Department of Computer Science, Faculty of Engineering, The University of Hong Kong

COMP4801 Final Year Project

AI Student Advisor

FYP18024 Final Report

By Kwok Cheuk Lum (3035269992)

Other team members:

Chan Lap Kiu (3035269784)

Chau Chun Wing (3035281275)

Woo Chong Yiu (3035271012)

Supervised by Dr. T. W. Chim
SUMMARY/ ABSTRACT

Which course to take is a common question that university students need to face. University students often want to take a course that they can score better but ironically, they generally do not know they can perform the best in which course. There are several online tools and databases providing information and advices on choosing majors or courses. Nonetheless, these tools only focus on ‘how to provide or display the collected data’ but not ‘utilizing the data to provide useful suggestions’. These tools simply show the data without processing them and leave the users to interpret the data into useful information on their own. A mobile application that utilizes a trained machine learning model for providing grade prediction function is developed in this project to help get university students out of this predicament. Instead of directly displaying the data, machine learning can be used to provide more precise advices on study path by utilizing the huge amount of data. Users can enter their course grade history into the mobile application to let it predict the grade he or she will get for an untaken course.

Besides, some students like to know whether the content of the courses fit their own interests. Therefore, an academic advising community is also implemented in the mobile application which allows users to exchange their views and first-handed experiences on different courses. Students can also discuss topic related to academic planning and course selection. Through the academic advising community, the students can have a better overview on different courses.

This report talks about the background, methodology, results and future works of the project, the AI Student Advisor. The final deliverables of this project include an Android application, an iPhone Operating System (iOS) application, a web server application and a trained machine learning model.
ACKNOWLEDGEMENT

I would like to express my greatest gratitude to the following people for their help. Without their help, this project cannot be completed.

Dr. T. W. Chim: I would like to thank Dr. Chim for being our project supervisor. He was always supportive and gave us timely and helpful advice.

Dr. S. M. Yiu: I would like to thank Dr. Yiu for giving us a lot of useful feedback during our interim presentation so that we can make appropriate amendments accordingly.

The staff of the department: I would like to thank the staff of the Department of Computer Science of the University of Hong Kong for scanning and providing us the past students’ transcripts so we can input to the system. Without the data, our machine learning model could not be trained.

HKU CS students: I would like to thank every HKU CS student who had contributed in our project by providing valuable real-world data of past academic performance and user feedbacks. Without the data, our project is not achievable.

My team members. I would like to thank my team members for the hard work they contributed to this project.
TABLE OF CONTENTS

LIST OF FIGURES ................................................................................................................6
LIST OF TABLES ....................................................................................................................7
ABBREVIATIONS ...................................................................................................................8
1. INTRODUCTION ..............................................................................................................9
   1.1. Background ................................................................................................................9
   1.2. Previous works ............................................................................................................10
   1.3. Objectives ..................................................................................................................11
   1.4. Scope ........................................................................................................................11
   1.5. Deliverables ..............................................................................................................12
   1.6. Outline of report ......................................................................................................12
2. METHODOLOGY ............................................................................................................14
   2.1. Development approach ............................................................................................14
   2.2. Software architecture ..............................................................................................15
       2.2.1. iOS ..................................................................................................................15
       2.2.2. Android .........................................................................................................16
       2.2.3. Web service ....................................................................................................16
   2.3. Programming languages, frameworks and services .................................................16
   2.4. Functions design ......................................................................................................18
       2.4.1. Grade Prediction ...............................................................................................18
       2.4.2. View posts in order and filter (My posts/ Bookmarked/ Hot/ Chronological) ..............................................................................................................19
       2.4.3. Create post .......................................................................................................21
       2.4.4. Bookmark post .................................................................................................22
       2.4.5. Add comment ....................................................................................................22
       2.4.6. Authentication ..................................................................................................23
   2.5. Data collection methods ...........................................................................................25
   2.6. Model training ..........................................................................................................25
   2.7. Project schedule ......................................................................................................26
3. USER INTERFACE .............................................................................................................27
   3.1. View posts in order and filter (My Posts/ Bookmarked/ Hot/ Chronological) ..................................................................................................................29
   3.2. Create post ................................................................................................................32
   3.3. Add/ view comments .................................................................................................34
   3.4. Edit post .....................................................................................................................36
   3.5. Login/ Register .........................................................................................................37
4. RESULTS .............................................................................................................................40
4.1. Machine learning training results..........................................................40
  4.1.1. Metrics .........................................................................................40
  4.1.2. Preprocessing ...............................................................................40
  4.1.3. Grade prediction results .................................................................40
  4.2. Encountered challenges ....................................................................41

5. CONCLUSION AND FUTURE WORKS ..................................................43
REFERENCES .................................................................................................44
LIST OF FIGURES

