

A Front-End Web Application for Automatic Meeting Scheduling

Iman Kalyan Majumdar

s1132294



4th Year Project Report

Computer Science

School of Informatics

University of Edinburgh

2016

Abstract

This is the project report for developing a front-end web application that automates the process of scheduling by integrating itself with an existing automated meeting scheduling algorithm. This was integrated through the use of a RESTful API. The algorithm and API used were designed by previous students of the University of Edinburgh as their undergraduate thesis. The application developed allows user's to schedule new meetings, set their preferences used by the automated algorithm to suggest a meeting time, respond to a proposed meeting time and monitor a different stages of the meeting during the process of scheduling a meeting through an user-friendly interface.

Acknowledgements

I would like to thank a number of people without whom this would have not been possible.

First and foremost I want to thank my supervisor, Dr. Michael Rovatsos, without whose proposal I would not have had the opportunity to take up such an interesting and challenging project. Furthermore, without his guidance on how to begin solving the problem at hand this project would have proved to be too difficult.

Secondly, I want to thank Zhenyu Wen without whose help I would have not been able to integrate the application successfully.

Thirdly, I would like to thank all the participants that took the time to participate in the experiments providing invaluable feedback, without which the project would not be successful.

Lastly, I want to thank my family and friends for their never ending support during the entire process.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

*(Iman Kalyan Majumdar
s1132294)*

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Literature Review	2
1.3	A Study of Doodle	4
1.4	Hypothesis	5
1.5	Objectives	6
1.6	Outline	6
2	System Design	9
2.1	Dependency Projects	9
2.2	System Architecture	10
2.2.1	Client Side	10
2.2.2	Server Side	11
2.2.3	Web Framework	11
2.2.4	Data Store	13
2.2.5	Model-View-Controller	14
2.3	Workflows	16
2.3.1	Registering and Connecting to Google Calendar	17
2.3.2	Creating a New Meeting	18
2.3.3	Responding to a Meeting Invitation	19
2.4	Database Design	21
2.4.1	User Collection	21
2.4.2	Organiser Collection	23
3	The Preference Model	25
3.1	Concept	25
3.2	Turning Responses into Numbers	27

4	Integration	29
4.1	Google Calendar Integration	29
4.2	Back-End Integration	31
4.2.1	Register New User	31
4.2.2	POST New Meeting	32
4.2.3	Send Preferences	32
4.2.4	GET Proposed Meeting Time	35
5	User Experience Design and Implementation	37
5.1	Concept User Experience Framework	37
5.1.1	Interaction Design (ID)	37
5.1.2	Controlled Feedback (CF)	38
5.1.3	Device Independent (DI)	39
5.2	Design of Web Pages	39
5.2.1	Registration and Login	39
5.2.2	Organiser and Participant	42
5.2.3	Settings	48
6	Evaluation	51
6.1	Methodology Used	51
6.2	Experiments	52
6.2.1	Scenario	52
6.2.2	Experiment 1 - Comparison Study	52
6.2.3	Experiment 2 - Independent Study	55
6.3	Summary	58
6.3.1	POD7	59
6.3.2	POD9	60
7	Conclusion	63
7.1	Summary	63
7.2	Major Limitations	64
7.3	Further Work	64
7.4	Lesson Learnt	65

Chapter 1

Introduction

1.1 Motivation

Meeting scheduling is a task that is performed by humans very often, both formally and informally, to gather people to discuss issues at hand. In the field of work, this is the primary method used to resolve issues in a collaborative environment. However, this task of scheduling can become more and more tedious as the number of participants grow. There are several published and on-going research attempting to develop a solution to this problem by automating as much of the process as possible. These solutions range from the use of experimental multi-agent systems, negotiation protocols and more. However, even though these systems are able to produce satisfactory results but lack an user-friendly front-end application that is able to incorporate it's methods into a system people can use today. On the other hand, the popular meeting scheduling software Doodle provides the public with an application that allows them to propose meetings with multiple participants by simplifying the process of manually scheduling meetings from an user interface stand point.

This brings to light a gap present in the area of meeting scheduling between the development of automated scheduling algorithms and front-end meeting scheduling tools. The succeeding sections will provide a more in depth analysis of existing literature and describe a study of Doodle's interface before moving on to describe how this project aims to bridge this gap.

1.2 Literature Review

This section highlights some of the approaches taken towards automatic meeting scheduling. The focus of this review is to give an overview of the different ways in which the problem of automatically scheduling meetings between multiple agents can be approached. The review also explicitly mentions the aspects of each approach that were difficult to understand given my knowledge in the area and the complexity of the approach itself.

One of the approaches taken to solving the meeting scheduling problem (MSP) in the presence of multiple users is presented by A. Grubshtein et al. (2010). They propose a model that computes a meeting time based on maximizing an utility function that accounts for a set of user preferences. The preferences are integers that represent the priority class (PC) and agent-meeting urgency (AMU). PC defines how important it is for a particular agent to attend the meeting. AMU defines how much importance the agent gives to the meeting. These preferences are combined with two other factors that further constraints the algorithm from computing a time outside the period within which it has to be scheduled. One of the methods taken to evaluate this approach was done by monitoring the ‘Effective Optimal Utility’ (EQU), the utility reached after the algorithm arrives at a solution for a meeting time, as the number of meetings for a given day increase from 1 to 10. The change in the utility as the the number of meetings increases is regarded as the cost of adding another meeting. It was difficult to understand how the external local search algorithm used went about reaching a solution that maximizes the EQU because little detail was given in the paper. However, it was stated that this model was able to compute a good result for scenarios consisting of several agents. [15]

Apart from A. Grubshtein et al., Elizabet Crawford et al. (2004 - 2005) has done a lot of work in this area. They have published three separate papers discussing four further approaches to solving different aspects of the MSP. One paper attempts to learn negotiation strategies suitable for each agent participating in a meeting. The chosen strategy is aimed to maximize an agent’s utility, based on a concept similar to the one discussed previously in A. Grubshtein et al. (2010). The propose that by being able to learn a negotiation strategy for each of the

agents they are able to choose the best possible strategy depending upon the scenario and the agents in concern. This concept is based of how a robot that is programmed to play football chooses a particular strategy based on factors such as who is in possession of the ball and how close the robot is to the goal. In the case of meeting scheduling, possession is synonymous to which agent it is being negotiated with and the distance to the goal is synonymous to how close the negotiation is to completion. The model proposed in the paper is able to learn which strategy to choose by updating a set of weights for each agent that determine the ideal strategy to be chosen. The approach taken towards how the weights are updated is quite complex and could not be completely understood. However, the idea is that the weights are updated on each round of negotiation, thereby being able to choose different strategies over all the rounds of negotiation that take place. This approach showed promise as the authors were able to show that by choosing a particular negotiation strategy for each agent depending on their preferences produces better results than the case where the same negotiation strategy is used for all agent's. Specifically, it was seen that the agent's preferences play a part in which strategy the algorithm converges to after the initial rounds of negotiation, where most of the learning takes place. [5]

The other paper discusses three other approaches, Meeting-Oriented Approach (MOA), Schedule-Oriented Approach (SOA), and Calendar-Oriented Approach (COA) to solving this MSP and their limitations. MOA gives each agent a number of points with which they bid for their preferred timeslot proposed for a meeting. A socially accepted choice is calculated depending on the amount of points bid for a timeslot but this mechanism doesn't account for combinatorial preferences. SOA allows agents to bid for timeslots that comprise of all possible combinations of timeslots and meetings. This approach may lead to so many options that the problem becomes intractable. COA allows agents to express their preferences over the period within which the meeting is to be scheduled. [6]

Finally, we will look at the algorithm used by the application developed for this project and discuss how it is different or similar to the above approaches. The algorithm for this project was developed by Marcinowski (2014), a graduate computer science student from the University of Edinburgh, for his undergraduate thesis. The approach taken uses similar ideas mentioned by the COA, such as

using an agent's preferences and solving a constraint satisfaction problem (CSP) through the use of an external Clojure library. The hard constraints that have to be satisfied include computing a time within the given period during which it has to be scheduled and the soft constraints are the preferences for each timeslot that must be accounted for. Preferences are set for a slot representing a range of free time with numerical rating. The numerical rating can be an integer from '1' to '5', where '1' denotes that the slot is the least preferred and '5' denotes that the slot is most preferred. [11]

Overall, all these approaches provide satisfactory results but are focused on the development of an algorithm used to solve the MSP. Additionally, they use low-level preference models where the agents are required to enter numeric values instead of more high-level model where the agent is able to specify preferences such as 'I like Mondays', 'I like Tuesday afternoons', and 'I don't like to have meetings on Wednesdays'.

1.3 A Study of Doodle

Doodle is a meeting scheduling tool that allows people to create public and private meeting polls with multiple participants while keeping the process of scheduling itself to be manual. It focuses on providing an user friendly experience that simplifies the procedure to receive responses from participants. It does this by creating a poll with a series of suggested timeslots manually chosen by the user from which the participants can pick their preferred one. Once, all the participants have responded the user can then send out an additional email to all participants notifying them of the most popular timeslot, stating that is the decided time when the meeting is to take place.

A small preliminary experiment was conducted on 5 subjects in an attempt to identify any issues with Doodle's interface. The subjects were asked to create a meeting poll to schedule a meeting within a given work week (Monday to Friday). Their behaviour with the interface was observed during each step of the four step process described below.

- Step 1 - Fill in form with Title and their email address.

- Step 2 - Select dates within the work week.
- Step 3 - Enter slots from which the participants can pick.
- Step 4 - Invite participants.

In addition to the observations made, a small set of questions were asked after the completion of the entire task. It was observed that step 1 and 4 were straightforward and clear. However, the subjects mentioned how cognitively they would think about inviting participants before completing steps 2 and 3.

Furthermore, it was observed that completing steps 2 and 3 required more time than the other steps. Some of the factors that cause this are unavoidable in a non-automated meeting scheduling tool because of the difficulty faced by subjects when manually picking the days and timeslots within those days. However, there were also some delays caused by the interface. One of them being that subjects who had not used Doodle before did not understand how to specify the duration of the meeting and instead entered timeslots with only a start time. Since, Doodle's interface does not require you to enter an end time, the poll was still created.

Furthermore, they judged the calendar interface used to be useful and that it may benefit from using the space to the right of the calendar (see below) to enter the slots rather than have an additional step.

Overall, it can be concluded that there are rooms for improvement in steps 2 and 3. But in general the interface was deemed to be simple and appropriate to complete the task at hand. In addition to the study, I thought it was important to note that when I tested the application for handling of erroneous input, it was seen that a user is able to enter incorrect time slots such as 1:30 PM to 1:00PM. The interface does not handle such cases by notifying the user that they have entered an incorrect slot.

1.4 Hypothesis

This project hypothesizes that by using the interface of Doodle as a base to develop a front-end web application that adopts an automated meeting scheduling

algorithm to schedule meetings, will provide an example of an approach that can be used to develop industrial standard front-end meeting scheduling applications which automate the process of meeting scheduling for real world users. Furthermore, it is expected that the findings of this project will highlight the areas of concern when taking such an approach.

1.5 Objectives

A set of objectives set out at the beginning of the project are mentioned below.

1. Create a user management system that stores an user's preferences and credentials required to access their Google Calendar data in addition to managing the session of the logged in user.
2. Create an interface allowing the user to set their preferences.
3. Integrate and format the Google Calendar data retrieved using the Google Calendar API into the format required by the RESTful API running in the back end.
4. Integrate the front-end with the RESTful API developed by Marijonas (2015) [12] to send and receive information required to compute an automatically proposed time.
5. Develop an easy to use interface for the user to schedule new meetings by creating a mechanism to add participants easily and to choose an interval within which the meeting is to be scheduled.
6. Create an interface that will allow the user to accept or reject the automatically proposed time based on their preferences.

