Automated Meeting scheduler using chatbot

Amar Singh[1]
Information Technology
K.J. Somaiya College of Engineering
Somaiya Vidyavihar University
Mumbai, India
amar.vs@somaiya.edu

Chitrang Goyani[1]
Information Technology
K.J. Somaiya College of Engineering
Somaiya Vidyavihar University
Mumbai, India
chitrang.g@somaiya.edu

Kunal Jain[1]
Information Technology
K.J. Somaiya College of Engineering
Somaiya Vidyavihar University
Mumbai, India
kunal28@somaiya.edu

Chinmay Lotankar[1]
Information Technology
K.J. Somaiya College of Engineering
Somaiya Vidyavihar University
Mumbai, India
chinmay.l@somaiya.edu

Sunayana Jadhav[1]
Information Technology
K.J. Somaiya College of Engineering
Somaiya Vidyavihar University
Mumbai, India
sunayanavj@somaiya.edu

***Abstract*— The main motive behind building the cross-platform application is to provide a fast and convenient way to schedule meetings with the help of an automated chatbot. The application also provides a platform to have a pre-discussion before the meeting in the chatroom which is created once a meeting is approved by its members and can send documents related to the meeting stored on their devices or use the inbuilt OCR scanner of the application.**

***Keywords*— Automated meeting scheduling, chatbot, OCR scanner, chat room,**

I. Introduction

In the present advanced age, it is more normal for project groups to work across various workplaces, states, or even numerous nations than in adjoining desk areas. Thus, scheduling meetings can be very tedious and furthermore disturb efficiency. When you spend so much time figuring out when and where to meet, that you forget what you are actually supposed to meet about, that's a problem! Along with this, maintaining the synchronization between teams should become much easier in order to build a sustainable office environment.

The endless back-and-forth communication and steps include – reviewing calendars, finding out the best place to meet, setting up a call-in number, adding the meeting to the calendar, and inviting all the necessary attendees. This process makes your meeting a hassle before it even starts. Hence, scheduling meetings can be quite time consuming and disrupt productivity. In order to reduce the time consumption, increase productivity and maintain the synchronization between the teams we came up with an application.

The basic purpose of this cross-platform application is to provide a common ground for working professionals to schedule meetings on the go using an automated system that is smart enough to schedule meeting reminders and set up an approval system for the officials conducting the meet. But that's not it, along with this we aim to include features necessary for meetings such as an attachments system along with a scanner enabled with OCR to send necessary information before the meeting is conducted. For maintaining the security, we have plan to include a bio-metric system that authenticates users belonging to a particular firm.

II. Literature Review

[1] Personalized Calendar Assistant (PCalM) is a tool to coordinate the interactions (both user to user and user to the tool agent) and to schedule events on the user's calendar. [7] PCalM is designed to build the user's trust in its capabilities so that the user will delegate significant responsibility: entirely transferring to its routine scheduling tasks.

[7] The functional goals of PCalM are to increase autonomy through use: as the user becomes more confident with the agent's ability, the agent can retain more decision-making responsibility, to have the agent learn when to involve the user in the decision as autonomy increases, to have the system

learn user preferences through a combination of passive learning and advice-taking.

The critical components in the methodology received are the formation of a process framework that catches potential co-operations with clients and different specialists, learning technology to catch the client's inclinations, and prudence to empower direct guidance by the client at different degrees of reflection. As the framework improves its model of the client after some time, dependence on client collaboration will diminish.

A process framework captures the key decision points in calendar management and allows the user to make choices or give advice within this clearly defined framework. These choices range from selecting an alternative process, such as negotiating or relaxing.

[2] As mentioned in the paper, the Calender.help works on a 3-tier architecture. The first tier is for automation of microtasks and relies on ML and NLP. It helps in automation of tasks such as gathering relevant information from email conversations such as meeting duration and time slots.

The second-tier deals with manual microtask execution, if the application fails to identify if an incoming email is related to an existing meeting then a worker is asked to classify and determine correct interpretation of the information.