Figure 1. gpa.ai registration page .................................................................10
Figure 2. An agile software development cycle diagram, showing the 6 major steps to go through during development with “MEET” as the first step to start with [5]...14
Figure 3. A Model-View-Controller (MVC) architecture diagram showing the interactions between the three different types of objects, namely model, view and controller [6]...........................................................................................................15
Figure 4. A REST architecture diagram showing the connection between components [9]..............................................................................................................................................................................................16
Figure 5. System flow of grade prediction ......................................................................18
Figure 6. System flow of viewing posts in order and filter .........................................................19
Figure 7. Algorithm of calculating hot posts [11]...............................................................21
Figure 8. System flow of create post ..............................................................................21
Figure 9. System flow of bookmark post ............................................................................22
Figure 10. System flow of add comment ..........................................................................22
Figure 11. System sequence diagram of authentication.......................................................23
Figure 12. Layout of Gmail app.......................................................................................27
Figure 13. Layout of iOS version Facebook app [13].........................................................28
Figure 14. View posts screen.........................................................................................29
Figure 15. UI for triggering “pull-to-refresh”....................................................................30
Figure 16. UI when triggering “pull up to load more” ......................................................30
Figure 17. Navigation drawer of AI Student Advisor .........................................................31
Figure 18. UI of create post.........................................................................................32
Figure 19. UI of the alert shown for adding a tag to the post. “Mobile App” is the tag entered by the user..........................................................................................................................32
Figure 20. UI showing how a tag is placed. ....................................................................33
Figure 21. UI of the add and view comments screen without comments .........................34
Figure 22. UI of the add and view comments screen without comments .........................34
Figure 23. Screen capture showing the input panel pushed up by the keyboard and the send button lightens up when there is input. .................................................................35
Figure 24. UI of the login screen ................................................................................37
Figure 25. UI of the forget password screen .................................................................38
Figure 26. UI for the screen to verify the token for resetting password .........................38
Figure 27. The UI of step 1 in registering an account .....................................................39
Figure 28. The UI of step 2 in registering an account .....................................................39
Figure 29. The UI of step 3 in registering an account .....................................................39
LIST OF TABLES

Table 1. Schedule of this project .................................................................26
Table 2. Training results without preprocessing ..............................................40
Table 3. Training results with preprocessing ..................................................41
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>iOS</td>
<td>iPhone Operating System</td>
</tr>
<tr>
<td>GPA</td>
<td>Grade Point Average</td>
</tr>
<tr>
<td>HKU</td>
<td>The University of Hong Kong</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>kNN</td>
<td>k-nearest neighbors</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-View-Controller</td>
</tr>
<tr>
<td>MVP</td>
<td>Model-View-Presenter</td>
</tr>
<tr>
<td>MVVM</td>
<td>Model-View-ViewModel</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>UX</td>
<td>User Experience</td>
</tr>
<tr>
<td>WALS</td>
<td>Weighted Alternating Least Squares</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1. Background

University students always face questions like “Which course should I take?” or “Which course will I perform the best?”. They often make wrong decisions due to the incapability of estimating themselves or insufficient planning. Around half of the university freshmen have no idea what to choose as their major when they get admitted to college [1]. According to the Minister for Education and Training of Australia, Simon Birmingham, around one third of the students do not complete the first subject they are admitted to because they do not think thoroughly before choosing the subject [2]. These lead to a waste of time and money of both the students and tax payers [2].

University students mostly prioritize courses that they are potentially going to take by how likely they are going to score high in a course. Students currently get advices by asking past students or advisors in person or viewing some unofficial online databases. Then they assess their own situation to see which courses they are most likely to perform the best in. As shown from the above statistics, these methods are faulty. The causes of this situation include unreliable online grade statistics, and the seniors’ advice might not fit the students’ own scenarios. These result in students ending up with unsuitable courses and having an undesirable result.

In this project, a mobile application that integrates machine learning to give suggestions is built to solve the problem. The course suggestion system can be viewed as a recommendation system, which is similar to those recommendation systems used in online shopping websites. If the course grade histories of different students are collected, they can be viewed as the user behavior in online shopping recommendation system. Therefore, the problem can be modeled as a collaborative filtering problem and the estimated grade results of untaken courses can be found using matrix factorization.

In addition, students need to know whether the course content is really interesting or suits their needs. Therefore, an academic advising community is also developed and added to the mobile application which allows users to exchange their views and first-
handed experiences on different courses. Students can help each other out on course selection problems via this platform as well.

1.2. **Previous works**

There are a few existing projects which have the similar idea of providing an online academic advisor. These projects include gpa.ai and “Next-Term Student Performance Prediction: A Recommender Systems Approach”.

![gpa.ai registration page](image)

**Figure 1. gpa.ai registration page**

The gpa.ai project provides all distribution of course grades in the university for the user. It requires the user to complete a registration progress which the user needs to login with Facebook and university account. Then gpa.ai uses the user’s information to login to the university account to check the user’s course results. Then the user can use the provided chatbot to give some suggestions on the service. Finally, the user can also view all the course grade distribution.

As mentioned earlier, gpa.ai is one of the services which just shows the data they have without processing them to give suggestions. Therefore, the users are left to interpret the data by themselves. An advantage of this project over some other projects is that it provides accurate data because it requires the user to login with university account and requires the course data to be verified. However, there is a severe security issue for this project. Since it is not using OAuth to let the user login to university account,
the user needs to directly hand the username and password to them. This can lead to information leakage and holds back users who do not trust them.

For “Next-Term Student Performance Prediction: A Recommender Systems Approach”, it explores the accuracy of different machine learning algorithms to predict a student’s performance in the next semester [3]. The problem was modeled in the same way as it is in this project. Course grade history of students were collected to form a matrix. The matrix was then factorized to estimated matrices to represent latent factors which were then used to make predictions on course grades. The project focus on which matrix factorization method produces the best result and did not explore in making a frontend application.

1.3. Objectives

To tackle the aforementioned dilemma effectively, three objectives were laid out for this project.

The first objective is to use a machine learning model for building a recommendation system in a form of mobile application that help students find courses they will perform the best easily and accurately.

The second objective is to create an online community that allows students to learn more about the content of different courses.

The third objective is to introduce intriguing social platform, gamified elements and attractive user interface (UI) to attract universities students to use it.

