+1 ,", Loss = "+self.Loss()," , RMSE train =
",rsme_train)
            print(it+1)
            print(self.loss())
            print(rsme_train)
            print(self.evaluate_RMSE(rate_test))
```

```
def pred(self, u, i):
    # predict the rating of user u for item i
    u, i = int(u), int(i)
    try:
        pred = self.X[i, :].dot(self.W[:, u]) + self.b[i] + self.d[u]
    except:
        return 0
    return max(0, min(5, pred))

def evaluate_RMSE(self, rate_test):
    n_tests = rate_test.shape[0]
    SE = 0
    for n in range(n_tests):
        pred = self.pred(rate_test[n, 0], rate_test[n, 1])
        SE += (pred - rate_test[n, 2])**2

    RMSE = np.sqrt(SE/n_tests)
    return RMSE
```

At this point, we have built into the Matrix Factorization class with the necessary methods.

## 5. Model input explanation

Input for Matrix Factorization will be:

Utility matrix Y is a numpy.ndarray which includes all existed ratings, there are 4 columns in Y, userId, movieId, rating, timestamp.

```
rs.Y
✓ 0.0s
Python

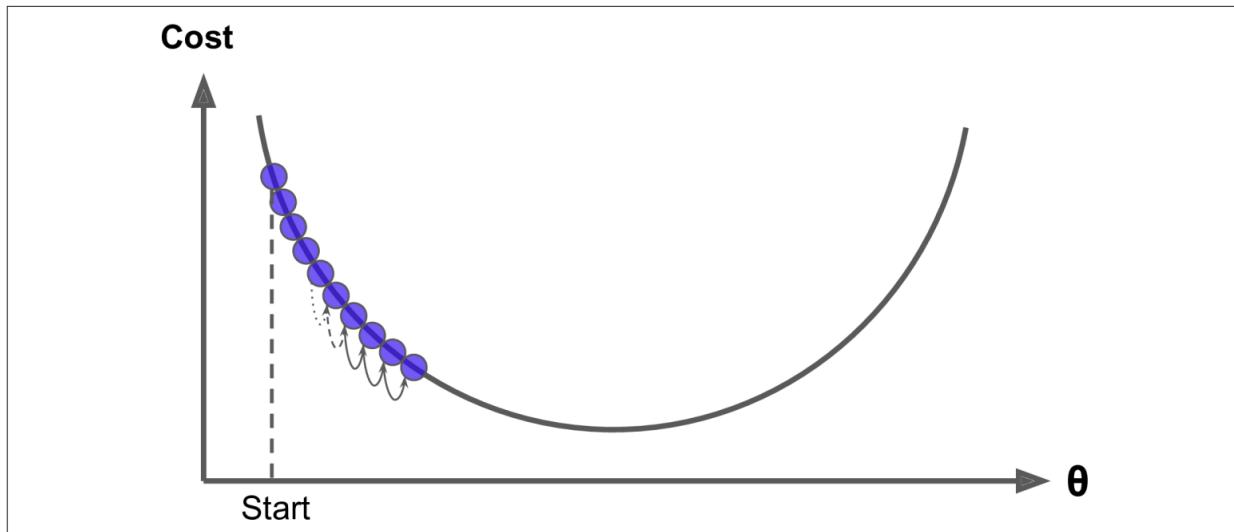