The third tier is for the execution of manual macro tasks where workers are asked to determine the best next course of action for the meeting and execute it. This includes actions such as sending a message to the participant asking for more details or sending/cancelling/updating a meeting invitation.

[4] In the Distributed Multi-agent Meeting scheduler paper, it states that they have used distributed multi agent architecture in which every person is an agent. The agents on behalf of the users autonomously and automatically assist to book meetings. In this system, the agents on behalf of the associated users help to find a solution that satisfies the user's meeting requirements and their preferences. The project has a automated meeting scheduler architecture that consists of various agents that on behalf of the users are used to manage, negotiated and schedule tasks, meetings, events and appointments. The system is client server based. The system is able to operate in online as well as in offline mode. In online mode host and guest agent are used. The host agent uses FCFS (First Come First Served) and HR (High Rank) strategies to request a meeting and once the meeting request is received the guest agent will use the same FCFS and HR strategies to handle the respective. Whereas in offline mode a single agent is used to perform the FCFS and HR strategies in its local scheduler until the system, is online again. Once, the system is online, the single agent is changed to two agents i.e. the Host and the guest agent. The prototype in this paper is implemented using Java. The paper further promises the incorporation of Machine Learning Techniques for better user experience.

III. Proposed Technology

Technology is ever changing and one must ensure to go hand in hand with the latest technology. For making the application available to each and every device available over various operating systems such as Android and iOS; the two most important OS platforms in the world right now, we decided to work on Flutter and build the entire application using Flutter SDKs and working on Dark language gives us to create a cross platform application. Helping people to automate a meeting with the help of a chatbot where each user shares its weekly free allotments that can be used to schedule a meet with other members is the basic idea of “Meeting Scheduler” app. Following modules are a minimum requirement on our application:

- User registration and login

Every user new to the application will have to register on the application once. The application is more towards a particular organization oriented so the user must provide details in accordance to the organization the user belongs to. The initial page of the application will have a Register and a Login button. Google sign-in will also be available wherein the email need not be verified. If the user registers using email, the email will be verified by an automatic mail sent to the user and user verifying the account manually. The user once verified can then login using email/username and password.

Responsibilities	Give an interface to the user to

	register so that the user can enter the system for the first time.
Constraints	The user should have a functioning(active) email address.
Composition	Frontend: Flutter SDK (dart programming language), Backend: Firebase Database, and Firebase Authentication.
Interaction	The user creates the account and enters the system.
Resources	Developer, Project Analyst, Software Tester

Table I

- Manage/Edit Schedule

Every user joining the application can manage or edit their previously stored schedule for the week. The user will have to provide a schedule of their own to schedule a meeting through the application. This schedule of every user is stored in the firebase real-time database and is used by the chatbot to prepare schedules using the algorithm. The manage/edit schedule page has dates represented for the next 7 dates from the current date and the user can allot hours where the user may be free. This 24hr x 7days schedule of the user is vital to create meetings wherein each user wants to have a meet with, the chatbot tries to find a common free slot among the selected members and provides a basis to schedule a meet.

Responsibilities	Give an interface to the user to view and update their account profile.
Constraints	The user should not be ambiguous and well authenticated. A session should be present within the app.
Composition	Frontend: Flutter SDK (dart programming language), Backend: Firebase Database, and Firebase Authentication.
Interaction	The user views the information provided which is saved into the database, the user can update the same as well.
Resources	Developer, Software Tester.

Table II

- Chatbot

Chatbot is an automated program that helps in creating meetings with a user by asking simple questions. The chatbot asks the user to search for other users to have a meeting with. The chatbot then asks the user on what day from the next 7 days the user wants to have a meeting on. After the desired output, the chatbot then runs the algorithm to find a common free slot among every member of the selected members by the user. Upon doing so, the user then provides a subject and a description in short about the meeting. If no common free slots are available the chatbot provides an error and asks to re-enter a new day of meeting. Once the entire process is completed, a meeting request is sent to the meeting members selected by the user.