1.4. Scope

The mobile application is only developed for the iOS and Android platform but not other platforms like Blackberry and Windows. This decision was made because iOS and Android take up more than 96% of the mobile OS market share [4], which covers the need of serving the students and meets the general standard of the industry as well. The mobile application is only introduced to computer science students studying in The University of Hong Kong (HKU). It is constrained so because the data of other departments in the HKU are not accessible. For future developments, the project can
be scaled up to give academic advices to all undergraduates of HKU or even other universities.

1.5. Deliverables

The final deliverables of this project include an Android application, an iOS application, a web server application and a trained machine learning model.

Features of the mobile application includes course grade prediction, course suggestions and an online community for course content sharing.

The course grade prediction feature allows a user to predict his or her course grade for an untaken course. The mobile application predicts the grade by using the course grade history given by the user.

The course suggestion is a feature similar to the course grade prediction. The AI student advisor predicts the grades that the student will get for all courses and sort the list in descending order of the predicted grades.

The online community allows users to ask and answer each other’s questions on course content. Users may also upvote or downvote one another’s answers to indicate they are useful or useless respectively.

1.6. Outline of report

The remainder of this report proceeds as follow. The methodology of how to build the mobile application and the corresponding UI design are provided and discussed in chapter 2. The whole project’s development approach will be covered first. After that, the choices of system architectures on Android, iOS and web service are discussed and choices are justified. Furthermore, the reasons for choosing Node.js as the backend framework and MongoDB as database are also provided in this chapter. Data collection methods, modeling training and project schedule are also discussed. In chapter 3, the UI of the project is discussed. The choices of the UI design pattern are justified, and the user interfaces are described in detail. Then the results of the project are discussed in chapter 4. It includes the metrics used to measure the accuracy, effect of using preprocessing and results of the machine learning model training and the difficulties that were encountered. In chapter 5, which is the final chapter, the report
ends with a conclusion of the project and future works that can be done.
2. METHODOLOGY

In this chapter, the technological choices and how the application is built are covered. The agile software development cycle is chosen due to its short and fast iteration cycles which are needed in this project. Software architectures, programming languages and frameworks were chosen carefully with the criteria of fitting the purpose of this project, compatibility with the respective platforms, stability and community size.

2.1. Development approach

![Agile software development cycle diagram](image)

*Figure 2. An agile software development cycle diagram, showing the 6 major steps to go through during development with “MEET” as the first step to start with [5]*

In this project, agile development method is used (see Figure 2) so that the project is more flexible and adaptive to changes comparing with traditional waterfall approach. It works by breaking the project into smaller parts and then plan, design, build and test for each of these smaller parts rapidly. Each of these smaller parts is called an iteration. After each iteration is completed, that part is deployed to get feedbacks and then improved in the next iteration according to the feedbacks.

This kind of flexibility is needed in this project because a lot of feedbacks will be given by our advisor. The users may also give us feedbacks on whether the application features are attractive and useful enough. The agile development approach allows us to adopt the feedbacks and make necessary changes effortlessly. It also
allows us to mitigate potential risks earlier as feedbacks are given earlier.

2.2. **Software architecture**

2.2.1. **iOS**

![Diagram of Model-View-Controller (MVC) architecture]

*Figure 3. A Model-View-Controller (MVC) architecture diagram showing the interactions between the three different types of objects, namely model, view and controller [6]*

For the iOS mobile application, the MVC architecture is used such that the objects in the application is a model, view or controller (see Figure 3). This architecture is officially suggested in the Apple Inc. documentation as a good design for a Cocoa application. In fact, many of the Cocoa Touch APIs provided by Apple were built using the MVC architecture [7]. Therefore, this architecture’s performance and compatibility with iOS is the best.

Other than the performance and compatibility advantages that are specific on iOS, the codes written in MVC architecture are generally more reusable, thus making the application more extensible. Moreover, the MVC architecture was introduced in Smalltalk-80 version of Smalltalk that was released in 1980 [8]. It proves that this architecture was widely tested out and so stable that it is still used nowadays.

However, it is commonly known that MVC architecture might cause the problem of having fat or massive ViewControllers in iOS applications. This is because the UIViewController classes sometimes consist of both the view and controller codes which makes the codebase huge inside these files. In light of this, service layer,
categories, subclassing and Interface Builder are used extensively to alleviate this problem in this project.

Even though fat ViewControllers might exist in the project, the MVC architecture is chosen due to its stability, extensibility and compatibility which outweighs the downside.

2.2.2. Android

The MVC architecture pattern is also enforced in the Android application. Although other variants like MVVM and MVP proposed were proposed, the MVC is chosen due to the team’s familiarity.

2.2.3. Web service

![REST architecture diagram](image)

Figure 4. A REST architecture diagram showing the connection between components [9]

The web services that are implemented on the backend conforms to REST architecture and Hypertext Transfer Protocol (HTTP). It lowers the coupling between the client and the server by communicating via a specified format so that it is more scalable (see Figure 4). The REST architecture is also simpler to use so the project can be developed faster.

2.3. Programming languages, frameworks and services

For the frontend mobile application, it is developed natively on both the Android and
iOS platforms. New mobile frameworks including React Native and Flutter were reviewed but they were not chosen because the communities of these technologies are relatively small comparing to the native codes communities. It means that there are not enough existing frameworks and support to suit the project’s rapid development. Native libraries can be used in React Native projects but some flaws were also spotted when applications are ejected to use these native libraries. Moreover, the major benefit of React Native and Flutter, that is the reusability of codes, cannot be fully utilized in this project because the UI design of the Android and iOS versions are different to facilitate the habit of different OS users. After these thorough considerations, the decision made was to fall back to using the widely tested and robust native code development.

