A Cloud-based Mobile App for Tutors and Parents - EasySchedule

CHAN Ka Lok 3035229502
LAM Siu Wai 3035229277
HO Hilary 3035229306

Supervised by
Dr. WU Chuan

Submission Date: 15/04/2018
ABSTRACT

*EasySchedule* is an android application which aims to assist scheduling problem between students and tutor. It provides auto scheduling, broadcast messaging, and reminder functions. *EasySchedule* can integrate with Google calendar to import and export events. *EasySchedule* use Firebase database [23] as the backend, and scheduling algorithm will be processed in Google Cloud Platform (GCP) [29]. Notifications will be sent to users through Firebase Cloud Messaging (FCM) [25] services. Design of system architecture, user interface, algorithm and database are discussed in the methodology chapter. Results chapter show the screenshot of the system and the finding of three approaches to solve the multi-way matching problem. Major difficulties are integrating EasySchedule with Wechat and Whatapps and implementing scheduling algorithm. Future direction of *EasySchedule* can be introduce a more efficient algorithm, building iOS version of the application, handle system failure at the middle of the scheduling and use machine learning to suggest class to the users.
ACKNOWLEDGMENT

I would like to express our gratitude to the following person who helped us to prepare this paper and provided generous support. I learn a lot of useful things throughout the project which is good for my future career.

1. Dr. WU Chuan, Final year project supervisor
2. Mr. Cezar Cazan, English class instructor
# Table of Contents

Abstract ........................................................................................................................................... 2  
Acknowledgment .......................................................................................................................... 3  
List of Tables ................................................................................................................................. 6  
List of Figures .................................................................................................................................. 7  
1. Overview ...................................................................................................................................... 10  
   1.1. Background ........................................................................................................................... 10  
   1.2. Research gap ......................................................................................................................... 10  
   1.3. Project Scope ......................................................................................................................... 11  
2. Related Works ............................................................................................................................. 14  
3. Methodology ............................................................................................................................... 15  
   3.1 System Architecture Design ................................................................................................. 15  
       3.1.1 Programming Language for Front-end Application and Backend ................................. 15  
       3.1.2 System Architecture Design With Cloud Database ......................................................... 16  
   3.2 User Interface Design ........................................................................................................... 18  
   3.3 Database Design .................................................................................................................... 20  
   3.4 Algorithm Design .................................................................................................................. 23  
       3.4.1 Proof Multiway-matching Problem is NP-Hard ............................................................... 23  
       3.4.2 Data Representation for the Algorithm ........................................................................... 24  
       3.4.3 Algorithm Design ........................................................................................................... 24  
4. Results ......................................................................................................................................... 27  
   4.1 Screenshot of the system .......................................................................................................... 27  
       4.1.1 Login Page and Register Page .......................................................................................... 27  
       4.1.2 Register Email and Verification ....................................................................................... 28  
       4.1.3 Home Page ....................................................................................................................... 28  
       4.1.4 Tutor create class page ...................................................................................................... 28  
       4.1.5 Search Page and Join Class .............................................................................................. 30  
       4.1.6 Tutor Accept/Decline New Joiners .................................................................................. 31  
       4.1.7 Schedule Page and Add New Timeslot ........................................................................... 31  
       4.1.8 Choose time and Choose Class for New Time Slot ......................................................... 32  
       4.1.9 Setting page and User Profile .......................................................................................... 32
4.1.10 Send broadcast Notifications ................................................................. 33
4.1.11 Notifications Reminders ........................................................................ 33
4.1.12 Sign up email and Emails Reminders ....................................................... 35
4.1.13 Integrate with Google Calendars ............................................................... 37
4.2 Result of the Algorithm .................................................................................. 38
  4.2.1 Test cases design ....................................................................................... 38
  4.2.2 Performance test set up ............................................................................ 40
  4.2.2 Performance of algorithms using different approaches ......................... 41
5. Major Difficulties Encountered ....................................................................... 49
  5.1 Integration with WhatsApp /WeChat ............................................................. 49
  5.2 Implementation of algorithm ....................................................................... 49
6. Recommendation ............................................................................................. 51
  6.1 Two-Way Suggestions ................................................................................ 51
  6.2 More Efficient Algorithm ............................................................................ 51
  6.3 Fault tolerance of the server ........................................................................ 51
  6.4 Build IOS application .................................................................................. 52
7. Conclusion ....................................................................................................... 53
Reference .............................................................................................................. 54
Appendix .............................................................................................................. 58
LIST OF TABLES

Table 1 World wide Smartphone OS Market Share in tabular format 8
Table 2: Comparison between similar scheduling applications 11
Table 3: Test Case Design 35-36
Table 4: Computer specification for performance test 37
LIST OF FIGURES