1.6 Outline

This report begins by stating the overall system design and technologies used to implement this web application in Chapter 2. This is followed by the description of a high-level preference model that is able to convert human responses onto

a numeric scale in Chapter 3. The model is then used to compute the preferences for a range of timeslots which are then sent to the back-end to compute a proposed time based on user preferences. This integration with the back-end is discussed in Chapter 4 along with how the application is able to retrieve data from the user's Google calendar. Then Chapter 5 goes on to introduce an User Experience framework that influences the design of each individual web page. The completion of the work in all these chapters leads to a prototype of a front-end web application for automated meeting scheduling. The prototype is then evaluated with a focus on it's usability through the use of contextual interviews. The process of evaluation and the key points of discussion that arose are discussed in Chapter 6. Finally, to conclude, Chapter 7 summarizes any findings of the report, delivers a judgment on the quality of the completed application, and discusses how such an application shows promising signs of an approach that can be used to develop more advanced applications of the same nature.

Chapter 2

System Design

This chapter discusses the system design decisions taken during the development of the project along with the solution to any conceptual problems that arose in the process.

2.1 Dependency Projects

The following servers are required to be running simultaneously for the system to work.

Front-End Flask Application	Port 80
Back-End Meeting-Scheduling (MS)	Port 3000
Back-End Peer-Manager (MS)	Port 3002

These components communicate with each other to implement all the functions provided by the system. A high level view of the communication between all the different servers in the system are shown below.

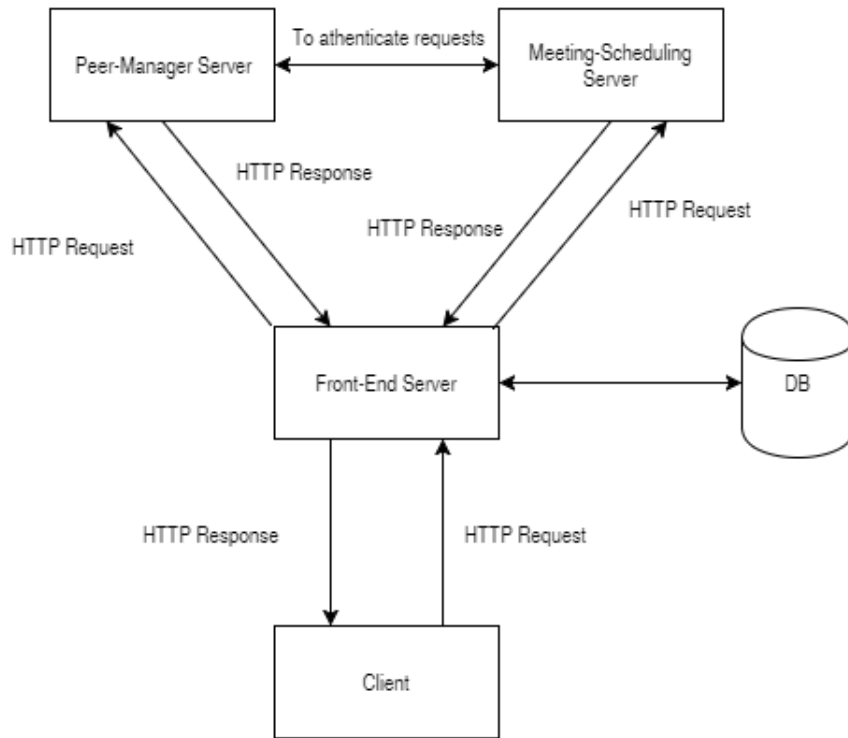


Figure 2.1: View of the interactions between the different servers and the client

2.2 System Architecture

In this web application, the browser of choice for the end user will act as the client and the web server will handle all requests made by the client through HTTP requests.

2.2.1 Client Side

The client side holds all the code responsible for displaying and interacting with the web page. The code is comprised of HTML, JavaScript and CSS. HTML is a markup language used to display web pages in all existing browsers.

The modern look and interactivity provided by each web page is created using CSS and JavaScript respectively. CSS is a stylesheet language that handles the presentation of a HTML web page and JavaScript is a programming language used primarily in web applications to provide dynamic functionality to HTML web pages.

The Bootstrap framework was used to develop modern, minimalistic, and responsive web pages. Using this popular HTML, CSS and JavaScript framework saved a lot of time that would have been spent in the styling of the web pages. Additionally, it also allowed the creation of responsive web pages that are able to scale appropriately depending upon the screen size of the device in use. [2]

2.2.2 Server Side

The server side holds all the code responsible for handling any HTTP request made from a client and storing any persistent data used by the application in a database of choice. The Web Framework and database chosen to develop this application were primarily driven by its compatibility with the RESTful API designed by Marijonas (2015) and my knowledge of the given language/area. Thus, increasing the chances of a successful integration of the front end web application with the existing back-end.

2.2.3 Web Framework

There are a number of existing web frameworks that exist today, such as Ruby on Rails, Django, and AngularJS to name a few. However, because the RESTful API was utilized and tested by Marijonas using Python, it was decided to look at only Python based web frameworks, allowing the application to use his python script used for testing as a template for utilizing the API. This choice also complemented the fact that I had prior experience using Python. Given these conditions, the search was narrowed down to the three most popular Python based web frameworks Django, Pyramid, and Flask. Some of the pro's and con's of each framework are described in table 2.1 on the next page.

Web Framework	Pro's	Con's
Django	<ul style="list-style-type: none"> • Includes varying modules which the developer can use to implement standard features. • Separates each aspect of the application into sub applications increasing the modularity while lowering coupling. • Large amount of resources with good documentation. 	<ul style="list-style-type: none"> • Added complexity because it is aimed towards bigger applications.
Pyramid	<ul style="list-style-type: none"> • Very flexible. The developer can choose the database, templating engine and method handling the routing within the application as they desire. • Connect to different types of databases within the same application. • Flexible bootstrapping tools available to quickly build large applications. 	<ul style="list-style-type: none"> • Little documentation.
Flask	<ul style="list-style-type: none"> • Aimed for small applications performing a specific task. • Allows the developer to hand pick each component of the application and only include what is necessary. 	<ul style="list-style-type: none"> • Debugging is difficult as it has to be done by explicitly looking at the error log produced by the server. • Little documentation.

Table 2.1: Pro's and Con's of popular Python Web Frameworks [3]

Django and Pyramid are primarily used for large scale web applications which perform several kinds of functions. These frameworks are suitable for developing a social network such as Facebook or an all in one email and calendar application such as Outlook. However, to develop a specific application such as the one in this project is more suitable to use a smaller framework like Flask. Additionally, due to the recent development of the Flask framework, there now exists a module, flask-ext-login, which handles the user login system and sessions like in Django. Thus, Flask was chosen as the framework to be used to develop the application.

2.2.4 Data Store

MongoDB was used as the main data store. MongoDB is a NoSQL database that stores data in documents structured as a JSON. Several documents make up a collection. In MongoDB a ‘collection’ is synonymous to a table in a RDMS ¹ and a document is synonymous to a table row in a RDMS. The key factors influencing this decision are outlined below. [8]

- **Avoiding Joins** - Joins are an expensive operation used in traditional RDMS to connect data between multiple tables. MongoDB allows documents to be embedded within another document instead of having to join tables to get the desired data.
- **Frequent Changes** - MongoDB allows the developer to frequently make changes to the structure of documents.
- **Compatibility** - The JSON data format is used to communicate between the front-end and back-end. Since, MongoDB uses the same format to store documents in a collection, the data exchanged between the servers can easily be inserted into the database with minimal formatting.
- **For the future** - As the data store used in the front and back-end are the same, in the future this data store can be combined removing the need for two separate data stores with reduced difficulty.

¹RDMS - “Short for relational database management system and pronounced as separate letters, a type of database management system (DBMS) that stores data in the form of related tables.” [1]

2.2.5 Model-View-Controller

The Model-View-Controller architecture is the most commonly used architecture pattern employed in the development of graphical front-end web applications. Therefore, such an architecture suited the development of rich user experiences aimed to be provided by this application. As the name suggests, the architecture is made up of three components which are described below.

- **Model** – The model is responsible for handling the data of the application. This includes inserting, updating and removing data depending on the instructions sent to it from the controller.
- **View** – The view is responsible for rendering the different screens that will be seen by the user when using the application. The user will then be able to interact with the view through the controller.
- **Controller** – The controller is responsible for ‘controlling’ the interactions between the View and the Model. Therefore the controller receives the input from the view, performs some actions on them such as validation and then updates the data in the model depending on the actions triggered by the user through the View.

The use of this architecture ensures that the application maintains low coupling between the different parts of the application while having high cohesion. Though Flask does not support this architecture out of the box, it was implemented using their ‘blueprint’ module.

Figure 2.2, on the next page, visualizes how all these components combine to represent the complete system architecture used to develop this application.

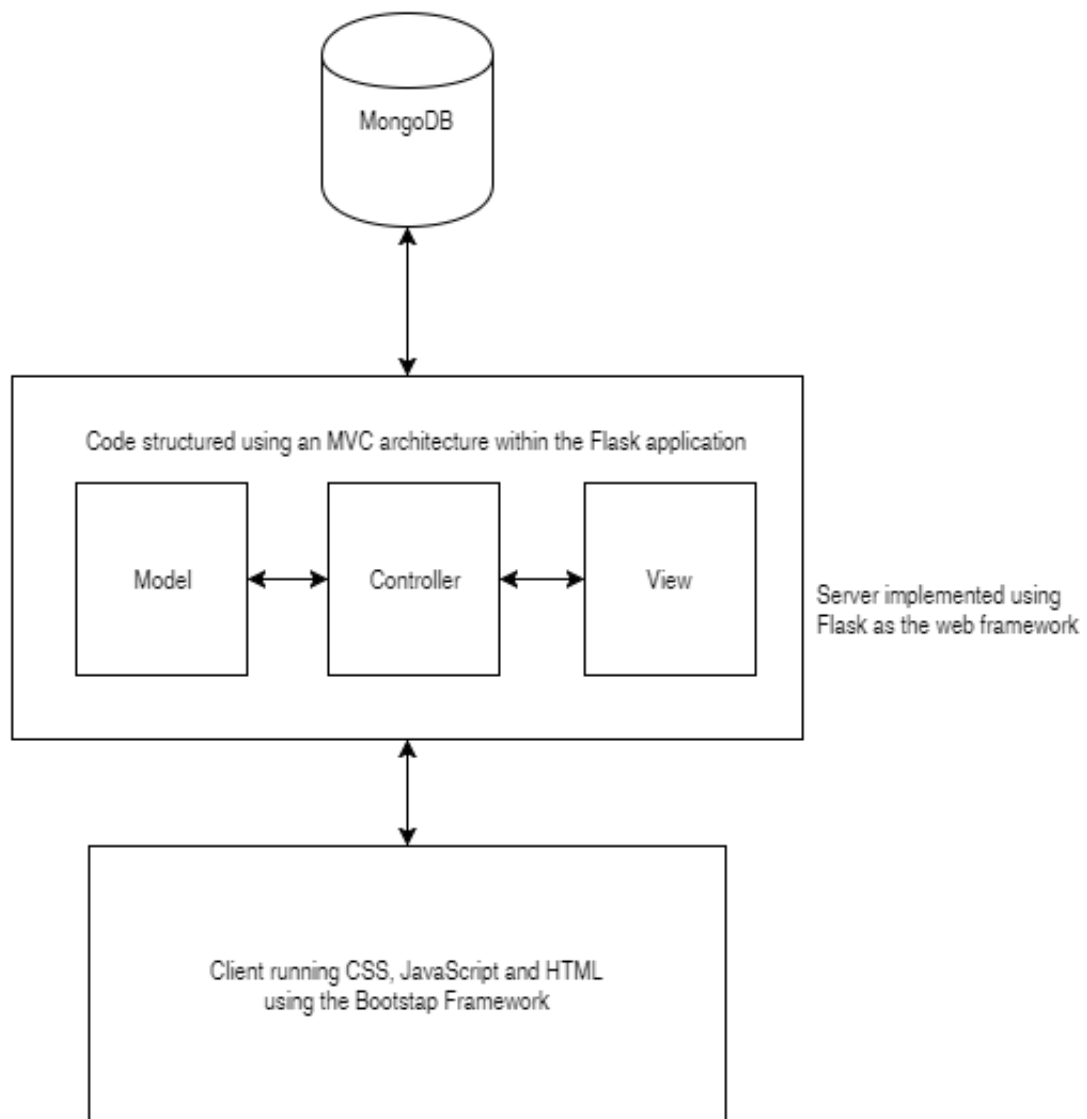


Figure 2.2: View of the full system architecture using the different technologies

The next section highlights some of the different ways in which the end user can interact with the application built on this architecture through the use of workflows.