matrix([[      0,          0,          5, 874965758],
       [      0,          1,          3, 876893171],
       [      0,          2,          4, 878542960],
       ...,
       [ 942,     1187,          3, 888640250],
       [ 942,     1227,          3, 888640275],
       [ 942,     1329,          3, 888692465]], dtype=int64)
```

K: number of features, this value is used to approximate the utility matrix  $Y \in R^{MxN}$  by the product of two matrices  $X \in R^{MxK}$  and  $W \in R^{KxN}$ .

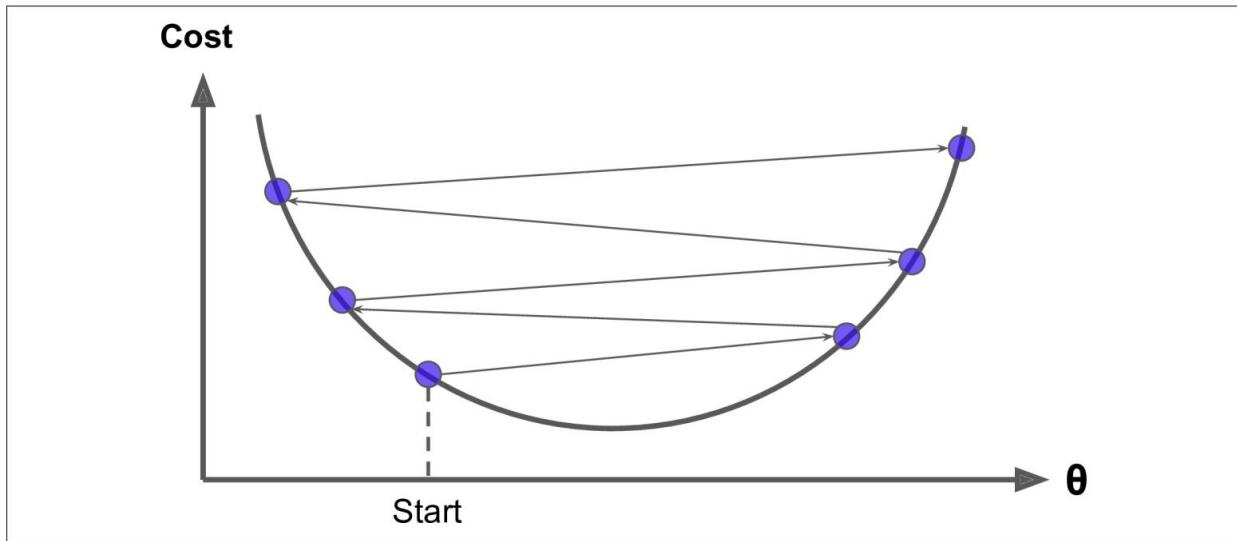
lam: lambda  $\lambda$

learning\_rate: learning rate of model

Example of learning rate too small:



Example of learning rate is too large



max\_iter: maximum iteration of gradient descent used to find  $X$  and  $W$ , it appears in fit().

print\_every: this value is for system to report processing during the training session.

n\_users: number of existed users

users\_ids: a list of all user ids

n\_items: number of existed movies

items\_ids: a list of all movie ids

n\_ratings: number of existed ratings

$X$ : a matrix has size  $n\_items \times K$

