

Criterion C — Development

Techniques:

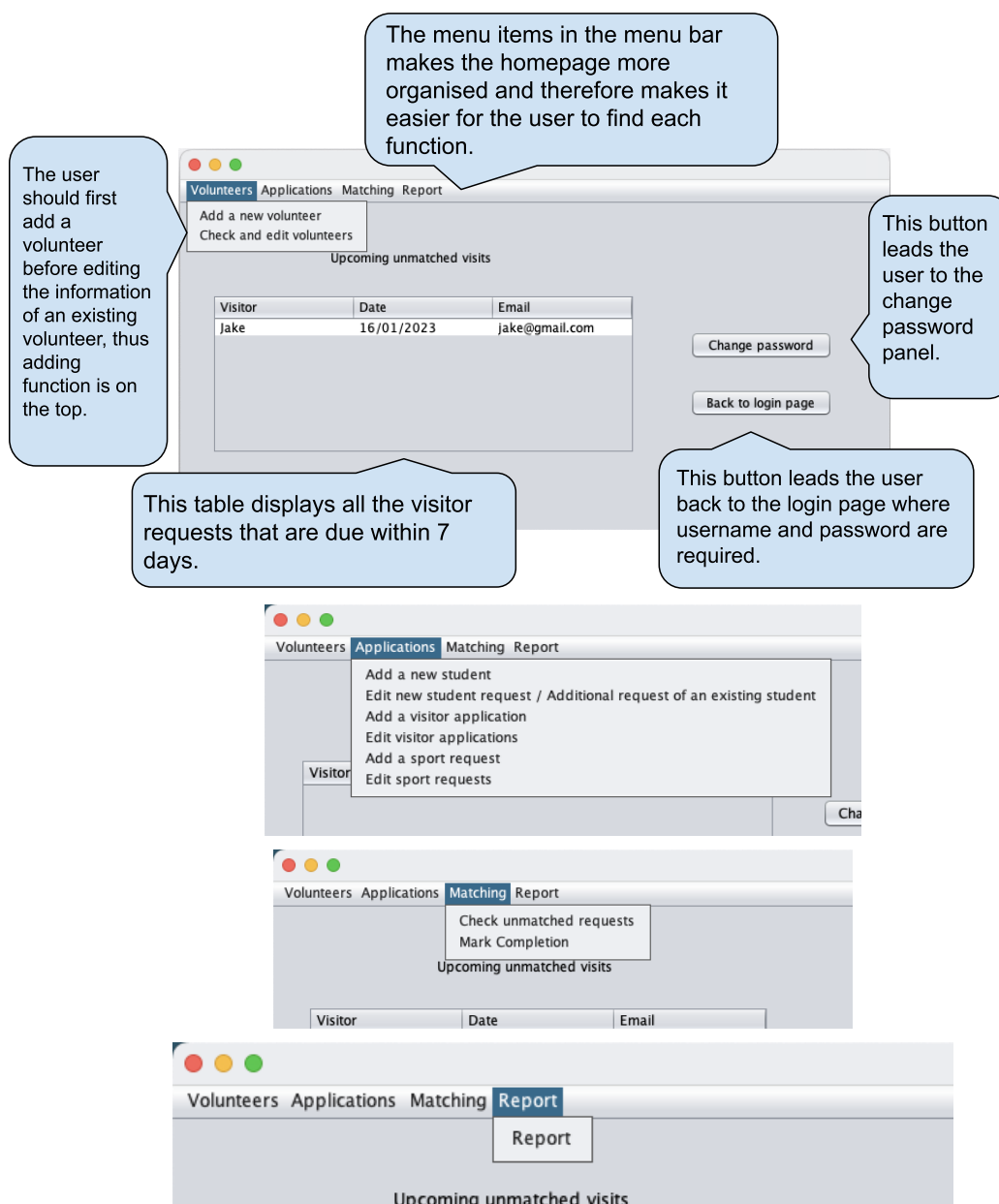
1. User friendly GUI
 - a. Menu system and organization of panels
 - b. Labels in the panels to guide user
 - c. Disabling of radio buttons and combo boxes
 - d. Minimal data entry
2. Modular development and organization
3. File handling
 - a. Storing data to text files (writing files)
 - b. Retrieving data from text files (reading files)
 - c. Complex searches and editing (selection and report panels)
 - d. Relational database model
4. Data validation and verification
 - a. Presence check
 - b. Range check
 - c. Format check
 - d. Uniqueness check
5. Algorithmic thinking
 - a. Automatic filtration of volunteers when matching applications and volunteers
6. Use of existing libraries
 - a. Date and time selection using JCalendar
 - b. Dynamic JTables
 - c. Sending emails via javax.mail.jar

User friendly GUI

a. Menu system and organization of panels

All the sub panels are linked to a single home page so that all these panels can be accessed through simple clicks on the dropdown menu at the top of the home page. The order of the menu items are organized in a way that follows the order of the working process from left to right, preventing the user from realizing that they need to first enter a sub panel before entering the panel that they are already in.

Example 1: homepage



b. Labels in the panels to guide user

There are labels throughout the program to guide the user on what should be done for each entry. Some are descriptions in sentences while others are short labels.

Example 2: add sports event

Add sport event module

Sport name

Number of volunteers needed for each section of the event

Score board ☐ Boundary line ☐

Sports equipments ☐ Venue layout ☐

Match arrangement ☐ Medical equipments ☐

The labels help the user to understand what data needs to be entered, and specify what data each textbox require.

c. Disabling of radio buttons and combo boxes

For the purpose of maintaining the consistency of data entry, certain radio buttons and combo boxes are disabled when the program is initialized and will only be accessible when appropriate options are chosen to prevent garbage data entry.

