COMP 4801 Final Year Project Final Report
2017/18
A Cloud-based Mobile App for Tutors and Students - *EasySchedule*

Written By

**HO Hilary** 3035229306

Group Members

**CHAN Ka Lok** 3035229502

**LAM Siu Wai** 3035229277

Supervisor

**Dr. WU Chuan**

Date of Submission

15 April 2018
ABSTRACT

In Hong Kong, many students are taking extra classes after school. They have to communicate with their private tutors in order to schedule their classes one by one manually. Without an automated scheduling platform, it takes a significant time cost in class scheduling. Therefore, it is necessary to develop an automated scheduling system for tutors and students to find out the most suitable class schedule for themselves in a faster and easier way.

In our study, a model of scheduling system with reminder function is demonstrated, namely EasySchedule. EasySchedule is an Android application written in Java by Android Studio and Google Firebase as the database. Data is updated instantly with the use of Firebase, a realtime database. Three algorithms for class scheduling are implemented and tested. The algorithm that schedules classes with the greatest availability timeslot among the students performs the best. EasySchedule takes a reference to the current trend of best mobile application design. Adopting material design, EasySchedule looks simple and comfortable with outlined icons and peaceful theme color.

In the future, we hope to further optimize our application with machine learning. By analyzing the huge amount of data collected, EasySchedule will provide two-way suggestions of classes to both tutors and students. Moreover, we plan to expand our reachability by developing website version and iOS application in the future.
ACKNOWLEDGEMENT

I would like to express my gratitude to all who have participated or given some comments and feedbacks to our final year project.

First, I would like to show my special gratitude to our supervisor, Dr. Wu Chuan, who guided us throughout the whole project. She provided us a very precious opportunity to develop this mobile application platform for scheduling classes in a more convenient and automatic approach. I would like to thank her for taking time on our project even though she has a busy schedule. Her feedback and advice are insightful and useful, especially for doing researches and figuring out the solutions.

Next, I would also like to express my sincere thanks to our project examiner, Professor Francis C.M Lau. He raised some important issues and gave us some helpful suggestions to make our application more user-friendly and practical.

Last but not least, I am very thankful to my group mates, Wai and Calvert. They made great efforts in the project, no matter in researching or developing the application. They also express their ideas and thoughts for further discussion in order to pursue for excellence.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>2</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>3</td>
</tr>
<tr>
<td>Table Of Contents</td>
<td>4</td>
</tr>
<tr>
<td>List of Figures</td>
<td>5</td>
</tr>
<tr>
<td>List of Tables</td>
<td>6</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>7</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>8</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>8</td>
</tr>
<tr>
<td>1.2 Problems / Research Gap</td>
<td>8</td>
</tr>
<tr>
<td>1.3 Scope</td>
<td>10</td>
</tr>
<tr>
<td>2. Related Work</td>
<td>13</td>
</tr>
<tr>
<td>3. Methodology</td>
<td>15</td>
</tr>
<tr>
<td>3.1 System Architecture Design</td>
<td>15</td>
</tr>
<tr>
<td>3.1.1 Programming Language for Front-end Application</td>
<td>15</td>
</tr>
<tr>
<td>3.1.2 Overview of System Architecture Design with Cloud Database</td>
<td>16</td>
</tr>
<tr>
<td>3.2 User Interface Design</td>
<td>17</td>
</tr>
<tr>
<td>3.3 Database Design</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Algorithm Design on Multiway Matching Problem</td>
<td>20</td>
</tr>
<tr>
<td>3.4.1 Proof of NP-Hard (non-deterministic polynomial-time hard) Problem</td>
<td>20</td>
</tr>
<tr>
<td>3.4.2 Algorithm Design</td>
<td>21</td>
</tr>
<tr>
<td>4. Result</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Research</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Prototyping</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Final Product</td>
<td>24</td>
</tr>
<tr>
<td>4.3.1 Registration</td>
<td>24</td>
</tr>
<tr>
<td>4.3.2 Login Page</td>
<td>26</td>
</tr>
<tr>
<td>4.3.3 Home Page</td>
<td>27</td>
</tr>
<tr>
<td>4.3.4 Class Creation by Tutor</td>
<td>28</td>
</tr>
<tr>
<td>4.3.5 Timeslot Indication</td>
<td>31</td>
</tr>
<tr>
<td>4.3.6 Scheduled Class</td>
<td>37</td>
</tr>
<tr>
<td>4.3.7 Class Announcement</td>
<td>41</td>
</tr>
<tr>
<td>4.3.8 Search Page</td>
<td>43</td>
</tr>
<tr>
<td>4.3.9 Class Joining Request by Students</td>
<td>44</td>
</tr>
<tr>
<td>4.3.10 Setting Page &amp; Profile</td>
<td>46</td>
</tr>
<tr>
<td>4.4 Algorithm Testing</td>
<td>47</td>
</tr>
<tr>
<td>4.4.1 Running Time</td>
<td>49</td>
</tr>
<tr>
<td>4.4.2 Distribution of Numbers of Successfully Scheduled Classes</td>
<td>50</td>
</tr>
<tr>
<td>4.4.3 Distribution of One-week Schedule</td>
<td>51</td>
</tr>
<tr>
<td>4.4.4 Distribution of Number of Participants</td>
<td>52</td>
</tr>
<tr>
<td>4.4.5 Overview of Total Participants</td>
<td>53</td>
</tr>
<tr>
<td>5. Difficulties</td>
<td>56</td>
</tr>
<tr>
<td>5.1 Integration with WhatsApp/ WeChat</td>
<td>56</td>
</tr>
<tr>
<td>5.2 Implementation of algorithm</td>
<td>57</td>
</tr>
<tr>
<td>6. Future Plan</td>
<td>58</td>
</tr>
<tr>
<td>6.1 Two-way suggestions</td>
<td>58</td>
</tr>
<tr>
<td>6.2 Web page version, iOS app</td>
<td>59</td>
</tr>
<tr>
<td>7. Conclusion</td>
<td>60</td>
</tr>
<tr>
<td>8. Reference</td>
<td>62</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1 Worldwide Smartphone OS Market Share in line graph representation..... 10
Figure 2 Overview of System Architecture Design Using Firebase.........................16
Figure 3 Facebook Mobile App Design (Before and After) ..................................18
Figure 4 The JSON tree of the database ..................................................................19
Figure 5 Simulation of 3-dimensional matching problem..........................................20
Figure 6a Registration Page.......................................................................................24
Figure 6b Login Page With Toast Message of Email Sent........................................24
Figure 6c Screenshot of Verification Email ...............................................................25
Figure 6d Screenshot of Successful Verification .......................................................25
Figure 7 Login Page with user input .........................................................................26
Figure 8a Home Page of Tutor..................................................................................27
Figure 8b Home Page of Student ............................................................................27
Figure 9a Screenshot for Class Creation Input Form ................................................28
Figure 9b Student List for Tutor to Select for This Class .........................................29
Figure 9c Deadline Selection for Scheduling ..........................................................29
Figure 9d Email to Notify Students and Tutor of New Class ..................................30
Figure 9e Notifications about New Class ..................................................................30
Figure 10a Weekly Schedule Timetable ....................................................................31
Figure 10b Screenshot of Timeslot Input Page ..........................................................31
Figure 10c Starting and Ending Time Selection .......................................................32
Figure 10d Classes Selection for Particular Timeslot Input .......................................32
Figure 10e Screenshot of All Timeslot Inputted .........................................................33
Figure 10f Overview of Indicated Free Timeslot .......................................................33
Figure 10g Permission Request for Google Calendar ...............................................34
Figure 10h Google Calendar View of a User .............................................................34
Figure 10i Dialog with All Possible Free Timeslots ..................................................34
Figure 10j Timeslot Indication through Google Calendar .........................................34
Figure 10k Reminder Notification for Timeslot Indication ........................................35
Figure 11a Email about Scheduled Class .................................................................36
Figure 11b Notification about Scheduled Class ........................................................36
Figure 11c Home Page of Tutor with Scheduled Classes ........................................37
Figure 11d Home Page of Student with Scheduled Classes ....................................37
Figure 11e Course Information with Scheduled Time ..............................................38
Figure 11f Detail Information of Tutor .....................................................................38
Figure 11g Google Calendar View with Newly Scheduled Class Information ..........39
Figure 11h Reminder before Class Begins ...............................................................39
Figure 12a Class Information with Message Button ...............................................40
Figure 12b Announcement Message with Title .......................................................40
List of Figures

