Meeting Scheduling

Final Report

Michael Jacob
4-19-2020
Student ID: 1625998
Computer Science BSc
Supervisor: Prof. Tomasz Radzik

ORIGINALITY AVOWAL

I verify that I am the sole author of this report, except where explicitly stated to the contrary.

ABSTRACT

The purpose of this project is to enable automatic scheduling of meetings within an organisation setting. The aim is to produce a user-friendly system that can be used to create and organise meetings with others, without the requirement of knowing all details of the meeting. The system can automatically suggest the missing details. For example, a user may wish to schedule a meeting at noon but not know which date to schedule it on. The targeted features of the meeting to be generated may be any or all of the date, time and location. The project also aims to produce a second type of automatic scheduling, in which many meetings can be automatically scheduled at once, with the intention of scheduling as many as possible in such a way that avoids clashes.

Part of the project involves the creation of an interface that allows users to interact with the system in various ways, such as to view their meetings in a calendar format and view the details of each meeting. Once a meeting is scheduled, users will be able to use the interface to see these details as well as an invitation to attend the meeting.

ACKNOWLEDGEMENTS

Prof. Tomasz Radzik

CONTENTS

1 INTRODUCTION

Meetings are an integral part of running a successful organisation. They provide a good opportunity for communication and planning future direction. Arranging meetings should be a quick and straightforward task. With the aid of technology, it is possible for an application to be created enabling different people to arrange meetings. This avoids the necessity of contacting each attendee to ensure availability.

The purpose of this project is to create a meeting scheduling application designed to be used by an organisation. It will improve the efficiency and speed of arranging meetings for the members compared to operating without this system. This will allow them to use their time more productively which should reduce cost and result in improved profitability for the organisation.

The system should enable users to manage their meetings as well as to create new meetings. It will be a simplified system that will provide quick access to their calendar. This will benefit members by introducing an easy way to manage their schedules. This system will save time by providing an opportunity to automate some features of a new meeting. It will also be possible to create many meetings and schedule a batch of them together. There are other existing systems to be examined later in further detail.

The chosen environment for this project is as a Java FX application with a database written in SQL. Therefore, the project includes a graphical user interface as well as the database back end. This will provide simple forms to create, edit and delete information in the database. Users will also be able to view the availability of rooms in a calendar form and respond to invitations to meetings.

2 BACKGROUND

The aim of this project is to create an application that would enable members of an organisation to schedule meetings.

It will be possible to schedule these meetings automatically based on constraints and preferences selected by the user. Constraints would include the availability of the room and attendees and the duration of the meeting. Preferences would include a preferred time of day.

Without such a system, members would have to spend time contacting those they wish to attend and room organisers to ensure availability. It would then be necessary to await responses and if any are negative, they would have to start the process again and this would prove very inefficient. Many existing applications only contain recording systems so the whole process would still be necessary.

With this new application it will also be possible to leave the scheduling to a separate batch scheduler that will be able to allocate requested meetings in a way that suits the greatest number of meetings possible. This kind of an application would be useful for large organisations with many venues and potential attendees to both keep track of meetings and schedule them in a less time-consuming manner.

Another advantage for potential users is the ability to schedule multiple meetings in co-ordination with each other through use of the batch scheduler. This helps to prevent a scenario in which previously confirmed meetings leave no space for a new, important meeting. Members then have to attempt to reschedule previous meetings, wasting valuable time. The application is intended to be general enough to support many different types of organisations.

The system could be used by large organisations which may hold several types of internal meetings such as staff meetings, annual reviews, disciplinary meetings or performance meetings. Meetings would also need to be arranged for discussions with external clients. It is crucial for the organisation to run efficiently to have in place a meeting scheduling application which would enable meetings to be speedily and accurately arranged.

The system could also be used on a smaller scale, for example by a student union which may wish to organise discussions on student events, visits by guest speakers or other committee meetings. The work done by the students for the union is voluntary work and therefore an efficient system will make less demands on their valuable time.

2.1 Existing Applications

There are several currently available applications that provide a meeting scheduling service. Some of these applications are briefly considered here.