Responsibilities	Give an interface to the user to select members, the time slots for the meeting, and provide a general purpose for the meeting.
Constraints	There should exist one free time slot among all the members.
Composition	Frontend: Flutter SDK (dart programming language), Backend: Firebase Database.
Interaction	The user selects the members and then views the available time slots, then selects the time slot for the meeting, and provides a document, if it exists.

Resources	Developer, Software Tester.

Table III

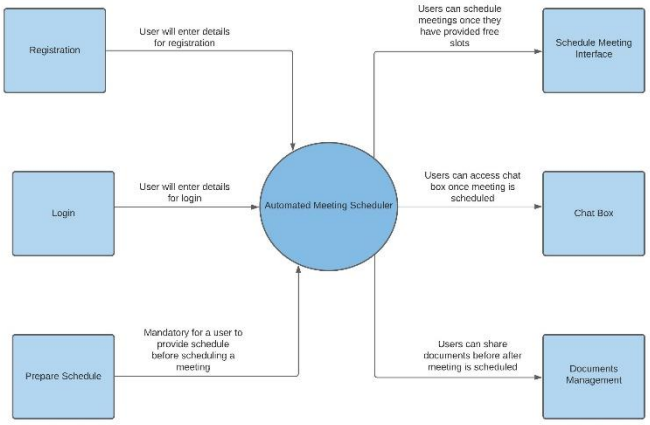
- Chat Room

Once the user initiates a meeting by selecting the members of the meeting, the members need to approve the meeting by accepting the meeting request. Upon doing so, once all the members approve the meeting, a chatroom is created for the same meeting. This chatroom is useful to send documents related to the meeting, setting meeting goals, providing various details related to the meeting.

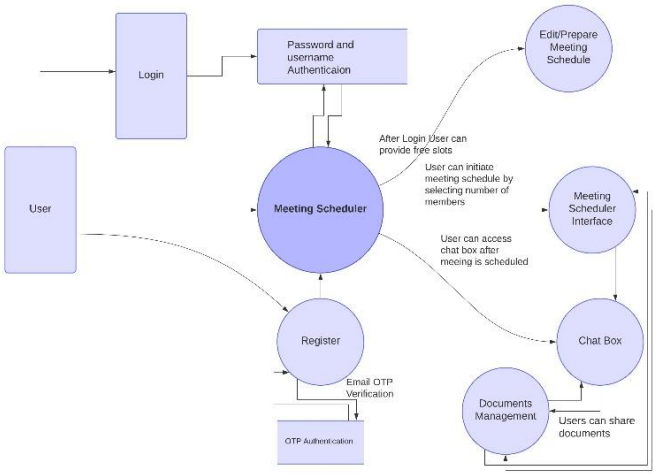
Responsibilities	Give an interface to the user to discuss the meeting details and send documents if any
Constraints	The meeting should be approved and confirmed.
Composition	Frontend: Flutter SDK (dart programming language), Backend: Firebase Database, and Firebase Authentication
Interaction	The user can discuss the meeting with the others, send documents related to the meeting.
Resources	Developer, Software Tester.

Table IV

The chosen system architecture for this project is Model View Control. [9]A Model View Controller pattern is made up of the following three parts: Model - The lowest level of the pattern which is responsible for maintaining data; View - This is responsible for displaying all or a portion of the data to the user; Controller - Software Code that controls the interactions between the Model and View, it also isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. This kind of architecture was decided to keep the project simple yet efficient. Since it can provide flexibility to the application's GUI based on the requests received by the user this would be an apt form of system architecture for this project. The User can start a new discussion in the main community, reply to previous discussions, request access to message in a public community, and update his profile.



Data Flow Diagram Level 0



Data Flow Diagram Level 1

On starting the application for the first time, the user will be asked to register. Having two ways of registration namely Google sign-in and Email sign-in. Upon doing the Google sign-in, the user will provide the Google credentials so the user won't have to verify the account for the same. Upon Email sign-in, a verification link is sent to the user's email address and only after verifying the user can use the

application. After entering the application, there are three tabs - Upcoming meetings, Pending meetings and chatroom. The + sign on the page helps to initiate a new meeting with the chatbot. Once a meeting is initiated, details of the meeting are seen in the pending meetings tab until the members of the meeting approve the meeting. Once the approval of all the members is received, the meeting details are shifted to the upcoming meetings tab and a chatroom for the same is created with all the members in the chatroom tab. The kebab menu present at the top right of the screen has options such as edit/manage schedule, my profile, contact us and help tabs. Under the edit/manage schedule, the user can change the free slots of the week. For first time users, they need to provide the free slots of the week first in order to initiate meetings. Under the "my profile" menu, the user can edit details such as name, phone number.