Figure 12c Notification of Announcement ................................................................. 41
Figure 12d Email of Announcement ............................................................................ 41
Figure 13a Part of Search Page .................................................................................... 42
Figure 13b Part of Search Page .................................................................................... 42
Figure 14a Class Information with Request Function .................................................. 43
Figure 14b Confirmation of Request ........................................................................... 43
Figure 14c Email about New Participants Request ...................................................... 44
Figure 14d Acceptance or Rejection of Request ......................................................... 44
Figure 15a Setting Page .............................................................................................. 45
Figure 15b User Profile Page ....................................................................................... 45
Figure 15c Updating User Details ............................................................................... 45
Figure 16a Running Time Analysis for minStudent = 50% of Class Size .................... 48
Figure 16b Running Time Analysis for minStudent = 75% of Class Size .................... 48
Figure 17a Number of Successful Classes for minStudent = 50% of Class Size ......... 49
Figure 17b Number of Successful Classes for minStudent = 75% of Class Size ......... 49
Figure 18a Distribution of One-week Schedule for Scheduling 100 Classes Simultaneously ........................................................................................................ 50
Figure 18b Distribution of One-week Schedule for Scheduling 150 Classes Simultaneously ........................................................................................................ 50
Figure 19a Number of Participants for Scheduling 100 Classes Simultaneously ..... 51
Figure 19b Number of Participants for Scheduling 150 Classes Simultaneously ..... 51
Figure 20a Total Number of Participants for 100 Classes ........................................... 52
Figure 20b Total Number of Participants for 150 Classes ........................................... 53

List of Tables

Table 1 Worldwide Smartphone OS Market Share in Tabular Format .......................... 10
Table 2 Percentages Distribution of Number of Class Students and Tutors Taking ... 47
ABBREVIATIONS

Here are some abbreviations used in report.

AVD: Android Virtual Device
CPU: Central Processing
GCP: Google Cloud Platform
IDE: Integrated Development Environment
NDK: Native Development Kit
NP: Non-deterministic polynomial-time
OS: Operating System
SDK: Software Development Kit
VCS: Version Control System
1. INTRODUCTION

1.1 Background

It is very common for Hong Kong students to have a very busy schedule after their school. “Winning at the starting line” – a belief of the majority of parents in Hong Kong recently. They hope to let their kids learn as much as possible to equip themselves for their future. According to the survey conducted by South China Morning post in 2016 [1], more than half of participants believed that parents should develop and train kids’ interest and strength in their childhood; while more than 20% of parents agreed that child has a higher learning ability. The most surprising figure is that the kids of about 50% of those respondents are attending numerous extra classes after school (i.e. at least two to three classes).

1.2 Problems / Research Gap

For some activities or classes organized by education institutions, most of them have a regular timetable. On the other hand, for those classes created by private tutors, classes may not have a regular and fix timetable. Some of them will be arranged according to the availability of tutors and students. To handle a number of different classes with tutors and students, the common way to communicate is through social media applications, such as WhatsApp, WeChat, and Messenger, etc. They have to tell each other their available timeslots for scheduling the next class and find out the common free time manually. Thus, they have to handle a lot of chats and groups with different social media applications.

By the approach of handling time schedules manually, some issues are found. First, it takes lots of time for students and tutors dealing with a suitable time schedule after every lesson. They have to manually extract some useful information from the conversation and set reminders on their phone calendar by themselves. Second, it may easily lead to some human mistakes. For instance, misunderstanding one another’s available time slots, missing the classes as reminders had not been set or had been set wrongly, etc. As a result, they may not have an optimal schedule, i.e. they cannot arrange as many classes as possible for a week or they cannot arrange classes with as many students as possible. If they figure out some mistakes before the class begins,
for example, they find out the time of Class A actually crashes with that of Class B, they have to reschedule Class A and B, or even more. In the worst case, all classes of a student or tutor can be affected. In the end, he/she has to reassign all those classes once again manually. Inspired by these troublesome and tedious tasks, we found that it has a significant value to improve the way of scheduling in an easier and convenient manner.