Objective-C and Java were the chosen languages to develop for iOS and Android platforms respectively. The more modern languages, Swift and Kotlin, were not chosen respectively because our team is more familiar with Objective-C and Java. The newer languages does not offer a significant improvement in this small project as well.

Some third party libraries and frameworks are also adopted to increase our development pace in frontend. For example, AFNetworking and OkHttp are used on the iOS and Android platforms respectively to make HTTP requests and handle the network errors easily. Network request codes can be reduced significantly in ViewControllers and Activities. This leads to a better separation of concerns and prevent the fat ViewController problem mentioned previously. DragRefreshAndLoadMoreTable and “Ultra Pull To Refresh with Load More” are used on the iOS and Android platforms respectively to implement the features of “pull-to-refresh” and “pull up to load more”.

For backend, Node.js is used together with the MongoDB database as an usual practice. Node.js is chosen because it supports async functions which help in improving the backend’s performance. MongoDB is chosen for its scalability as a NOSQL database. Since settings up servers is a cumbersome job, a PaaS called Heroku is used in this project for setting up the backend environment. The backend environment can be set up conveniently with this service which shortens the development time a lot. This is particularly important because the development time
span of this project is quite short.

Gitlab is used for managing the source codes of this project, no matter the frontend or the backend codes. It is chosen because it is free to create a private project on GitLab. The branching tools of GitLab also allow the team to develop and write codes simultaneously and combine them in the future seamlessly. GitLab was also designed in a way to facilitate continuous integration and continuous development that fits the agile development approach well.

Tensorflow together with Cloud Machine Learning Engine in Google Cloud Platform are used in this project. They were chosen because they use the state-of-the-art weighted alternating least squares (WALS) algorithm for constructing a recommendation system [10].

2.4. Functions design

Six major functions are included in the mobile application: Grade Prediction, View posts in order, Create post, Bookmark post, Add comment, Authentication. This section proceeds with detailed breakdown of the user flow for each function and data flow of the mobile application for each function

2.4.1. Grade Prediction

![System flow of grade prediction](image)

*Figure 5. System flow of grade prediction*

A user can get a grade prediction for an untaken course through the mobile
application. The data of past course grade results, which is inputted by the user, is sent from the mobile application to the server (see Figure 5, step 1). Then the server gets all the trained user data from the database (see Figure 5, step 2) and finds the index of another user that is the most similar with the current user (see Figure 5, step 3). After that, this record is used to get the predicted grade of an untaken course using the trained machine learning model stored on the server (see Figure 5, step 4). At last, the server passes the result back to the mobile application (see Figure 5, step 5).

The machine learning model is not updated and trained again each time a new user demands a grade prediction due to performance issue. If the model is trained every time, it will take approximately 10 minutes for each prediction, which is undesirable. Therefore, another similar user’s record is used instead. Details of this problem is discussed in chapter 2.6.

2.4.2. View posts in order and filter (My posts/ Bookmarked/ Hot/ Chronological)

![System flow of viewing posts in order and filter](image)

*Figure 6. System flow of viewing posts in order and filter*

A user can view the posts with the one of the following filter or order: viewing self-created posts, viewing bookmarked posts, viewing hot posts and viewing posts in chronological order. For self-created posts, the mobile application filters the posts and only show the posts created by the user. For bookmarked posts, the mobile application filters the posts and only show the posts bookmarked by the user using the bookmark function. For hot posts, the mobile application shows all posts but order the posts according to their hotness. For chronological order, the mobile application shows all posts according to the time they were created.

For all filters and orders, the data flow works the same. Only the parameters in the
mobile application’s request is different. The mobile application sends a request to the server that specifies how the posts should be ordered or filtered first (see Figure 6, step 1). Next, the server finds the posts first ten posts in the corresponding order or filter (or less if there are less than ten posts in total in the database that fulfill the filter) from the database (see Figure 6, step 2). Finally, the server sends a response of the posts taken from the database back to the mobile application (see Figure 6, step 3). When the user scrolls to the bottom of the community screen, the mobile application will send a request to get the subsequent ten posts with the same order or filter. The user can continue to scroll until there are no more posts to load.

To order the posts according to their hotness, a hotness algorithm is used. This algorithm references the hotness algorithm (see Figure 7) of the popular online open source forum, Reddit. A slight modification is introduced to fit the purpose of this project. Instead of upvotes and downvotes, the variable x represents the number of comments. The modification is made because there are no upvotes and downvotes for posts in this project. Although a modification is needed, this algorithm is still chosen as reference due to two features of it.