Figure 1 Worldwide Smartphone OS Market Share in line graph representation 9
Figure 2: Overview of system architecture design using Firebase 13
Figure 3: Facebook Mobile App Design (Before and After) 15
Figure 4a: Database Design: UserSchedule node 18
Figure 4b: Database Design: Class node 18
Figure 4c: Database Design: Category node 18
Figure 4d: Database Design: ClassSchedule node 18
Figure 4e: Database Design: Notifications node (1) 19
Figure 4f: Database Design: Notifications node (2) 19
Figure 4g: Database Design: SchedulingDate node 19
Figure 4h: Database Design: Users node 19
Figure 5: Multi-way matching problem 20
Figure 6a Screenshot of Algorithm Design (1) 22
Figure 6b Screenshot of Algorithm Design (2) 23
Figure 7.1a Screenshot of Register Page 24
Figure 7.1b Screenshot of Login Page 24
Figure 7.2a Screenshot of Register Email 25
Figure 7.2b Screenshot of Account Verified 25
Figure 7.3a Screenshot of Home Page (Tutor) 25
Figure 7.3b Screenshot of Home Page (Student) 25
Figure 7.4a Screenshot of Tutor Create Class 26
Figure 7.4b Screenshot of Choosing Participants 26
Figure 7.4c Screenshot of Choosing Deadline 26
Figure 7.5a Screenshot of Search Page 27
Figure 7.5b Screenshot of Request Join Class 27
Figure 7.5c Screenshot of Confirm Send Join Class Request 27
Figure 7.5d Screenshot of Toast Message for Join Class 27
Figure 7.6a Screenshot of Tutor Accept Join Class Request 28
Figure 7.7a Screenshot of Schedule Page 28
Figure 7.7b Screenshot of Add Time Slot 28
Figure 7.8b Screenshot of Choose Time Dialog 29
Figure 7.8b Screenshot of Choose Class Dialog 29
Figure 7.9a Screenshot of Setting Page 29
Figure 7.9b Screenshot of User Profile 29
Figure 7.10a Screenshot of Send Broadcast Announcement 30
Figure 7.11a Screenshot Notifications: Remind add timeslot 30
Figure 7.11b Screenshot Notifications: Broadcasts Message from Tutor 30
Figure 7.11c Screenshot Notifications: New Class Created 31
Figure 7.11d Screenshot Notifications: Class Scheduled 31
Figure 7.11e Screenshot Notifications: Remind Participants to attend lesson 31
Figure 7.12a Screenshot of Email Verification for new users 32
Figure 7.11d Screenshot of New Join Class Request 32
Figure 7.12c Screenshot Email: Broadcasts Message from Tutor 32
Figure 7.12d Screenshot Email: New Class Created 32
Figure 7.12d Screenshot Email: Class Scheduled 33
Figure 7.13a Screenshot of export to google calendar 34
Figure 7.13b Screenshot of import schedule from google calendar 34
Figure 8.1a Distribution of number of successful class (min student = 50% class size) 38
Figure 8.1b Distribution of number of successful class (min student = 75% class size) 38
Figure 8.2a Running Time Analysis (min student = 50% class size) 39
Figure 8.2b Running Time Analysis (min student = 75% class size) 39
Figure 8.3a Overview of total participants (100 classes) 40
Figure 8.3b Overview of total participants (150 classes) 40
Figure 8.4a Distribution of number of participants for 100 classes. 41
Figure 8.4b Distribution of number of participants for 150 classes. 41
Figure 8.5a Distribution of one–week schedule for 100 classes. 42
Figure 8.5b Distribution of one–week schedule for 150 classes. 42
1. OVERVIEW

1.1. BACKGROUND

As Hong Kong is a highly competitive city, kids or primary students often need to participate in different playgroups like piano class or swimming class after school to maintain their competitiveness. Study conducted by South China Morning Post in 2016 [1], showed that more than 50% of the respondents agreed that parents should find and build their kids’ interest in their early age and more than 20% of the parents believed that kids can learn better when they are young. In addition, 50% of these respondents will schedule at least two to three tutorials after school activities for their kids.

1.2. RESEARCH GAP

Since playgroups or extracurricular classes are offered by private tutors. It is difficult for tutors to arrange the teaching time with students manually. Tutors often need to handle multiple chat groups from WhatsApp, Messenger and WeChat etc., extract available time slots of the participants and then find out the feasible time for the next class. In addition, students and tutors are joining different playgroups at the same time, it is time consuming for them to look for a suitable timeslot.

There are problems with the manual arrangement of time schedules. First, both students and tutors need to communication with different groups and update information with each other, which may consume lot of times. For example, if a tutor wants to update time schedule of a class, he/she need to chat with all the participants of that class and such amendment may affect other class’s schedule as well. Second, they need to manually set the appointment in their schedule book/ calendar which may cause mistake easily. For instance, they may forget to set the calendar or wrongly set the time and venue in their calendar. Thirdly, as tutors only consider the timeslot of their participants, which mean they do not have the full pictures of others tutor’s available time slot. The classes are then schedule in sub-optimal solution, which mean less profit will be earned by the tutor. With the above problems, this brings us the idea of creating an application which can automatic the scheduling process and avoid these human mistakes.
1.3. **Project Scope**

In this project, we will deliver a native android application, which named as *EasySchedule*. Although there are frameworks like React Native [2], Xamarin [3] or Ionic [4] which can help us to develop a cross platform application and attract more users, this increase time cost for implement this application as this team has less experience on using these languages. In addition, application which written in native language will result in better performance and user experience compare with hybrid app or web app. This is because hybrid app always comprised of different interface which is used for communicating the native component inside Android OS or Apple OS, which result in slow speed in loading the app and loading the screens. In fact, java is an object-oriented programming language which enables this team to create modular program and program codes can be reused [20]. Moreover, there are already many support libraries for this team to import and use it directly for saving the development time.

According to the research about Smartphone OS Market Share conducted by International Data Corporation in 2017 [5], the market shares of Android have more than 80% through the year 2016 and the first quarter in 2017, which mean that Android phone have more users than iOS users or Windows Phone (See Table 1). In the same report, the market share of Android and iOS users are very stable in the past few years. Android users are roughly 85% of the whole market while iOS will have 15% users. It is expected that this situation will be the same for the coming few years (See Figure 1). Therefore, this project will deliver a native android application.