To be more specific, this paper proposes the automation of scheduling process between students and tutors. We introduce a heuristic approach to solve the multi-way matching algorithm, a method to match with multiple tutors and students and find out the best-fit schedule efficiently.
1.3 Scope

![Worldwide Smartphone OS Market Share](https://www.idc.com/promo/smartphone-market-share/os)

Figure 1 Worldwide Smartphone OS Market Share in line graph representation

Source: [https://www.idc.com/promo/smartphone-market-share/os](https://www.idc.com/promo/smartphone-market-share/os)

<table>
<thead>
<tr>
<th>Period</th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016Q1</td>
<td>83.4%</td>
<td>15.4%</td>
<td>0.8%</td>
<td>0.4%</td>
</tr>
<tr>
<td>2016Q2</td>
<td>87.6%</td>
<td>11.7%</td>
<td>0.4%</td>
<td>0.3%</td>
</tr>
<tr>
<td>2016Q3</td>
<td>86.8%</td>
<td>12.5%</td>
<td>0.3%</td>
<td>0.4%</td>
</tr>
<tr>
<td>2016Q4</td>
<td>81.4%</td>
<td>18.2%</td>
<td>0.2%</td>
<td>0.2%</td>
</tr>
<tr>
<td>2017Q1</td>
<td>85.0%</td>
<td>14.7%</td>
<td>0.1%</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

Table 1 Worldwide Smartphone OS Market Share in Tabular Format

Source: [https://www.idc.com/promo/smartphone-market-share/os](https://www.idc.com/promo/smartphone-market-share/os)

In this project, a native Android application is developed, but neither iOS (iPhone OS) nor cross-platform application. With a cross-platform application, our target market can surely be expanded. However, a newer programming language is required. Two common examples are React Native (available since 2015) [2] or Xamarin (available since 2011) [3]. In our past experience, we mainly worked on Java for mobile application while we have less experience and knowledge in these cross-platform languages, this would probably increase the time cost for implementation while learning a new language at the same time. Therefore, we decided to create an application for one particular OS (Operating System). According to the research on
Smartphone OS Market Share conducted by International Data Corporation in 2017 [4], nearly 90% of the global market uses Android OS while only less than 20% of them use iOS (see Table 1). This means the majority of the world uses Android OS instead of iOS.

From Figure 1, it is observed that the market shares of both Android and iOS have been stably maintained since 2014. Only slight fluctuation is found within the range of 80% and 20% respectively. Therefore, we made a reasonable prediction on the future trend of the worldwide smartphone OS market share. We believe the trend would not change significantly in the coming few years. Due to these researches, we decide to implement an Android application.

In the meantime, we choose to develop a native app, but neither web app nor hybrid app. Using native app, users can have the fastest, the most responsive and reliable user experience [5]. On the contrary, less satisfied performance may be experienced in both web app and hybrid app as the content or images have to be downloaded by a browser. This results in a slower performance than using native app. After comparison, we believe a native Android application can suit our project requirement and expectation.

Three main features will be introduced in this application, including automatic class scheduling, reminder function and class searching function. Classes will be scheduled after the specific enrollment deadline provided by tutors. Students have to input their available timeslots before the deadline. The schedule generated by EasySchedule is just a suggestion. Students may adjust their timetable with the tutors afterwards if they do not prefer the assigned timeslot. Class announcement allows tutors to broadcast important messages to the students in their classes, like typhoon arrangement and reminder of lesson. Class searching function allows new incomers of our application to find a class and join. They can make a request to the corresponding tutor of the class for joining.

This paper makes three major contributions. First, it introduces the user interface of our application. It enables users to input their available time slots for
matching use. It also displays users’ enrolled and scheduled classes clearly once they open the application. Second, it demonstrates the algorithm to find the best schedule for users, especially for tutors. The algorithm will run right after the enrollment deadline of his/her course(s) is over. When more than one course having the same deadline, all these classes will be considered at the same time. Third, it shows the result of different algorithms we proposed, including running time, number of successfully scheduled classes, the total number of participants assigned, etc. All these data helps us to find out the best-fit algorithm for our application.

The remainder of this paper proceeds as follows. First, to show the significant importance of this paper, we start with a survey of related work in the current market and study the research gap. Then, we present our system architecture design, user interface design, database design and algorithm design. Next, we discuss our result with some testing data for better understanding of its strength and limitations. Lastly, we close with our thoughts and future plan for further improvement.
2. RELATED WORK

In the current market, three related applications are available in Android market, including, itzTutor Partner [6], Synkers [7] and TuLi [8]. They are all used as a private tutor platform. With these applications, students can find suitable private tutors and arrange their classes.

In general, we have a similar purpose with those applications mentioned above. There are some similarities between these applications and EasySchedule. First, itzTutor Partner and TuLi allow users to indicate their free timeslots for a week. EasySchedule also provides this function, our system can then filter out the busy timeslots and make the matching process less complicated and faster. Second, itzTutor Partner and Synkers also display the scheduled class list for users. It allows tutors and students to clearly view what classes they have enrolled chronologically as an overview. Third, students can search the classes they want. For itzTutor Partner and TuLi, tutor can decide whether to accept or reject the request from students. Students in Synkers can directly book the timeslot of tutors without their confirmation, tutors are relatively a passive user Synkers. However, EasySchedule is designed at tutor’s view, we will take the first approach which is to allow tutors to confirm the request from students.

However, these three applications are mainly for students to find new private tutors and enroll classes manually. Unfortunately, EasySchedule also targets at the existing classes. In other words, tutors and students are already known to each other and classes have been started. Therefore, EasySchedule allows tutors to create the class and invite his/her students to join the class. Furthermore, itzTutor Partner and Synkers allow users to chat with tutors privately for further discussion. EasySchedule only provides the function of class announcement for tutors, as we would like to avoid implementing too many existing functions from other chatting apps.
The most important difference between those applications and EasySchedule is that we aim at facilitating the class scheduling procedure in an automatic way instead of a manual approach. This explains the importance value of this paper even though there are some related works in the current market.

Despite the fact that these three applications do not have the automatic schedule function, they still have their merits and reference values. Therefore, this paper takes the reference to the user interface design and workflow of these related applications and customizes them for EasySchedule.
3. Methodology

In this section, four aspects of our methodology will be explained in details, including system architecture design, user interface design, database table design as well as algorithm design.

3.1 System Architecture Design

Our application is designed to a two-tier architecture [9]. It means it consists of a front-end application (i.e. client) and a back-end server (i.e. data source).