First, the creation time of a post has an impactful effect on its hotness. It is undesirable to have old posts sticking at the top of hot posts for a long time in a forum. Newer posts should preferably rank higher than most old posts. Therefore, the great impact of creation time on the hotness is an advantageous feature. It allows newer posts to have a head-start because they are newer. Another feature of the algorithm is that the impact of new comments degrades when the number of comments increase because logarithm is used. For example, the first ten upvotes and downvotes difference (after the modification, the first 10 comments) is valued as high as the next hundred (Because \( \log_{10} 0 = 1 \), \( \log_{10} 10 = 1 \) and \( \log_{10} 100 = 2 \)).
Given the time the entry was posted at 7:46:43 a.m. on December 8, 2005, we have $t_i$ as their difference in seconds:

$$t_i = A - B$$

and $x$ as the difference between the number of up votes $U$ and the number of down votes $D$

$$x = U - D$$

where $y = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$

and $z$ as the maximal value of the absolute value of $x$ and $1$

$$z = \begin{cases} |x| & \text{if } |x| \geq 1 \\ 1 & \text{if } |x| < 1 \end{cases}$$

we have the rating as a function $f(t_i, y, z)$

$$f(t_i, y, z) = \log_{10} z + \frac{y t_i}{45000}$$

**Figure 7. Algorithm of calculating hot posts [11]**

### 2.4.3. Create Post

A user can create a post in the community. The user can do so by clicking the create post button first. Then the user needs to enter the details of the post, including title, content, and tags, and submit them in the mobile application. The mobile application then sends a request containing the above details together with the user id to the server (see Figure 8, step 1). Next, the server will add the document to the database (see Figure 8, step 2) and return a successful message if it successfully adds the document to the database (see Figure 8, step 3). If the mobile application receives the successful message, the mobile application will return to the home page which is the page that shows the latest posts, so the user is able to see his post is on the top of the page. Otherwise, the mobile application will pop up an error message according to the
error message received from the server.

2.4.4. **Bookmark post**

![Figure 9. System flow of bookmark post](image)

A user can bookmark a post so that it can be viewed again conveniently via the “view post with filter” function. When the user clicks the bookmark button of a post in the mobile application, a bookmark request with the post id will be sent to the server (see Figure 9, step 1).

Then the backend server will add the user id to the post document in the database based on the post id received and return a successful message if it successfully adds the document to the database (see Figure 9, step 2). If the client-side receive the successful message (see Figure 9, step 3), the client-side application will change the bookmark button to black to show the post is bookmarked. Otherwise, the client-side will pop up an error message according to the error message received from the backend server.

2.4.5. **Add comment**

![Figure 10. System flow of add comment](image)

A user can reply to a post by writing a comment. The user needs to fill in the content
of the comment and submit it. A comment request is sent from the mobile application to the backend server (see Figure 10, step 1). The user id and post id are added to the request by the mobile application. After the backend server receives the reply post request, it inserts the document into the comment database (see Figure 10, step 2). If the record insertion is successful, a successful message is sent to the mobile application (see Figure 10, step 3). Next, the mobile application appends the new comment to the existing comments locally, so no additional requests are required. If the record insertion is unsuccessful, the mobile application will pop up an error message according to the error message received from the backend server.

2.4.6. Authentication

![System sequence diagram of authentication](image)

Figure 11. System sequence diagram of authentication

The authentication function allows users to register or log in the application. The system sequence diagram shows how the sequence of events occur during the authentication flow (see, Figure 11).

A user needs to register for an account before logging in. The user needs to fill in his
email, password, and other personal information in the mobile application. Then, the mobile application makes some preliminary checks. It makes sure that the entered email and password are not empty and the password is longer than six characters. Afterwards, a registration request with the entered information is sent to the backend server. Next, the backend server checks whether the email is already in use or not in the database. If not, the password is hashed and encrypted and inserted into the database. Then, a one time token is sent by the backend server to the user’s email to verify his or her identity. Since hackers might try to guess the one time token, this unauthenticated account and one time token are only stored in the database for 15 minutes. If the time limit reaches and the user does not enter and submit the correct token, the unauthenticated account and the one time token are deleted. The user needs to restart the whole registration process. If the one time token verification is successful, the user is automatically logged in and can start using the application directly.

Passwords are hashed before saving into the database. A hashed password cannot be recovered into the original password. Therefore, a hacker cannot get the original password even if he or she has access to the hashed value stored in the database.

Bcrypt is used to hash and encrypt the passwords rather than other alternatives in the market as it was specifically designed for passwords and thus have better performance. Furthermore, a 128-bit salt [12] is included and the iteration count can be modified in the algorithm of bcrypt so it is immune to rainbow attacks and brute force attacks.

A user cannot use other functions before logging in. The user needs to type in the email and password of the account, which he or she registered previously. Then, the mobile application makes the same preliminary checks as during the registration process. The mobile application sends a login request with the credentials to the backend server. The backend server then checks whether the credentials are correct. Since the passwords are hashed as mentioned, the backend server needs to use the bcrypt algorithm to encrypt the received password again to check. A response with successful status is sent back to the mobile application. If the login is successful, the mobile application shows the community screen. If not, the mobile application will pop up an error message according to the error message received from the backend
2.5. **Data collection methods**

Since a machine learning model needs to be trained in this project, students’ grades need to be collected.

First, there is an initial set of data used to train the machine learning model. The initial set of data is collected through two channels. The two channels are the Department of Computer Science of HKU and questionnaires. The Department of Computer Science of HKU provided electronic transcripts of graduates for the training. A simple website is built to host a questionnaire for collecting and entering data into the database. The website URL is also distributed to students studying computer science in HKU. However, most of the students only filled in their grades for a few courses which make the data not too useful.

Another source of data is the from the users of the mobile application. Every time a new user uses the grade prediction function, grades are inputted, and a new record is received. As mentioned (see Chapter 2.4.1), the machine learning model cannot immediately train each new record because of performance issue. Therefore, the new records are saved in the database first. The usage of these new records is discussed in the next chapter. Since our mobile application is not fully completed yet, no user uses the mobile application. However, data is expected to be received in through this channel when the mobile application is published.