2.3 Workflows

Workflows describe how an action initiated by the end user will be completed by the system to result in the end goal. These workflows were used as a aid to design the application to have a logical flow of information from the client to the web server and back. Additionally, it was important to signify the different stages a meeting, being scheduled, can take over it's lifecycle when using an automated approach. To do this a list of states were used to signal the system when to initiate the processes of the next state and to notify the user of the state. These possible states that a meeting can take and their transitions between them are described in table 2.2 below.

State	Description	Criteria for Completion	Next State
Pending	Waiting for the participants to send their preferences.	ALL participants have sent their preferences.	Proposed
Proposed	A meeting time has been computed based on the preferences of each participant.	ALL participants have either Accepted or Rejected the proposed time.	Final/Cancelled
Final	ALL participants have accepted the proposed meeting time.	-	-
Cancelled	A participant has rejected the proposed meeting time.	-	-

Table 2.2: Different States of a Meeting

It was realized to account for the transition between Pending to Proposed and Proposed to Final/Cancelled, it was needed to store the different states of each participant invited during the scheduling process. Table 2.3 summarizes all these states and the requirements which have to satisfied to transition between them.

State	Description	Criteria for Completion	Next State
Invited	The participant has been invited to the meeting.	The participant has sent his preferences for the period within which the meeting is to be scheduled.	Pending
Pending	The participant is waiting for a meeting time to be proposed	The participant has Accepted or Rejected the proposed meeting time	Accepted/Rejected
Accepted	The participant has Accepted the proposed meeting time.	-	-
Rejected	The participant has Rejected the proposed meeting time.	-	-

Table 2.3: Different States of a Participant

The initial state is ‘Pending’ for a meeting and ‘Invited’ for a participant. The final states of a meeting are ‘Final’ or ‘Cancelled’. These final states cannot be reached until all the invited participants reach a final state of ‘Accepted’ or ‘Rejected’. The workflows in the following sections describe when these states are caused to change during the scheduling process.

2.3.1 Registering and Connecting to Google Calendar

To register with the service, an user is required to enter a set of details that are used as their credentials to login to the system in the future as well as register themselves with the peer-manager. It is important to complete the registration with the peer-manager successfully because it is responsible for authenticating the user when they system attempts to initiate a function specified by the RESTful API. Therefore, the system will first attempt to register the user with the peer-manager, and only if that succeeds then proceed to storing their details on the front-end.

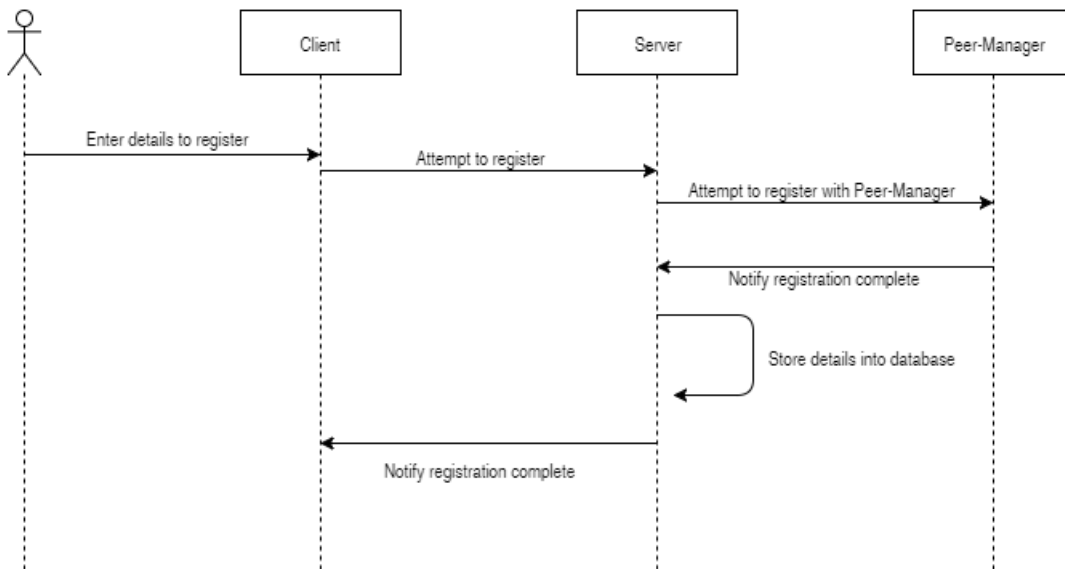


Figure 2.3: Registration workflow

The set of details to be entered by the user is kept to a minimum so as to make it easy to register with the service. These details were as follows.

- First Name, Last Name, Email ID, and Password

Upon entering a valid set of details, the user will be prompted to login to their Google Account to establish a connection with their Google Calendar. This will generate a time sensitive token that has to be used in any future calls made to receive data from their personal calendars. This token is stored in the database so as to reduce the number of times the user is prompted to login to his Google Account. Once it has expired, the user has to reconnect with their Google Account to receive a new token. The details of how this connection is established are discussed in section 4.1 of Chapter 4.

2.3.2 Creating a New Meeting

After an user has successfully registered with the service, they will be able to schedule meetings with a number of participants. Similar to the registration process, the user will be prompted to enter the following details to create a new meeting.

- Title

- A List of Participants
- Duration of the meeting
- An earliest date before which the meeting cannot be scheduled
- A latest date after which the meeting cannot be scheduled

After entering these details the client will make a request to the MS server to create the meeting on the back-end. After this is complete, the front-end will prompt the participants for their availability by setting the state of the participants to ‘Invited’ and the state of the meeting to ‘Pending’.

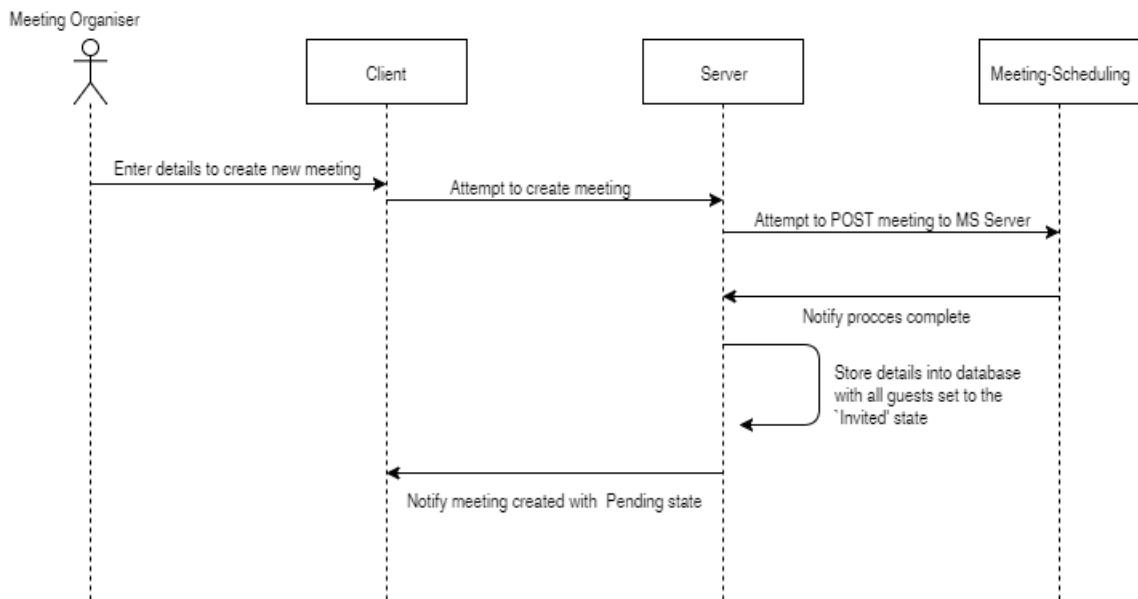


Figure 2.4: Workflow of organiser when creating a new meeting

This meeting will then become available to all the participants, who can then respond with by the process discussed in the next section.

2.3.3 Responding to a Meeting Invitation

Participants are able to respond to a meeting invitation in a two step process. Initially, they are shown the earliest and latest date within which the meeting is to be scheduled and prompted to send their availability based on their preferences to the MS server. After all the participants have sent their availability the back-end will compute a proposed meeting time which the participants can then accept

or reject.

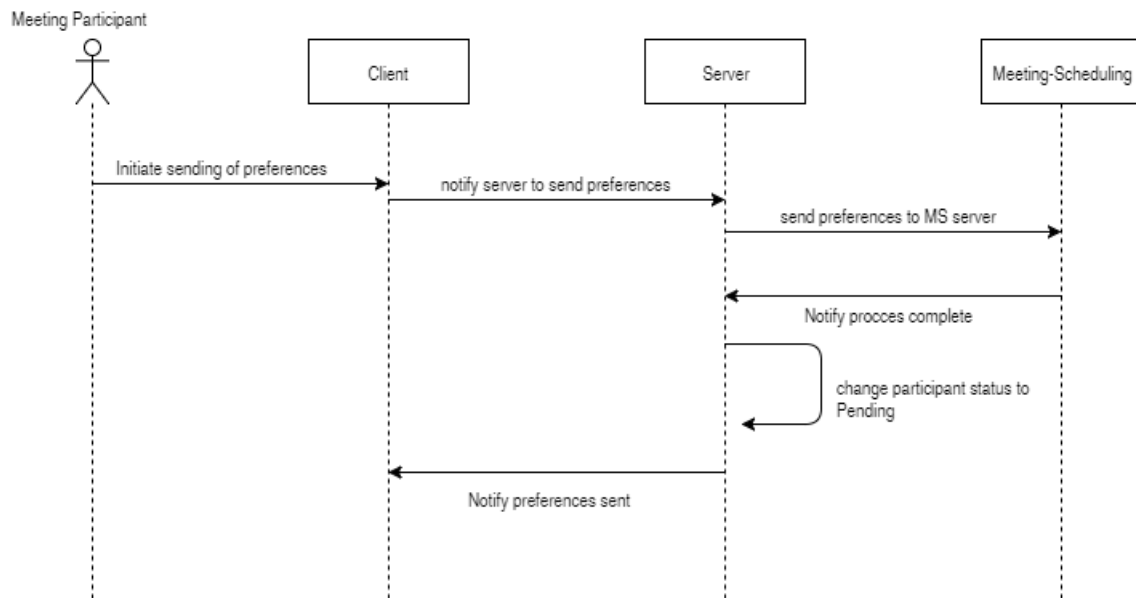


Figure 2.5: Workflow of sending preferences

Once all the participants have sent their availability and a proposed meeting time has been computed by the MS server, the status of the meeting is changed to 'Proposed'. At this point a participant can accept or reject the proposal.

