TimeScape

The solution to scheduling group meetings

Nick Hess
Computer Science
Virginia Tech
Blacksburg VA, United States
hessnt30@vt.edu

Tim Vadney
Computer Science
Virginia Tech
Blacksburg VA, United States
vtim@vt.edu

Thomas Tran
Computer Science
Virginia Tech
Blacksburg VA, United States
thomastran@vt.edu

Gio Romero
Computer Science
Virginia Tech
Blacksburg VA, United States
gioromeroruiz@vt.edu

Yasir Hassan
Computer Science
Virginia Tech
Blacksburg, VA, United States
yasirh@vt.edu

ABSTRACT

Our project idea is called TimeScape. When trying to meet up with friends or other people, it can often be difficult to determine optimal times that work for everyone. Popular existing tools merely serve as a way to block out specific time slots without giving users many options to communicate their own availability, making it difficult for project managers, club leaders, and event organizers to ascertain the necessary information.

As such, our project aims to take in the available times of all members, and visually display the times that align with the most number of people (e.g. time slot X has ¾ members free, time slot Y has 4/4 members free, etc.). We aim to provide a means to easily visualize optimal time slots in an easily digestible format will make it more efficient to help arrange meetups.

1 Introduction

When scheduling team meetings, miscommunication or lack of communication can often lead to teams/groups not being aware of the best times to meet. This problem can lead to the following problems arising:

- People missing important meetings
- Deadlines being missed
- Meeting at tedious hours

Furthermore, the people who schedule these meetings are tasked with understanding everyone's availability in order to determine optimal time slots where the most people can meet. Such a task can often be laborious, requiring a plethora of messages back and forth on different group messaging platforms—Slack, Discord, email, etc.—to keep track of and plan accordingly.

TimeScape aims to overcome these problems by implementing a web application that anyone can easily use. With its primary features aiming to serve both organizers and participants alike, the goal is to provide a seamless experience for managing and scheduling events, meetings, and gatherings.

1. Unified platform: TimeScape centralizes the process of scheduling meetings, reducing the need for cross-platform communication. It enables organizers to easily gather availability information and streamline the scheduling process.
2. Visual representation: The tool visually displays time slots that accommodate the most participants, minimizing manual effort and confusion. This helps organizers quickly identify and select optimal meeting times, boosting team productivity
3. Seamless experience: TimeScape is designed to be intuitive and user friendly, reducing the learning curve for both organizers and participants. It integrates features from existing tools while offering a novel solution through its scheduling algorithm.

By addressing these problems, TimeScape aims to simplify scheduling for software engineering teams and other

groups, ultimately improving communication. coordination, and productivity.

2 Motivating Example

It is common for organizations to use tools such as Microsoft Outlook to block out specific time ranges on specific dates and invite attendees to the meetings. However, asking attendees for feedback and ascertaining their availability is another issue altogether. To solve this issue, employees can often communicate with each other on platforms such as Slack. This poses a challenge in that numerous threads can be created, messages can fill up quickly and prevent past ones from being visible, and the organizer has to sift through each one and calculate which time ranges would work best manually. TimeScape aims to solve this issue as described further in the report, but here is an example of its strength in action via a scenario.

2.1 Scenario

Setting: Work

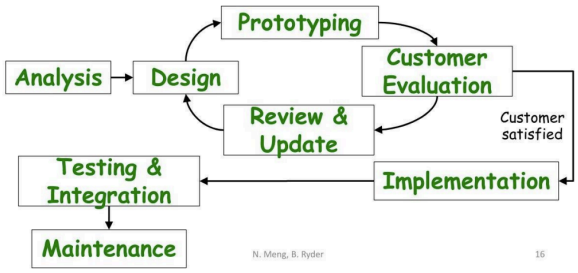
Bill is a project manager at a medium-sized software company. He wishes to schedule a department-wide meeting to get up-to-date on the status of all the teams he's overseeing. In the past, Bill would ask in the department's Slack channel for everyone's availability, and would get flooded with a multitude of messages. Furthermore, he would then have to sift through each one and manually determine a good meeting time to accommodate the most amount of people. As such, he uses TimeScape to visualize the availability windows of 15-20 employees to determine the most optimal meeting time. Ideally, he wishes to find a time slot where no employees are excluded from this meeting. Bill sends out the invitation link to the schedule, awaits his employees' responses, and selects an optimal meeting time.

Brad is a software engineer working on a team overseen by Bill. Brad has multiple meetings during the day—within his team and with potential clients. Additionally, his schedule changes often, so he does not have a static window of availability week-to-week. Brad receives the invitation link from Bill to the TimeScape schedule and inputs his available time windows for specific days, changing them as needed as his obligations adjust throughout the week. In the past, he would have to keep messaging in the department's Slack channel to notify Bill, but he now has a centralized place to do it.

As aforementioned, integrating TimeScape into existing business workflows to schedule events such as group check-ins, Scrums, etc. can be viewed as seamless and simple. Rather than wasting time going back and forth between Slack and Outlook, event organizers can view participant feedback in real-time, and use TimeScape's algorithm to determine the best meeting time automatically. As such, software engineering teams following things like the Agile methodology or the Prototyping model (in the case of scheduling meetings with customers to get feedback) can save time and energy when looking to collaborate and meet-up with one another and clients alike.