2.6. **Model training**

The problem is being modeled as a collaborative filtering problem like the recommendation systems in online shopping websites. A grades matrix $G$ is formed by having entries $g_{ij}$ as the grade of user $i$ for course $j$. This matrix is then factorized into two estimated matrices representing the latent factors using the WALS algorithm.

Data collected are being filtered first. Student records which have less than 10 computer science courses are filtered to reduce the sparsity of the grades matrix. The data are divided into 2 sets in each iteration, with 90% of them as a training set and 10% of them as a testing set.
Due to performance issues the machine learning model cannot be trained every time when a new grade prediction request is received. However, when using the WALD algorithm, the model cannot predict using a data that is not used in training. This is called a cold start problem. Therefore, to address this cold start problem, the k-nearest neighbors (kNN) to find the nearest neighbor in the existing records for estimation.

In order not to waste the new records received from grade prediction requests, new records that are received from users using the grade prediction are temporarily stored in the database. A cron job is set to train the model again every week using all the records including both the new and old ones. After training, the temporary records are cleared.

### 2.7. Project schedule

<table>
<thead>
<tr>
<th>Month</th>
<th>Deliverables of Phase 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sept</td>
<td>Development of mobile app (Basic UI for registration and login)</td>
</tr>
<tr>
<td>Oct</td>
<td>Development of mobile app (Forum feature)</td>
</tr>
<tr>
<td>Nov-Dec</td>
<td>Deliverables of Phase 2</td>
</tr>
<tr>
<td>Jan-Feb</td>
<td>Data Collection (Questionnaire and Scanning transcripts) + Machine learning algorithm research + training + mobile app testing</td>
</tr>
<tr>
<td>Feb-April</td>
<td>Machine learning algorithm research + training + migration with mobile app</td>
</tr>
<tr>
<td>April</td>
<td>Deliverables of Phase 3</td>
</tr>
</tbody>
</table>

*Table 1. Schedule of this project*
3. USER INTERFACE

UI is always a crucial part of any mobile application because most mobile applications only have simple guides or even no guides at all. Therefore, users only learn how to use the mobile application through the UI. To facilitate the user’s usage, the UI of the mobile application follows the norms and common practice in the industry so that users do not need to adapt to new designs again. Other than facilitating the users’ usage, the UI needs to be attractive to fulfill the project’s objective (see Chapter 1.3).

There are two common designs for the core mobile application layout. They are the navigation drawer and the bottom tab bar.

![Gmail App Layout Image](image)

*Figure 12. Layout of Gmail app*

The navigation drawer layout summarizes all the major functions of the mobile application in a “drawer” which is accessible from the menu button (usually represented by three horizontal stripes) on the top left-hand corner of the screen. Mobile applications using the navigation drawer layout mostly have a few different functions but sharing the same layout. Taking the Gmail app as an example (see Figure 12), choosing the options in the drawer (starred, snooze, important etc.) yields a same layout.
Figure 13. Layout of iOS version Facebook app [13]

The bottom tab bar layout has a tab bar docked at the bottom of the mobile application, so users can switch to different functions simply with one single click. The number of functions included in the tab bar is also commonly constrained to below five. The mobile applications often have a few major functions that needs to be accessible anywhere conveniently. However, most of the major functions in these applications have different layouts. Taking the Facebook application as an example (see Figure 13), all functions (news feed, requests, messages, notifications and more) have different layouts.

Most screens of the AI Student Advisor mobile application display posts in a list, only the filtering and ordering of the posts are different. Although there are two screens including the grade prediction screen and the settings screen uses a different layout, the majority of them are the same. Moreover, there are seven functions that needs to be accessed conveniently which exceeded the convention of having five functions in the tab bar. Therefore, the navigation drawer layout is chosen for this project.

The UI of all screens of the mobile application of this project are designed and implemented successfully. This chapter follows with the detailed design breakdown of all screens for each function.
3.1. View posts in order and filter (My Posts/ Bookmarked/
Hot/ Chronological)

Figure 14. View posts screen

The viewing posts screen is shown in (see Figure 14). This screen is also the home
screen of the mobile application. If a user logged in to the app on the phone before, a
session is maintained, and this screen is shown on app launch. Users can get back to
this screen via the navigation drawer whenever it is needed. By default, this screen
shows all posts in chronological order. Each post in the screen displays the core
information of the post, including the post title, post content, tags, bookmark status
and the number of comments in the post.

Taking the top post as an example (see Figure 14), “Should I study machine learning”
is the post title, “I got C in Artificial Intelligence, should I study Machine Learning?”
is the post content and “AI” and “ML” are the tags. The number of comments is
indicated at the bottom right hand corner of the post next to the comment icon (IALOG).
There are zero comments in this post.
A user can pull-to-refresh the posts or pull up to load more. An extra view, which indicates it is loading and the last updated date, is added to the top when the user pulls down the list (see Figure 15). This extra view hides automatically when the loading finishes. A similar extra view is appended to the list when the user pulls up the list at the bottom (see Figure 16). The only difference is that it does not include a last updated date.

A user can add a post by pressing the button on the top right-hand corner of the screen. A detailed description of that screen will be discussed in Chapter 3.2.

The bookmark button on the bottom left hand corner of each post indicates the bookmark status of the post. A highlighted bookmark button (■) indicates that the post is already bookmarked. On the other hand, a normal bookmark button (□) indicates that the post is not bookmarked. A user can negate the bookmark status of a
post by pressing the bookmark button.

A user can enter a post to view comments of it or add comments to it. In addition, the core information of a post as mentioned above is also included in that screen. The user can do so by pressing on any area of the post, except for the bookmark button. A detailed description of that screen will be discussed in Chapter 3.3.