<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 World wide Smartphone OS Market Share in tabular format*

*Source: 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

*EasySchedule* have three main functions, they are auto class scheduling, notifications reminders, search class function. This application allows parents to input their available time slot before the deadline set by the tutor. Students receive reminders before the deadline to indicate the available time slot. Once they deadline is over, *EasySchedule* performs scheduling algorithm to find out the best time schedule for having the class. *EasySchedule* sends notifications to the participants before the class begin so that they will not miss the class. In case of emergency, when strong typhoon is coming, tutors can also send broadcast message to students to cancel the class. *EasySchedule* integrates with google calendar so that tutors and students can import the calendar from google calendar and export a class event to the google calendar. Search class function allows new joiners to find a suitable class and send a join request to that class tutor.

This paper makes three contributions. First, it introduces the user interfaces of the complete system, like how to input a new available time slot, display of scheduled class and user setting etc. Second, it explains the matching algorithm in detail like how the data of free slot is represented and how this algorithm is implemented. Third, it compares the performances of three different approaches to solve the multi-way matching problem and summary the finding from the experiment.

The remainder of this paper proceed as follows. In the related work chapter, different exiting applications are being compared to show the need for building this application. In the
methodology chapter, system architecture design outlines the overview of the application and language chosen for front end and back end. User interface design highlights the crucial features and some design principles which apply to this application. Algorithm design explain scheduling procedures and the data representation of the time slot. Database design shows the data structure which is used for the application and how we retrieve the data from firebase. In the result chapter, screenshot of the different parts of the application will be shown, tests cases and experiments which have been carry out to compart the three approaches of the algorithm will be explained. Important findings from these experiments will also be discussed. Difficulties encountered chapters summarize the main difficulties which this team encountered throughout the project. Future work chapter suggests the works which can be done for further improve the system. This paper close with highlight the main features of EasySchedule, and future direction of this project.
2. RELATED WORKS

itzTutor Partner [7], Synkers [8], and TuLi [9] are three famous tutoring applications which can be downloaded in the Google Play Store. However, these applications focus on one to one private tutoring scenario and users need to manually to book a session with tutor. In addition, only TuLi [9] seems to have an auto scheduling function and it may be abandoned as its website cannot be reached.

Apart from the tutor-parent centric applications, there are some scheduling applications which aims to finding the free slot for most participants (see Table 2). Among these applications, most of them need people to do scheduling manually. For the communication, Boomz [32] and Remind [33] provides the chat function while other applications do not support this function. To summarize, there may not exist an application which meets the current needs for students and tutors. Therefore, this team is going to develop an application to solve these problems.

<table>
<thead>
<tr>
<th>Name</th>
<th>Platform</th>
<th>Scheduling</th>
<th>Communication</th>
<th>Used for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doodle [30]</td>
<td>Web + app</td>
<td>Manually</td>
<td>N/A</td>
<td>Any activities</td>
</tr>
<tr>
<td>Calendly [31]</td>
<td>Web</td>
<td>Manually</td>
<td>N/A</td>
<td>Meetings</td>
</tr>
<tr>
<td>Remind [33]</td>
<td>Web + app</td>
<td>N/A</td>
<td>Message</td>
<td>School</td>
</tr>
<tr>
<td>SignUp.com [34]</td>
<td>Web</td>
<td>N/A</td>
<td>N/A</td>
<td>Any activities</td>
</tr>
</tbody>
</table>

Table 2: Comparison between similar scheduling applications
3. METHODOLOGY

This chapter illustrated the overview of the system in different aspects, which includes system architecture design, user interface design, database design and matching algorithm design.

3.1 SYSTEM ARCHITECTURE DESIGN

3.1.1 PROGRAMMING LANGUAGE FOR FRONT-END APPLICATION AND BACKEND

There are five programming languages which can use used for mobile application development, which including Java, JavaScript with HTML5, Swift, C# and C++ [12].

For the frontend, Java is chosen for this project as it is the most popular language and there are many libraries, sample codes and documentation provided by Google, which can help this team to build the application easily. Although C++ can be chosen for building a native android application, it requires more time set up the NDK (Native Development Kit) and C++ may not be able to run in the virtual machine. This means this team need to connect to real devices during the development time which is not convenient.

For the backend, node.js [22] is chosen for implementation of the matching algorithm and the cloud functions to communicate the devices and the backend. It is because node.js doesn’t have the datatype concept when initializing variables, which mean elements with different data types can be put into a list easily by writing some simple line of codes. In addition, node.js uses less coding to perform tasks as compared to java. Node.js can work perfectly with cloud environment and therefore firebase cloud functions are written in node.js / Typescript.
3.1.2 System Architecture Design With Cloud Database

*Figure 2: Overview of system architecture design using Firebase*

*EasySchdule* consists of different components to form the whole system (See Figure 2). Google Cloud Platform (GCP), Firebase, Google calendar are services provided by Google. Since Firebase cloud functions not able to store complex algorithm, the algorithm is hosted on GCP to handle the request from Firebase. GCP is used for application server to perform the matching algorithm while Firebase is the database to store the application data.

Client devices are android mobile phones which has been installed this application. Users can input their available time slots by manually input or get it from their google calendar. It will then connect with Firebase database [23] to store the data. When there is new class created, the cloud function inside the Firebase will invoke a http request to the GCP to add a new schedule time, which mean once it reaches the time limit which have been set by the tutor, the scheduling algorithm will be invoked.

The scheduling algorithm retrieve available time of the targeted tutor and the students from the Firebase database [23] to perform further computation and then return the result back to Firebase. Clients device retrieve scheduled class information from Firebase. Users can then store the event to their Google calendar.
Some of the cloud functions [21] are responsible for sending notifications to the client’s device through Firebase cloud messaging (FCM) [25]. The node.js¹[22] script in the firebase functions gets the necessary data including token of the devices and the payload before sending the notifications. Once the client devices receive the FCM messaging, it will build the notifications to the users by using the Notification Builder object. AlarmManager object will be used for reminding the students to input the available time slot and attending the class before the class starts.