3.1.1 Programming Language for Front-end Application

Java is selected for building our front-end application. At present, the five best programming languages for mobile app development are Java, JavaScript with HTML5, Swift, C# and C++ [10]. As we hope to shorten the learning time in our development, we evaluated two languages that we are familiar with, namely, Java and C++. For Android application development, Java is the most popular and the official programming language [11]. It is supported by Google and it has the largest number of libraries (i.e. codes that we can freely and easily use), official tutorials and documentation. On the other hand, Android NDK (Native Development Kit) is required to run the mobile application using C++. In other words, codes do not run on a virtual machine, but directly on user’s device. However, it is relatively hard to set up and more bugs may exist when using NDK with C++. After comparing with these strengths and limitations, we believe Java is the best choice to implement EasySchedule.
3.1.2 Overview of System Architecture Design with Cloud Database

Figure 2 Overview of System Architecture Design Using Firebase

The system consists of four components - client, database, application server and Google Calendar (see Figure 2). EasySchedule is an Android platform in this stage, so the client refers to a native Android application written in Java. Google Calendar is one of the channels for inputs and outputs. Firebase is used as a cloud database (refer to subsection 3.3 for data modeling). Google Cloud Platform (GCP) acts as an application server.

Google provides the services of GCP and Firebase. They are chosen as part of the system due to their stability, scalability and well-documented APIs. Firebase does not support executing complex algorithm on its Cloud Function feature, so GCP is necessary for running our complicated scheduling algorithm.

In our system design, client devices, such as smartphones or tablets, will connect to Firebase database and send or retrieve data, such as user information, class details, etc. Before the enrollment deadline, users have to input their available timeslots to our system. They can do this in two ways. One way is to input on their own, the other way is by retrieving available timeslots from user’s Google Calendar with permission. In order to run the matching algorithm for scheduling, we have written a script on the GCP for monitoring enrollment deadline, alongside with our scheduling algorithm. The script will invoke the scheduling algorithm once the enrollment deadline is over (refer to subsection 3.4). All the data needed for the algorithm, such as the minimum requirement of participants for a class, students
information, etc., will be read from Firebase directly during the scheduling process. After scheduling, the scheduled timeslots will be passed back and stored in Firebase. Then, clients can retrieve the scheduled classes information from Firebase. If user permission is granted to access their Google Calendar, the scheduled timeslots will be written to the Google Calendar automatically.

Moreover, EasySchedule provides push notification for any class announcement and reminders. A Firebase function written in Node.js is required. It will be triggered whenever EasySchedule sends a notification to users and the messages are sent through cloud messaging.

3.2 User Interface Design

Other than system architecture design, user interface design is also a crucial issue as it greatly affects user experience and satisfaction. EasySchedule targets at students and tutors, while both kinds of the user will perform slightly different, so EasySchedule introduces two main views, one for the tutors and one for the students. With a simple and neat interface, we hope to provide a pleasant user experience to users. Therefore, we have done some researches based on mobile app design trend in 2017.

Material design is a very common and prevailing trend in no matter mobile application or website. Its simplicity and conciseness are the major reason for being popular [12][13]. The card-based design is used to display data and information. In other words, the whole screen is divided into smaller chunks. Each chunk organizes and displays different information. This is an effective way of delivering and spreading messages to users.

Apart from that, simple icons are adopted [12][13]. In the past, we would love to use gorgeous and complex graphics or icons; yet, simple icons with thin outlines become a new trend.
EasySchedule has several pages with different functions, for example, Home page, Schedule page, Search page and Setting page. Therefore, we are aware of how to display these pages well. In the past, hamburger menu was very common and widely used, but some drawbacks were found. It had a lower efficiency and discoverability [14]. In addition, users have to deliberately open the menu to view the options. This undoubtedly affects the user performance, as users have to figure out the hidden menu in the application. If they are not familiar with the items on the menu, they will easily get lost or confused as they have no idea how to perform the task they want. As a result, hamburger menu is replaced by tab bar. For instance, Facebook mobile app has changed its user interface from hamburger menu to tab bar at the bottom (See Figure 3). This brings convenience and improves the performance [15].

In the recent trend, tab bar is commonly used [16]. It allows at most five sections at the top or bottom of the screen. What you see is what you get. Users can quickly spot on what they want as the buttons are in their sight. One limitation of tab bar is that it cannot fit with a long menu. Only five sections/buttons can be displayed. However, it is sufficient for EasySchedule to show all its main functions.

Other than that, the use of color is also an important consideration in mobile application design. According to color psychology [17], a human can focus on calm color (i.e. green or blue) for a longer time since it provides a calm and restful environment. Therefore, we design to use the color between blue and green as our theme color.
3.3 Database Design

Firebase is chosen among various kinds of cloud database services. After some evaluations, we found that Firebase is more suitable for EasySchedule. First, our collected data is not workload analytics. No data analysis is required but the data are only used for the matching algorithm. Second, our data is non-relational. We may have a massive amount of new data with varying data types in the horizontal scaling after released. By Firebase, it will be easier in the expansion of the database with fewer restrictions [18]. With these considerations, we selected Firebase.

At this stage, Firebase provides three different types of database, including Realtime Database, Cloud Firestore and Storage. Since Cloud Firestore is a beta release service, Realtime Database is chosen for storing data while Storage is chosen for storing files.

![JSON tree of database](image)

Figure 4 The JSON tree of database

In Firebase Realtime Database, data is stored in JSON key-value pair (see Figure 4). In EasySchedule, a huge amount of data and messages have to be stored, so
we design our database based on our expected data. Different kinds of data are stored in different nodes. A few major nodes will be introduced as follows. A *Users* subtree stores personal information, such as name, user type (tutor or student), email, phone number, profile icon, etc.; a *Class* subtree stores all classes details, including class name, tutor ID, enrollment deadline, etc. (see Figure 4); a *ClassSchedule* subtree stores the details of scheduled classes, including the assigned timeslot for each class, list of students ID, etc.

### 3.4 Algorithm Design on Multiway Matching Problem

#### 3.4.1 Proof of NP-Hard (non-deterministic polynomial-time hard) Problem

![Figure 5 Simulation of 3-dimensional matching problem](image)

Assume that the scheduling problem in this paper is not an NP-hard problem. This means this scheduling problem can be solved by an algorithm with linear time. However, this problem can be reduced to a well-known NP-hard problem – 3-dimensional matching problem.