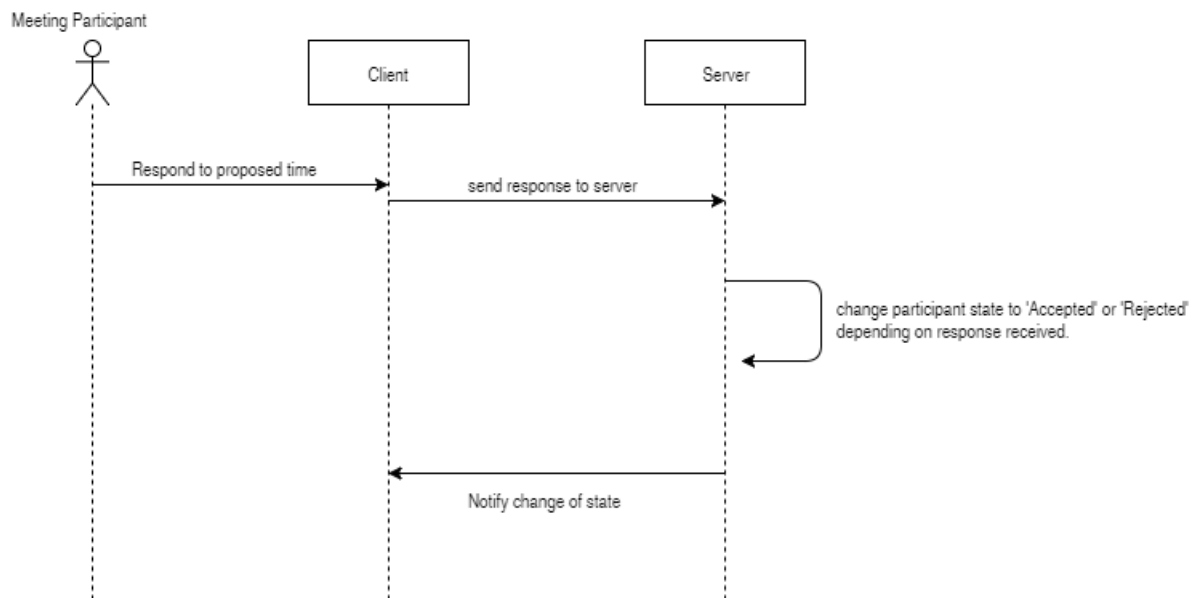


Figure 2.6: Workflow of responding to proposed meeting

The state of the participant is changed according to their response. (Refer to

Table 2.3) After all the participants have either accepted or rejected this proposed meeting time, the state of the meeting is changed to either ‘Final’ or ‘Cancelled’ depending upon the responses. (Refer to Table 2.2)

2.4 Database Design

The database design went through two key changes during the development process. These changes are highlighted in this section along with the motivation behind them to produce the final structure of the documents.

Initially, the database consisted of four types of collections, User, Organiser, Participant, and Preferences. This was reduced to only the User and Organiser collections.

2.4.1 User Collection

The User collection had the following initial structure based on the details entered during the registration process described in section 2.2.1.

```
{
    _id: ObjectID(...),
    createdAt: DateTimeField(required=True),
    firstName: StringField(required=True),
    lastName: StringField(required=True),
    emailID: StringField(required=True, unique=True),
    password: BinaryField(),
    salt: StringField(max_length=255),
    calcookie: StringField()
}
```

Listing 2.1: User Collection Structure

The ObjectID for each document in the collection is generated automatically by ongoDB during the insertion process. Most of the other fields are self explanatory as to what they contain except for ‘calcookie’. ‘calcookie’ contains the token

generated by Google's OAuth2Client that allows the system to access the user's Google Calendar. Additionally, for security purposes the password is encrypted using the PBKDF2 algorithm. The PBKDF2 is a key derivation function that uses a randomly generated salt to hash the unencrypted password entered by the user. This same salt is then used upon login to hash the password entered during login and compare it against the encrypted password stored in the database to check for a match. If they are the same, then the user is logged in, otherwise they are notified that the incorrect details were entered.

Due to my lack of experience using NoSQL databases, it was initially planned to create another Preferences collection which would hold the preferences set by each user. This idea was guided by my experience in using traditional RDMS where it is not possible to store a table within another table. However, MongoDB allows one to be able to store embedded documents within a field of a document. This allowed the number of collections to be reduced by embedding the Preferences collection within the User collection. To do this, a new field 'preferences' was added to the user collection which holds the data in the structure of the original Preferences collection. The revised User collection is shown below.

```
{
    _id: ObjectID(...),
    createdAt: DateTimeField(),
    firstName: StringField(),
    lastName: StringField(),
    emailID: StringField(),
    password: BinaryField(),
    salt: StringField(),
    calcookie: StringField(),
    preferences: {
        w_day: IntField(),
        day: ListField(),
        w_tod: IntField(),
        tod: ListField(),
    }
}
```

Listing 2.2: Revised User Collection Structure

The significance of the fields in the embedded ‘preferences’ document is described in detail in Chapter 3.

2.4.2 Organiser Collection

The design of the Organiser collection was based on the details entered during the workflow described in section 2.2.2. At first, a Participant collection was to be used to store the state of a participant for a particular meeting. The state field was used to determine the state of the meeting during the scheduling process as described in section 2.2. However, by adding the state field in the guests field of the Organiser collection there was no longer a need for the Participant collection. The guest field held the email id of the invited guests along with their state. The final structure of the Organiser collection is shown in Listing 2.3 on the next page.

```

{
    _id: ObjectID(...),
    createdAt: DateTimeField(),
    organiser: StringField(),
    title: StringField(),
    description: StringField(),
    url: StringField(),
    duration: IntField(),
    earliestDate: DateTimeField(),
    latestDate: DateTimeField(),
    suggestedTimeslots: [
        {
            startTime: DateTimeField(),
            preference: IntField(),
            busyGuests: ListField()
        }
    ],
    proposedTimeslot: {
        startTime: DateTimeField(),
        preference: IntField()
    },
    finalTimeslot: DateTimeField(),
    guests: [
        {
            email_id: StringField(),
            statusGuest: StringField()
        }
    ],
    statusMeeting: StringField()
}

```

Listing 2.3: Organiser Collection Structure

Thus, by adopting MongoDB’s ability to embed documents within another document the application is able to store all the information required to schedule a meeting using the application in just two collections.

Chapter 3

The Preference Model

This chapter introduces a high-level preference model that takes an intuitive approach to converting the responses to a set of questions onto a numeric scale that is used to set the preferences for an user.

3.1 Concept

Initially, it was thought that the user would enter a preference for each individual time slot. But then it was quickly realized that such a method would take a lot of time and be very tedious, defeating the purpose of the application. Additionally, it was also not intuitive for a user to rate each free time slot with a preference ranging from ‘1’ to ‘5’ as required by the algorithm used in the back-end. Where ‘1’ represents that it is least preferred and ‘5’ represents that it is most preferred. Cognitively, this would mean that the user has to think about not only about whether he/she prefers the time slot but how much he/she prefers it. Thus, to reduce the cognitive load on the user, it was decided to use a more intuitive model. This model was designed to keep the use of numbers away from the user and perform computations server side based on the answer to a set of questions. The kind of questions that could be used are described below.

- Which days do you prefer to schedule meetings on?
- Do you prefer to keep a particular day free of meetings? If so which day?
- What time of day do you prefer?

- Do you like gaps between your meetings?

The questions mentioned above are more intuitive for a human to answer. It also allows them to save time by only having to answer a few questions while letting the system compute their numeric preference based on their responses. During implementation, the questions were formalized to the following.

- Q1: Which days do you prefer to schedule your meetings on?
- Q2: Which days do you NOT prefer to schedule your meetings on?
- Possible Responses to Q1 & Q2: Monday - Sunday
- Q3: What time of day do you prefer to schedule your meetings at?
- Q4: What time of day do you NOT prefer to schedule your meetings at?
- Possible Responses to Q3 & Q4: Morning, Afternoon, and Evening

Divisions of the Time of day

The 12 hours scheduling day was divided into three parts. Where each part represented an interval of 4 hrs.

8 AM ————— 12 hrs ————— 8 PM		
8 AM — 4 hrs — 12 PM	12 PM — 4 hrs — 4 PM	4 PM — 4 hrs — 8 PM
Morning	Afternoon	Evening

Table 3.1: Division of ToD preference

As you can see from the table above that this division corresponded to three natural times of a day, Morning, Afternoon, and Evening. It was considered breaking this down further into 2 hr intervals representing ‘Early Morning’, ‘Before Lunch’, ‘After Lunch’, ‘Afternoon’, ‘Evening’, and ‘Late Evening’ to produce better results. But at this point a large portion of the development had already taken place and due to time constraints this was not implemented. Thus, the original four hour intervals were kept.

3.2 Turning Responses into Numbers

This section outlines the mathematical function used to convert the responses of a user to the questions stated in the previous section on to a numerical scale. This was done in a simple and logical way.

- For the first question, the selected days were given a preference of ‘5’ and the rest were given a preference of ‘2.5’. This data was stored in an array named day_{pref} .
- For the second question, the selected days were modified in the day_{pref} array to have a preference of ‘1’.
- Similarly, the selected time of day for the third question (ToD) was given a preference of ‘5’ and the selected time of day for the fourth question was given a preference of ‘1’. The others were given a preference value of ‘2.5’. This data was stored in another array tod_{pref} .

The weights given to each of the parameters day_{pref} , tod_{pref} , and $prox$ was controlled by the priority given to them by the user. The user is able to assign one of three priority levels high, medium and low to each parameter. Each priority level corresponded to a weight that is shown below.

High	0.66
Medium	0.33
Low	0.1

The total weight to be distributed among the parameters has to equal 1, to ensure that the preference computed for a timeslot is in the required 0 - 5 range. In the case, where all the parameters are given the same weight, the weight is distributed equally among all the parameters.

Finally, to get the final preference value for each range of free time slots, it was put through the following function.

$$pref_{timeslot} = w_{day} * pref_{day} + w_{tod} * pref_{tod};$$

where w_x is the weight given to parameter $pref_x$

This model is used to send the preferences required for a set of timeslots to the back-end meeting scheduling server. These preferences are then used by the automated algorithm developed by Marcinowski to compute a proposed meeting time.

Chapter 4

Integration

This chapter discusses the integration of this application with the user's Google Calendar and with the existing back-end responsible for computing a meeting time based on user preferences. To recap the ports running the different servers refer below.

Front-End Flask Application	Port 80
Back-End Meeting-Scheduling (MS)	Port 3000
Back-End Peer-Manager (MS)	Port 3002

4.1 Google Calendar Integration

To implement this calendar integration, an external python library was used, `google-api-python-client`. This library allows the application to complete the OAuth 2.0 authorization protocol used by the Google Calendar API. The following steps are performed to complete the protocol in order to gain access the user's calendar.

1. Register the application using the Google Developers Console to acquire a web application client ID that is required to get access tokens for each user.
2. Redirect the user to the Google OAuth 2.0 server along with the client ID acquired in the previous step. The application can then gain consent of the user to access their requested data.

3. If the user grants access to their data, then the OAuth 2.0 server generates an authorization code and returns to the application.
4. This authorization code is then exchanged with the OAuth 2.0 server to get an access token for the user.
5. The authorization protocol is now complete and this access token can be used to retrieve data from the user's Google Calendar.

Now that the user has granted the application to access their data, the next step is requesting data from Google Calendar and parsing it's response to extract only the information required. The google-api-python-client library greatly reduces the difficulty of this task by providing access to a set of functions which are described below.