Furthermore, the cloud functions [21] for emails notification will be invoked when a Class is created by a tutor and it use the nodemailer [28] library to send emails to the class participants to notify them indicate the timeslot before the deadline. Similar emails will be sent to class participants when the class is schedule successfully.

¹ Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient.
3.2 User Interface Design
User interface design is important for the application. Better user interfaces can result in better user experience and user satisfaction. This application will have tutor and student view so that it can better to fit their needs.

According to the mobile app design trend in 2017, material design is widely used because of its simplicity and conciseness [13] [14]. To deliver messages to the users effectively and display information clearly, Card UI designs are used. This design aims to split chunks of information into small chunks so that user can digest the information easily. Moreover, hand gestures like swiping, pulling down and flipping can also enhance user experience and therefore these principles will be applied to this application.

Simple icon [12] [13] instead of complex graphic are used so that users can easily remember the main functions of the application.

Nowadays, hamburger menu is replaced by tab bar in the recent trend [16] as tab bar is more efficient than hamburger menu. In the past, Facebook mobile version was used hamburger menu (see Figure 3), which hide the important functions inside the menu. This design will brings low efficient and discoverability problems because users need one extra step to enter the main function. In Recently, Facebook used tab bar to display some of the main functions, which can improve performance and more convenience to the users. Although there are limitations on the number of icons on the tab bar (usually at most 5), it is already enough to display the most frequently used functions to the users.

Figure 3: Facebook Mobile App Design (Before and After)
Source: https://www.lukew.com/ff/entry.asp?1945
Research done by color psychology [17] found that clam color like green or blue provides a more clam and restful environments to the users. Using this two colors can allow users to focus certain amount of time. Therefore, the color scheme for EasySchedule is blue and green, which aims to provide a peaceful environment to users.
3.3 DATABASE DESIGN
In this project, data are stored in Firebase database [23] while images are stored in Firebase storage. They are free backend service provided by Google. It is a free so that this team no need to worry about the development costs. In addition, it provides authentication API [24] and Cloud Messaging API (25) which can reduce the development time. It also provides many sample codes in their website so that this team can take references and easily build this application with firebase.

Firebase stores data in JSON format, which means data are stored as key value pairs. Firebase database is using the concept of NoSql concept so that this team can define the structure themselves to meet the needs of building this application. Below are screenshots of the data structure which are used by EasySchedule. The available timeslots of a user for a specific class in add under UserSchedule/{userId}/{classId}/FreeSlot. Class node stores the class details like class name, duration of a class and the price etc. Category node stores different category of the class and the images of the category. ClassSchedule node stores the details of the class which has been scheduled, i.e. class time and the reference to studentId. For Notifications node, the type is used to distinguish different type of notification event like ClassNewParticipants and ClassAnnouncement. ScheduleDate node stores the date which the scheduling algorithm will be involved. Users node stores the details of each users and their setting like the status for email notifications and their display name etc.

---

1 Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.
Figure 4a: Database Design: UserSchedule node

Figure 4b: Database Design: Class node

Figure 4c: Database Design: Category node

Figure 4d: Database Design: ClassSchedule node
Figure 4e: Database Design: Notifications node (1)

Figure 4f: Database Design: Notifications node (2)

Figure 4g: Database Design: SchedulingDate node

Figure 4h: Database Design: Users node
3.4 Algorithm Design

3.4.1 Proof Multiway-Matching Problem is NP-Hard

The goal of this study is to schedule different class at a specific time such that the number of classes is the maximum after scheduling. In addition, tutor and student are not able to attend more than one class at a specific time. In fact, this problem can be reduced to 3-dimentional matching problem [19], a known np-hard problem. 3-dimentional matching problem is a generalization of bipartite matching to 3-uniform hypergraph. There are three set finite and disjoint set $X,Y,Z$. $T$ is subset of $X \times Y \times Z$. $M$ is subset of $T$ and it is 3 -dimentional matching such that for any two distinct triples $(x_1,y_1,z_1)$ in $M$ and $(x_2,y_2,z_2)$ in $M$, where $x_1 \neq x_2$ and $y_1 \neq y_2$ and $z_1 \neq z_2$.

The scheduling problem is this application can be reduced to 3-dimentional matching as follows (see Figure 5). It is assumed that student and tutor can join one class in this proof to simplify the analysis. The blue dots represent the tutors (Set $X$), green dots represent the available time slot (Set $Y$) while the red dots represent the students (Set $Z$). Assuming this problem is not an NP hard problem, which implied that it can be in polynomial time, but this problem can be reduced to a known NP hard problem (3 -dimentional matching) even for that special case. This lead to a contraction of assumption because NP hard problem cannot be solved in polynomial time. This completed the proof.

![Figure 5: Multi-way matching problem](image-url)
3.4.2 Data Representation for the Algorithm

The first task to solve the algorithm is to represent time using data structure. In this project, this team define a time slot as 30 minutes interval to save the memory space and make the computation easier. 2D array is used to represent a user’s availability, name it as freeSlot. The size of this array is 7 * 48 as we have 7 weekdays and 48 slots per day. In addition, the counting is starting from Sunday midnight 12:00 am to 11:30 pm Saturday night. The boolean value in each cell can represent users’ availability. If “ freeSlot[6][18] = 1 ” mean the user is free at Saturday 9 am.