Use Case Diagram

The major components of the application's UI are:

- Registration page: The user needs to provide details such as Full name, Username, Email, Password, Organisation name, Gender, Phone number.
- Login page: The user can login in the app using the email/username and Password. Logging in to the application is possible only one the email is verified.
- Upcoming meetings: Provides the details of the meetings that have been approved by it's members.
- Schedule a meet icon: This triggers the initialisation of scheduling a meeting with the help of chatbot.
- Pending meetings: This tab provides details about the meetings that have been initialised but yet to be confirmed by the members.
- Chat-Room: Provides group chats of meetings that are approved and present in the Upcoming meetings tab.
- Edit/Manage schedule: Present in the Kebab Menu, this helps the user to make changes in the weekly schedule of the user and provide free slots.
- My Profile: Personal details can be viewed by the user.

IV. IMPLEMENTATION

- **Authentication**

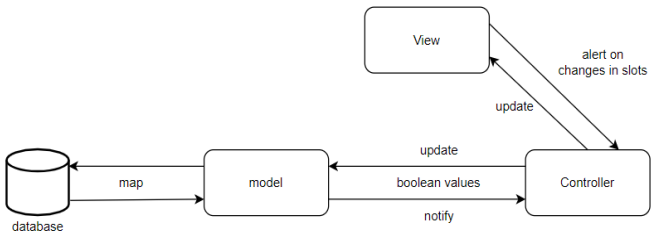
Users are authenticated/identified with the help of the email address and password provided by the user. Authentication is implemented using the Firebase Authentication API. Users can either authenticate using email and password or by Google Authentication provided by the Firebase Authentication. The input fields provided on the client-side (Frontend) and the validation (form-validation) on the same ensures accurate data is provided to the Firebase API to carry authentication. In case of authentication failure due to a number of reasons, the application handles the error that causes the failure and prompts the user about the error using the float messages. (The user needs to provide accurate personal information on either of the sign-up methods, the personal data is securely saved in the database inside the node (randomly generated user id) where the security is ensured using NoSQL rules).

- **Prepare/Manage Schedule**

Preparing the Schedule: The user needs to provide his/her schedule for the week (which can be altered

at any point in the future). The page contains 7 sections for each day of the week, and each section has 24 time slots for the entire day (worst case). Selecting a particular slot for a particular section implies that the user is free to attend a meeting for that slot and on that day of the week. The page is mainly UI rendering until the user clicks the "Submit Schedule" button.

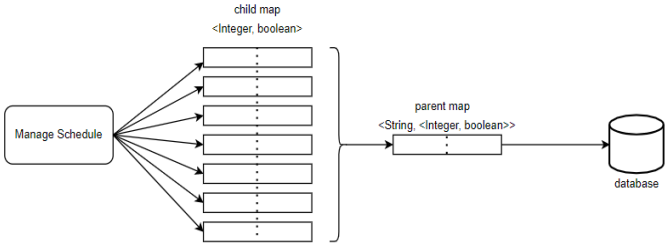
On clicking the "Submit Schedule" button, the boolean value for each slot in each section is saved in a map (one map each for each day in the week) and these maps are then saved in the parent map. The child map has the key-value identifier (<Integer, boolean>) and the parent map has the key-value identifier (<String, Map<Integer,boolean>>). The parent map is then iterated 7 times through the keys and each child map is iterated 24 times through each time slot while simultaneously sending the data structure on the cloud database.



MVC architecture for managing the schedule

Viewing/Editing the Schedule: The user can view/edit the schedule provided by him/her. The same page is rendered as above for implementing this functionality, but first the schedule is fetched from the cloud database (if provided initially), and then the UI is rendered according to the schedule fetched for the user in the system. The same child maps are used to store the incoming boolean values from the cloud database and render the UI state according to the value for the day and the time slot.