Example 3: add new volunteer

Add new volunteer module

ID number

Name

Email address

Grade level

Volunteer type (you are welcome to select more than one type of volunteer) :

☒ Peer helper If selected, please fill out speciality(ies) below

☐ Marketing helper If selected, please fill out speciality(ies) below

☐ PE helper If selected, please fill out speciality(ies) below

☐ Score board ☐ Boundary line ☐ Sports equipments

☐ Venue layout ☐ Match arrangement ☐ Medical equipments

Please select the time period(s) that you are available to be a volunteer in the table below (maximum 2 periods)

All the disabled combo boxes are only activated once their corresponding type of volunteer has been selected.

d. Minimal data entry

Based on the options the user has chosen, some options are removed from the comboboxes by filtering and thus the user doesn't have to worry about choosing the unsuitable volunteer. Meanwhile, selecting volunteers instead of typing prevents data entry errors.

Example 4: matching requests and volunteers

The screenshot shows a 'Matching module' window with three main sections: 'Application filter', 'Eligible volunteers', and a bottom section with 'Save pairing' and 'Exit' buttons. The 'Application filter' section includes a table for 'Application details below' and a 'Select application type' dropdown. The 'Eligible volunteers' section includes a table for 'Volunteer details below' and a 'Select volunteer' dropdown. Three callouts provide additional context:

- Callout 1:** This combo box selected by user specifies the type of request being selected. (Points to 'Select application type')
- Callout 2:** The specific request of the selected type is chosen in this combo box, identified by visitor's name and email. (Points to 'Select application')
- Callout 3:** The ID of the volunteer in this combo box is already filtered by the system, which means not all the marketing helpers are loaded into this combo box, but instead those who are listed here are available during the requested time, and are able to speak the same language as the visitor. (Points to 'Select volunteer')

Application filter

Application details below

Name	Phone No.	Email	Date	Period	Language
Wayne	12345678	wayne@gmail.com	17/01/2023	9:30 am to 10:50 am	Chinese

Eligible volunteers

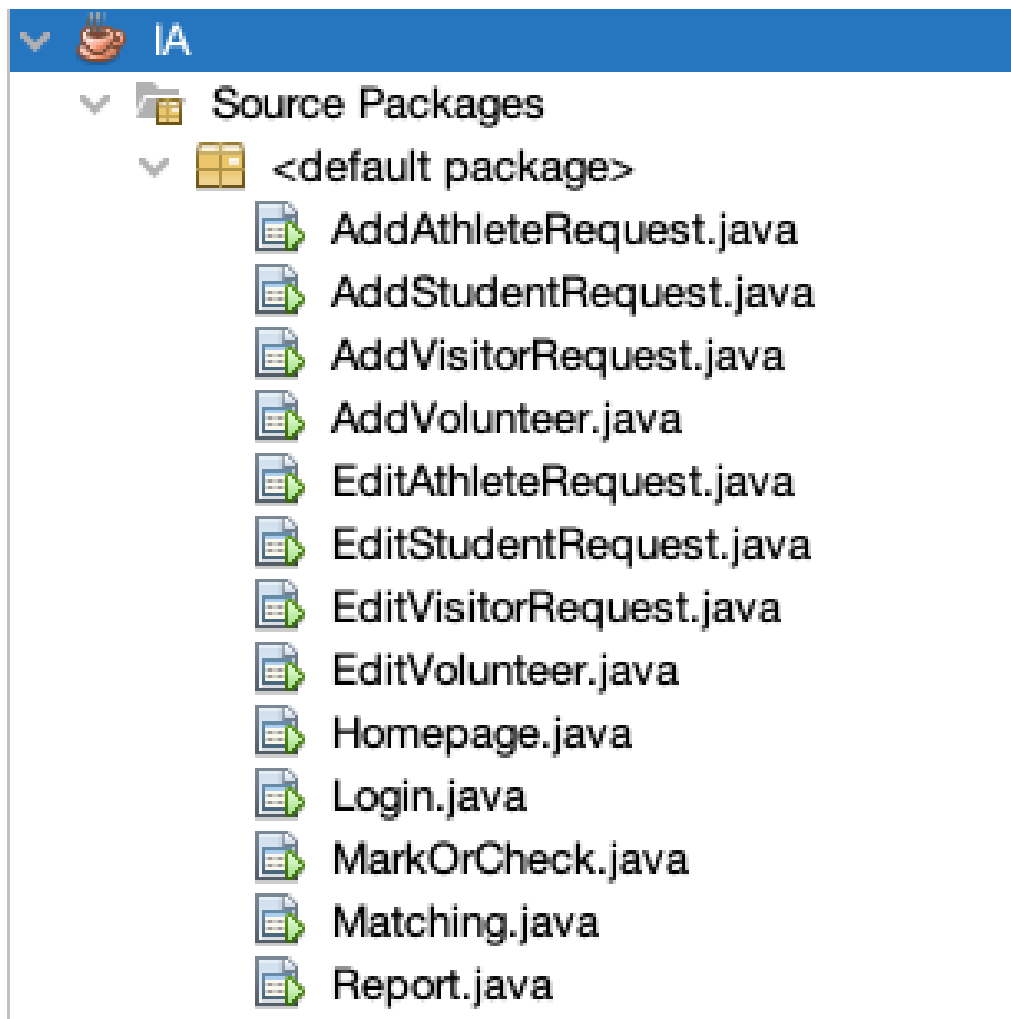
Volunteer details below

ID	Name	Language 1	Language 2	Period 1	Period 2
----	------	------------	------------	----------	----------

Buttons: Save pairing, Exit

Modular development and organization

All the functions within the program are connected to the menu options from the menu bar in the homepage. Each function within the program has their corresponding individual panels which allows them to function independently and does not interfere with each other. This leads to the easy modifiable and updatable characteristics of this program.