3.4.3 Algorithm Design

This team use heuristics approach to solve the scheduling problem. There are three different approaches to solve the problem. They are Schedule with the greatest availability time slot, Schedule sequentially and Schedule with less conflict with other available slots by a tutor. These three approaches will be evaluated in the result chapters. The ideas of these approach are similar, the design is as follows.

As mentioned in the System Architecture Design. The scheduling algorithm will be invoked when it meets the deadline which has been set by the tutor previously. Firstly, it will gather the class information like class participants, minimum participants, class name etc. and the available time slots of all the users which is related to the classes which are going to be scheduled. It will then organize the list into structuredUnscheduledList,

\[
\text{ie } \{ \text{classid: -L9KYXnbskOidSTDhWO0, availableSlot: \{xCoordinate, yCoordinate, studentIndicationNumber\}, studentNumber: \{2DarrayWithNumber\}, minRequirement: 3, classLength: 60, creator: T-046e8ee080855f354a0f421fa21ef1, className: abc, currentMax: 7 \} }.
\]

This structuredUnscheduledList will be used to compute the globalReference, which will be used to determinate the order to schedule the class. The globalReference function will return the unScheduledClassNumber, globalMinReq and the classScheduleOrderList. The structure of the classScheduleOrderList: \[ [xCoordinate, yCoordinate, studentIndicationNumber, Crush among the slots, randomNumberofSameClass, randomNumber] \]. The classScheduleOrderList is sort according to different approach of the algorithm. For example, if the approach is Schedule with the greatest availability time slot. The order will sort with respect to
studentIndicationNumber, and the randomNumber is used to determinate the order if two class having the same studentIndicationNumber.

Once the classScheduleOrderList is ready, the scheduling can be started (see Figure 6a & 6b). If schedulingMethod =1 (Schedule with the greatest availability time slot), the while loop will be terminated when (currentPointer < globalMinReq) or (unScheduledClassNumber <=0).

When entering the while loop, first calculate the time slot for the class and the minimum number for that class.ie if the class need 60 mins, we need 2 slots. Then it will check whether the tutor is free and check if there is enough student indicate that he/she is free at that slot. It will be followed by count the number of students who really can join the class, i.e. student can join the class if he/she has less than 2 class at that day and having enough free slots. If there is numberOfStudentCanJoinClass > minimumReqStudent, the class can be scheduled, then it stores to firebase and update those participant’s availability. Finally, it will update the student’s time slot and tutor’s time slot which is temporality stored in the RAM and recomputed the computeGlobalReference to continue with the scheduling. If the schedulingMethod =2 (Schedule sequentially) or 3(Schedule with less conflict with other available slots by a tutor), it is almost the same except that it will loop through the entire classScheduleOrderList.

---

**Figure 6a Screenshot of Algorithm Design (1)**
```javascript
var updatedResults = computeGlobalReference(structuredUnscheduledList);
if (updatedClassScheduleOrderList.length > 0) {
    updatedCurrentPointer = updatedClassScheduleOrderList[0][2];
} else {
    updatedCurrentPointer = 0;
}
unScheduledClassNumber = updatedUnScheduledClassNumber;
globalMinReq = updatedGlobalMinReq;
currentPointer = updatedCurrentPointer;
classScheduleOrderList = updatedClassScheduleOrderList;
} else {
    console.log("Class cannot be scheduled");
    // update current pointer
    currentPointer = classScheduleOrderList[0][2];
} else {
    console.log("Not enough student, test next slot");
    currentPointer = classScheduleOrderList[0][2];
}
else {
    console.log("Tutor is not free");
    currentPointer = classScheduleOrderList[0][2];
}
else {
    while ((unScheduledClassNumber > 0) && (classScheduleOrderList.length > 0)) {
        // same logic as above
    }
```

Figure 6b Screenshot of Algorithm Design (2)
4. RESULTS

4.1 SCREENSHOT OF THE SYSTEM

4.1.1 LOGIN PAGE AND REGISTER PAGE

Figure 7.1a Screenshot of Register Page

Figure 7.1b Screenshot of Login Page
4.1.2 **Register Email and Verification**

Figure 7.2a Screenshot of Register Email

Your email has been verified

You can now sign in with your new account

Figure 7.2b Screenshot of Account Verified

4.1.3 **Home Page**

Figure 7.3a Screenshot of Home Page (Tutor)

Figure 7.3b Screenshot of Home Page (Student)
4.1.4 Tutor Create Class Page

Figure 7.4a Screenshot of Tutor Create Class

Figure 7.4b Screenshot of Choosing Participants

Figure 7.4c Screenshot of Choosing Deadline
4.1.5 Search Page and Join Class

Figure 7.5a Screenshot of Search Page

Figure 7.5b Screenshot of Request Join Class

Figure 7.5c Screenshot of Confirm Send Join Class Request  Figure 7.5d Screenshot of Toast Message for Join Class
4.1.6 Tutor Accept/Decline New Joiners

Figure 7.6a Screenshot of Tutor Accept Join Class Request

4.1.7 Schedule Page and Add New Timeslot

Figure 7.7a Screenshot of Schedule Page

Figure 7.7b Screenshot of Add Time Slot
4.1.8 CHOOSE TIME AND CHOOSE CLASS FOR NEW TIME SLOT

Figure 7.8b Screenshot of Choose Time Dialog

Figure 7.8b Screenshot of Choose Class Dialog

4.1.9 SETTING PAGE AND USER PROFILE

Figure 7.9a Screenshot of Setting Page

Figure 7.9b Screenshot of User Profile
4.1.10 Send broadcast notifications

Figure 7.10a Screenshot of Send Broadcast Announcement

4.1.11 Notifications Reminders

Figure 7.11a Screenshot Notifications: Remind add timeslot
Message from Tutor

Figure 7.11b Screenshot Notifications: Broadcasts
Figure 7.11c Screenshot Notifications: New Class Created

Figure 7.11d Screenshot Notifications: Class Scheduled

Figure 7.11e Screenshot Notifications: Remind Participants to attend lesson
4.1.12 Sign up email and Emails Reminders

Figure 7.12a Screenshot of Email Verification for new users

![Verify your email for EasySchedule](image)

Hello,

Follow this link to verify your email address:

https://easyschedule-23ad3.firebaseapp.com/_auth/action?mode=verify&email=abc@email.com&confirm=true

If you didn’t ask to verify this address, you can ignore this email.