Finding a largest 3-dimensional matching is similar to our problem, i.e. finding the largest number of classes that can be scheduled by matching timeslots of tutors and students. Figure 5 illustrates the 3-dimensional matching $M$ with $|M| = 7$, i.e. there are 7 possible matchings in this scenario. Reduced to our problem, let blue node represents tutor; green node represents timeslot; red node represents student, this simulates a student indicating his/her free timeslot (i.e. blue node links to green node) and a tutor indicating his/her free timeslot (i.e. red node links to green node). If the same timeslot is indicated (i.e. the same green node links to red and blue node), that means the connected student and tutor are free for that timeslot.
To conclude, our scheduling problem can be reduced to this 3-dimensional matching problem, which is a well-known NP-hard problem. Since NP-hard problem cannot be solved in a linear time, our scheduling problem cannot be solved in linear time too. By proof of contradiction, our scheduling problem is a NP-hard problem.

3.4.2 Algorithm Design

In this paper, matching algorithm is the core of the application. It determines whether our idea can stand out from all other similar applications or not. Users expect EasySchedule can find out the best schedule for them simply by inputting their available time. To implement this, a matching algorithm is essential.

Our algorithm first gathers information from the database and then starts scheduling. Whenever a new class is created, an HTTP request will be sent to Google App Engine and a scheduler is created. When the specified time in a scheduler is reached, the algorithm will be triggered.

The main idea of our algorithm is to assign the class to a timeslot which has the highest number of available time indication from students. The algorithm will be explained in the following part.

First, our algorithm gets the data of available timeslots of all participants, including tutors and students, for all classes having the same enrollment deadline from the Firebase. Before checking some conditions, the algorithm creates a list of timeslots that all tutors have indicated as free. Then, for each class, the algorithm will check if the tutor is free for a certain timeslot. If the tutor is free, then it will check if the number of students enrolling the class meets the minimum requirement set by the tutor; otherwise, the algorithm will check for the next timeslot from the list. If the number of enrolled students is enough, then it will find out the students who have the common free timeslot with the tutor. If the maximum number of these available students is more than the minimum requirement of a class, then this class will be scheduled and updated to Firebase.
Also, the algorithm will constrain each student can only have at most 2 classes assigned on the same day. For example, if both classes A and B are assigned for student X on Sunday, then student X can no longer be assigned to any other classes on Sunday. This avoids a student from having a very busy schedule on a particular day. Moreover, there must be at least half an hour break time between two scheduled classes on the same day. This simulates the situation in reality. Tutors or students have to travel from place to place, so it is relatively impossible to attend two classes consecutively due to geographical location or other reasons. Whenever any condition is not satisfied, it will have an early termination.

Other than this algorithm, two other algorithms are implemented to solve the same problem for performance testing. The main difference between these algorithms is the sorting of data. By running three algorithms for the same set of data, we can find out whether our current algorithm performs the best or not.

In the second algorithm, instead of finding the timeslot which has the highest number of indication by students, i.e. the most number of students are available for that timeslot, it assigns classes sequentially from Sunday morning to Saturday night. The algorithm focuses on the free timeslots starting from Sunday morning to night. If the basic condition, like the minimum requirement of students, is satisfied, then the class will be assigned. In other words, it has the possibility that many classes will be assigned on Sunday as a result.

In the third algorithm, it sorts the data according to tutors’ free timeslot. If a particular timeslot has the least number of tutors claiming that they are free, that means fewer tutors are available for that timeslot, then this timeslot will be considered first. By doing this, it can ensure that there will be relatively less conflict between tutors. Thus, it can probably assign more classes.
4. Result

Up to the last stage of our project, we have done certain tasks including research work, prototypes and application development. The following subsections will demonstrate the work we have done in details.

4.1 Research

To start with our project, we have done some researches regarding to this paper, including mobile OS market share, related applications, development software, programming language, cloud database storage, interface design and algorithm design. All these have been mentioned above.

4.2 Prototyping

According to our research, we designed a simple and neat user interface for EasySchedule with material design. Simple icons are used for the tab bar at the bottom of the screen. Also, we chose navy blue as our theme color so as to create a relaxing atmosphere for users.
4.3 Final Product

This section will display the screenshots of all major functions with a brief description in EasySchedule.

4.3.1 Registration

![Registration Page](image)

**Figure 6a Registration Page**

![Login Page with Toast Message of Email Sent](image)

**Figure 6b Login Page With Toast Message of Email Sent**
Before using our application, a user has to register an account with some personal information, including username, password and email, etc. (see Figure 6a). After registering, a verification email is sent to the email of the user (Figure 6b). The user has to click the link in the email in order to verify the email address (Figure 6c). After clicking the link, the user will see the screenshot of Figure 6d.
4.3.2 Login Page

After the account is successfully verified, the user can login by his/her registered email and password (see Figure 7).

Figure 7 Login Page with user input

Email
demo@gmail.com
Password
*****

LOGIN

Forgot your password?

Register now!
After logging in to EasySchedule, different users will view the different homepage. Tutors will see the homepage with three tabs on the bottom of the screen (see Figure 8a); students will see the homepage with four tabs on the bottom of the screen (see Figure 8b). In the Home Page, it will display the brief information of enrolled classes, such as class name, venue and enrollment deadline. For tutor view, there is a blue floating button for tutors to create a new class.
4.3.4 Class Creation by Tutor

![Create Class Form](image)

**Figure 9a Screenshot for Class Creation Input Form**
When a tutor wants to create a new class, he/she has to input some class information, including class name, class description, category, tuition fee and duration, etc (see Figure 9a). Also, the tutor has to select his/her students for this class through the list of student users who have already registered in EasySchedule (see Figure 9b). The last step is to set the enrollment deadline for class scheduling (see Figure 9c). The enrollment deadline is used for triggering the scheduling process.
After the tutor has created a class, an email and notification will be sent to notify all the participants of that class and ask them to indicate their free timeslot (see Figure 9d and 9e).
### 4.3.5 Timeslot Indication

#### Schedule

<table>
<thead>
<tr>
<th></th>
<th>MO</th>
<th>TU</th>
<th>WE</th>
<th>TH</th>
<th>FR</th>
<th>SA</th>
<th>SU</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### AddTimeslot