File handling

a. Storing data to text files (writing files)

In order to utilize data of volunteers and applicants, data must be stored in a database. Using the FileWriter and PrintWriter method, the program takes the validated data inputted by the user and transforms it into a text of a consistent format.

Example 5: add volunteer

Add new volunteer module

ID number
12345

Name
Peter

Email address
peter@gmail.com

Grade level
12

Volunteer type (you are welcome to select more than one type of volunteer) :

☒ Peer helper If selected, please fill out speciality(ies) below
Science Music

☐ Marketing helper If selected, please fill out language(s) you speak below
N/A N/A

☐ PE helper If selected, please select the section(s) that you can help below
☐ Score board ☐ Boundary line ☐ Sports equipments
☐ Venue layout ☐ Match arrangement ☐ Medical equipments

Please select the time period(s) that you are available to be a volunteer in the table below (maximum 2 periods)

Monday Period 4 Wednesday Period 4

Add a new volunteer Back

```
String na = name.getText();
int ID = Integer.parseInt(id.getText());
String em = email.getText();
int gr = Integer.parseInt(grade.getSelectedItems()[0]);

String period1 = p1.getSelectedItems()[0];
String period2 = p2.getSelectedItems()[0];

FileWriter vol = new FileWriter("Volunteer.txt", true);
PrintWriter volp = new PrintWriter(vol);
if(error == 0)
{
    volp.println(na+","+ID+","+em+","+gr+","+period1+","+period2);
}
volp.close();
```

Volunteer.txt

Peter,12345,peter@gmail.com,12,Monday Period 4,Wednesday Period 4

```
if(peer.isSelected())
{
    String spe1 = s1.getSelectedItems()[0];
    String spe2 = s2.getSelectedItems()[0];

    FileWriter p = new FileWriter("Peer.txt", true);
    PrintWriter pp = new PrintWriter(p);

    if(error == 0) pp.println(ID+","+spe1+","+spe2);
    pp.close();
}
```

Peer.txt

12345,Science,Music

b. Retrieving data from text files (reading files)

For the purpose of eliminating entry errors, and also for the convenience of the user, the program could restore the saved data of volunteers and applicants, which requires reading data from the database. Using the FileReader, BufferedReader, for loop and StringTokenizer, the program is able to read, retrieve and parse the required data from the storage files.

Example 6: edit volunteer information

The screenshot shows a Java Swing window titled "Edit volunteer module". It contains several input fields and buttons. Callouts provide context:

- Top-left callout:** "The volunteer is selected here by name, ID number and email address" points to the "Select volunteer" dropdown menu which currently shows "Jack,12345,jack@gmail.com".
- Bottom-left callout:** "By clicking this button, all the selection made by the user when adding this volunteer will be loaded, as shown in the other combo boxes" points to the "Read volunteer data" button.
- Right callout:** "By using the same file reading method, the individual txt file for each type of volunteer is read, and if the selected volunteer is found, data is restored" points to the "Specialties" and "language you speak" dropdown menus.

The form includes fields for "Grade level" (set to 12), "Volunteer type" (with checkboxes for Peer helper, Marketing helper, PE helper), "Specialties" (Science, Music), "language you speak" (Chinese, Japanese), and "Please select the time period(s) that you are available to be a volunteer in the table below (max 2)" (Monday Period 2, Tuesday Period 2). Buttons at the bottom are "Save changes", "Remove this volunteer", and "Back".

```
FileReader readVul = new FileReader("Volunteer.txt");
BufferedReader rr = new BufferedReader(readVul);

String temp = "";
while((temp=rr.readLine())!=null)
{
    StringTokenizer tok = new StringTokenizer(temp, ",");

    String na = tok.nextToken();
    int id = Integer.parseInt(tok.nextToken());
    String em = tok.nextToken();

    String gr = tok.nextToken();
    String pr1 = tok.nextToken();
    String pr2 = tok.nextToken();

    if((id+ "").equals(idnum))
    {
        grade.setSelectedItem(gr);
        p1.setSelectedItem(pr1);
        p2.setSelectedItem(pr2);
    }
}
rr.close();
```

c. Complex searches and editing (selection and report panels)

Using the methods introduced in the previous section (section b), the program is able to conduct complex searches by parsing data from the database and displaying them into the tables, and in turn allowing the user to easily conduct viewing and filtration.

Example 7: matching volunteer and request

After selecting the specific application (request), the corresponding txt file will be read and data will be taken and displayed in the table on the right hand side.

Matching module

Application filter

Select application type

Visitor

Select application

Wayne.wayne...

Select volunteer

12345

Application details below

Name	Phone No.	Email	Date	Period	Language
Wayne	12345678	wayne@gmail.com	17/01/2023	9:30 am to 10:50 am	Chinese

Eligible volunteers

Volunteer details below

ID	Name	Language 1	Language 2	Period 1	Period 2
12345	Jack	Chinese	Japanese	Monday Period 2	Tuesday Period 2

Save pairing

Exit

```
FileReader readSR = new FileReader("Visitor.txt");
BufferedReader SR = new BufferedReader(readSR);

String temp = "";

while ((temp = SR.readLine()) != null) {
    String[] arr = temp.split(",");

    if ((appkey.getSelectedItem() + "").equals(arr[0]+","+arr[2])) {
        res = arr;
        break;
    }
}

SR.close();

tab.insertRow(tab.getRowCount(), new Object[]{res[0], res[1], res[2], res[3], res[4], res[5]});
```

FileReader readSR = new FileReader("Visitor.txt");
BufferedReader SR = new BufferedReader(readSR);

String temp = "";

while ((temp = SR.readLine()) != null) {
 String[] arr = temp.split(",");

 if ((appkey.getSelectedItem() + "").equals(arr[0]+","+arr[2])) {
 res = arr;
 break;
 }
}