Thanks,
Your EasySchedule team

Figure 7.12b Screenshot of New Join Class Request

![New Participants](image)

Dear tutor,

participant Student Ho want to join class: Badminton

Best Regards,

Easy Schedule

Figure 7.12c Screenshot Email: Broadcasts Message from Tutor

![Class Announcement](image)

Dear students,

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

Best Regards,
Tutor Ho

Figure 7.12d Screenshot Email: New Class Created

![The Class Drawing Class has been created](image)

Dear students,

Please indicate the timeslot before the deadline.

Best Regards,
Tutor Ho
Figure 7.12e Screenshot Email: Class Scheduled
4.1.13 Integrate with Google Calendars

Figure 7.13a Screenshot of export to google calendar

Figure 7.13b Screenshot of import schedule from google calendar
4.2 RESULT OF THE ALGORITHM

4.2.1 TEST CASES DESIGN
Testing plays an important role for system development cycle. Test cases are used to ensure that the application is run as we expected. If the application cannot pass all the test cases, which mean the application is not working properly and people will not use it. Below are the test cases which were carried out to ensure the functionally of the algorithm.

<table>
<thead>
<tr>
<th>No.</th>
<th>Descriptions</th>
<th>Test Cases</th>
<th>Expected Behavior</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The class will not be scheduled if the student enrolled &lt; min class size</td>
<td>Create a class with class size = 5, and the no. student can attend = 4</td>
<td>Class cannot be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>The class will not be scheduled if the participants do not have common slots</td>
<td>Create a class with 1 slots need, and minRequire student = 2, one student indicates free at slot = a while another student indicate free at =b</td>
<td>Class cannot be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td>3</td>
<td>The class will not be scheduled if the slots require &lt; the student’s available slots</td>
<td>Create a class with 3 slots need, and minRequire student = 2, one of the student has 3 slots available while another student only has 2 slots available</td>
<td>Class cannot be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td>4</td>
<td>The class will not be scheduled if the tutor does not have enough free slot.</td>
<td>Create a class with 2 slots need and minRequire student = 1, tutor is having 1 free slot while the student has 2 free slots</td>
<td>Class cannot be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td>5</td>
<td>Student will be scheduled for more than 2 classes in the same day</td>
<td>Assume the Class A, Class B have been scheduled on Mon, student A attend these two classes. When scheduling the next Class C and student A is one of the participants</td>
<td>Student A will not be scheduled into Class C if Class C will be scheduled at Mon.</td>
<td>OK</td>
</tr>
<tr>
<td>6</td>
<td>Class B will not be scheduled at the same time of Class A if the participants are the same</td>
<td>Assume Class A need 1 slot and Class B also need 1 slot. Participants of Class A and B are the same, and participants are available at 1 only slots</td>
<td>Either Class A / Class B can be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td>7</td>
<td>Class A /Class B can be scheduled if the participants are the same, but they have enough free slots.</td>
<td>Assume Class A with 1 slot and Class B with 1 slot need. Participants of Class A and B are the same, and participants are available at slot x and slot y</td>
<td>Class A and Class B can be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td></td>
<td>The Class will not be scheduled at multiple time frame</td>
<td>Create 5 different classes, and assign different students to those class</td>
<td>The 5 Classes either scheduled at one of the time frame provided by these tutors / cannot be scheduled</td>
<td>OK</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------------------</td>
<td>-----------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
<td>----</td>
</tr>
<tr>
<td>9</td>
<td>Class is scheduled according to the greatest available time slot (the most student who are free at the slot)</td>
<td>Create 5 different classes, and assign different students to those class</td>
<td>The Order of scheduling is according to the most student who are free at the slot</td>
<td>OK</td>
</tr>
<tr>
<td>10</td>
<td>Class is scheduled sequentially</td>
<td>Create 5 different classes and assign different students to those class</td>
<td>The Order of scheduling is according to the order of the time slots</td>
<td>OK</td>
</tr>
<tr>
<td>11</td>
<td>Class is scheduled with less conflict with other classes</td>
<td>Create 5 different classes and assign different students to those class</td>
<td>The Order of scheduling is according to the order less conflict with other classes</td>
<td>OK</td>
</tr>
</tbody>
</table>

Table 3: Test Case Design
4.2.2 PERFORMANCE TEST SET UP

To measure and compare the performance of the three approaches of the algorithm, several tests have been carried out. These tests are simulated the real – life situation. It is because if real – life situation is not considered; the result is not realistic and the application may not be worked well in real practice. Therefore, the parameters were set as follows:

For student, the distribution for joining 1 class = 30 %, 2 classes = 30 %, 3 classes = 15 %, 4 classes = 15 %, 5 classes = 10 % of the total number of classes. For tutor, the distribution for joining 1 class = 40 %, 2 classes = 30 %, 3 classes = 20 %, 4 classes = 10 %, 5 classes = 0 % of the total number of classes. In addition, the class size was set to be between 5 students to 7 students and class duration was between 1 hour to 3 hours. We have done two different sets of test to compare their performance. One set was set minimum class size requirement = 50% of the class size while the other set = 75%. The tests were carried out when the class size = 10 up to 200 with 10 classes increases each time. Since the results were vary every time, each test was carry out 5 times and the graphs shown in the report are after taking average.