1. Choose the weekday(s)
   - M  T  W  T  F  S  S

2. Add the timeslot(s)
   - add

3. Select class(es)
   - add

**Figure 10a Weekly Schedule Timetable**

**Figure 10b Screenshot of Timeslot Input Page**
On the *Schedule* page, users will see a blank timetable if they have not inputted any timeslot before (see Figure 10a). Users can input their timeslots by a long click on the timetable. After a long click, users can select multiple weekdays at the same time (see Figure 10b). Afterwards, users have to add the timeslots with starting and ending time (see Figure 10c) and select the classes they want to specify (see Figure 10d).
After adding all available timeslots (see Figure 10e), user can save the information to the timetable (see Figure 10f). The class name will be shown on the timetable according to users’ input.
Figure 10g Permission Request for Google Calendar

Figure 10h Google Calendar View of a User
Other than manual input, users can choose to input the timeslot by Google Calendar. EasySchedule will ask for permission to access the Google account (see Figure 10g). When the permission is granted, users do not need to input manually, EasySchedule will retrieve the timeslots without events in users’ Google Calendar. For example, a user has marked as busy on Wednesday from 10 a.m. to 11 a.m. (see Figure 10h). EasySchedule will retrieve all the possible free timeslot from user’s Google Calendar, except Wednesday (from 10 a.m. to 11 a.m.) and a dialog will pop up for the user to select the timeslot he/she wants to indicate (see Figure 10i). If the user just selects Wednesday from 8 a.m. to 10 a.m., then the timeslots will be marked in EasySchedule (see Figure 10j).
If tutors or students have not indicated their timeslots when the deadline is approaching, there will be notification reminder (see Figure 10k), for example, an hour before the deadline. It depends on the setting of a user (refer to subsection 4.3.10).
4.3.6 Scheduled Class

The Class How to relax from stress has been scheduled.

Dear all,
Please check the class time for How to relax from stress.
Best Regards,
EasySchedule

Figure 11a Email about Scheduled Class

Figure 11b Notification about Scheduled Class

When the enrollment deadline is over, the algorithm will be triggered and the class-scheduling task will be performed. For any successfully scheduled classes, an email and notification will be sent to all participants, including tutors and students to inform them about the scheduled time for particular class (see Figure 11a and 11b).
Before the class is scheduled, it will be displayed under section *Enrolled.* Once the class is successfully scheduled, it will be displayed under the section of *Scheduled* for users to easily identify (see Figure 11c and 11d).
Also, before class is scheduled, the course information will show the class scheduling date. But once the class is scheduled, the course information page will update with its scheduled time (see Figure 11e). Furthermore, students can view the detail information of tutor (see Figure 11f) by clicking his/her profile icon.
As mentioned before, users can link EasySchedule with their Google Calendar. Therefore, if classes are scheduled, and with permission granted, EasySchedule will directly update the classes information to Google Calendar (see Figure 11g).

To avoid tutors and students missing the class time, there will be notification reminder (see Figure 11h), for example, an hour before the deadline. It depends on the setting of a user (refer to subsection 4.3.10).
4.3.7 Class Announcement

Badminton

**Description**
Learn to play badminton with correct posture

**Fee**
$350 for every 60 minutes

**Schedule Date**
Class will be scheduled after 2018-4-21

**Location**
Mong Kok

**Participants**
Siu Wai, Student Ho

---

Class Cancelling Due to Bad Weather

Sorry to inform you all that our class has to be cancelled due to Typhoon Signal No. 8.

---

Figure 12a Class Information with Message Button

Figure 12b Announcement Message with Title
If a tutor has an announcement for the class, he/she can click the message button on the right side of the class information (see Figure 12a) and compose an email with title and message content for all students in the class (see Figure 12b). After the announcement is sent, all students in that class will receive application notification as well as an email with the message sent by the tutor (see Figure 12c and 12d).
In *Search* page, students can search classes by categories (see Figure 13a and 13b). There are currently 10 categories available in EasySchedule. They can find new classes by selecting the category they are interested in.
4.3.9 Class Joining Request by Students

If a student is interested in a class, but he/she is not one of the participants added by its tutor, he/she can make a request to the tutor for joining this class (see Figure 14a). After the student clicks to join the class, EasySchedule will ask he/she to double confirm if he/she really wants to join the class (see Figure 14b).
Once the student confirms the request, the corresponding tutor will receive an email about a new participant’s joining request (see Figure 14c). The tutor can accept or reject the request in his/her class information (see Figure 14d). If the tutor accepts, then the name of this student will appear in the participants list.
4.3.10 Setting Page & Profile

In Setting page, it displays user’s profile icon, username and some customized settings (see Figure 15a). Users can set if they want to receive notification and/or email from EasySchedule; set to show their phone number and email address to other users or not; set the reminder time for class and enrollment deadline. Users can view their detailed personal information by clicking on their icon or username (see Figure 15b). They can also update any data on their personal profile (see Figure 15c).
4.4 Algorithm Testing

To prove our algorithm perform well, we conducted some testings with different algorithms. Our algorithm for EasySchedule schedules the timeslots with the greatest availability among students. Two other algorithms are implemented for comparison with our algorithm, in terms of running time and number of classes that are successfully scheduled. One of them is to schedule the classes sequentially from Sunday morning to Saturday night; another one is to schedule the classes with less conflict of tutors’ free timeslot.

In the following part, a few statistics will be shown, including the number of successfully scheduled classes, running time, time distribution of a one-week schedule, distribution of the number of participants and total participants involved.

First, we generated some classes with some conditions for each class.
Class size: 5-7 students
Class duration: 1-3 hours
Minimum number of students: 50% or 75% of actual class size

With 50% or 75% of actual class size as the minimum number of students, it can ensure more classes will be successfully scheduled. On the contrary, if it is set to be 100% of the actual class size, that means the minimum number of students is 5 when the actual class size 5. In other words, if anyone of these 5 students does not have the same free time slot with the others, then this class can no longer be scheduled.