Microsoft Outlook is multipurpose software primarily known as an email application. It has features to enable meeting scheduling. Its Scheduling Assistant shows current calendars of people and room availability and can be used to auto fill emails to invite attendees. It also has a useful graphical view for choosing an appropriate time and place. It does not make suggestions for scheduling new meetings and does not provide opportunities to filter or add constraints. Each room has an appointed administrator to approve meetings in that room. Outlook can suggest the creation of meetings based on the wording of received emails. Outlook is widely used with an average score of 4.3/5 from 943 reviews [1], although these are reviews of the application as a whole.

Google Calendar is an online calendar linked to the Gmail email service that provides the ability to record events and reminders. An event may be a meeting and guests can be added to it provided they have a Gmail account. Guests can be authorised to edit the event. Meeting times can be suggested by the system dependent entirely on the calendars of the guests to find a time all guests are available. It can be used to book rooms if detailed information has been entered in a correctly structured way. If that is the case, it can also suggest rooms based on the locations of attendees, history of meetings, and equipment and capacity requirements. Meetings can be scheduled to occur at regular intervals. It has a simple but clear layout in the form of a single calendar with screens for creating a new event. Google Calendar is a very popular application and receives many positive reviews with an average score of 4.6/5 from 985 reviews [2].

Calendly is an application in which users set up an account and adjust their settings to create preferred meeting types. Other authorised users can then book meetings with the user in accordance with these preferences. It integrates with other calendar software such as Google Calendar to make the user's current schedule known and update the calendar as new meetings are scheduled. Calendly is quite different from Outlook and Google Calendar as the perspective is geared primarily towards the user receiving an invitation. It enables a user to set up slots with specified characteristics for others to book meetings with them. This design choice makes scheduling meetings easier as the application is less reliant on the scheduler's knowledge of the user's requirements. Calendly is not as high profile an application but nonetheless is well regarded with an average score of 4.6/5 from 650 reviews [3].

X.ai makes use of artificial intelligence to communicate on behalf of users with others to arrange meetings. The AI is able to construct and send emails including responses to requests for adjustment from the recipient. This helps save time by removing the need for the user to involve themselves with back and forth negotiation over meeting times and places. It integrates with other software such as

Google Calendar and office 365 and uses this software to book rooms and know schedules. X.ai is gaining popularity with an average score of 4.2/5 from 49 reviews [4].

All the above applications are popular and widely used, each providing their own different ways to schedule meetings. However, none of these applications provide the ability to set aside a batch of meetings to be scheduled with consideration of each other. This project aims to produce an application that includes more customisable features allowing for greater flexibility as well as automatic scheduling to utilise time in an efficient way.

3 REQUIREMENTS

The project requires the creation of a system suitable for use by members of an organisation. The members need to be able to schedule meetings with each other in venues commonly used by the organisation. In order for the system to run smoothly and efficiently, certain criteria have to be met.

Only members should be able to access the system. There should be security measures in place to limit access.

Only some members should have authority to manage and set up the system. There should be another level of access rights permitting this.

Members should be able to view their current meetings. It is important that the system contains some method for this that is user friendly.

Members should be able to create new meetings. This should be possible without a reliance on the automatic scheduling.

Members should be able to create new meetings automatically without needing to supply all the exact information.

Members should be able to create new meetings to be scheduled taking in to account other meetings that need to be scheduled. It should be possible for an automatic schedule to be done that focuses on optimising the scheduling for a number of meetings rather than just a single meeting.

Members should be able to view availability of rooms. It would be inconvenient for members to attempt to schedule meetings in rooms if they were unable to check if that room is available.