- **build** - This function allows the application to create a service object to access the services API. It takes two required arguments, API name and version. For the purpose of this application the API being accessed is 'calendar' and API version being used is 'v3'.

```
service = build('calendar', 'v3', ...)
```

- **events.list** - After creating the service object using build, the application is able to query the user's calendar for events using the list function. Among the many arguments that list can take, the two of concern are 'timeMin' and 'timeMax'. Using these two parameters the application is able to retrieve all events scheduled between 'timeMin' and 'timeMax'.

```
service.events().list(calendarId='primary',  
                        timeMin='2016-03-12T08:00:00',  
                        timeMax='2016-03-12T20:00:00',  
                        singleEvents=True,  
                        orderBy='startTime')
```

The 'timeMin' and 'timeMax' have to hold datetime values in the ISO format.

It is assumed that the user uses their primary calendar for their meet-

ings. However, this is a limitation of the application where it is not able to access data stored in multiple calendars by the user.

4.2 Back-End Integration

The integration with the back-end is done primarily through the RESTful API developed by Marijonas (2015). However, the API does not have the ability to register with the PM service. To do this a call was made directly to the PM server to register the user.

4.2.1 Register New User

After the user initiates the registration process, a connection is made with the PM on 'localhost:3002'. Then a HTTP POST request is made to 'localhost:3002/users' with the following header and body to register the user with the peer-manager.

```
header = {  
    Authorization: 'Basic username:password',  
    APP_KEY: app_key,  
    APP_SECRET: app_secret  
}  
  
body = {  
    username: email,  
    password: password,  
    email: email,  
    emailConfirmation: False  
}
```

Listing 4.1: POST Request to Register User

If the peer-manager responds with 201, then the registration was successful and the application continues executing the rest of the process. Otherwise, the user is notified that an error occurred during the registration process and is prompted to try again.

4.2.2 POST New Meeting

To post a new meeting to the MS server running on 'localhost:3000' the following function is called as specified by the script to test the RESTfulAPI using an HTTP POST request.

```
requestMS('meetings', 'POST', {
  earliestDate: 'a date time string in the ISO format',
  latestDate: 'a date time string in the ISO format',
  duration: 'duration in minutes',
  guests: 'array of email addresses'
})
```

Listing 4.2: POST New Meeting

If the response from the MS server was 200 then the meeting has been successfully posted on the MS server and the application continues to save the meeting on the front-end with an initial meeting state of 'Pending'.

4.2.3 Send Preferences

One of the most challenging parts of this project was implementing the algorithm that is used to send an user's preferences a meeting being scheduled. The process can be broken down into the following steps.

1. Compute the range of free slots from the earliest to the latest date within which the meeting has to be scheduled.
2. Compute the preference for each free slot using the preferences set by the user and using the model described in Chapter 3.
3. Append the list of busy slots during this period to the interval array from the previous step. Any slot where an event is already scheduled is regarded as a busy slot.

The difficulty of this process lies in step 1. Where the free range of time slots have to be divided further into smaller ranges that are divided into the different times of day used by the model. For example, if an user was free from 10:00 AM

to 18:00 PM on a particular day then this free slot had to be split into three ranges, one for each time of day resulting in three free slots, 10:00 - 12:00, 12:00 - 16:00, and 16:00 - 18:00. This task becomes more difficult as the free range of timeslots span over a period of days. The first approach attempted to perform this operation is shown below in pseudo-code.

```
for each event in the range:

    if next event - current event > 12 hours:
        divide free slot for current day until 20:00
        divide the remaining time left for the next day

    elif next event - current event > 8 hours:
        divide free slot into three further divisions

    elif next event - current event > 4 hours:
        check at what tod the event starts
        divide free slot into two further divisions

    else:
        append the end of the current event to array FS
        append the start of the next event to FE
```

Listing 4.3: Approach 1

This approach worked for only a small set of possible free slot ranges. However, to account for all possible free slot ranges over 12 hours would require adding an indefinite number of if-then statements. Therefore, another approach was taken. Though this new approach required more time computationally, $O(\text{no. of free ranges} * \text{no. of divisions made})$ instead of $O(\text{no. of free ranges})$ taken by the previous approach, it was able to account for all possible scenarios of free slot ranges.

```
for each event in the range:

    if not the last event:
```

```

Append event to FS
Check tod of event

while next event - current event > 4 hours or next event tod !=
    current tod of free range:

    Append end of the current tod to FE

    if not the last tod:
        Increment tod
        Append start of the current time of day to FS
    else:
        Reset tod
        Increment date by 1
        Append the current tod for the current day to FS

Append start of next event to FE

```

Listing 4.4: Approach 2

This approach executes an additional while loop as long as the free time in-between two events is greater than 4 hours or the next event starts at a different time of day. This is because each division in the time of day (tod) array is at an interval of 4 hours. Therefore, if the gap between events is more than 4 hours then the free slot is split into the necessary number of divisions. When the gap is less than four hours it is checked if the smaller free range slot starts and ends during the same time of day.

After this is completed, the application proceeds to complete steps 2 and 3. Once all the steps have been completed, the following HTTP PUT request is made to the MS server with an user's preferences.

```

requestMS(meeting.url + '/guestPreferences/' + emailID, 'PUT', {
    'intervalAvailability': interval
})

```

Listing 4.5: PUT Guest Preferences

‘interval’ represents the array holding items representing a duration of time and the corresponding preference for that duration.

The MS server responds with a 200 response if the guest preferences have been successfully added.

4.2.4 GET Proposed Meeting Time

Once all the participants of a meeting have sent their availability, the application retrieves the proposed time calculated by the MS server. This action is completed using two GET requests with a gap of 5 seconds in the middle. This was the time gap used by Marijonas in his test file to allow the back-end to finish computing a proposed time and it was decided not to change this value.

```
requestMS(meeting.url, 'GET'})  
time.sleep(5)  
requestMS(meeting.url, 'GET')
```

Listing 4.6: GET Proposed Meeting Time

The MS server responds with a list of suggested timeslots and the suggested timeslot with the highest preference is set to be the proposed timeslot of the meeting. These details are then save to the relevant document in the *Organiser* collection and the corresponding state of the meeting is changed to ‘Proposed’. The updated view can then be seen by the user on the browser.

The rest of the changes in meeting state are handled by the front-end and does not require communicating with the back-end servers.

Chapter 5

User Experience Design and Implementation

This chapter discusses how the actions in the previous chapter are triggered by the user using the interface and how their results are displayed in the browser with a focus on User Experience.

5.1 Concept User Experience Framework

The basis of the User Experience (UX) framework used to develop this application was inspired by a number of sources. There were more than 10 factors mentioned between the resources and it would have been impossible to address all of them. But the factors that were identified to be the most important with regard to this project have been grouped into three main components. This allowed for a more focused development, producing a better quality of work in the selected areas. The reason behind the choice of each component is divided into two parts, what each component represents and why it was chosen. [14] [13]

5.1.1 Interaction Design (ID)

- **Description:**

There is an increasing pressure to create rich UXs and at the heart of

this experience is interaction design. Interaction design tries to address a number of questions that help define the UX. Some of these questions and areas are shown below.

1. What layout pattern would work best?
2. How to draw the user's attention to the action at hand to reach their end goal? Conversely how do you draw the user's attention away from things that are not required to complete the action at hand?
3. What user interface components should be used to show the necessary information in a manner which the user can comprehend?
4. Striking the balance between familiarity and innovation.

- **Reason:**

It can be seen that by addressing the above questions the developer is able to mould an UX for users catered to performing the task at hand. If there had been other applications that performed the same task, which were widely accepted then they would have provided a good example to follow. However, because there are not any existing applications made for automatic meeting scheduling this was an extremely important aspect of the design process.

5.1.2 Controlled Feedback (CF)

- **Description:**

This aspect is responsible for determining the amount of feedback that should be relayed back to the user to help them interact with the application and be aware of the status of a particular completed action. Furthermore, it also addresses how convey this feedback in order to bring about a prompt reaction, if necessary, from the user.

- **Reason:**

Since there are several stages in the process of scheduling a meeting, this was a vital aspect of the experience as it was responsible for notifying of the user state of a meeting being scheduled and the states of participants

in the meeting. This feedback gives the user a better picture of what stage of scheduling process a meeting is in.

5.1.3 Device Independent (DI)

- **Description:**

This aspect determines the how well the application scales and functions on different devices with varying screen sizes. The application should be able to perform all the expected functions on each device while having a consistent visual design across the pages for all screen sizes.

- **Reason:**

This is an aspect that has become more important than ever before, with society moving towards a multi-device culture where devices vary in screen size. It is important that the application delivers the same UX regardless of the device it is being used on.

It is important to note that the above choices were personal choices and can vary from developer to developer. Other aspects such as Ease of Use and Accessibility are also considered to be important parts of developing an overall UX.

5.2 Design of Web Pages

Overall, each web page was designed to have a minimalistic appearance and focus on the framework discussed in the previous section. Thereby, attempting to create an application with a clear flow of content and convey appropriate feedback where necessary. Additionally, by using the bootstrap framework described in section 2.1.1 the application provides a consistent appearance across different screen sizes.

5.2.1 Registration and Login

These pages are designed to replicate any registration and login pages of popular web applications such as Gmail, Facebook, and Outlook with a few slight modifications which are described below.

5.2.1.1 Register Page

To keep a minimalist design, the form only consisted of input boxes with place holders indicating to the user what is to be entered into the text field. But because the function of this page is to allow the user to register with the service, the user input had to be validated. This validation was handled server side to comply with the Model-View-Controller architecture being followed. If any errors are identified during the validation process, they would be conveyed back to the user through text describing the error below the corresponding field. In the case where there are no errors present in the user input, it will initiate the registration process and taken to the Organiser Page upon completion of the process. This transition from the Register Page to the Organiser page upon clicking the Register button provides feedback to the user that they have been successfully registered without the need to display explicit text stating the same.

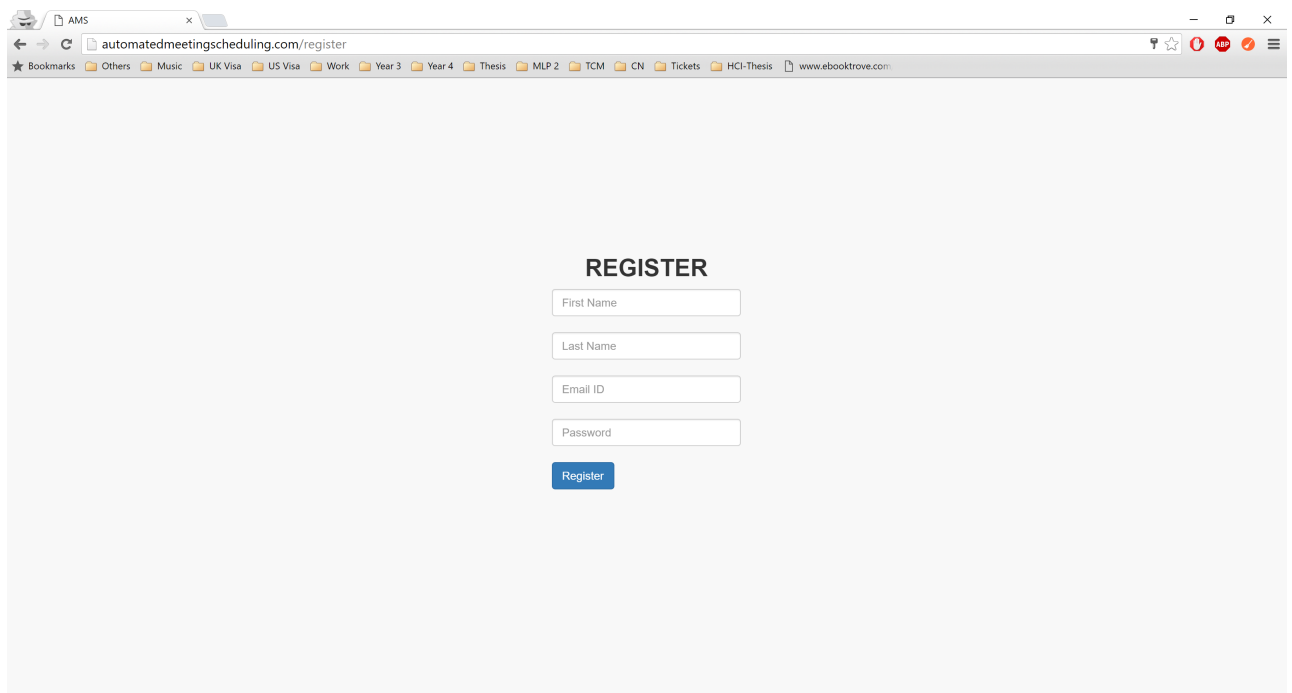


Figure 5.1: View of Register Page

Additionally, because the registration involves communicating with the peer-manager this process may not be instantaneous. Therefore, to notify the user that the process is still ongoing an animated spinner is used. This spinner continues to spin until the user is redirected to the Organiser page. Furthermore, to draw the user's attention to the button, the colour of the button is changed from

blue to yellow. All these decisions were taken with regard to the CF factor of the UX framework.