3 Background

Our team, TimeTeam, plans to use the Prototyping model, which is an iterative model. Since we likely won't get very far into implementation of our solution, we feel that the iteration of the prototyping process will best fit our team.



Visualization of Prototyping Process

The Prototyping model will help us easily change our requirements as we receive feedback from customers and users. Utilizing this constant stream of feedback, we can rapidly iterate TimeScape within a short time-frame to help develop a presentable final deliverable.

Our team will use Kanban concepts, such as online tools via the website Trello, in order to visually display all of our tasks that need to be completed. These concepts will help us manage and measure workflow, and limit work in progress.

We will also use key Scrum concepts such as sprints and backlog which will keep us on track to complete our goals. TimeTeam will also meet at least once weekly for a standup meeting to discuss new developments, what each member has done, and problems encountered.

A critical aspect of TimeScape is its algorithm, designed to streamline the process of scheduling meetings. Here are some key concepts of our algorithm.

- Time slot evaluation: The algorithm gathers and processes participant's availability data, dividing it into time slots. It then evaluates each slot to determine the number of participants, available, highlighting those that accommodate the most attendees
- Conflict resolution: The algorithm considers various constraints such as recurring events, time zones, and existing appointments to minimize scheduling conflicts. This ensures optimal time slots align with participants' schedules
- Visual representation: The algorithm's results are displayed in a clear, intuitive interface. Participants can hover over time slots to see how many attendees can make it, allowing for easy identification of suitable meeting times.

This background section provides an understanding of our algorithm project management methodologies, and integration strategies through our key concepts.

4 Related Work

A research paper investigating the methodologies behind scheduling meetings presented its findings on the incorporation of web-based scheduling applications. Oftentimes, failures that prevent successful meetings pertain to relying on a software that does not properly "inform and remind participants about the meeting", and the "lack of selecting a suitable time for most participants…". Both of these failures result in wasted time and unproductive meeting results for all parties involved. Furthermore, the study shows that 46.3% of online meeting scheduling applications and its users want a way to "[generate] a convenient time for most of the participants." 43.2% desire reminders about it through SMS and 23% desire reminders through emails. While there are many existing applications out there with some of these features, the study states that "no application has yet been created to freeze [all the problems]" users experience with these software [2].

Related software engineering tools to our app idea are Google Calendar, when2meet, and Microsoft 365 [1]. What makes TimeScape unique is that we are going to improve these previous software engineering apps by designing an algorithm to determine optimal times between peoples' schedules and be able to visualize optimal times to the users when scheduling meetings with other people.

For example, websites like when2meet are not visually appealing and it can be very difficult to gain a grasp of when to meet at a certain time [1]. Using TimeScape can facilitate this process of scheduling meetings with multiple people by providing users a smooth experience and a visually appealing UI to help determine times to meet. A user organizing a meeting can hover over different time slots on the calendar interface and TimeScape will highlight compatible time slots based on the schedules of all participants.

With mainstream calendars like Google Calendar and Microsoft 365, TimeScape will have its own calendar system that will improve upon Google Calendar and Microsoft 365 to help enable scheduling and notifications for meetings [1].

In summary, TimeScape aims to solve the challenges presented by the study. Existing tools such as Google Calendar and Microsoft Outlook serve as a simple way to schedule meetings and invite attendees, while when2meet can help with getting participant feedback. However, there is not one tool that can do everything—an appealing and understandable UI for meeting times, manage and schedule multiple meetings, invite participants and get feedback, and have an algorithm that can calculate the optimal time range(s) on specific date(s)—that is available to consumers. More specifically, the algorithmic aspect of TimeScape is what sets it apart from existing tools, in addition to its front-end foundation with the UI.

5 Implementation

TimeScape's implementation uses Creational Patterns for code flexibility and reusability, while an Event-Driven architecture ensures real-time responsiveness to user actions and scheduling complexities. The mock user interface shows how we would handle event creation and catered time recommendations. Our testing strategies include unit and integration testing for robust functionality and error handling.

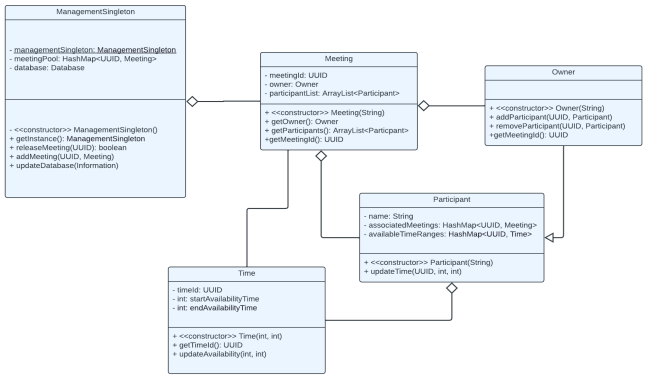
5.1 Low-Level Design

We chose to go with a Creational Pattern since this pattern focuses on object creation, flexibility, and code reuse. This low-level design pattern aligns nicely with TimeScape as it offers flexibility and reusability in building out the web application.