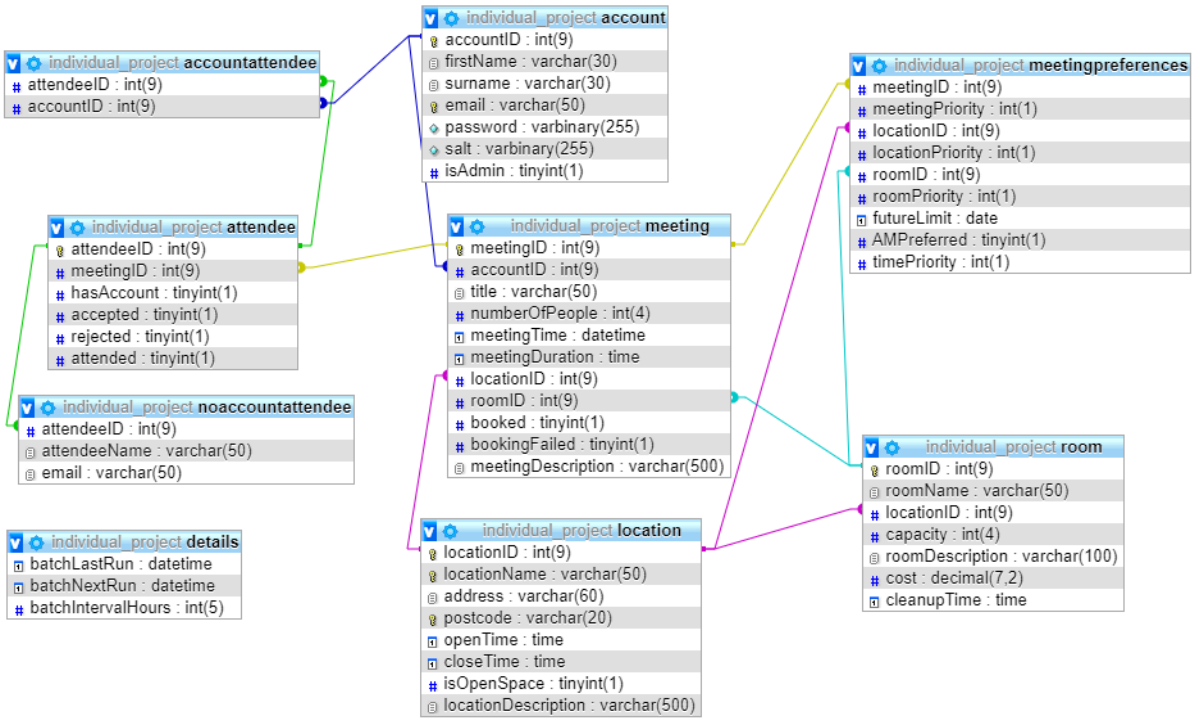
Meetings should be scheduled at a time when all attendees and the required room are available, to avoid clashes.

A system set up with the above features will enable members to successfully manage their schedules.

4 SPECIFICATION

The system will have a database to store the necessary information about the environment.

The database specification is shown in the diagram.



Account

This entity models the user accounts. It contains attributes for the name, email and password of the user and whether the account has administrator privileges. These privileges will enable the account to manage and set up the system. The password itself is not stored in the database, rather the hash and associated salt.

Location

An area or building of the organisation. It has a name, address and postcode as well as opening hours and a description of the location. Open space is not currently in use. It was intended to indicate a large space that does not require a room to schedule a meeting.

Room

A particular room within a location. Locations can have many rooms. Rooms have a name, location, capacity and description. There may also be a monetary cost

involved in booking the room and a clean-up time during which a room will be unavailable after the end of a meeting.

Meeting

A meeting scheduled in the system. Meetings have an organising account, a time and place and a description. The time and place are chosen when the meeting is entered to the database and finalised when the meeting is booked. Meetings will always be entered as already booked unless left to the batch scheduler, in which case the entered time and place are simply place holders until the meeting is booked. The attributes booked and booking failed indicate the current status of the meeting. The number of people intended for the meeting is also recorded.

Meeting Preferences

This entity is for storing the preferences and their priorities for an individual meeting. If a meeting is not scheduled immediately, these constraints will be used to find an appropriate time and place. The preferences are the location and room, whether a morning or afternoon meeting is preferable, and the latest date the meeting can take place. The overall priority of the meeting itself is also stored.

Attendee

A person attending a meeting. They may or may not possess an account. The system records whether the attendee intends to go to the meeting and can also be used to track their attendance.

Account Attendee

Attendees who have an associated account.

No Account Attendee

A guest at the meeting who is not a user, for example an external client. Their name and email are stored.

Details

An entity to store details of the system. It currently contains the last and next run times of the batch scheduler as well as the specified interval for it to run. It should only ever have a single row. At the moment only last run time is in use as the batch scheduler must be run manually.

4.1 SCHEDULERS

The immediate scheduler will attempt to schedule the meeting in a way that satisfies the user specified details, automating the rest. It will then present the user with a choice of possible options for the meeting, one of which can be selected.

The batch scheduler will use the priorities of the constraints as a heuristic to schedule each meeting in the best conditions. It will attempt to satisfy all constraints but will settle for highest score if this is not possible. Certain constraints such as any preferences given maximum priority and the latest date constraint will be considered unbreakable. The batch scheduler will use the overall priority of the meeting to determine which meeting to schedule if it is not possible to schedule them all.