<table>
<thead>
<tr>
<th>Number of Classes</th>
<th>Percentage of Students taking</th>
<th>Percentage of Tutors teaching</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30%</td>
<td>40%</td>
</tr>
<tr>
<td>2</td>
<td>30%</td>
<td>30%</td>
</tr>
<tr>
<td>3</td>
<td>15%</td>
<td>20%</td>
</tr>
<tr>
<td>4</td>
<td>15%</td>
<td>10%</td>
</tr>
<tr>
<td>5</td>
<td>10%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 2 Percentages Distribution of Number of Class Students and Tutors Taking
Table 2 shows the percentages of students and tutors participating in a certain number of classes. For example, if there are a total of 100 students for some classes, there are 30 students (i.e. 30%) taking 1 class only and 15 students (15%) taking 3 classes, etc. The percentages in Table 2 simulate the real situation in Hong Kong. According to a survey conducted by Hong Kong Young Women’s Christian Association [20], about 60% of students take part in 1 to 2 classes; nearly 30% take part in 3 to 4 classes, around 10% of students take part in 5 classes or more in their school terms. For tutors, we simulate those percentages in a reasonable and acceptable distribution.

For each student and tutor for these classes, they should indicate their available free timeslots. In our test, we also simulate realistic free timeslots for them. In normal school terms, we expect tutors would have more free time than students. Tutors have random free timeslots from 9 a.m. to 10 p.m., with at least 2 consecutive hours free once they are available. Students have random free time slots from 9 a.m. to 10 p.m. every Sunday and Saturday, with at least 2 consecutive hours free once they are available; while they have random free timeslots from 4 p.m. to 10 p.m. every Monday to Friday, with at least 1 hour free once they are available.

In order to conduct our testing more comprehensively and more reliable, we tried to schedule by three different algorithms for the same set of classes. For instance, we created 10 classes with the same enrollment deadline and the conditions mentioned above. Then, we ran the three algorithms one by one. The result of each algorithm tells us some statistics, including running time, the number of participants for classes, the number of successfully scheduled classes, etc. After running from 10 classes to 200 classes, we have got the following result.

In the following part, the performance of three different algorithms will be demonstrated and explained. For convenience, algorithm 1 represents the algorithm that schedules with the greatest availability timeslot of students (blue color); algorithm 2 represents the algorithm that schedules sequentially from Sunday to Saturday (red color); algorithm 3 represents the algorithm that schedules with less conflict of tutors’ free timeslots (green color).
4.4.1 Running Time

From Figure 16a, it shows that the running time of algorithm 2 has an extreme fluctuation, especially when the number of classes increases. For algorithm 1 and 3, the running times are similar up to scheduling 180 classes together. However, running time of algorithm 3 increases sharply at the end of the graph.
From Figure 16b, it shows that the running times of algorithm 2 and 3 are similar. They increase gradually in the first half of the test, yet they increase quickly starting from 120 classes. Obviously, it is observed that algorithm 1 has the shortest running time among them.

In general, Figure 16a and 16b conclude that algorithm 1 performs the best in terms of running time.

**4.4.2 Distribution of Numbers of Successfully Scheduled Classes**

![Figure 17a Number of Successful Classes for minStudent = 50% of Class Size](image)

![Figure 17b Number of Successful Classes for minStudent = 75% of Class Size](image)
From Figure 17a and 17b, we found that three algorithms give a similar result. The only implication from these two figures is that, when the minimum requirement of students for each class is smaller, the more classes can be successfully scheduled. In order to identify which algorithm performs better, we have an in-depth study on some details of the scheduled classes in next subsection.

4.4.3 Distribution of One-week Schedule
This part shows the allocation of the scheduled classes within a week.

Figure 18a Distribution of One-week Schedule for Scheduling 100 Classes Simultaneously

Figure 18b Distribution of One-week Schedule for Scheduling 150 Classes Simultaneously
From Figure 18a and 18b, the major observation is that algorithm 2 tried to assign as many classes as possible for Sunday while relatively few classes from Monday to Saturday. This result is not satisfactory, as this will cause a very busy schedule on Sunday for both tutors and students, but relatively free for the rest of a week. On the other hand, algorithm 1 and 3 generate an acceptable schedule.

4.4.4 **Distribution of Number of Participants**

This part shows the number of scheduled classes with a particular class size.
From Figure 19a and 19b, the line graphs are similar. Algorithms 2 and 3 have nearly the same result while algorithm 1 has a line graph that is plotted at a lower position. This means that algorithms 2 and 3 assigned the most classes with 3 students while only few classes have 4 to 7 students. For example, there are about 35 to 50 classes having 3 students while about 13 to 20 classes having 4 students. There is a sharp drop when the class size is 4. However, for algorithm 1, it shows an average and reasonable trends, e.g. there are about 30 to 35 classes having 3 students while about 20 classes having 4 students. We can observe that there is no sudden increase or drop in the number of scheduled classes with different class sizes. In another way, there will be more students participating in classes if algorithm 1 is used, because there is at least certain number of classes having a larger class size.

### 4.4.5 Overview of Total Participants

This part shows the total number of participants involved for 100 classes as well as 150 classes.
This implication derives from subsection 4.4.4. From Figure 20a and 20b, they clearly show that algorithm 1 can schedule the most number of students successfully. There is a difference of about 50 to 100 number of students with algorithm 2 and 3. Therefore, we will say that algorithm 1 can bring the most benefit and revenue to our tutors. To be more specific, with the same number of classes, if algorithm 1 is used,
tutors can have about 50 to 100 students more than that of algorithm 2 and 3 in this example.

To conclude our testing results, we found that algorithm 1 will be the most suitable algorithm for our application. It has a stable and the shortest running time; it does not assign a busy schedule on Sunday only; it assigns the most number of students successfully.
5. DIFFICULTIES

5.1 Integration with WhatsApp/ WeChat

In our initial application design, we planned to integrate EasySchedule with WhatsApp or WeChat in order to allow users to chat easily when using our application. However, we met some difficulties in implementing this function after some studies on the documentation of WhatsApp and WeChat.

We found that there is no official API for integrating WhatsApp with other Android application. Only a third-party API is available for integrating similar functionality. However, it raises the risk that phone number will be blocked as messages are sent and received through the third-party platform. Another main concern is its security. Hackers may attack the third-party APIs and it will cause data leakage, especially some personal particulars.

Fortunately, WeChat provides an official API to integrate with other application. Yet, we have to first register a WeChat official account. In other words, we have to set up a company with business registration so as to register the account.