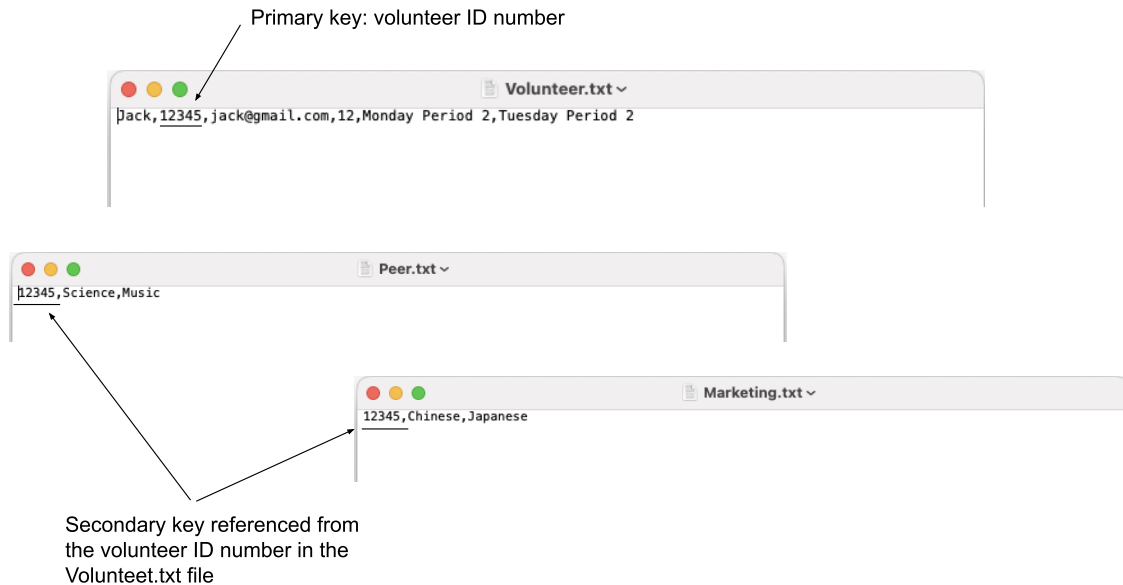
SR.close();

tab.insertRow(tab.getRowCount(), new Object[]{res[0], res[1], res[2], res[3], res[4], res[5]});

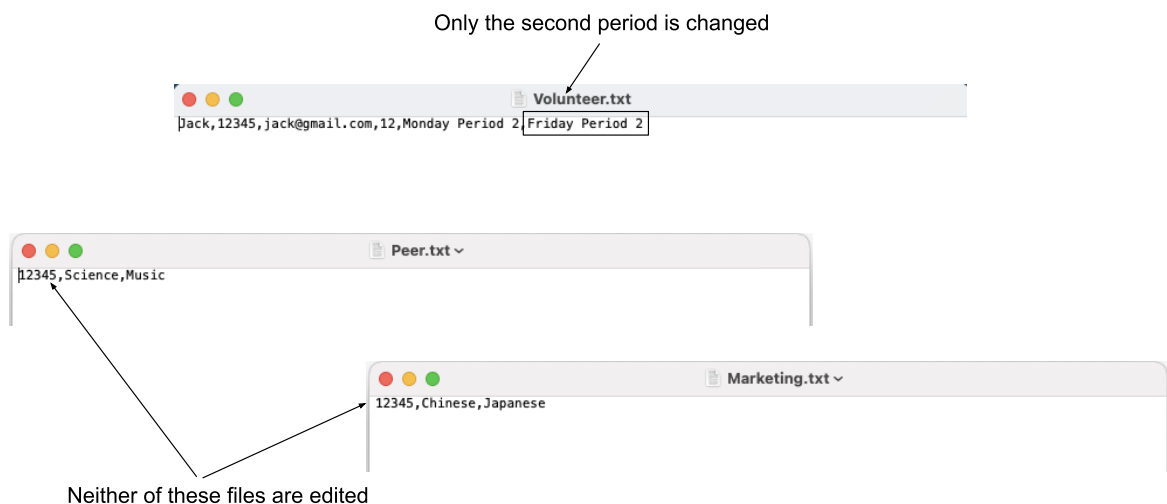
d. Relational database model

Relational database is employed in this system to help eliminate data redundancy and therefore save storage space. In addition, it also saves users from the trouble of changing a single data repeatedly over multiple files, and therefore makes the editing process earlier.

Example 8: interlinked Volunteer.txt, Peer.txt and Marketing.txt files



If the volunteer changes his second period from “Tuesday Period 2” to “Friday Period 2”, only the file containing time periods is changed, no editing is done on any other files, as demonstrated below:

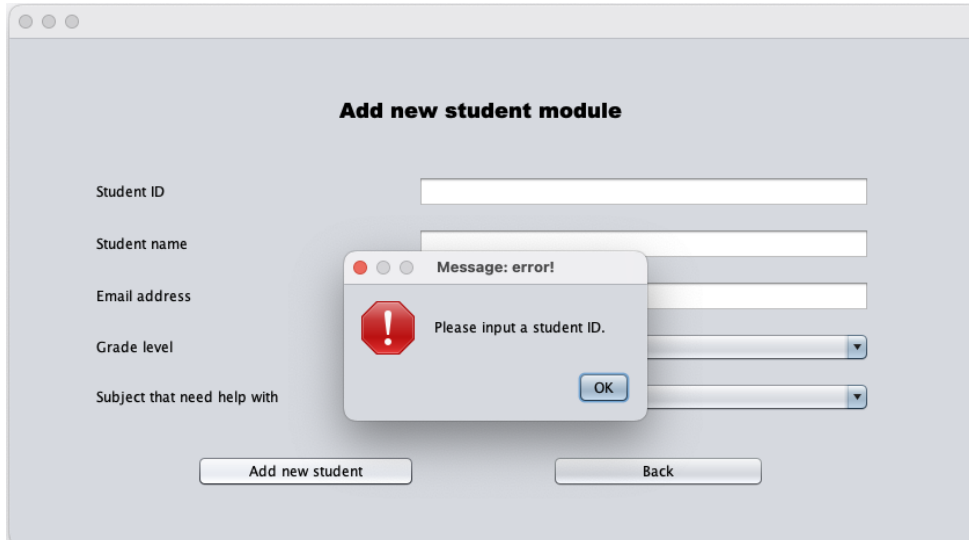


Data validation and verification

a. Presence check

This data validation technique is employed to prevent the user from leaving necessary fields blank and thus ensures data consistency. Data is stored only if it is complete.

Example 9: add new student

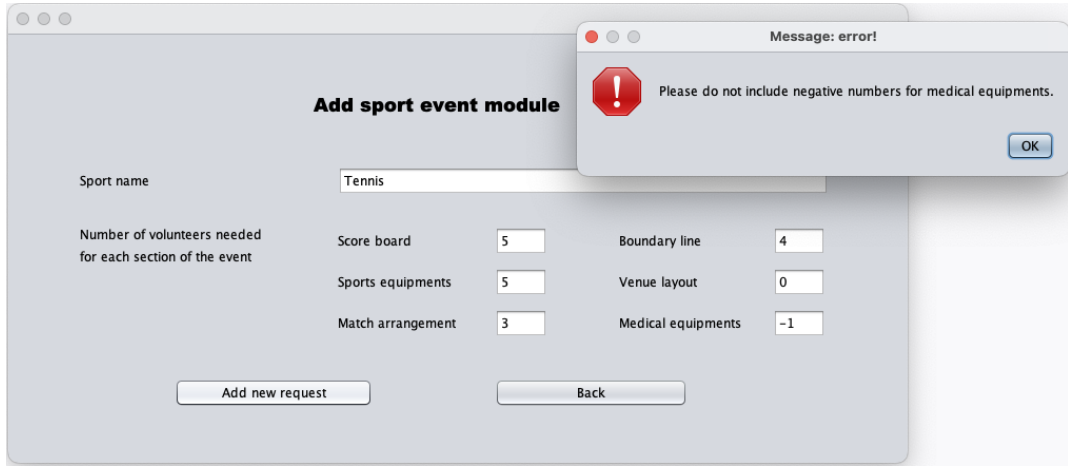


```
if(stuid.getText().equals(""))
{
    javax.swing.JOptionPane.showMessageDialog(null, "Please input a student ID.", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
    error++;
}
if(name.getText().equals(""))
{
    javax.swing.JOptionPane.showMessageDialog(null, "Please input the student's name.", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
    error++;
}
if((grade.getSelectedItem()+ "").equals("Select grade"))
{
    javax.swing.JOptionPane.showMessageDialog(null, "Please select a grade level.", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
    error++;
}
if((help.getSelectedItem()+ "").equals("Select subject"))
{
    javax.swing.JOptionPane.showMessageDialog(null, "Please select a subject.", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
    error++;
}
```

b. Range check

This data validation technique checks if the inputted numerical data is within the reasonable range.

Example 10: add sports event



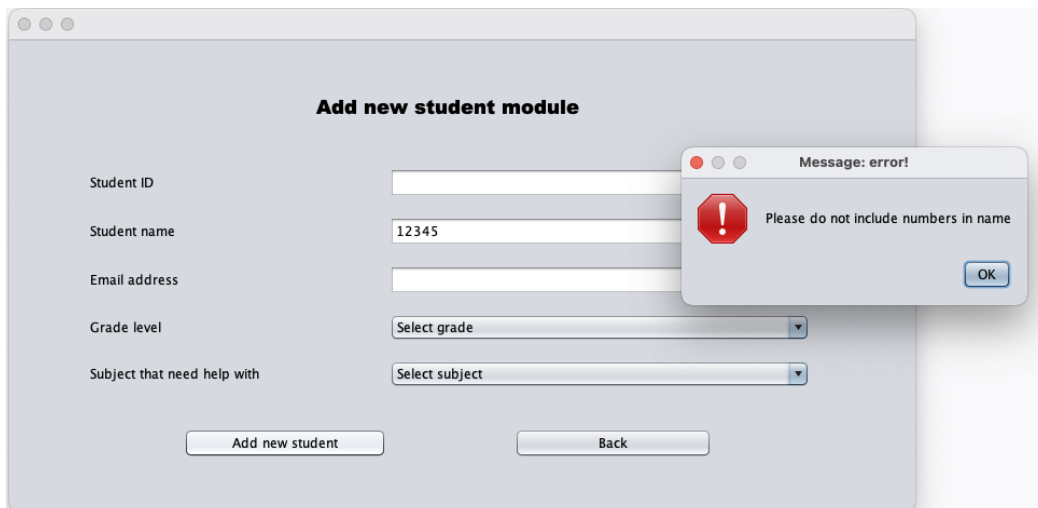
The screenshot shows a Java Swing window titled "Add sport event module". It contains several text input fields and numeric input fields. The "Sport name" field contains "Tennis". The "Number of volunteers needed for each section of the event" is a label for a group of numeric input fields. These fields are: "Score board" (5), "Boundary line" (4), "Sports equipments" (5), "Venue layout" (0), "Match arrangement" (3), and "Medical equipments" (-1). At the bottom are "Add new request" and "Back" buttons. An error message dialog box is overlaid on top, titled "Message: error!", with a red warning icon and the text "Please do not include negative numbers for medical equipments." and an "OK" button.

```
if(Integer.parseInt(score.getText())<0 || Integer.parseInt(boundary.getText())<0 || Integer.parseInt(sports.getText())<0 || Integer.parseInt(venue.getText())<0 || Integer.parseInt(match.getText())<0 || Integer.parseInt(medical.getText())<0)
{
    javax.swing.JOptionPane.showMessageDialog(null, "Number of volunteers needed cannot be negative.", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
    error++;
}
```

c. Format check

This data validation technique checks if the inputted data is of the reasonable type and only allows reasonable data to be entered into the database.