```
rs.X
✓ 0.0s
Python
array([[ 1.62476921, -0.18699334, -0.2235872 , ..., -0.17356157,
       1.20560839, -0.0139317 ],
      [-0.46995351, -2.11226004,  0.71141291, ...,  0.08181904,
       0.14493894, -1.84104401],
      [ 0.47721143,  1.29224774, -2.03788142, ..., -0.84664553,
       -0.02935711,  1.67643089],
      ...,
      [ 1.20005374, -1.23662352,  1.00701845, ..., -0.42077396,
       -2.2942159 , -1.59070611],
      [-0.01463026,  0.81649247,  0.36473521, ..., -1.58257472,
       0.28027851, -0.9680526 ],
      [ 1.24440303, -2.55109522,  1.05026741, ...,  0.83776664,
       0.12984461,  0.42833674]])
```

W: a matrix has size  $K \times n\_users$

```
rs.W
✓ 0.0s
Python
array([[-0.70215581, -1.38512364,  0.20842535, ...,  1.59358238,
       1.27215489,  1.74206167],
      [-1.96754498, -0.82362947,  1.02092632, ..., -0.17511587,
       0.5934071 , -1.81855495],
      [-0.57283232, -1.203771 , -2.40503652, ...,  0.7468749 ,
       0.30285366, -0.09902802],
      ...,
      [ 1.51474366, -1.3398714 ,  1.30850692, ..., -1.25718713,
       -0.14669722, -1.92611484],
      [ 0.98760926,  0.35513189, -0.82111377, ..., -0.12451902,
       -1.28860538, -0.38356291],
      [-0.82390424, -0.49598753,  1.1977729 , ..., -0.24305448,
       0.22599195,  0.91993007]])
```

b: item biases, an array with length of  $n\_items$ .

d: user biases, an array with length of  $n\_users$ .

## 6. Model explanation

With the input  $Y$ , we try to determine  $X, W, b$  and  $d$  such that the formula  $y_{mn} \approx x_m^T w_n + b_m + d_n$  will come close to value in  $Y$ .

Loss function is a way we calculate the difference between the value in  $Y$  and value calculated by  $X, W, b$  and  $d$ .

The way we find the right  $X, W, b$  and  $d$  is gradient descent with the update formula shown before.

When  $X, W, b$  and  $d$  are optimized, we can use them to calculate the prediction for unknown value in utility matrix  $Y$ .

## 7. Functions that affect the model

### FR.1.1: User Registration

When a new user signs up, the model needs to update with  $n\_users$ ,  $users\_ids$ ,  $W$  and  $d$ .

### FR.3.1: User Create Rating, FR.3.3: User Update Rating, FR.3.4: User Delete Rating

When a new user handles a rating, the model should update  $W$  and  $d$  for that user.

### FR.7.3: Admin Add New Movie

When an Admin adds a new movie, same as FR.1.1, we need to update the model.

## 8. Model application

When Backend component started, there will be an object of matrix factorization in extensions.py.

There are 5 initial inputs: Y, X, W, b, d.

For X, W, b and d, we load it with saved trained model in database, if there is nothing in database, we will load it in the csv file I prepared in Data Preparation section.

```
# Load models for mfcf model if the model does not exist in the database
W = np.asarray([])
X = np.asarray([])
d = np.asarray([])
b = np.asarray([])

if len(list(db["model"].find())) == 0:
    W = np.loadtxt('./data/W.csv', delimiter=',')
    X = np.loadtxt('./data/X.csv', delimiter=',')
    d = np.loadtxt('./data/d.csv', delimiter=',')
    b = np.loadtxt('./data/b.csv', delimiter=',')

    wf = {"name": "W", "value": bson.Binary(pickle.dumps(W, protocol=2))}
    xf = {"name": "X", "value": bson.Binary(pickle.dumps(X, protocol=2))}
    df = {"name": "d", "value": d.tolist()}
    bf = {"name": "b", "value": b.tolist()}

    db["model"].insert_one(wf)
    db["model"].insert_one(xf)
    db["model"].insert_one(df)
    db["model"].insert_one(bf)

else:
    for model in db["model"].find():
        if model['name'] == "W":
            W = np.asarray(pickle.loads(model['value']))
        elif model['name'] == "X":
            X = np.asarray(pickle.loads(model['value']))
        elif model['name'] == "d":
            d = np.asarray(model['value'])
        elif model['name'] == "b":
            b = np.asarray(model['value'])
```

For Y, we need to load all ratings in database

```
# Load all ratings as utility matrix for mfcf model
ratings_cursor = db['rating'].find()
ratings_dataframe = pd.DataFrame(list(ratings_cursor), columns=[ 
    'userId', 'movieId', 'rating', 'timestamp']).astype({'userId': int,
'movieId': int, 'rating': int, })
ratings_matrix = np.asmatrix(ratings_dataframe)
rate_train, rate_test = train_test_split(
    ratings_matrix, test_size=0.2, random_state=10)
```

Finally, we can create a model object that live when server running:

```
# Load all ratings as utility matrix for mfcf model
ratings_cursor = db['rating'].find()
ratings_dataframe = pd.DataFrame(list(ratings_cursor), columns=[ 
    'userId', 'movieId', 'rating', 'timestamp']).astype({'userId': int,
'movieId': int, 'rating': int, })
ratings_matrix = np.asmatrix(ratings_dataframe)
rate_train, rate_test = train_test_split(
    ratings_matrix, test_size=0.2, random_state=10)

# mfcf machine Learning model
mfcf_model = MatrixFactorization(
    Y=ratings_matrix, K=50, lam=.01, Xinit=X, Winit=W, bInit=b, dInit=d,
learning_rate=50, max_iter=30)
```

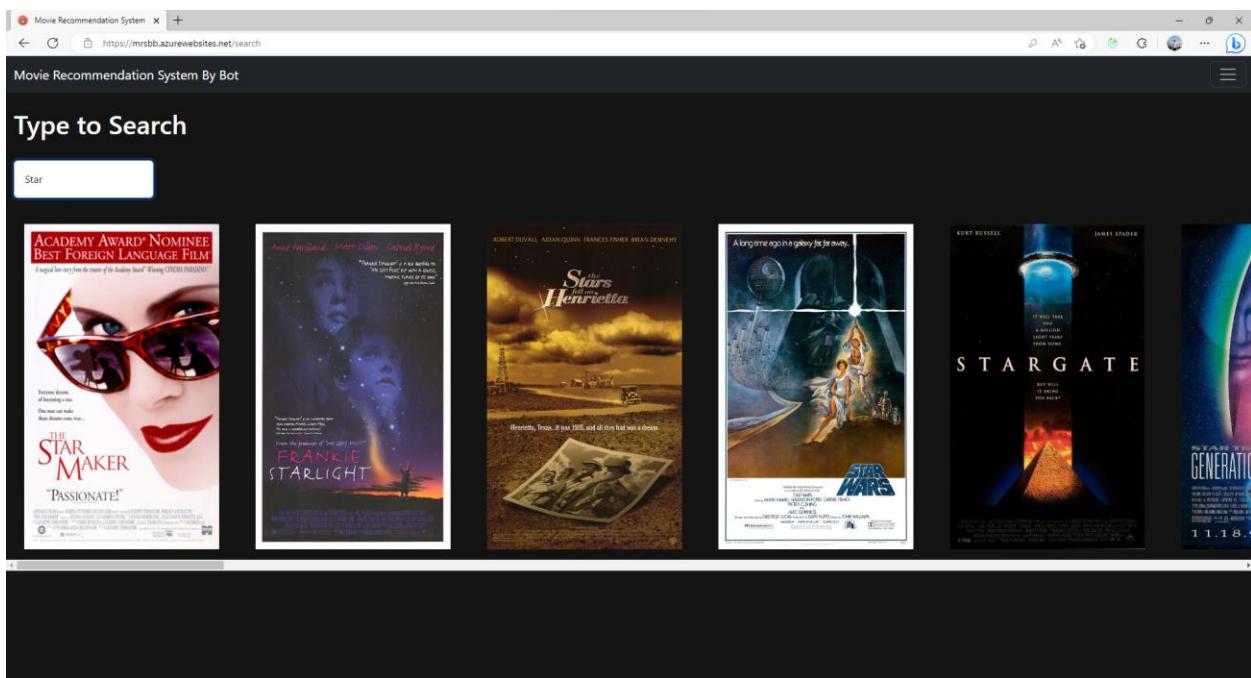
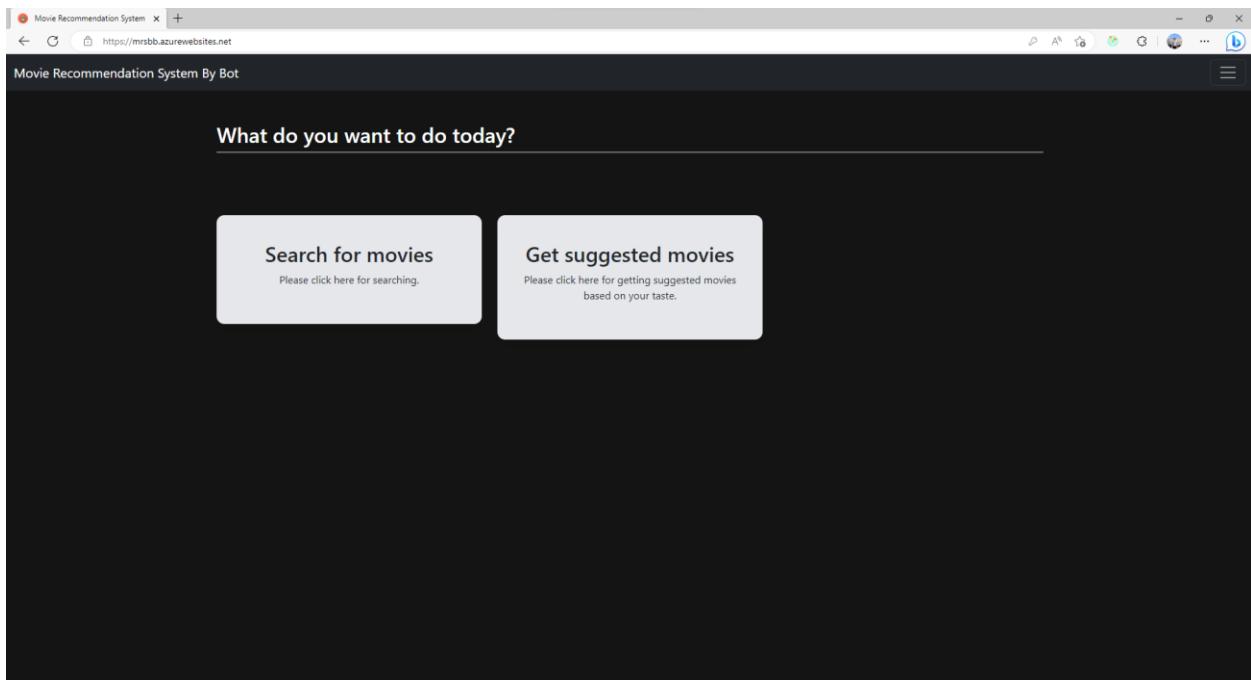
Whenever a function that affects model is called, it will call directly to the only one live object.

For example, please refer to the register method in People class.