5 DESIGN

5.1 SCREENS

The application has been created for Windows 10 using Java FX and a model view controller style. It has various screens for different aspects of the project.

The login screen is the screen displayed on starting the application. It contains fields for entering an email and password to log in to the system.

Upon logging in the user is presented with their calendar. The calendar screen displays a calendar style view which shows all meetings relating to the user. This includes meetings organised by the user as well as any meetings they have been invited to attend. These meetings can be selected to view additional details.

Users are also able to change their own account details. Administrator accounts have the ability to create, edit and delete other accounts, locations and rooms. They will be unable to delete their own account but can create and delete other administrators. The screen for each of these actions is a form containing the attributes of the respective entity in the database. When no accounts currently exist, an account must be added upon starting the application, and will be created as an administrator.

All users have the option to create a meeting. The meeting creation process involves progressing through three screens. The first allows the user to enter the title, duration and description of the meeting as well as whether the meeting should be scheduled immediately. The second is for the selection of attendees. Attendees who have accounts may be added by searching by name or email. Attendees who do not have accounts can be added by clicking a link on the screen and then entering a name and email.

The third screen varies depending on whether the meeting is to be scheduled immediately. If that is the case, the user is given the option to enter the location, room, date and time requested, or automate any/all of these aspects. The result, or possible results in the case of an automated meeting, will be displayed in a dialogue. The user may then select a preferred meeting and confirm their choice or refuse all options and try again if they wish to do so.

If the meeting is not to be scheduled immediately the third screen instead provides option to enter how important certain attributes of the meeting are. The user can select a preferred location and room and choose whether an AM or PM meeting is preferred. Each choice is given a priority ranging from 0 - ignore to 5 - unbreakable. The user can also give one of three priorities, ranging from 1 – least important to 3 – most important, to the meeting itself to indicate the importance of scheduling this meeting. The last preference specified by the user is the latest date the meeting can occur. This is always unbreakable.

Meetings created by way of these preferences are put aside to be scheduled by the batch scheduler at a later date. This batch run can be done by administrators from the batch screen. After clicking schedule, all currently unscheduled meetings in the database are attempted to be scheduled by the algorithm. The results are presented to the user in a colour coded form with green to indicate meetings that were scheduled and red for those that were not. The user can decide whether to accept these results or to try again specifying 'victim' meetings that will not be attempted to schedule. This may change the results of the algorithm to something more preferable.

Once scheduled, meetings cannot be changed, although they can be deleted by either an administrator or the organiser. This is done from the details screen of the meeting.

Users are also able to view the meetings currently scheduled in a particular room in a similar calendar view to that of the user calendars. They are able to view details of meetings they are involved in. Users will also be able to view the details of the room itself.

Additionally, users can view the invites they have sent or received and act on them. Received invitations can be accepted or rejected. Sent invites can be marked as attended or did not attend, allowing the organiser to record the attendance at their meetings.

5.2 CLASSES

View related details are mostly handled by .fxml files. In general, each connection class makes use of its model class and the model classes use the connection classes. Structures and utilities are used throughout.

Connection classes

Connect governs the connections to the database. It makes use of connection pooling to help prevent ports from overloading. There are connection classes for account, location, room, meeting, attendee, details and meeting preferences. Each of these contains queries relating to their respective entities in the database.

Structures

There is a structured class for each entity apart from details. Attendee is an abstract class extended by account attendee and no-account attendee. There is also a class for invites. Provisional meeting models a meeting in the process of creation that has yet to be added to the database.

List classes

The controller and model classes for the accounts screen maintain a list of the accounts in the database and display it on screen in a table format. Users are able to search the list to reduce the size of the table. Actions can be taken for each user. Their details can be edited, their password can be changed and they can be deleted. This is an administrator only screen.

The rooms and locations controllers and models behave similarly to accounts. Locations allows the actions edit and delete, rooms also allows the user to view the room calendar. As a result, rooms is the only one of these screens that is not administrator exclusive. The normal user version does not provide delete as an option and rather than edit, allows the user to only view the details, although this is enforced only in the edit room controller.

Add/edit

There are controller and model classes to add and edit accounts, locations and rooms. These screens are presented as forms. The user input is entered in to the database if it is considered valid. All of these screens are administrator only with the exception of edit room for which the normal version is view only.