5.2.1.2 Login Page

To have consistency between the pages, the style of the text fields and the feedback received upon validation errors were kept the same as the Register Page. The page consisted of two text fields, one for the email address and the other for the password. The input to the password field was kept hidden as expected so as to prevent people nearby the user from obtaining their password by glancing at their screen.

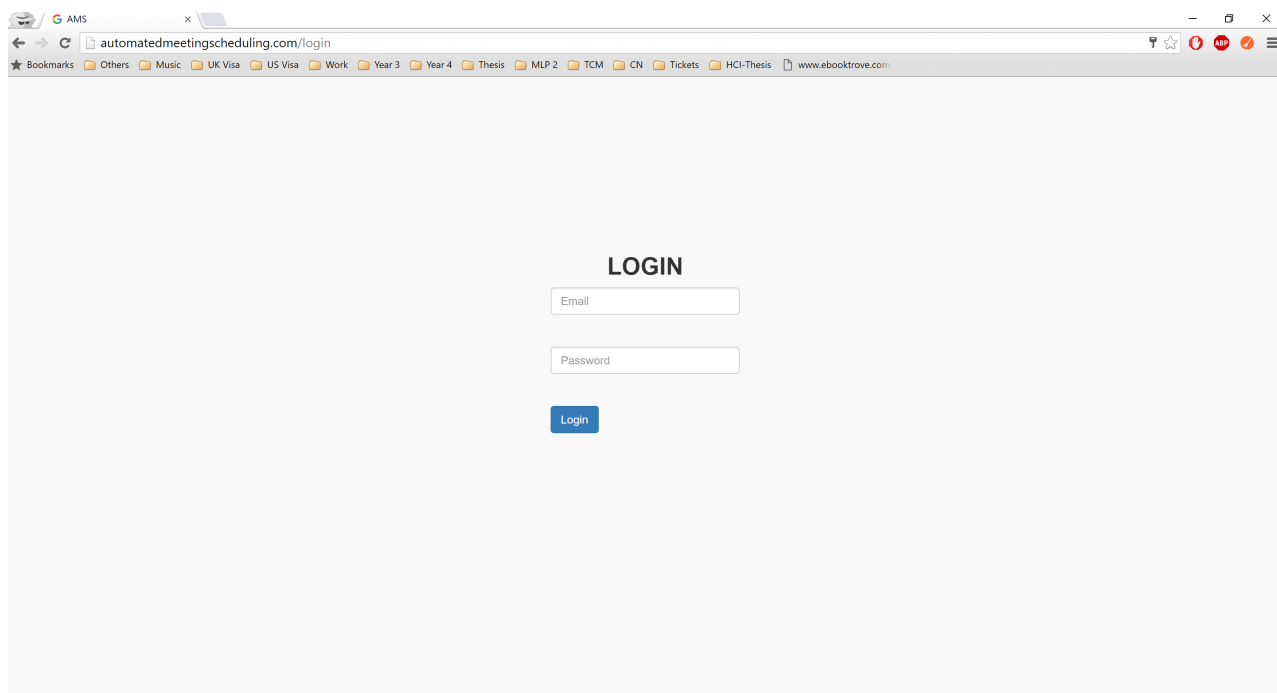


Figure 5.2: View of Login Page

Upon clicking the Login button, the user is redirected to the Organiser page. Since, during the login process there is no communication required between the peer-manager or the meeting-scheduling server, this process happens instantaneously to the human eye. Therefore, there is no need to notify the user when the process is ongoing.

5.2.2 Organiser and Participant

The layout of these pages were divided into three columns where each column had a specific function. This decision to separate the layout into the following columns resulted from considering the ID factor of the UX framework.

- **Navigation Pane** - The left most column was responsible for navigation between different parts of the application and was made visible in the same location of every page with the same options. This column takes up 20% of the screen width space when the screen width is larger than 768px. But for smaller screens this column would collapse into a button on the top right that allows the user to toggle the visibility of this pane and display the content at the top of the page instead of the left. this pane also had the option to allow the user to logout of the application.

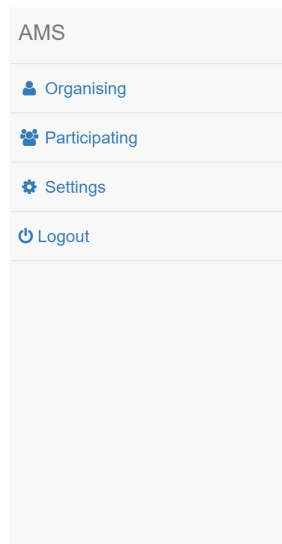


Figure 5.3: View of navigation pane

- **Meeting List Pane** - The second column displayed a list of meetings being organised by the user or in which the user is participating in. At the top of the list the user has access to three separate controls.
 1. Search - A grey search icon that upon clicking searches through the displayed meetings in the list and resets the view of the list to only display the meetings that match the search. However, due to time constraints this feature could not be implemented.

2. Add New Meeting - A blue plus button which upon clicking causes a panel to appear by sliding down from the top of the content pane, so that the user can schedule a new meeting. After successfully creating a meeting, the panel disappears by sliding up from the bottom.
3. Delete Meeting - A red garbage icon that is only available in the Organising page to allow the user to delete any meetings they are organising or have organised.

An important aspect of this pane is representing the different states of each item in the meeting list. The colour combination used to do this is shown below.

Yellow	The initial state and other non-final states
Green	A positive final state has been reached
Red	A negative final state has been reached

In addition to the colour, the literal string of the state is displayed. It was chosen to display the literal string of the state instead of icons to avoid any confusion in the interpretation of the icon by the user and removing the need for the user to learn the meaning of each icon used to represent the different states.

- **Content Pane** - The content of the third column changes dynamically to display the content of the meeting selected from the Meeting List Pane or to display the form used to create a new meeting. This column appears only when required so as to draw the user's attention to it only when needed, otherwise it disappears so as to not distract the user from their current task.

A similar layout is used in other popular web applications such as Gmail and Outlook. Hence, enhancing the familiarity the user feels while using the application. Such a layout also provides a gradual flow of information from left to right that further re-enforces the familiarity felt by the user as the same approach is followed in the applications mentioned earlier. Additionally, this is also the flow of information presented in most languages apart from a few such as Arabic and phonetic languages. But as support for multiple languages was not a part of the objectives of this application, it was designed for users whose first language is English.

5.2.2.1 Organising Page

Each item in the meeting list pane displays the title, number of guests, a date and state of the meeting. The date displayed shows the proposed date of the meeting if it has been proposed, otherwise it shows ‘TBD’, a common abbreviation standing for ‘To be decided’ that is assumed to be known by all users to use the system.

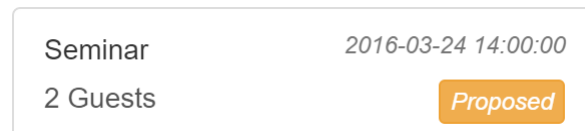


Figure 5.4: Example of an item in the meeting list pane in the Organising Page

Upon clicking an item from the meeting list pane, a view is displayed in the content pane that summarizes all the information pertaining to that meeting. The design of this content is discussed with reference to Figure 5.5.

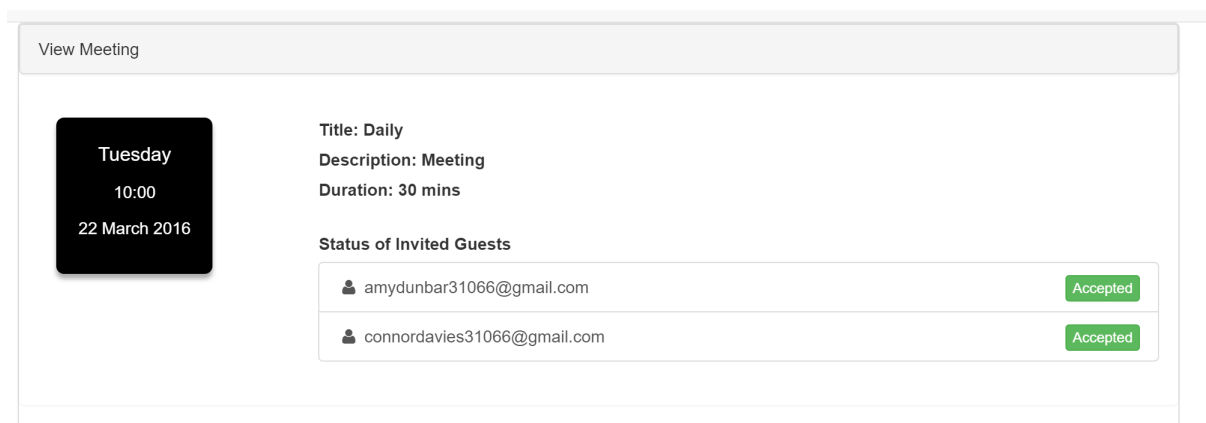


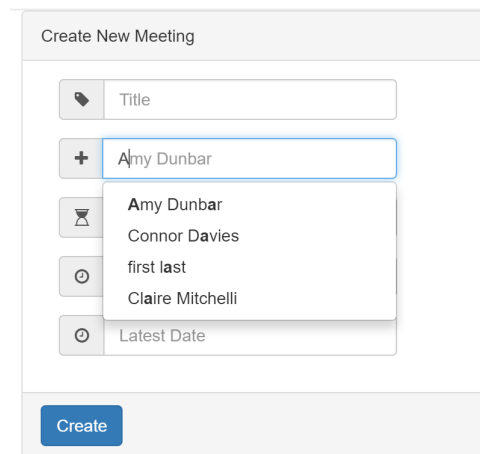
Figure 5.5: View of a Meeting

At the top left of the pane, the date of the meeting is shown if it has been proposed. This is an important piece of information. Therefore, it is displayed in a separate column with a large size and different background to draw the user’s attention.

The remaining space shows the title, duration and status of each participant in the meeting using a list. Each item in this list consists of a user icon, email

id of the participant and the corresponding state of the participant. The most important pieces of information being conveyed to the user in this list are the states of the participants. Thus, it was decided to display the state in a manner consistent to that used in the meeting list pane with the same colour combination.


The content pane is also used to display the form to create a new meeting. This form makes suggestions to the user as they type in the names of a participant. This was achieved with aid of an external JavaScript library, typeahead.js. [9]




The image shows a web form titled "Create New Meeting". It contains several input fields with icons to their left: a key icon for "Title", a plus icon for a participant list, a clock icon for a time field, a circular arrow icon for a repeat field, and another circular arrow icon for a "Latest Date" field. The participant list field is active, showing a dropdown menu with suggestions: "Amy Dunbar", "Connor Davies", "first last", and "Claire Mitchelli". A blue "Create" button is at the bottom of the form.