Example 11: add new student



The screenshot shows a Java Swing window titled "Add new student module". It contains several input fields: "Student ID" (empty), "Student name" (12345), "Email address" (empty), "Grade level" (a dropdown menu showing "Select grade"), and "Subject that need help with" (a dropdown menu showing "Select subject"). At the bottom are "Add new student" and "Back" buttons. An error message dialog box is overlaid on top, titled "Message: error!", with a red warning icon and the text "Please do not include numbers in name" and an "OK" button.

```

boolean containNum = false;
for(int i = 0; i < 10; i++)
{
    if(name.getText().contains(i+""))
    {
        containNum = true;
    }
}
if(containNum==true)
{
    javax.swing.JOptionPane.showMessageDialog(null, "Please do not include numbers in name", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
    error++;
}

```

d. Uniqueness check

This data validation technique ensures that certain data which are required to be unique cannot be entered when it is the same as any of the previously entered data.

Example 12: add new student

```

FileReader r = new FileReader("New.txt");
BufferedReader rr = new BufferedReader(r);

String current = "";
while ((current = rr.readLine()) != null) {
    String[] arr = current.split(",");

    if(arr[0].equals(stuid.getText()))
    {
        javax.swing.JOptionPane.showMessageDialog(null, "ID already exist. Please create a new ID", "Message: error!", javax.swing.JOptionPane.ERROR_MESSAGE);
        error++;
    }
}

```

Algorithmic thinking

a. Automatic filtration of volunteers when matching applications and volunteers

The program allows the user to match the applications and volunteers. However, the user doesn't randomly pick volunteers; instead, he needs to ensure that suitable volunteers are selected. This is done by finding marketing helpers with the same free period and language as a visitor, finding peer helpers in the same grade and same subject as a new student, etc. The algorithm in the program does this automatically in order to simplify the matching process for the user.

Example 13: pseudocode algorithm for automatic filtration of volunteers

```
eligible // an empty list that will be filled by the records of filtered volunteers
language // a given string indicating the language that the visitor speaks
period // a given string indicating the time period that the visitor requested, transferred into
          school class periods (i.e. Period 1)
date // the date that the visitor asked for a visit, transferred to the form of weekdays (i.e Monday)

n // integer value indicating number of lines in the file "Marketing.txt"

fileVariable openread("Marketing.txt")
Loop i from 1 to n
    fileVariable.ReadLine(i)

    arr[] = fileVariable.ReadLine(i).split(",")
    arr[0] // the field recording the ID number of the volunteer
    arr[1] // the field recording the 1st language that the volunteer speaks
    arr[2] // the field recording the 2nd language that the volunteer speaks

    If arr[1] equals language
    or arr[2] equals language
    then
        fileVariable openread("Volunteer.txt")

        p // integer value indicating number of lines in the file "Volunteer.txt"
        Loop k from 1 to p
            fileVariable.ReadLine(k)

            colle[] = fileVariable.ReadLine(k).split(",")

            colle[1] // recording the ID number of the volunteer
                     *will be compared with arr[0] to parse the right volunteer
            colle[4] // a time period that the volunteer is available
```

```

        colle[5] // another period that the volunteer is available

        If arr[0] equals colle[1]
        and colle[4] equals period
        or colle[5] equals period
        then
            eligible.addItem(arr[0])
        end if
    end loop
end if
end loop
output eligible

```

Below is part of the actual code of automatic filtration of volunteers

```

public ArrayList getMktHelpers(String lang, String period, String date) {
    ArrayList<String> eligible = new ArrayList<String>();
    try {
        FileReader read = new FileReader("Marketing.txt");
        BufferedReader rr = new BufferedReader(read);

        String curr = "";
        while ((curr = rr.readLine()) != null) {
            String[] arr = curr.split(",");

            if (arr[1].equals(lang) || arr[2].equals(lang)) {
                FileReader readMain = new FileReader("Volunteer.txt");
                BufferedReader rrr = new BufferedReader(readMain);

                String cu = "";
                while ((cu = rrr.readLine()) != null) {
                    String[] colle = cu.split(",");

                    Date weekday = new SimpleDateFormat("dd/MM/yyyy").parse(date);
                    SimpleDateFormat form = new SimpleDateFormat("EEEE");
                    String wd = form.format(weekday);

                    String per = "";
                    if (period.equals("8:00 am to 9:20 am")) {
                        per = "Period 1";
                    } else if (period.equals("9:30 am to 10:50 am")) {
                        per = "Period 2";
                    } else if (period.equals("12:10 pm to 13:30 pm")) {
                        per = "Period 3";
                    } else if (period.equals("13:40 pm to 15:00 pm")) {
                        per = "Period 4";
                    }

                    String dape = wd + " " + per;

                    if (arr[0].equals(colle[1]) && (colle[4].equals(dape) || colle[5].equals(dape))) {
                        eligible.add(arr[0]);
                    }
                }
            }
        }
        rr.close();
    } catch (Exception e) {
    }
    return eligible;
}

```

Matching module

Application filter

Select application type
New student

Select application
169504,Sam

Select volunteer
12345

Application details below

ID	Name	Email	Grade	Subject
169504	Sam	sam@gmail.com	12	Music

Eligible volunteers

Volunteer details below

ID	Name	Email	Subject 1	Subject 2
12345	Jack	jack@gmail.com	Science	Music

Save pairing Exit

The peer helper must have a same subject as the new student

Matching module

Application filter

Select application type
Visitor

Select application
Wayne,wayne...