Create meetings

The add meeting controller is for the first page of the add meeting form into which the duration, title and description are entered. This is also when it is decided whether the meeting will be scheduled immediately or left to the batch scheduler. It creates a provisional meeting instance to be used by the rest of the form.

Add meeting attendees and add no account attendee serve the next stage of meeting creation, adding attendees. Account attendees can be searched for, selected and added whereas those that do not have an account must have their name and email entered into a form. The currently selected attendees are displayed in a list and can be removed before progressing to the next screen.

The add meeting now classes are used to finalise an immediate scheduling of a meeting. The user can enter a location, room, date and time and decide which to automate. If all are automated the meeting is scheduled as long as constraints are met. Otherwise the automation is left to the algorithm.

Add meeting later provides the opportunity to add the meeting directly to the database with an accompanying set of preferences. The meeting is not marked as booked as it is instead left to the batch scheduler.

Batch

The batch classes allow administrators to run the batch scheduler. The display shows the list of currently unscheduled meetings and the last time the batch scheduler was run. When the algorithm is run the results are presented for approval.

Calendar

The calendar controller can be accessed as a user calendar or as a room calendar. The relevant information is loaded in by the model class.

Invites

Invites contains two tables representing the invites sent and received. Received invites can be marked as accepted or rejected. Sent invites can be marked as attended or not. The resulting status is displayed once the meeting is in the past.

Login

Login model checks the user input of an email and password, hashes the password using the salt associated with the email and checks if it matches the hash stored in the database.

Meeting Details

The model class generates a message that contains the details of the meeting for output to the screen. This is also where the meeting can be deleted by the organiser or an administrator.

Primary/Scene/Interface

The primary controller governs the switching of screens and the sharing of data between them. To do this it makes use of the controller interface. The interface is implemented by scene controller which is an abstract super class for all of the other controllers. Scene controller also contains navigation methods that use primary controller to switch between screens.

Utilities

Password manager contains methods for generating the hash and salt of a password.

Location utility provides useful methods for manipulating certain data regarding rooms and locations.

5.3 ALGORITHMS

Automatic immediate scheduling

This algorithm is run when a user wishing to schedule a meeting immediately chooses to automate some attributes. It is the auto schedule method of the add meeting now class. The algorithm stores a list of solutions and returns up to five of them. Solution is a nested class in add meeting now that stores a location, room, date and time. The algorithm begins by finding appropriate locations. If the user selected the location manually it is considered the only reasonable location. This is also true for the other attributes. Otherwise all locations in to which a meeting can be scheduled are stored. To be able to schedule the meeting, the location must be open long enough and at the time specified by the user if they have chosen to do so.

Having chosen a list of locations, the algorithm then looks for which rooms within these locations can host the meeting. This requires that the room is large enough and that there is a vacant time slot large enough for the meeting.

The next attribute selected is the date. If automated, the date is limited to between tomorrow and eight days' time inclusive. This was chosen as a reasonable period to prevent the algorithm returning an excessively large number of solutions. Dates are chosen for each solution by searching for vacancies during that day in the room associated with the solution.

Times are found in a similar way. The only start times considered are every hour, the opening time of the location and a start time that causes the meeting to finish at the closing time.

Once the list of solutions is found, the algorithm randomly selects solutions to display to the user. Each solution is checked to see if it produces a clash for any of the attendees. Once five solutions that do not clash are found, the algorithm terminates. Five was chosen as a suitable amount to show the user. Although the user cannot change this number, it can be easily changed in the code. The solutions are transformed into objects of type provisional meeting for display. The random behaviour means that were the user to run the algorithm again, they are likely to receive a different selection of solutions.

```
AUTOMATIC IMMEDIATE SCHEDULING

Solutions = new list

If automatic location

    Add solutions for all suitable locations

Else

    Add solution for specified location
```

```
If automatic room
    Add solutions for all suitable rooms for each solution
Else
    Add solution for specified room
If automatic date
    Add solutions for all available dates in the next week for each solution
Else
    Add solution for specified date for each solution
If automatic time
    Add solutions for all available times for each solution
Else
    Add solution for specified time for each solution
i = 0
toReturn = new list
While i < max_number
    If no remaining solutions
        Break
    Select random solution and remove it
    If attendee clash for solution
        Continue
    Add solution to toReturn
    ++i
return toReturn
```

Batch scheduling

The algorithm begins by collecting the currently unscheduled meetings in the database. It then schedules each one as best as it can. After this, it selects a meeting randomly to undo the scheduling and then tries again to schedule the meetings. It compares the new result with the previous and keeps only the one with the best score. The score is calculated by taking the sum of the scores of each individual meeting multiplied by the importance of that meeting. The score of the individual meeting is equal to the weighted percentage of the preferences met in its current scheduling.