Figure 5.6: Suggestions of Participants Available to Invite


After selecting a participant, they are displayed below the field. The user can add more participants using the same process and remove any participant, if required, afterwards. Additionally to select the date a mini-calendar shows up for the current month from which they can select a date.





Test





Invite Participants

 Amy Dunbar




 Connor Davies





30



Earliest Date

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Figure 5.7: View of selected participants and mini-calendar

After selecting a date the user can proceed to selecting a slot from the displayed timeline view by selecting on the checkboxes. This timeline shows any scheduled meetings between the range of free slots for the selected day. This allows the user to better visualize their day.

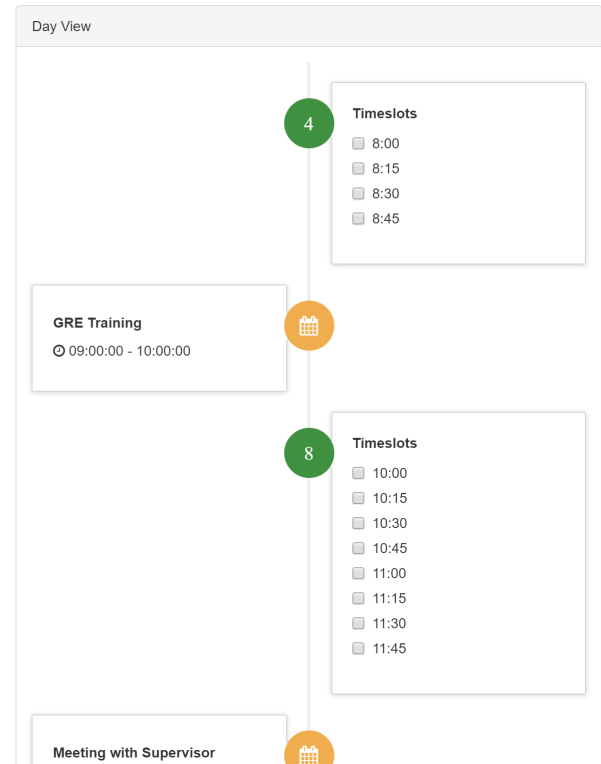


Figure 5.8: View of the timeline view to select a slot for the earliest and latest date

5.2.2.2 Participating Page

Each item in the meeting list pane displays the title, email id of the organiser, a date and state of the participant. The value of the date field is the same as described in the previous section.

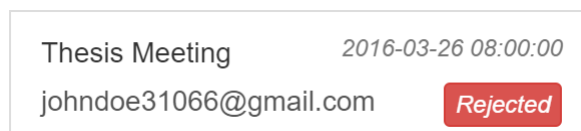


Figure 5.9: Example of an item in the meeting list pane of the Participating Page

Upon clicking an item from the meeting list pane, a view is displayed in the content depending on the status of participant for that meeting. If the meeting state is 'Proposed' then the user is prompted to respond using the timeline view shown in the figure below.

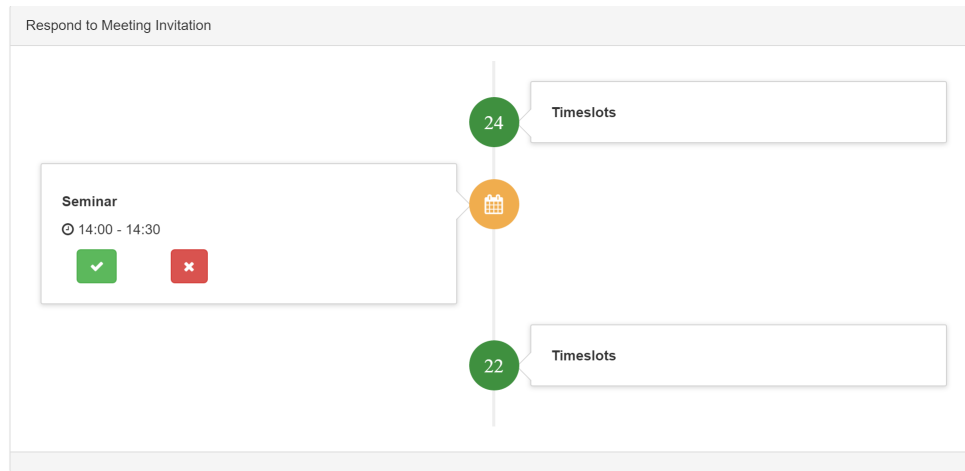


Figure 5.10: Interface to Respond to Proposed Meeting time

By clicking the green check button, the user can accept the proposed meeting time. Conversely, by clicking the red cross button they can reject the proposed meeting time. These icons and colours are a standard for representing positive and negative responses.

5.2.3 Settings

The layout of this page was different from the previous two pages due to the difference in content. This page contains only two columns. The first column is the same Navigation Pane used in the previous two pages but the second column is a form.

The form consists of a series of questions that the user can respond to by selecting the buttons below. Additionally, they can set the priority using the combo-box used to the left of a set of questions. A sample input is shown in the figure below.

AMS

Organising

Participating

Settings

Logout

High

Which days do you **PREFER** to schedule your meetings on?

Monday Tuesday Wednesday Thursday Friday Saturday Sunday

Which days do you **NOT PREFER** to schedule your meetings on?

Monday Tuesday Wednesday Thursday Friday Saturday Sunday

High

What time of day do you **PREFER** to schedule meetings at?

Morning Afternoon Evening

What time of day do you **NOT PREFER** to schedule meetings at?

Morning Afternoon Evening

High

Would you rather have all your meetings one after another or with a gap?

Consecutively Gap

Save

Figure 5.11: Interface to Set Preferences

The last setting did not affect the preference as it was an attempt to further advance the features during development but due to time constraints was left incomplete.

Chapter 6

Evaluation

This chapter discusses the two experiments to evaluate the usability of the approach used to develop this application. The points of discussions (POD) that were raised during the evaluation are elaborated on and identifies a way to resolve the issue.

6.1 Methodology Used

Contextual interviews are a method used to test the usability of websites. This is a more informal approach that is considered a more natural way of gathering data and can lead to more realistic results. Usually, a subject is not required to perform any particular task during this process and interact with the website as they want to. However, in an attempt to formalise the interviews to get a more focused feedback from the subjects, in this project they were asked to complete a certain series of tasks upon which they were asked questions. One of the major disadvantages of using contextual interviews as a method for evaluation is its inability to collect quantitative data. Instead, this method allows for the collection of only qualitative data. Since, I wanted to evaluate the usefulness and judge the cognitive load required to perform the tasks given using the application, which required gathering more qualitative than quantitative data, this method for evaluation was deemed suitable. This method also proved to be able to highlight the different cognitive load experienced by subjects when using the two different approaches trying to achieve the same end goal. [4]

6.2 Experiments

Two experiments were conducted, where the first experiment was a comparative study where 10 subjects were asked to compare the difficulty of each task against Doodle and the second experiment was conducted among a further 10 subjects to evaluate the application without a comparison made with Doodle.

6.2.1 Scenario

No. of Participants	Independent Variable	Dependent Variable	Control Variable	Random Variables
20	Application Interface	User Behaviour	Meeting being scheduled	Subject's Mood, Keyboard Layout, Operating System

6.2.2 Experiment 1 - Comparison Study

6.2.2.1 Goal

The goal of this experiment was to identify the similar steps in the approaches taken to schedule a meeting using the prototype and Doodle. These steps were then compared against each other to evaluate the usability of performing each step on both applications. In this case, the questions asked in the contextual interview focused on the user interface and the cognitive load experienced in both approaches.

6.2.2.2 Tasks and Questions Asked

Task 1

Complete the following steps on both applications to create a new meeting to be scheduled.

Step	Doodle	Prototype
1	Fill in title	Fill in title
2	Select days	Select earliest and latest date
3	Enter three time slots	-
4	Invite Participants	Invite Participants

Each subject was asked to complete each of the steps on both applications. Before moving on to the next step they were asked a set of questions. This was repeated till all the steps were completed. Then a final round of questions were asked.

Task 2

Complete the following steps to respond to a meeting invitation.

Step	Doodle	Prototype
1	Click on doodle poll link	Click on specific meeting
2	Choose preferred slots	Accept or Reject proposed time

Questions

After the completion of each step on both applications, the subjects were asked the following questions.

- Q1: How different or similar was each step on both applications?
- Q2: Which application had a better interface for completing it's step?
- Q3: Did the step cause you to experience some kind of cognitive load? If so, on which application did you feel it required more of a cognitive load?

Finally, after completing all the step to a task, the following to questions were asked to the subjects.

- Q4: Which approach do you prefer and why?

- Q5: Would you like to highlight any other other concerns?

6.2.2.3 Discussion of Key Observations and Interview Responses

- **POD1 - Task 1 similarities** - Steps 1 and 4 were deemed to be similar in both approaches and put a negligible amount of cognitive load on the subject.
- **POD2 - Removal of step 3 in Task 1** - The approach used by the prototype does not require to have step 3, where the subject is asked to enter slots from which they would like the participants to pick from. This was well received and reduces the cognitive load felt by the subjects. However, a few subjects brought up that they feel more comfortable when using Doodle where they have a greater control over the time slot entries even though it increases the cognitive load.
- **POD3 - View of Meeting details** - the subjects said that they preferred being able to see the input to all the steps at times during the process of completing the task.
- **POD4 - Task 2 interface** - All the subjects preferred the interface of the prototype than Doodle. Subjects did not understand the point of the calendar view in Doodle, where they were only able to view the different choices in their calendar but could not accept their preferred ones from that view. To accept their preferred choice they had to revert back to the table view and select the corresponding checkbox. This reflected a dissatisfaction among users caused by the interface not being able to match the user's expectation. On the other hand, subjects liked the interface used by the prototype where they can accept or reject the proposed time in the timeline view.

Overall, the responses from the subjects indicated that the prototype had a better interface with regard to the visual design and feedback received upon completing a task. Additionally, all the subjects preferred the approach taken by the prototype to complete Task 2 but the same was not the case for Task 1. The approach preferred for Task 1 varied among the subjects. The common reason given by the user's as to why they chose Doodle was because they had more control over the

scheduling. Though this is a valid point, it also showed how this experiment lacked in replicating the actual difficulty of the overall task of scheduling a meeting. With a focus on the user interface and cognitive load of each step, it was unable to replicate the difficulty faced in coming to a common meeting time among multiple participants. It is true that subjects given the choice would like to have more control, but when this task becomes more and more tedious as the number of participants grow it can be expected that the approach taken by the prototype would be preferred so as to save time. However, the experiment succeeded in identifying the preferred interface between the two applications and identify areas where the calendar integration of Doodle lacked functionality.

6.2.3 Experiment 2 - Independent Study

6.2.3.1 Goal

The goal of this experiment is to study the usability of the prototype without comparing it to Doodle. Without a comparison, I believe that it is possible to identify the positives and negatives felt by the subjects when taking this new approach to the design of a front-end web application implementing an automated meeting scheduling algorithm. Additionally, it also asks questions to further improve the interface.

6.2.3.2 Tasks and Questions

Task 1 - Create a new meeting with three participants

The subjects were given a dummy account with which they were to perform this task. They were also told which three participants to invite along with the duration and title of the meeting. However, it was up to the subject to choose the period within which this meeting had to be scheduled.

Task 2 - View meetings that have been scheduled.

The subjects were asked to view a Finalized meeting and a Cancelled meeting.

Task 3 - Set preferences.

The subjects were asked to login as the user specified to them and set their preferences.