Select volunteer
12345

Application details below

Name	Phone No.	Email	Date	Period	Language
Wayne	12345678	wayne@gmail.com	17/01/2023	9:30 am to 10:50 am	Chinese

Eligible volunteers

Volunteer details below

ID	Name	Language 1	Language 2	Period 1	Period 2
12345	Jack	Chinese	Japanese	Monday Period 2	Tuesday Period 2

Save pairing Exit

The marketing helper and visitor must be able to speak the same language

The marketing helper and visitor must share the same free period

Use of existing libraries

a. Date and time selection using JCalendar

A date has 3 components: day, month and year, and the arrangement of these components may vary due to different habits of people from different regions. The program utilizes the JCalendar library which allows the user to select the dates on a calendar instead of manually inputting the date, and in turn eliminates entry errors as well as the trouble of validating dates of different formats.

Example 14: add visitor

The screenshot shows a web application window titled "Add visitor request module". It contains several input fields: "Visitor name", "Phone number", "Email", "Date & Time period", and "Language". The "Date & Time period" field is currently selected, and a JCalendar date picker is displayed over it. The calendar shows the month of February for the year 2023. The days of the week are listed at the top: Mon, Tue, Wed, Thu, Fri, Sat, Sun. The dates are arranged in a grid. The date 22 is highlighted in red, indicating it is the selected date. A "Back" button is visible next to the calendar. The "Language" field has a dropdown menu.

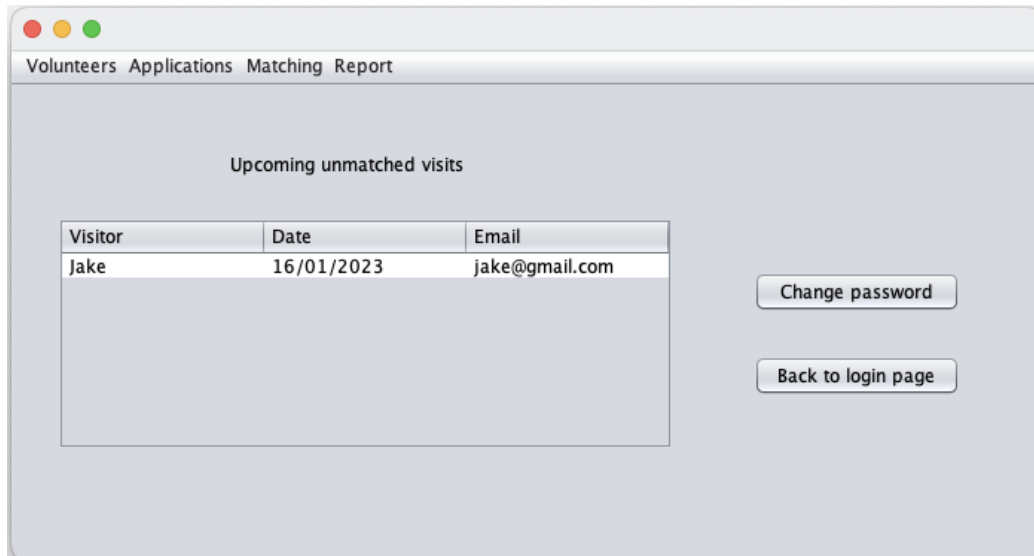
```
Date t = sdate.getDate();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
String ti = sdf.format(t);
```

Code to disable dates prior to the current date to maintain data integrity:

```
sdate.setMinSelectableDate(new Date());
```


The date function is also used in the homepage, which loads all the visitor requests due in 7 days in a table.

Example 15: homepage



```
try
{
    DefaultTableModel mod = (DefaultTableModel) unmatched.getModel();
    mod.setRowCount(0);
    mod.setColumnCount(0);
    mod.addColumn("Visitor");
    mod.addColumn("Date");
    mod.addColumn("Email");

    FileReader read = new FileReader("Visitor.txt");
    BufferedReader rr = new BufferedReader(read);

    String x = "";
    while((x=rr.readLine())!=null)
    {
        String[] line = x.split(",");
        String rqtime = line[3];

        Date date = new SimpleDateFormat("dd/MM/yyyy").parse(rqtime);
        Date today = new Date();

        double days = 0;

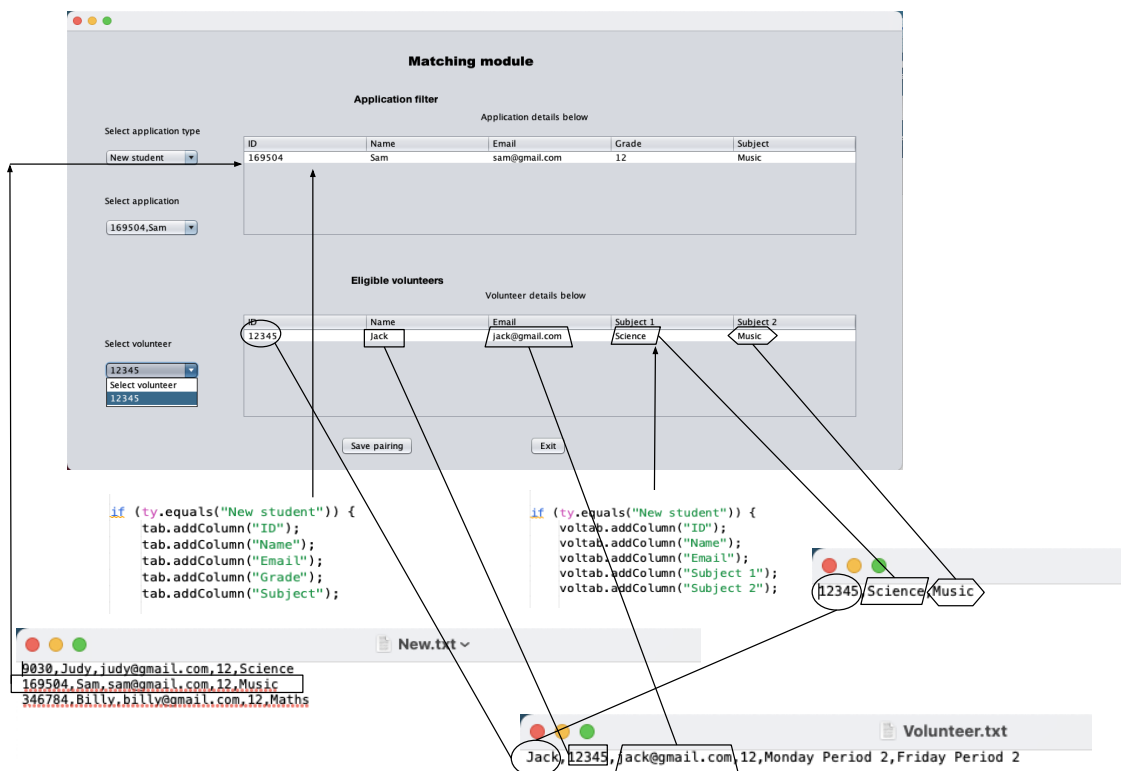
        try
        {
            long diff = date.getTime() - today.getTime();
            days = diff/(1000*60*60*24);
        }
        catch(Exception e)
        {
        }

        if(days<=7 && days>=0)
        {
            mod.insertRow(mod.getRowCount(), new Object[]{line[0], line[3], line[2]});
        }
    }
}
catch(Exception e)
{
}
```