Each meeting is scheduled by first attempting to satisfy all three preferences, the location, room and time of day, and then allowing them to be dismissed in order of least importance. It does this in a way that effectively maximises the total priority of the satisfied preferences.

The batch scheduler runs through a number of iterations depending on the value of a set number. This number is currently a constant within the code and cannot be varied by the user.

```
BATCH SCHEDULE

Current = best = next = list of unscheduled meetings

For chosen number of iterations

    For each unscheduled meeting in current

        Schedule meeting

        If score of scheduled meeting > score of unscheduled

            Remove unscheduled meeting from next

            Add scheduled meeting to next

If score of next > score of best

    Best = next

If not last iteration

    Choose victim from next

    Set victim to booking failed

    Move victim to end of list

    Current = next


SCHEDULE MEETING


Order preferences by priority //strongest, middle, weakest


Schedule with all three preferences

  If success return meeting


Schedule for strongest two preferences

  If success return meeting
```

Schedule for strongest and weakest preferences

If success return meeting

Compare priorities strongest with weakest two

If strongest > weakest two

Schedule for strongest

If success return meeting

Schedule for weakest two

If success return meeting

Else

Schedule for weakest two

If success return meeting

Schedule for strongest

If success return meeting

End

Schedule for middle

If success return meeting

Schedule for weakest

If success return meeting

Schedule ignoring preferences

If success return meeting

Return original meeting unscheduled

6 EVALUATION

6.1 INTERFACE

The application has been tested throughout development, mostly through manual tests. All the features are functional. On opening the application, the user is presented with the login screen. This successfully prevents access to the rest of the system without a valid login. Upon login, the user's calendar is displayed. The user is likely to wish to visit this screen every session and it is therefore a sensible choice for the entry screen. Having logged in, it is now possible to use the navigation bar to visit the various other screens of the application. The navigation bar is limited for regular users compared to administrator users who have access to additional features. The other screens consist of forms and tables.

The forms in the system are consistent in layout and clear to understand. The data entered is validated and any issues are highlighted by red error texts. Changes made are effective immediately and following submission, the user is navigated to the relevant screen which shows the updated information. Some forms require an input of a location or room, both of which must be entered in the correct format. Suggested inputs are listed when the user begins typing. However, suggested rooms are not limited to those within a chosen location which could cause some confusion as the validation would prevent these rooms from being submitted alongside the chosen location. The user is informed by error text if such a choice is made. There is adequate room available in the description boxes for detailing a description of the meeting, room or location.

The table style pages are also consistent in layout and the familiarity will help the user. The invites page displays two different tables in a tab view, each with a similar style. The table headers are accurate descriptions and some of the tables contain an 'action' column with buttons for various tasks to be performed on a row of the table. This layout is clear, functional and easy to manoeuvre. Some of the tables are searchable which will help to find desired information when the database increases in size.

There are other more sophisticated applications available. Some are quite advanced as larger funds and more manpower have been invested in developing them. This system is simple but effective and user friendly. The development costs have been minimal.

6.2 ALGORITHMS

Automatic immediate scheduling

Automatic tests were carried out using a test case generator class to randomly populate a database with accounts, locations, rooms and meetings. The table below shows the average time taken to automatically schedule a two-and-a-half-hour meeting with five attendees. The average time was taken for one hundred attempts. The location, room, date and time were all set to be automated. The database was populated with thirty accounts and two locations, each containing ten rooms. The number of meetings inserted to the database was varied. Each meeting was assigned a random selection of attendees and was scheduled to take place in the next week.

Meetings	Average time
10	0.29 s
20	0.34 s
40	0.64 s
60	1.91 s
80	2.30 s
100	11.36 s

As can be seen from the results, the response time was fast. As the number of meetings already arranged increased the results took longer, as there were more meetings in the database to avoid. With 80 meetings already booked the response time was still perfectly acceptable. With one hundred meetings, the response time had reached over eleven seconds, which could be considered a ceiling due to the high frequency with which this algorithm will likely be run.