The time complexity to perform both the above



Data Structure to store the entire schedule on database

functionalities is O(D*S), where 'D' is the number of working days (7 in worst-case) and 'S' is the number of working hours per day (24 in the worst case). (A constant and stable network connectivity is necessary to perform the following functionalities).

- **Requesting a meet**

Selecting Members for the meeting: The scheduling interface provides a search bar for the users to search the members for the meeting. The user can select/deselect a particular member from the list using the simple interface provided in the application. The user then needs to confirm his/her selections to proceed to further steps to schedule the meeting. The application uses this set of members to make calculations further in application. The client basically maintains a set of lists, to store both the names of the members and their user identification respectively. The client makes use of the lists to send notifications about the meet, update the database regularly regarding the meeting, and so on.

Providing the subject for the meeting: The scheduling interface provides an input field to the user to enter the subject for the meet he/she is intending to schedule. This subject is a String that is stored in the database and hence serves as the reference for the meeting on the client-side of the other members of the meeting and provides the idea to other members about the intention for the meeting. The Subject is stored as a String and the user must provide one in order to proceed further to request the meeting.

Selecting the day for meeting: The scheduling interface provides a drop-box for the user to select the day for the meeting. As the user selects the day, it is stored as String on the client-side of the application, and it is used to fetch the schedule of each member of the meeting from the database. The Sets store the schedule of each member, and therefore, the intersection of these sets provides the common free-slots amongst all the members of the meeting. The algorithm initially stores the schedule for the first member and then progressively finds the intersection with the schedule of the next member and then updates the resultant common free slots.

Selecting the Time-slot for the meeting: The algorithm returns a list of common free time slots which is used to render the UI for the user to select the time slots for the meeting. The user needs to select the time slots such that there does not exist any gap amongst the slots selected. An algorithm ensures that the user selects the slots according to the constraints. The algorithm sorts the list of common free time slots using the most optimal sorting technique, and then the sorted list is iterated over to check if there exists any gap. The error is handled if the user selects slots with gaps within.

Confirming the meeting: The user can click on the "Confirm meeting" button after selecting the time slots for the meeting. The application stores all the data regarding the meeting that it has gathered on the client-side (frontend) to the database. The application after making sure that the entry has been made in the database, sends notification regarding the meeting to all the members of the meeting using their latest device TokenID stored in the database.

- **Chatbox**

Messaging System: The application provides an interface for the user to communicate with other members of the meeting. The messages entered by the users are stored as String and as the user clicks the "Send" button, the messages are securely stored in the Firebase Cloud Firestore, which allows real-time rendering of the user interface on the devices of all other members that are part of the conversation. The client-side code handles the user interface rendering for both the incoming and outgoing text messages.

Document Attachment: The system makes use of flutter plugins to access files that are present in the device. The application asks for the permission of the user the first time he/she makes use of this functionality. The user can select one or multiple files to send as attachments in one go. The files sent as attachments are stored in the Firebase Storage bucket, and the storage token is stored in the database and the attachment history is stored in the Firebase Cloud Firestore. This ensures that the files and the chat are readily available to the user at all times.

- **Biometric Authentication**

The biometric authentication functionality implemented in our application is sourced from an inbuilt flutter package named "local auth". This package enables the application to use the local biometrics setup on device and integrate them in the application. The available biometrics type are Iris, Face ID and Fingerprint Authentication.

In our application we have setup the biometrics to first detect the type of hardware present to support one of the 3 methods, if hardware is available on the device a user is asked to unlock the application using that particular biometric type. In-case a user has not setup a biometric locally we have also integrated a prompt for the user to setup biometrics on their device.