## X. Production Screenshots

The screenshot shows a web browser window for the "Movie Recommendation System". The URL in the address bar is <https://mrsbb.azurewebsites.net/signup>. The page title is "Movie Recommendation System By Bot". On the left, there is a logo and text: "MRSBB" in bold, followed by "Movie Recommendation System By Bot". On the right, there is a "Sign Up" form titled "Create your Account". The form includes fields for First Name\*, Last Name\*, Email\*, Password\*, Confirm Password\*, Birthday (set to 21 Apr 2022), Gender (Female or Male), and a "Sign Up" button. Below the form is a link "Already have an account? [Login](#)".

The screenshot shows a web browser window for the "Movie Recommendation System". The URL in the address bar is <https://mrsbb.azurewebsites.net/login>. The page title is "Movie Recommendation System By Bot". On the left, there is a logo and text: "MRSBB" in bold, followed by "Movie Recommendation System By Bot". On the right, there is a login form with email input ("iamtienn@gmail.com"), password input ("....."), a "Log In" button, a "Forgot password?" link, and a "Create new account" button.



Movie Recommendation System x + https://mrsbb.azurewebsites.net/movie/259

Movie Recommendation System By Bot

## Star Wars: Episode IV - A New Hope (1977)



--  
Genres  
Action|Adventure|Fantasy|Sci-Fi  
--  
Your rating for this movie is:  
 Delete Rating  
--  
You can change your rating by clicking on the stars above  
--

Movie Recommendation System x + https://mrsbb.azurewebsites.net/suggest

Movie Recommendation System By Bot

## Top 10 Recommended Movies For You



Generations  
Two Captains, One Destiny  
11.18.94

ALFRED HITCHCOCK'S Production of LIFEBOAT JOHN STEINBECK  
TALLULAH BANKHEAD BENDIK SLEZAK ANDERSON HODIAK HULL ANGEL CROWNE LEE RITCHIECK

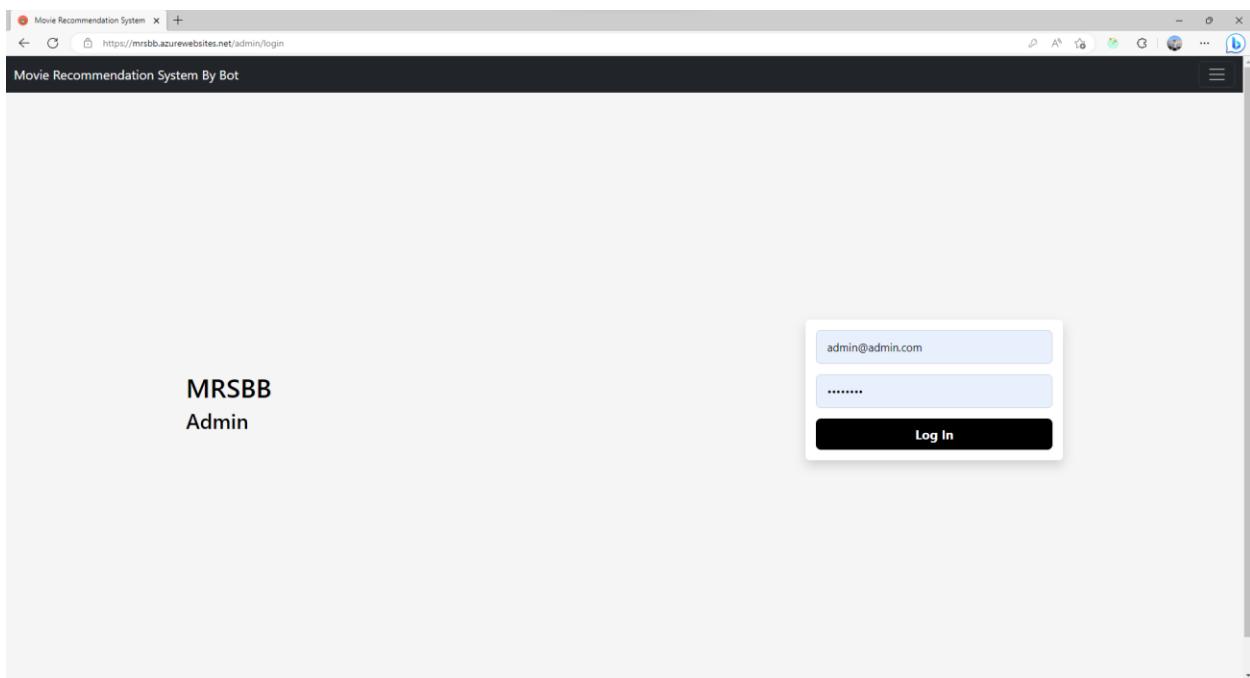
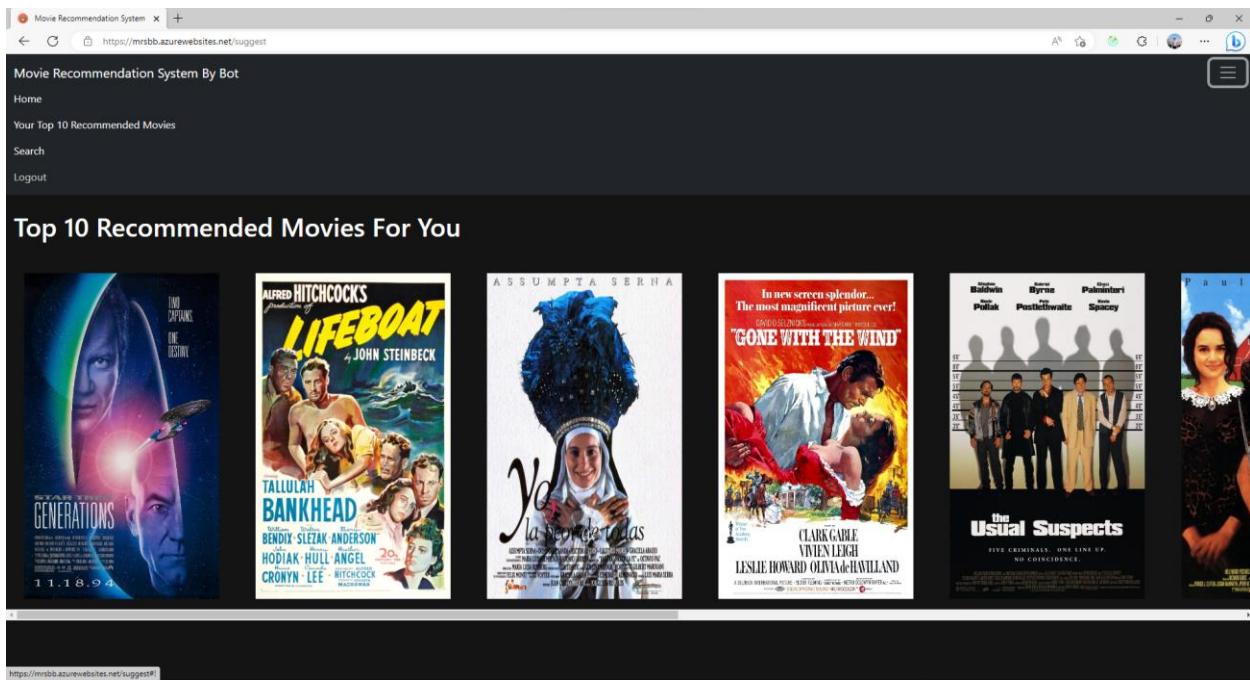
ASSUMPTA SERINA  
Yo la novia de tus padres

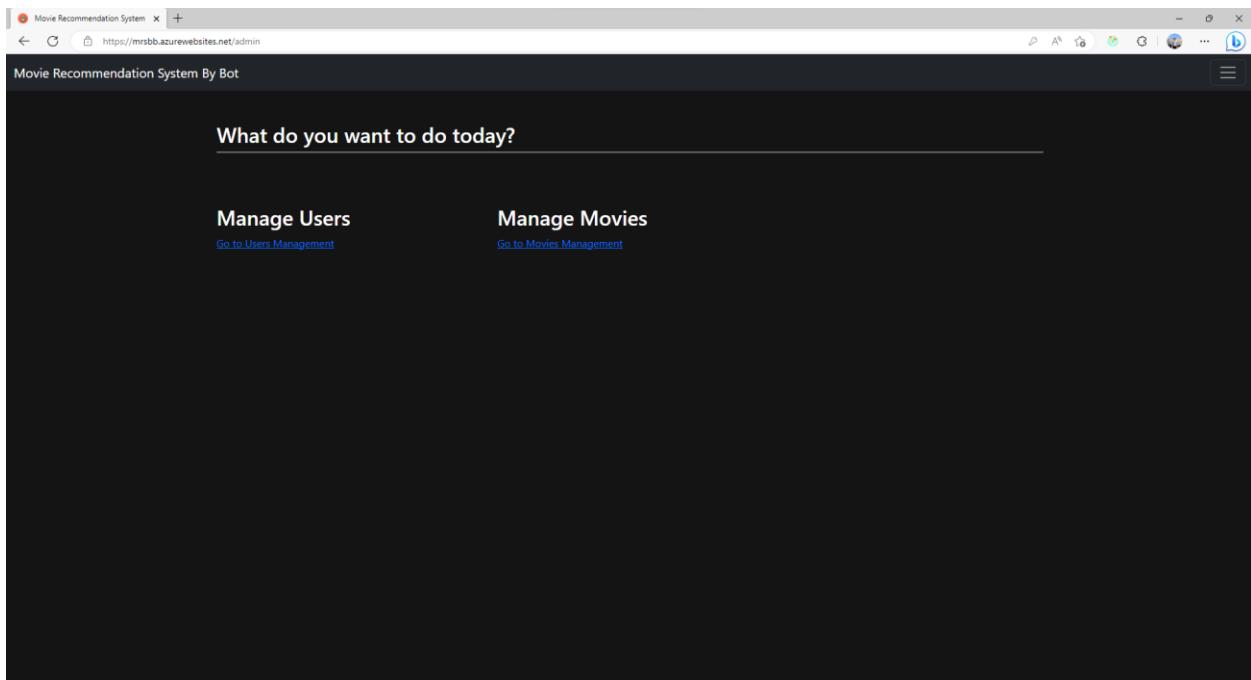
In new screen splendor... The most magnificent picture ever!  