The algorithm is slowed considerably by the requirement to check for clashes involving the attendees. This is shown by data collected with this requirement removed. Under otherwise the same circumstances, with one hundred meetings in the database the average time was just 0.85 seconds, a vast improvement. A likely cause of this is that the check for attendee clashes currently carried out involves checking every meeting that may clash that each attendee is involved in, for each solution, before it is reported to the user. Although attempts have been made to reduce the number of iterations of these loops, this is still a greater number of nested loops than otherwise seen in the algorithm.

Batch scheduling

As with the immediate scheduler, automatic tests were carried out using a test case generator class to randomly populate a database with accounts, locations, rooms and meetings. The table below shows the average, shortest and longest time taken to automatically schedule varying number of meetings in a batch and the number of times the iterative improvement of selecting a victim meeting was effective. These improvements typically registered a 1 - 6 percent increase of the total score. The data was taken for one hundred attempts. The database was populated with thirty accounts and two locations, each containing ten rooms, and thirty already booked meetings. The number of meetings inserted to the database was varied. Each meeting, either booked or unscheduled, was assigned a random selection of attendees and was scheduled to take place in the next week. The unscheduled meetings were also given a random list of preferences and priorities. The batch algorithm was run for three iterations, meaning that two improvements were attempted.

Unscheduled Meetings	Fastest Time	Slowest Time	Average Time	Iterative Improvements
5	0.21 s	50.9 s	4.60 s	2
10	0.84 s	72.4 s	10.8 s	2
15	1.46 s	169.8 s	17.3 s	7
20	2.66 s	182.6 s	26.9 s	7
25	2.17 s	93.2 s	21.2 s	5
30	2.43 s	100.7 s	21.7 s	8

The data shows an overall trend that the average time increases with the number of meetings to be scheduled. This is as expected, as more meetings to schedule means more iterations of the algorithm and increases the chance of encountering a meeting that is difficult to schedule.

The latter point is especially relevant as can be seen by the difference between the slowest and fastest times throughout the table. This high range shows that a particular selection of meetings may take longer to schedule and the worst case is much higher than the average time.

The exception to the overall trend is when 20 meetings were scheduled. The average time taken was higher than the average time taken for 25 and 30 meetings. This may be explained in part due to the exceptionally high slowest time which would have increased the average and may be due to a particularly difficult selection of meetings. This random nature of testing is useful for identifying scenarios in which

the algorithm performs badly but can lead to surprising results. This was mitigated somewhat by running 100 tests but it is still likely that the algorithm would run quicker at lower number of meetings to schedule.

The times taken are higher than that of the immediate scheduler which is to be expected as more meetings are being scheduled. Additionally, a higher run time is acceptable due to the reduced frequency at which this algorithm will be run. However, the time taken is still fairly high as the table shows recorded data in excess of 3 minutes and it is possible that different set-ups could increase that time further. Therefore, it would be a positive addition to the project to include a loading screen whilst the algorithm is running to report progress to the user. This would give the user confirmation that progress is taking place and would provide an estimate of time remaining before completion.

It can also be seen from the table that iterative improvements that increase the score collected on the first iteration do indeed take place when running the algorithm. These improvements are generally quite small mostly ranging from 1-6%. However, these may make a difference to the organiser of the improved meeting, as it will make the difference between one of their preferences being met or not. It is likely that increasing the number of iterations would increase the amount of times an improvement is seen. This comes with a trade off as increasing the number of iterations would reduce the performance of the algorithm. Therefore, it may be an improvement to the system to provide the user the option of deciding on the number of iterations when running the application.

7 LEGAL, SOCIAL, ETHICAL AND PROFESSIONAL ISSUES

When creating this project, it was important to give consideration to a number of important issues. Programs should be secure, user friendly and run efficiently.

The system stores personal information about individuals in a database. Therefore, it is important that the database server used is secure. Whilst the only personal information held is the user's name, email address and calendars, it would still be a concern if this information was leaked to unauthorised persons. During development xampp web server was used with MariaDB as the database. Whilst this can be made fairly secure as a development environment, it is not appropriate for a production environment [5]. As such, it would be recommended to make use of a different environment for a production ready version.