![Navigation drawer of AI Student Advisor](image)

*Figure 17. Navigation drawer of AI Student Advisor*

The posts in this screen can be filtered or ordered in four different ways as mentioned in Chapter 2.4.2. The way of filter or order can be changed via the navigation drawer’s options (see Figure 17) which can be opened by pressing the menu button on the top left-hand corner of this screen (see Figure 14). “My Posts” in the navigation drawer means viewing the self-created posts. “Bookmarked” in the navigation drawer means to filter and only view the bookmarked posts. “Hot” in the navigation drawer means to view posts ordered by hotness as explained in Chapter 2.4.2. “Forum” in the navigation drawer means to view all posts in the chronological order, which is also the
default ordering. No matter viewing in which order or filter, the layout of the posts maintain the same. The only changes are the posts shown and their order. They are deliberately designed the same so that users do not need to adapt the layout for different orders or filters.

3.2. Create post

The create post screen (see Figure 18) is shown after the user presses the add button in the view posts screen. To create a post, the user needs to enter the post title in the small textbox at the top and the post content in the big textbox in the middle. These two fields are essential and must be filled in when creating a post.

A user can optionally add tags to the post via the “Add” button. An alert with text field is shown to prompt the user for the tag when the “Add” button is pressed (see
Figure 19). The user can then add the tag by pressing the “Add” button of the alert (see Figure 19) after filling in the tag or cancel it by pressing the cancel button. The tag is then added on the screen, it can also be removed simply by pressing the tag directly. Tags can be repetitively added by repeating the procedures above.

![Image](image.png)

**Figure 20. UI showing how a tag is placed.**

After entering all the information, the user can submit the post by pressing the “Post” button on the top right-hand corner of the screen (see Figure 20). Contrarily, the user can press the back button ((cancel button) on the top-left hand corner of the screen to return to the view posts screen without posting. By pressing the back button (cancel button), all temporary post content is discarded.
3.3. Add/ view comments

The add and view comments screen is shown after the user presses a post in the view posts screen (see Figure 21). The user can press the back button (กด) at the top-left hand corner of the screen to go back to the view posts screen anytime.

The core information of the post is shown at the top of the screen in a similar manner as the posts in the view posts screen. The only difference is that, the post creator’s name is shown above the post title. The bookmark button (Bookmark) in this screen serves the same function of the bookmark button (Bookmark) of each post in the view posts screen (see Chapter 3.1).

All comments are shown in a list form below the core information of the post. If there are no comments for the post, nothing is shown below (see Figure 21). Each comment record shows the information of the comment, which includes the comment creator,
the comment content and the votes for the comment (see Figure 22). A user can upvote or downvote a comment by pressing the upvote button (▲) or the downvote button (▼) respectively. The upvote button (▲) and downvote button (▼) is highlighted if it is selected by the user. The number of votes is calculated by deducting the number of downvotes from the number of upvotes.

![Screen capture showing the input panel pushed up by the keyboard and the send button lightens up when there is input.](image)

To add a comment, the user needs to enter the comment in the text field of the input panel docked at the bottom of the screen and press the send button to submit it. When the text field is pressed, a virtual keyboard is shown on the screen which pushes the input panel to the top of it (see Figure 23). The virtual keyboard can be hidden anytime by pressing any part of the screen (except the virtual keyboard and input panel), and the input panel docks at the bottom of the screen again. The send button (►) is only highlighted and enabled when the user entered the comment in the text field. This technique is common in chatting or forum mobile applications to prevent empty comments. The screen is automatically scrolled to the bottom to show the
newly added comment after the comment is added successfully. An alert is shown if the comment submission failed.

If the user is the creator of this post, an edit button (∇) is shown on the top-right hand corner of the screen (see Figure 21). The user can edit the content of the post by pressing this button. Details of the edit post screen is discussed in the following chapter (see Chapter 3.4).

3.4. **Edit post**

If a user is the creator of a post, the user can edit a post by pressing the edit button on the top-right hand corner of the add or view comments screen (see Figure 21). The edit post screen is shown afterwards.

This screen is designed in a way that it looks exactly the same as the create post screen. The reason behind such design is to prevent users having the need to adapt to a new layout again. However, there is a slight difference that the original content is automatically filled in for the user. The post title and content are filled in the corresponding textboxes. Tags of the post are also added to the bottom, if any.

The user can press the “Post” button at the top-right hand corner of the screen to submit the amendment. To the contrary, the user can discard any changes made by pressing the back button (↩) at the top-left hand corner.
3.5. Login/ Register

The login screen is shown on app launch if the user has not logged in to the mobile application before (see Figure 24). The login screen is also shown after logging out from the mobile application. A user can login by typing in the email and password and press the “Login” button. The textbox for typing password automatically converts the characters of the password into black dots such that the password is securely hidden from people near the user. If the user has not registered yet, he or she can press the “Register” button to do so. The user can also press the “Forgot Password” button if he or she forgets the password.

Figure 24. UI of the login screen
The user can specify the email to receive a verification token (see Figure 25). Then the user needs to enter the token and the new password in the textboxes of the next screen (see Figure 26).
The registration process consists of 3 steps and thus 3 screens are needed. In the first screen (see Figure 27), the user needs to enter the email and password in the corresponding textboxes. Unlike the password textbox in the login function, it does not change the password characters into black dots to facilitate the users confirming their password. In the second screen, the user can type in the nickname he or she wish to show in app. The text in step 2 tells the user that the nickname is shown in the mobile application, so that the user knows it is not a private name to prevent privacy issues. In the last step, the user can enter the token.