Task 4 - Send preferences and respond to a proposed meeting time

The subjects were asked to go to the Participating page after completing Task 2. On this page they were asked to send their preferences and then respond to the proposed meeting time.

Questions

- Q1: Was the task easy to perform?
- Q2: Are there any concerns that you have regarding this task? If so, why?
- Q3: Is the interface clear?

6.2.3.3 Discussion of Key Observations and Interview Responses

- **POD5 - Subjects are able to remove participants added by accident** - All subjects were able to invite participants without any difficulty. The subject were also able to correct themselves if a wrong participant was added using the interface.
- **POD6 - Format of input required in duration field unclear** - The duration field caused confusion in subjects where the input varied from '1 hour', just '1' and '60 mins'. This is caused due to the interface not clearly stating how the duration should be entered. It was suggested to provide a list of valid options so as to not allow the user to enter an incorrect value. Adding this feature would also remove any possible hesitation by the user during the entry of the duration into the form, given they have already decided for how long they want the meeting to take place.
- **POD7 - Reaction to timeline view** - During Task 1, when the subject was required to select an earliest and latest date, the subject were able to understand what the timeline is showing but were confused by the large

number of slots to pick from. The large number of slots were due to the fact that the automated algorithm represents a single slot as a 15 minute window. This was believed to be too short because it results in a single 12 hour day to have 48 timeslots. The number of slots can be reduced by the prototype by taking into account the duration of the meeting. In Task 4, where the subject responds to a proposed time highlights a concern caused due to this same issue. Since, the timeline displays the total number of free slots on the day of the proposed meeting, it is not representative of the amount of time one is free for the day. The subjects highlighted that instead of using the number of free slots to use the actual span of time it represents to give them a better idea of how free they are on the day of the proposed meeting time. For example, if the subject is indicated 8 free slots before a proposed meeting, then that corresponds to a 2-hour window. By showing the subject that they have two hours free before the proposed meeting time will induce a better response.

However, in terms of the visual interface used to show the timeline, was well liked by the subjects. They said that it was clear and intuitive as to what they are expected to do when presented with the view.

- **POD8 - Preference model is intuitive but may be too simple** - The subjects liked the simplicity of the questions asked to set their preferences in Task 2. They said it reflected how they would cognitively think about when they would like meetings to take place.

However, one of the subjects brought up an interesting point. The subject said that it would make more sense if he was able to set what time of day he prefers for each day rather than the same time of day being preferred regardless of the day. For example, a person could prefer Monday mornings but not Friday mornings. Though the prototype is able to partially account for this by setting a low preference for Fridays to counter-act the high preference given to mornings, it indicates how the current preference model may be too simple. Another concern in integrating this into the prototype would increase the number of options from the subject has to pick his preferences from 14 to 42 and Hick's law states that as the number of options grow, the time required to respond increases. [10] Therefore, for people who tend to change their preferences often, this would become a tedious task.

Conversely, for people who tend not to change their preferences for a coming week, this approach may lead to better proposed times.

- **POD9 - Giving the organiser more control** - In Task 3, a subject asked what happens in the case that only one participant has rejected the proposal upon seeing the status of the individual participants on a Cancelled meeting. This raised an area of concern where the subject suggested on implementing a way to give the user more control.
- **POD10 - State monitoring done well** - All subjects said that they knew the state of each meeting during the entire process and of the individual participants of a meeting, in the case where the subject was an organiser. The only suggestion was that participants should also be allowed to see the status of other participants in the meeting. This was not included so as to avoid influencing the decision of an user depending on the state or responses of other participants. There have been studies which show that behaviour can be affected by the response of other participants.
- **POD11 - Organiser and Participant page well designed** - The layout of each page was praised by the subjects stating that the division of the different parts of the web page made for easy viewing. It was also observed that in general subjects were comfortable navigating around the webpage.

6.3 Summary

Overall, the evaluation process reflected that the user-interface provided by the prototype was better than that of Doodle and was able to highlight issues faced by subjects when scheduling meetings with approach used by this front-end meeting scheduling application. One of the successes of the application was in the creation of a system which monitors the different stages of a meeting being scheduled and inform user's so that they know how close or far they are from the meeting being scheduled.

Additionally, taking into account all the POD, an outline of how some of them were addressed are discussed below.

6.3.1 POD7

Though the issue raised in POD6 could not be tackled entirely, a small adjustment was made to the timeline view. Instead of showing the total number of timeslots available, the icon was changed to signal to the user that these are free slots. To not allow the confusion in the user due to the interpretation of what the number represents.

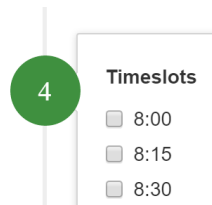


Figure 6.1: View before change of icon

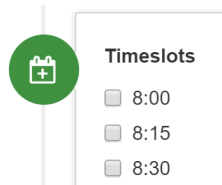


Figure 6.2: View after change of icon

6.3.2 POD9

The organiser could be given more control by using an additional state in the meeting life cycle that occurs in the scenario where all the participants do not have the same response to proposed meeting time. This was cause the state transition of a meeting from ‘Proposed’ to ‘Final/Cancelled’ mention in Chapter 2 to change. A modified transition is shown below, with the addition of an ‘Attention’ state.

State	Description	Criteria for Completion	Next State
Proposed	A meeting time has been computed based on the preferences of each participant.	ALL participants have either Accepted or Rejected the proposed time.	Final/Cancelled/Attention
Cancelled	ALL participant has rejected the proposed meeting time.	-	-

Table 6.1: Changes to the state transtitions

The attention state would allow the organiser of a meeting to react to the situation in three ways by scheduling the meeting knowing that all the participants cannot attend, reschedule the meeting and ask the participants to respond to a new meeting time, or cancel the meeting. These options would result in three further state transitions which can occur in the lifecycle of a meeting being scheduled. (See below)

State	Description	Criteria for Completion	Next State
Attention	One or more participants have rejected the meeting time.	The organiser chooses to reschedule the meeting.	Proposed
Attention	One or more participants have rejected the meeting time.	The organiser chooses to cancel the meeting.	Cancelled
Attention	One or more participants have rejected the meeting time.	The organiser chooses to finalize the meeting time.	Final

Table 6.2: Possible transitions from ‘Attention’ state

This also adds a new type of workflow where the organiser is able to take an executive decision to indicate to the system what to do next. The prototype was modified to account for the ‘Attention’ state but not in time to get any feedback from subjects. Unfortunately, due to the time constraints the actual actions taken upon response of the organiser could also not be implemented.

Chapter 7

Conclusion

7.1 Summary

The approach developed to create a front-end meeting scheduling application which automated the process of meeting scheduling highlighted the following areas that have to be taken into account in the development process.

- Integrating the application with an automated meeting scheduling algorithm.
- Creating a state model to monitor all the states of a meeting during the entire scheduling process.
- Choosing a model to compute preferences of timeslots that are used by the automated algorithm to compute a proposed meeting time.
- Designing and developing an interface that is able to perform all the steps required to initiate and complete the scheduling process.

This approach was initially evaluated against the approach taken by Doodle that highlighted the differences in the approach in terms of usability. The second evaluation of the approach without a comparison highlighted that the state monitoring and the interface used to send the preferences as a success. The report also identified several areas of the approach that caused issues experienced by an user and discussed possible solutions to fix these issues in the given approach. However, only a few of them were implemented into the application due to time constraints.

Overall, this project can be deemed as a success in terms of developing an approach that can be further developed to build more robust front-end web applications for automated meeting scheduling.

7.2 Major Limitations

1. High-level Preference Model too simple - The use of such a simple preference model was not able to account for preferences changing due to the number of meeting in the proximity of a range of free slots. However, this can be added to the model by adding the corresponding w_{prox} and $pref_{prox}$ parameters. The difficulty lies in developing a method to compute the $pref_{prox}$ parameter that represents the preference depending on the number of other scheduled meetings before and after the range of free slots.
2. Not accounting for meetings scheduled using the application - The application is only able to send the preferences of the user based on existing events in their calendar and does not consider the inclusion of proposed meeting times that they are scheduled to attend from within the application.

7.3 Further Work

There are many areas for further work as it is a new approach for developing front-end meeting scheduling application which automate the process of scheduling. Some of these areas correspond to developing ways in which the limitations discussed in the previous section could be overcome.

One other specific area that can be investigated further is a way to record a history of the reasons for rejection of proposed times and update the preferences set for time slots based on this history. Additionally, by providing the organiser with the reason for rejection by participants, they take a more informed action to the 'Attention' state.

7.4 Lesson Learnt

In designing a novel approach to such a problem, the application has to go through an evaluation process repeatedly to identify and fix all the concerns related to the approach. Therefore, since time was poorly managed during the development of this application, the completed approach evaluated was still in its infant stages and resulted in some issues raised that could have been easily fixed. The development of the approach through repeated phases of evaluation would have produced a more robust approach to tackle the problem of building front-end web applications for meeting scheduling.

Bibliography

- [1] Vangie Beal. *RDBMS - relational database management system*. URL: <http://www.webopedia.com/TERM/R/RDBMS.html>.
- [2] -. *Bootstrap Framework*. URL: <http://getbootstrap.com/>.
- [3] Ryan Brown. *Django vs Flask vs Pyramid: Choosing a Python Web Framework*. URL: <https://www.airpair.com/python/posts/django-flask-pyramid>.
- [4] Darko Čengija. *Usability: What is contextual inquiry?* URL: <http://www.uxpassion.com/blog/usability-contextual-inquiry/>.
- [5] Elisabeth Crawford and Manuela Veloso. “Learning to Select Negotiation Strategies in Multi-Agent Meeting Scheduling”. In: *Computer Science Department, Carnegie Mellon University NA.NA* (2005), pp. 1–7. DOI: <https://www.aaai.org/Papers/Workshops/2005/WS-05-09/WS05-09-005.pdf>.
- [6] Elisabeth Crawford and Manuela Veloso. “Mechanism Design for Multi-Agent Meeting Scheduling”. In: *Computer Science Department, Carnegie Mellon University NA.NA* (2004), pp. 1–11. DOI: <http://www.cs.cmu.edu/~mmv/papers/06IAT-liz.pdf>.
- [7] Google Inc. *Google Calendar API*. URL: <https://developers.google.com/google-apps/calendar/>.
- [8] MongoDB Inc. *The MongoDB manual*. URL: https://docs.mongodb.org/manual/?_ga=1.8709372.890382677.145942268.
- [9] Twitter Inc. *typeahead.js*. URL: <https://twitter.github.io/typeahead.js/>.
- [10] Lepper MR. Iyengar SS1. “When choice is demotivating: can one desire too much of a good thing?” In: *J Pers Soc Psychol* 79.6 (2000), pp. 995–1006.
- [11] Bartłomiej Marcinowski. “Flexible Meeting Scheduling”. In: *Computer Science, University of Edinburgh NA.NA* (2014), pp. 15–24.

- [12] Marijonas Petrauskas. “Integrating and improving an automated meeting scheduling service”. In: *Computer Science, University of Edinburgh* NA.NA (2015), pp. 1–45.
- [13] Luis Olsina Philip Lew and Li Zhang. “Quality, Quality in Use, Actual Usability and User Experience as Key Drivers for Web Application Evaluation”. In: *Web Engineering, 10th International Conference* NA.NA (2010), pp. 218–233.
- [14] Steve Psomas. *The Five Competencies of User Experience Design*. URL: <http://www.uxmatters.com/mt/archives/2007/11/the-five-competencies-of-user-experience-design.php>.
- [15] “Scheduling Meetings by Agents”. In: *Department of Information System Engineering, Ben-Gurion University of the Negev Beer-Sheva, Israel* NA.NA (2010), pp. 1–15. DOI: <http://www.jmir.org/2015/7/e175/>.