V. Result Analysis

The architecture used in the Calender.help is oriented towards the agent potentially learning user behaviour patterns overtime and focuses on automation of the microtasks. One potential pitfall that Calender.help faces is its service module only limited to emails. In this current generation of technology, chat interfaces are used for interaction more than the traditional way of emailing and hence our proposed methodology focuses on providing a solution that goes hand in hand with technology used in the current generation. However, our methodology is not focused on creating a module that would potentially learn user behaviour patterns and predict time slots. But instead, it focuses on simply automating the task of scheduling meetings in order to reduce the time spent in textual communication to setup the meeting.

The Personalized Calendar Assistant is an agent to handle several interactions which include, firstly, user to user and secondly, user to the agent interactions. Our system provides all the information and key data required to request/schedule an event, while the confirmation of the event rests in the hands of the users who are involved in the same. The involvement of users at all times ensures no particular events remain unchecked and an accurate notification and chat system ensures smooth interaction (both between user and agent and amongst users).

Our project follows the first come first served methodology for scheduling meeting instead of the host agent using FCFS and HR strategies for scheduling meetings for the purpose of fast and easy scheduling. In our project the meeting is allowed to be scheduled in online mode only, whereas in the distributed multi agent meeting project meeting is scheduled in both online and offline which is a drawback because there's a possibility of two users (one in online and one in offline mode) selecting the same time slot and the user which has a continuous internet connection will be the one to get the meeting scheduled.

The prototype developed by the distributed multi agent meeting project uses java while the application developed by us using flutter (dart) which provides hot reload, cross platform, performance, fast development and fast prototyping.

VI. Conclusions

This paper scrutinizes various existing techniques that provide the feature of scheduling meeting. It compares the existing application in the similar domain with this application and states the features that standout from other applications. The main motive of the study is to provide a fast and convenient way to schedule meetings with the help of an automated chatbot. Although, the idea of scheduling meetings is not new but with the technological advancement there are still less options for scheduling meeting especially through mobile phones. The paper tries to incorporate various modern technologies such as Google login & signup and authentication using E-mail. The application provides a simple, speed and cross platform feature that will interest everyone. The application aims to provides a platform to have a pre-discussion before the meeting in the chatroom which is created once a meeting is approved by its members and can send documents related to the meeting stored on their devices or use the inbuilt OCR scanner of the application. To sum up, Scheduling meeting can be quite time consuming and disrupt the productivity of an individual which is very valuable considering today's fast developing world. When you spend so much time figuring out when and where to meet, that you forget what you are actually supposed to meet about, that’s a problem, hence, we provide a solution to this through our modern application and paper.

Acknowledgment

Authors would like to acknowledge the management of K.J. Somaiya College of Engineering, for supporting this research work.

References

[1] Pauline M. Berry, Melinda Gervasio, Tomás E. Uribe, Karen Myers, and Ken Nitz, “A Personalized Calendar Assistant”, California, USA, January 2005

[2] Justin Cranshaw, Emad Elwany ,Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, Andrés Monroy-Hernández, “Calendar.help: Designing a Workflow-Based Scheduling Agent with Humans in the Loop”, Microsoft Research, Redmond, WA, USA, January 2007

[3] Andrés Monroy-Hernández and Justin Cranshaw, "Virtual Scheduling Assistant (Microsoft)", Microsoft Research, Redmond, WA, USA, April 2016

[4] Elhadi Shakshuki, Hsiang-Hwa Koo, Darcy Benoit, Daniel Silver, "A distributed multi-agent meeting scheduler", Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, Canada B4P 2R6, April 2007

[5] Chanan Glezer, Surya B. Yadav, "A Conceptual Model of an IntelligentMeetingScheduler", Journal of Organizational Computing and Electronic Commerce, Department of Information Systems and Quantitative Sciences, Texas Tech University, USA, September 1999

[6] Leonardo Garrido-Luna(1), Katia Sycara(2), "owards a Totally Distributed Meeting Scheduling System", KI-96: Advances in Artificial Intelligence, 20th Annual German Conference on Artificial Intelligence, Dresden, Germany, September 1996

[7] https://apps.dtic.mil/sti/pdfs/ADA460158.pdf

[8] https://www.impactplus.com/blog/meeting-schedule-apps-tools

[9] https://www.tutorialspoint.com/struts_2/basic_mvc_architecture.htm