In all 3 steps, the screens have a consistent layout so the user knows he or she is still in the registration process. The continue button is put in the center and is made in a large deliberately to prompt the user to go to the next step.
4. RESULTS

4.1. Machine learning training results

4.1.1. Metrics

Two metrics are used in this project to measure the error of the grade prediction results. They are the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). They are calculated by cross validation of the testing set and the training set for each iteration in each training as mentioned in chapter 2.6.

The MAE represents the range of the predicted grade. For example, if the MAE is 0.6 and the predicted grade is 2.3 then the true grade is in the range of 2.3±0.6.

4.1.2. Preprocessing

The data received from questionnaires and scanned transcripts are preprocessed. As mentioned earlier, the records of grade results with less than ten courses are filtered in order to reduce the sparsity of the training data.

4.1.3. Grade prediction results

There is a total of 128 students’ course histories being collected and trained. Out of these 100 of them are from the scanned transcripts of graduates provided by the Department of Computer Science of HKU and 28 of them are from online questionnaire. All of them were used to train for 10 times without preprocessing (see Table 2) and 10 times with preprocessing (see Table 3).

<table>
<thead>
<tr>
<th></th>
<th>Training set RMSE</th>
<th>Testing set RMSE</th>
<th>Training set MAE</th>
<th>Testing set MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>0.51 – 0.52</td>
<td>0.68 – 1.05</td>
<td>0.4 – 0.41</td>
<td>0.56 – 0.69</td>
</tr>
<tr>
<td>Average</td>
<td>0.516</td>
<td>0.798</td>
<td>0.408</td>
<td>0.597</td>
</tr>
</tbody>
</table>

*Table 2. Training results without preprocessing*
Comparing the results of trainings with preprocessing and that of trainings without preprocessing, there is a slight improvement. The average MAE of testing set in all our trainings without processing is 0.597 and that of trainings with preprocessing is 0.547 which leads to a 0.597 – 0.547 = 0.05 improvement.

4.2. Encountered challenges

There are a few challenges encountered in this project. They are the insufficient amount of data, sparse data and inaccurate project planning.

The amount of data collected is not enough because there are only 128 training data. Although the result seems to be converging, the result might have only reached a local minimum. A larger amount of data can increase the accuracy of the kNN algorithm used in solving the cold-start problem. That is because the more amount of data used, the user used as reference in the trained data set is likely to have more similar features as the user in request.

The data collected from the website questionnaire are mostly very sparse. Most of the data collected through this method is filtered out during preprocessing. The reason behind the sparsity is that the students do not have the patience to fill in all their grades. Since the data is sparse, it makes the matrices factorized using the WALS algorithm not too accurate. Therefore, after using preprocessing to filter out data with less than 10 course grades improved the training results.

The first two problems can be solved by providing some incentives to students filling out the questionnaires. For example, coupons can be provided to students filling in all their grades. Also, requesting more scanned transcripts for the Department of Computer Science also solves the problem, because the graduate transcripts provide very dense data which are very useful in training.

The project is also planned in a wrong order. The data collection should start at the start of the project and the research on machine learning training algorithms should also be done earlier so that the schedule is not so tight at the second half of the
project.
5. CONCLUSION AND FUTURE WORKS

The problem of university students unable to choose a suitable course for themselves is severe. The current services, including large course grade databases and in person consultation, provided to them does not suffice to help them make a correct decision.

This project aims to harness the technology of machine learning to provide university students better suggestions when choosing courses. The main deliverable of this project is a mobile application. It can utilize machine learning technology to provide course grade prediction and it also has an online community for discussion on course content. The mobile application development has been fully completed. All designed function, including Grade Prediction, View posts in order, Create post, Bookmark post, Add comment and Authentication, has been successfully implemented. The results of the machine learning model are fair. A MAE of 0.547 for the testing set means that the difference between the predicted grade and the true grade the student might get is less than plus or minus 2 subgrades. This error range is considered acceptable when being used in estimating grades. Therefore, the predicted grade has some reference value and can serve as a rough estimation.

All 3 of the objectives laid out are accomplished. A recommendation system is successfully built using the WALS algorithm. A working online community is also built in the mobile application successfully with all the designed functions. Social platform elements, including bookmark, upvote, downvote, are added into the mobile application to attract users as well.

There are improvements that can be done on both the online community and the machine learning model. The mobile application can introduce features like “Best Answer”, which a user can choose the answer which he or she thinks is the most useful for his or her post. A “Login via HKU Portal” feature can also be made so that users can register easier and reduce the friction of them to register a new account. Other matrix factorization algorithms and be explored to see if they can improve the accuracy of course grade prediction. The solution can also try a hybrid approach of using both collaborative filtering and content-based filtering.
REFERENCES


[2] "大學生選科後中途變卦 僅三分一在六年內畢業," Sing Tao Daily, 20 1 2017. [Online]. Available: http://std.stheadline.com/daily/article/detail/1537743-%E6%B5%B7%E5%A4%96%E8%8F%AF%E4%BA%BA-%E5%A4%A7%E5%AD%B8%E7%94%9F%E9%81%B8%E7%A7%91%E5%BE%8C%E4%B8%AD%E9%80%94%E8%AE%8A%E5%8D%A6+%E5%83%85%E4%B8%89%E5%88%86%E4%B8%80%E5%9C%A8%E5%85%AD%E5%B9%B4%E5%85%A7.