To ensure that only authorised users are able to access personal information, each user has an account, opened in their name, enabling them to log in with their registered email and a password issued by the administrator. They are then able to change their password and should be encouraged to do so. Passwords should be updated regularly. Administrators should be carefully chosen and trustworthy in view of their ability to access and modify most of the data in the system. Passwords are stored in the database as their hashed versions with a salt and they must contain at least six characters. In order to improve the level of strength of passwords, users could be encouraged to use longer passwords and include more variation such as symbols. Whilst logged in, users are able to access only their own calendar and only administrators and the user themselves are able to access and modify the user's data. It is therefore important that screens are not left logged in unattended, to prevent unauthorised access.

This report contains user documentation giving information regarding the operation of the system, including workings of the algorithms. This would be of interest to users as it provides an insight into the capabilities and limitations of the system. The application can be used by all without discrimination [6].

8 CONCLUSIONS AND FUTURE WORK

This project was started with the intention of creating a meeting scheduling application. It had to provide users with an interface that enabled them to conveniently add, view, edit and delete data from the system. This would include accounts, locations, rooms within locations and the meetings themselves.

It was an important aspect of the task that the system should be capable of automatically scheduling meetings. It would do this by offering suggestions where the user did not specify a chosen value for a particular attribute. At the start of the project it was clear that there were already many existing meeting scheduling applications, some of which already provide the opportunity for some automation. However, this project also sought to introduce a batch scheduler to improve the efficiency of scheduling multiple meetings at once, in a way that reduces the likelihood of the chosen time and place of one meeting preventing the scheduling of another, possibly more important, meeting.

Meeting scheduling is a vital part of running an organisation be it large or small. It is important because meetings take place regularly and have to be organised. Thus, it is necessary that organisations have access to a method of performing this task smoothly and competently.

The introduction of a batch scheduler could transform the way meetings are currently scheduled. It enables the possibility of organising multiple meetings in a way that makes them compatible with each other. This could lead to less meetings being cancelled or rescheduled and would save organisers time by reducing their workload.

With the completion of this project it can be seen that it is possible to implement an efficient system that meets these requirements. Using such a system would provide immediate benefits in costs, time saving and efficiency to an organisation.

As with any system there are always improvements to be sought. Whilst the algorithms work well, improving the speed would be a big step forward. This would make the software more suitable for larger scale use.

Another consideration that could be introduced would be email notifications. Emails could be sent to organisers to inform them of changes to the state of their meeting, such as the result of an attempted batch schedule, or a response to an invite. Emails could also be sent to attendees, including those who do not have an account, to invite them to the meeting when it is scheduled and to notify those with accounts that they have a new invitation to respond to. The introduction of email functionality would also be useful for resetting a password if forgotten, which currently would require an administrator's assistance. The ability for administrators to change other user's passwords would then no longer be necessary and could then be removed. As all emails are stored in the database, implementing these improvements would be straightforward, through use of a library such as javax mail.

Another possible improvement would be for the batch scheduler to be run automatically at a set interval rather than requiring an administrator to manually start the process. There are attributes in the database for the interval and next run time for this reason. However, the results of the algorithm would still require approval from a user to avoid over reliance on the algorithm. Therefore, such a change may not be worthwhile.

There are other attributes in the database not currently in use. The cost attribute could be used in the future to account for restricted budgets. The open space attribute could be used for meetings that have no limit on the number of attendees such as those that take place in an open area. It may also be helpful in modelling online meetings which are becoming more popular.

Another improvement would be the ability to view meetings in a list format. The only way currently possible to view a meeting is via a calendar, either the user's personal calendar or a room calendar. As both calendars only display booked meetings, it is not currently possible to check the status of meetings created for the batch scheduler, unless the user is an administrator and can view the batch screen. This also means that some users will not be able to delete their created meetings for the batch scheduler. A meeting list would also be useful for administrators to be able to find meetings they are not involved in as well as to delete meetings if required.

BIBLIOGRAPHY

[1] https://www.g2.com/products/microsoft-outlook/reviews

[2] https://www.g2.com/products/google-calendar/reviews

[3] https://www.g2.com/products/calendly/reviews

[4] https://www.g2.com/products/x-ai/reviews

[5] https://www.apachefriends.org/faq_windows.html

[6] https://www.bcs.org/membership/become-a-member/bcs-code-of-conduct/