To obtain a fair result, all the tests were carried out using the same computer. The computer specification is shown as below:

<table>
<thead>
<tr>
<th>Name</th>
<th>MacBook Pro (Retina, 15– inch, Mid 2014)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>2.2Ghz Intel Core i7</td>
</tr>
<tr>
<td>Memory</td>
<td>16GB 1600Mhz DDR3</td>
</tr>
<tr>
<td>Graphics</td>
<td>Intel Iris Pro 1536Mb</td>
</tr>
<tr>
<td>Serial #</td>
<td>C02ND192G3QC</td>
</tr>
</tbody>
</table>

Table 4: Computer specification for performance test
4.2.2 Performance of Algorithms Using Different Approaches

Figure 8.1a Distribution of number of successful class (min student = 50% class size)

Figure 8.1b Distribution of number of successful class (min student = 75% class size)
RUNNING TIME ANALYSIS
FOR MIN STUDENT = 50 % OF CLASS SIZE

Figure 8.2a Running Time Analysis (min student = 50% class size)

RUNNING TIME ANALYSIS
FOR MIN STUDENT = 75 % OF CLASS SIZE

Figure 8.2b Running Time Analysis (min student = 75% class size)
Figure 8.3a Overview of total participants (100 classes)

Figure 8.3b Overview of total participants (150 classes)
Figure 8.4a Distribution of number of participants for 100 classes.

Figure 8.4b Distribution of number of participants for 150 classes.
Figure 8.5a Distribution of one–week schedule for 100 classes.

Figure 8.5b Distribution of one–week schedule for 150 classes.
In term of number of successful class, there is not much difference between the three approaches (see Figure 8.1a & Figure 8.1b). Number of successful class is more when if we set the min student requirement= 50% class size when compare to min student requirement = 75% class size. This is expected behavior as higher percentage of min student requirement will have less chance that the class to be scheduled. In addition, the number of successful class are increase steadily. i.e. about 80% of the classes can be scheduled for min student requirement= 50% class (Figure 8.1a) while about 60 of the classes can be scheduled for min student requirement= 75%(Figure 8.1b). This number is a rough figure indicate how many class can be scheduled, we need to study a class in detail for further comparison of these algorithms.

In term of running time analysis, it seems that there is not much difference between the three approaches at the beginning. However, when the number of input size is getting larger, it is obvious that Schedule with the greatest availability time slot is the best (see Figure 8.2a & 8.2b). Although this observation is not clear for min student requirement= 50% class (Figure 8.2a), it is clear when min student requirement= 75% class (Figure 8.2b). As you can see, the running time for much slower for Schedule with the greatest availability time slot when the class size increase from 130 to 200. When class 200, it requires about 35% less computation time than Schedule Sequentially and Schedule with Less Conflict with other classes. It is expected that such difference will be much significant when the input size even larger.

For running time of Schedule Sequentially and Schedule with Less Conflict with other classes are quite similar. The explanation of this phenomenon is because the there is an early terminated in Schedule with the greatest availability time slot. It sorts the scheduling order with respect to the number of indication of different class. If there is a case that the indicationNumber < global minStudent requirement for all the classes, the looping will be end. However, this technique cannot be applied to other two approaches because the algorithm needs to scan the entire free slots to ensure there is no more class can be scheduled. Therefore, Schedule with the greatest availability time slot did better in term of running time.
As there is no much insight from the successful number of class, two of the classes are taken to find out more information. One was classSize = 100, min student requirement = 50% class size and other one was classSize = 150, min student requirement = 50% class size. To summarize, the result is consistence with each other.

For the overview of the number of participants, Schedule with the greatest availability time slot schedules more participants than other two approaches (see Figure 8.3a & 8.3b). For example, the total number of students which will join the class = around 450 for Schedule with the greatest availability time slot while for Schedule Sequentially and Schedule with Less Conflict = around 350. Recall that number of successful class are almost the same for the three approaches (Figure 8.1a & Figure 8.1b), which mean the total profit for tutor is more if there is more number of participants.

In addition, the distribution of participants is tending to schedule at a larger class size for Schedule with the greatest availability time slot (Figure 8.4a & 8.4b). As you can see from the two graphs, when the class size is increase, Schedule with the greatest availability time slot have more classes than other two approaches. This means that tutors are going teach a class with more people when using Schedule with the greatest availability time slot approach. This helps tutors to generate more income if he/ she can teach more students at a time. The explanation of this phenomenon is because Schedule with the greatest availability time slot are always schedule class as the highest indications of which participants are free at those slots.

For the distribution of one –week schedule, most of the classes are scheduled at Saturday or Sunday for the three approaches (see Figure 8.5a & 8.5b). It is because we set most of the student free at Sunday and Saturday which is according to the real-life situation for these tests. However, most of the class are scheduled at Sunday for algorithm Schedule Sequentially. For instance, for class size =150, the number of class almost double in Schedule Sequentially when comparing to other two approaches (Figure 8.5b). If most of the tutors are having their class at Sunday, they may be overload at that day and do not have enough rest. This situation happens because algorithm Schedule Sequentially is scheduled classes from the beginning which is Sunday to the end, and finally most class are scheduled at Sunday. There is not much difference between Schedule with the greatest availability time slot and Schedule with Less Conflict in terms of distribution of one week schedule.
To sum up the performance of the three heuristic approaches for solving the multiway matching problem, *Schedule with the greatest availability time slot* is better in terms of running time, distribution of the class size, distribution of class over one week. Therefore, the application will be used that algorithm for building the backend.
5. **Major Difficulties Encountered**