GONE WITH THE WIND CLARK GABLE VIVIEN LEIGH LESLIE HOWARD OLIVIA DE HAVILLAND

ROBERT REDFORD DENNIS HOPPER ROBERT DUVALL JAMES CAAN SCOTT WALTERS RICHARD DREYFUSS RON LIU PAUL NEWMAN

the Usual Suspects FIVE CRIMINALS. ONE LINE UP. NO CONCIENCE.

Paul





The screenshot shows a dark-themed web application window titled "Movie Recommendation System By Bot". The title "Users Management" is prominently displayed at the top. A sub-instruction "Click to a row to turn into specific user page." is visible. Below this, a table lists user information:

ID	Email	Surname	Name	Birth Day	Gender
6040	iamtiennng@gmail.com	Nguyen	Tien	21-04-1998	M

A small "Snipping Tool" window is overlaid on the bottom right of the screen, indicating that a screenshot has been taken and saved. The tool's status bar shows "Screenshot copied to clipboard and saved" and "Select here to mark up and share the image".

Movie Recommendation System x + https://mrsbb.azurewebsites.net/admin/movies

## Movies Management

Click to a row to turn into specific movie page.

Add new Movie

ID	Title	Genres
0	Toy Story (1995)	Animation Children's Comedy
1	Jumanji (1995)	Adventure Children's Fantasy
2	Grumpier Old Men (1995)	Comedy Romance
3	Waiting to Exhale (1995)	Comedy Drama
4	Father of the Bride Part II (1995)	Comedy
5	Heat (1995)	Action Crime Thriller
6	Sabrina (1995)	Comedy Romance
7	Tom and Huck (1995)	Adventure Children's
8	Sudden Death (1995)	Action
9	GoldenEye (1995)	Action Adventure Thriller
10	The American President (1995)	Comedy Drama Romance
11	Dracula: Dead and Loving It (1995)	Comedy Horror
12	Balto (1995)	Animation Children's
13	Nixon (1995)	Drama
14	Cutthroat Island (1995)	Action Adventure Romance
15	Casino (1995)	Drama Thriller
16	Sense and Sensibility (1995)	Drama Romance

Movie Recommendation System x + https://mrsbb.azurewebsites.net/admin/movies/1

## Movie ID: 1



Movie Title: Jumanji (1995)

Movie Genres: Adventure|Children's|Fantasy

Delete