In our point of view, it may not be worthwhile to do so. The majority (75%) in Hong Kong uses WhatsApp for communication while only 44% for WeChat [21]. Even if we can integrate with WeChat, it can only take care of a small part of users. Therefore, we decided to use another possible approach for communication. Tutors can broadcast their announcement through emails. Students can also receive emails and notifications for information updates. For example, EasySchedule will send a reminder email if students have not indicated their available timeslots when the enrollment deadline is coming.
5.2 Implementation of algorithm

In this paper, the algorithm is the most important and complicated component in our application. In the beginning, we believe that this scheduling problem cannot be solved in linear time due to its complexity and variety, so we make a guess that this problem maybe an NP-hard problem. However, it is inappropriate to claim that it is an NP-hard problem without proof. Therefore, we tried to think if our problem could be reduced to any well-known NP-hard problem. We encountered difficulties in the proof with our basic and limited knowledge in NP-hard problem, we found it hard to recall any NP-hard problems we know that can match to our situation. Eventually, we sought help from our supervisor and got some ideas from her experience. Other than that, we did some researches on Google based on the advice of our supervisor and tried to understand the theory behind. We finally found that our problem can be reduced to a 3-dimensional matching problem.

Another difficulty we met regarding to our algorithm is the choice of data structure. For example, to store the data of free timeslot for each tutor and student, there are 48 slots in one day as we let 30 minutes as a unit. That means, there will be 48 slots x 7 days = 336 slots in total for one person. We have thought of two ways to store the timeslots, one is by a string; another one is by a multidimensional array. By a string, we can represent the timeslot of 10 a.m. to 11:30 a.m. on Monday as “110001130”. The 1st digit represents the day of a week (i.e. Monday); the 2nd to 5th digits represent the starting time while the remaining 4 digits represent the ending time. By a multidimensional array, there is an array with size 7 (i.e. Sunday to Saturday), then there is an array of size 48 for each array element to represent from 12 a.m. to 11:30 p.m. If the tutors or students are free at that timeslot, it will be indicated as ‘1’; otherwise ‘0’.

Different ways have their own strength and weakness. For example, it stores less data by a string while no data parsing is needed for the multidimensional array. At first, we have a different point of views among our group mates. Fortunately, after some experiences in implementation, we found that using multidimensional array will be clearer and easier to understand. Finally, we could make our decision on the data structure. We understand that different people may have different perspectives, but when we have more practical experiences, we will easily know what data structure will be suitable in the future.
6. Future Plan

At this stage, all major functions in EasySchedule have been completed. To keep our application stand out in the current market, we pursue excellence. Therefore, it will be desirable to have some further development and scaling in the future.

6.1 Two-way suggestions

Big data becomes popular and prevalent in any industry these days. We plan to analyze the data collected from all users. By machine learning, EasySchedule will be able to figure out some possible patterns of behavior and give recommendations to users.

In students’ point of view, EasySchedule will recommend some courses that they may want to take. For example, it is found that some students usually take English class as well as Drama class for certain reasons, then when a new student is searching for an English class, we will also recommend Drama class for him/her. Another example is that a tutor is relatively popular for English class, i.e. he creates many English classes and there are many students joined for each class, we may suggest this tutor for new students who are searching for a new English class.

In tutor’s point of view, tutors want to know what kind of classes students like or the demand of particular classes. Therefore, EasySchedule will analyze the data and show some statistics or results for tutors. For example, it is found that there is a great demand for Programming Class. Then, EasySchedule will display this information. If tutors have the knowledge of programming, then he/she can create a new class to meet the demand. In the end, tutors can increase their income.

All in all, EasySchedule plans to provide two-way suggestions to both students and tutors to facilitate their choices. However, we have to collect a certain amount of realistic data for analysis. Therefore, this function will be implemented after the application is published for a certain time.
6.2 Web page version, iOS app

As mentioned in our scope (see section 1.3), we decided to develop Android application due to Android OS’s wide market share coverage and limited time for learning a new cross-platform language. However, we hope to take care of iOS users in the future. Therefore, we plan to develop an iOS application based on our current version. With all the designs, requirements and algorithm ready, we believe that it will not take too much time for iOS development.

Other than this, we would also like to build a web page version of our EasySchedule. There are tons of apps for every single company nowadays and mobile phones are full of application. It is found that some users may not want to download application due to the limited storage space on the mobile device or some other reasons. With the website version, users do not need to download any data to their devices. Therefore, we mainly target for users who use our website by desktop or notebook computer, but not target for mobile users. The disadvantages of web app have been explained in section 1.3. Another advantage of using website version is that the screen will be much larger than a mobile device, it will definitely facilitate the user performance, especially when viewing class information or inputting available timeslots, etc. Furthermore, a website has a higher reachability [22]. Any platform, browsers and search engines can reach our website. It can also be shared easily with friends by sending the website URL.
7. **Conclusion**

In this paper, we introduce the importance of simplifying class-scheduling task that can alleviate the problems brought by human errors or misunderstandings between users in their conversations.

According to some researches made, the major principles of a favorable mobile application design are simplicity and conciseness. With the use of clean user interface, simple icons and calm color, users will gain more satisfaction and tend to stay longer on the application. Also, tab bar replaces hamburger menu which places the main functions in the visible part of the screen. This enhances the visibility and convenience of usage. We designed our user interface of EasySchedule based on the current trend of a good mobile application design. Our user interface allows users to input their availability and displays the class information for scheduled and unscheduled classes.

Moreover, the algorithm design is demonstrated and tested for solving the multi-way matching problem with a reasonably good performance in terms of running time and the results of the scheduled classes. Our algorithm first eliminates the unavailable timeslots indicated by tutors, then it assigns a particular class to the timeslot with the most students available for it. Once the class is assigned, the corresponding time will be set as unavailable for all related tutors and students. By this algorithm, tutors and students can organize their classes more efficiently and accurately.

In our paper, some difficulties and limitations are encountered. With the obstacles in integrating WhatsApp and WeChat, we decided to use email and notifications as our alternative solutions to facilitate communications between users. The most significant issue is the implementation of our algorithm. We spent relatively long time on algorithm design and implementation as all possible scenarios of real-life situations have to be considered. Also, we struggled with the choice of the data structure as different data representations have their own merits.
An important avenue for future work is to optimize EasySchedule with an additional functionality with the study of machine learning. Suggestions of classes will be provided according to the behavior of current users analyzed from the collected data. We believe this functionality can be one of the significant features to increase the competitiveness of EasySchedule.

Last but not least, we plan to share our idea and help as many users as possible. Therefore, a website version and iOS application are planned for further development. This can expand our target market to the largest extent. We believe EasySchedule can improve or even solve the current problems of manual class scheduling.
8. Reference