5.1 **Integration with WhatsApp/WeChat**

At first, this team would like to integrate this application with WhatsApp or WeChat to exchange and share information to enhance user experience. However, it is not possible to do it after doing research on the documentation of WhatsApp and WeChat. Although there is third-party API which can send messages to different users using extra library written in Java, it has a possibility that this account will be blocked by WhatsApp in the future because of using unofficial API. Moreover, it is not safe to send messages through this channel in terms of security. Assume that User A wants to send some message to User B, hackers may steal some important information from User A by hacking the API. In addition, hackers can also modify the content from User A and the forward to User B. For integrate with WeChat, company detail must be provided to obtain a service account provided by WeChat according to the documentation, which means it is not possible to do it. Therefore, this team decided to use another channel like email and Android notifications to broadcast message to users. For private messaging between one student and one tutor, they can use the normal communication channel like WhatsApp, Wechat.

5.2 **Implementation of Algorithm**

This team faced several difficulties regarding the algorithm. Firstly, this team was lack of knowledge to prove the multiway matching problem is a NP hard problem. In addition, this team did not know how to formulate the problem and design an algorithm to solve it. However, when the project progress, this team consulted with Dr. Wu, she gave an important insight that bean counting could be helpful for this problem and she also introduced 3-dimensional matching (a known NP-hard problem). This team then started design and implementation the backend for this application.

During the implementation of the algorithm, I attempted to use Java language to implement the logic, but Java is not a suitable language to implement complex algorithm and working with Firebase. It is because Firebase use asynchronous task to get data but Java itself does not have a good framework to make the asynchronous request become synchronous request. In addition, when I need to declare a complex data structure like a list which contains different objects in java, extra works need to be done to obtain the same result when using node.js.
Therefore, I decided to use node.js language to complete the algorithm and Promise [26] library can make the request become synchronous

As mentioned in section 3.1.2, the algorithm need to be invoked at a certain time, like a specific date. At the beginning, I did not have a complete solution on it, I can make use of the node-schedule [27] library to achieve it after doing some research. In addition, with the help of Express.js, the backend endpoint looks more like a restful request. The cloud functions in firebase can then send a HTTP request to invoke the backend logic.
6. RECOMMENDATION

6.1 TWO-WAY SUGGESTIONS
Big data and machine learning are hit area in recent trend for any industries. Therefore, this technique should be applied in this application to attract more tutors and students. After collecting the data, it is useful to figure out the user behavior patterns and then provides the best recommendation to the users. For example, if people who join singing class will join the dancing class, we can pop up an advertisement when people search for singing class. This trick is widely used to in some online shopping shops like Amazon. On the other hand, we could analyze the profile of the tutors from time to time, if a tutor uses this application frequently, we could suggest more classes to him so as to increase his revenue. In general, two-way suggestions could be suggested to students and tutors. However, to apply these techniques, we need to have large user base so that the analyze will be more accurate.

6.2 MORE EFFICIENT ALGORITHM
Although this team has introduced three different approaches to solve the multi-way matching problems, the running time and the scheduled result when using schedule with the highest people which are free among the available slots by a tutor is the best among these three algorithms in terms of running time, distribution of the class and the number of class participant. This algorithm can be further improved by reducing the size of the 2D array, because some of the timeslot like 12:00 am to 06:00 am will not have classes usually. In addition, there may exists other heuristic approach which gives better running time. Therefore, it provides an opportunity for further research to make this algorithm works more efficient.

6.3 FAULT TOLERANCE OF THE SERVER
The server may experience hardware fails or system with other applications during the computation of the matching algorithm. Fault tolerance is very important to handle this situation. It is because the scheduling is a long-lasting process, if system need to recompute the previous result whenever there is problem occurs during the middle of the computation, it wastes a lot of time. It is expected the when the customer bases become more and more, the scheduling task will take more longer time than before. Therefore, we need to introduce a
mechanism to store the intermediate result so that if there is system failure, the system can roll back to previous result and continue the computation.

6.4 BUILD IOS APPLICATION
When native android version of EasySchedule is running after a few months’ time, it is good to develop a native iOS version of that. It is because this can attractive more people to use this application. To develop iOS version of EasySchedule, the database can be shared with the two applications which mean we can use swift language to retrieve data from Firebase database. If one of the users change his /her phone from Android to iOS in the future, he can continue using this application with the same account.
7. CONCLUSION

*EasySchedule* provides auto scheduling which facilitates the process of finding common free time slot to conduct lessons between students and tutors. Broadcast message can be sent to the participants when there is emergency or important message. Notifications are sent to students and tutors before the lessons for reminding them to attend the lesson. Users can input their availability from Google Calendar and export the event to Google Calendar. New joiners can search a class under different category and send a join request to the class tutor.

To summarize, this paper introduces the needs for building *EasySchedule*, the methodology of how to build the application in terms of system architecture, user interface, database and algorithm. The multi-way matching problem in *EasySchedule* is an NP hard problem which can be used by a heuristic approach. Different tests are carried out to study the performance between *Schedule with the greatest availability time slot*, *Schedule sequentially* and *Schedule with less conflict with other available slots*. *Schedule with the greatest availability time slot* is better than other two approaches. One of the limitation of this system is that the scheduling results may not an optimal schedule for all the tutors and students and it is difficult to prove the result from the approach is optimal. Further study is needed to improve the performance of the algorithm. Two ways suggestions, fault tolerance and iOS version of this application can be introduced in the further to enhance the application.
REFERENCE