b. Dynamic JTables

Data from different types of requests could be different. When they are displayed on a table, the title and number of the columns could vary. To prevent the user from the trouble of frequently switching between tables, the program contains a dynamic table that could flexibly change according to the type of requests or volunteers. This is achieved by manipulating the JTable function.

Example 16: matching module



c. Sending emails via javax.mail.jar

Once an application and volunteer are matched, a match is completed or canceled, the program is able to send email to both the applicant and volunteer to notify them.

Example 17: code for function SendEmail

```
public class SendEmail {  
  
    public static void sendingEmail(String toApplicant, String toVolunteer, String subject, String content) {  
        final String username = "yhyang23@iskl.edu.my";  
        final String password = "odkbgnoiwcguziyc";  
  
        Properties prop = new Properties();  
        prop.put("mail.smtp.host", "smtp.gmail.com");  
        prop.put("mail.smtp.port", "465");  
        prop.put("mail.smtp.auth", "true");  
        prop.put("mail.smtp.socketFactory.port", "465");  
        prop.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");  
  
        Session session = Session.getInstance(prop,  
            new javax.mail.Authenticator() {  
                protected javax.mail.PasswordAuthentication getPasswordAuthentication() {  
                    return new javax.mail.PasswordAuthentication(username, password);  
                }  
            });  
  
        try {  
            Message message = new MimeMessage(session);  
            message.setFrom(new InternetAddress("yhyang23@iskl.edu.my"));  
            message.setRecipients(  
                Message.RecipientType.TO,  
                InternetAddress.parse(toApplicant)  
            );  
            message.setRecipients(  
                Message.RecipientType.CC,  
                InternetAddress.parse(toVolunteer)  
            );  
            message.setSubject(subject);  
            message.setText(content);  
  
            javax.mail.Transport.send(message);  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Matching module

Application filter

Select application type

Visitor

Select application

Wayne,wayne...

Select volunteer

196832

Application details below

Name	Phone No.	Email	Date	Period	Language
Wayne	13426885439	wayne@gmail.com	27/02/2023	8:00 am to 9:20 am	Chinese

Eligible volunteers

Volunteer details below

ID	Name	Email	Language 1	Language 2	Period 1	Period 2
196832	Luke	luke@gmail.com	Chinese	English	Monday Period 1	Monday Period 4

Save pairing

Exit

```
String to = tab.getValueAt(0, 2) + "";
String cc = voltab.getValueAt(0, 2) + "";
String subj = "Marketing helper assigned to " + tab.getValueAt(0, 0);
String cont = "Dear " + tab.getValueAt(0, 0) + ",
+ \"\nWe found a marketing helper that is suitable for you. Below is the information of this helper:\"
+ \"\nName: \" + voltab.getValueAt(0, 1)
+ \"\nEmail: \" + voltab.getValueAt(0, 2)
+ \"\nDate: \" + tab.getValueAt(0, 3) + \" from \" + tab.getValueAt(0, 4)
+ \"\nLanguage: \" + tab.getValueAt(0, 5);
SendEmail.sendEmail(to, cc, subj, cont);
```

Marketing helper assigned to Wayne

yhyang23@iskl.edu.my

to wayne,luke

Dear Wayne,
We found a marketing helper that is suitable for you. Below is the information of this helper:
Name: Luke
Email: luke@gmail.com
Date: 27/02/2023 from 8:00 am to 9:20 am
Language: Chinese

Word count: 836

Works cited:

Mkyong. "JavaMail API - Sending Email via GMAIL SMTP Example." *Mkyong.com*, 10 Apr. 2019, <https://mkyong.com/java/javamail-api-sending-email-via-gmail-smtp-example/>.

"Send Mail to Multiple Recipients in Java." *Stack Overflow*, 1 Nov. 1959, <https://stackoverflow.com/questions/13854037/send-mail-to-multiple-recipients-in-java>.