For instance, our Meeting class is an **Abstract Factory** - a one stop shop for setup. Since the web application is aimed at users being able to create meetings on an on-demand basis, being able to build out all of the necessary classes that's involved in such a task makes it reusable and convenient to do from a back-end perspective

Meetings that are archived or have passed could be viewed as no longer necessary from a maintenance perspective—keeping track of them in an **Object Pool** when the meeting hasn't happened, and releasing it afterwards makes our application flexible and efficient

For users looking to quickly boot up meetings with/without configuration, having a **Prototype** or template instance to build off of creates a nice abstraction to the application
Since the web application aims to serve multiple users simultaneously, using a **Singleton** for the server or manager class that communicates all the updates, data, and information to things like the database is necessary—having one instance prevents the need for solving concurrency-related issues.



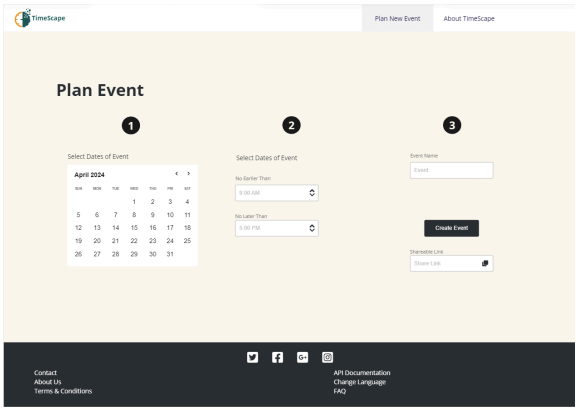
UML Class Diagram

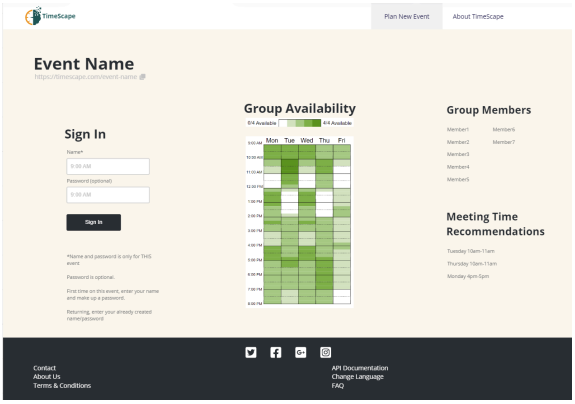
5.2 High-Level Design

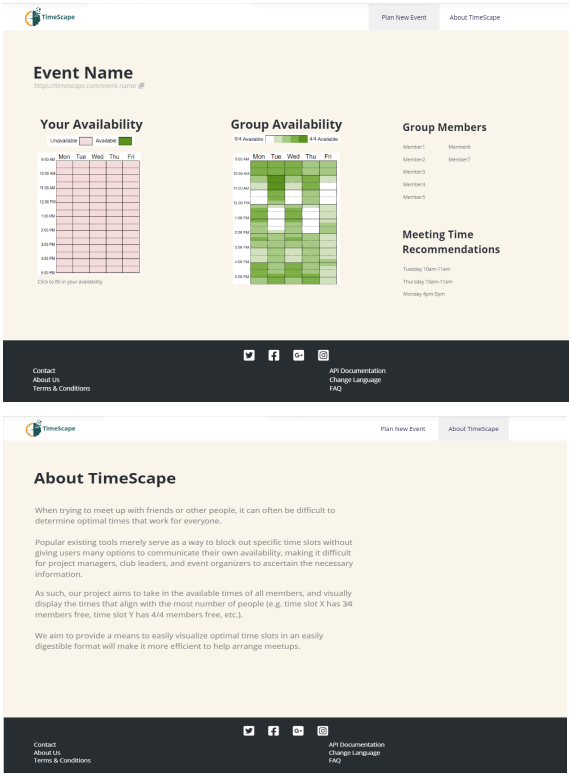
Our project focuses on scheduling a meeting and deciding the best meeting time for a group of people. Various events occur at the same time, whether that be users indicating their availability, changes in availability, scheduling conflicts, and new additions or removals from the meeting team. An **Event-Driven** architecture will allow the system to react to these events in real time and update the list of best meeting times and the overall meeting schedule accordingly. Requirements for meeting scheduling may evolve and using this pattern allows the project to be scalable and flexible. It is the ideal choice for building a system that efficiently schedules meetings for large groups of people.

This pattern promotes the production, detection, consumption of, and reaction to events. Some of the benefits of Event-Driven architecture are anonymous handlers of events, and support for reuse and evolution, with new features being easy to add. One disadvantage of Event-Driven architecture is that components have no control over the order of execution.

5.3 Mock User Interface







TimeScape Mock User Interface

The first frame of our wireframe is the landing page where users can create a TimeScape meeting event table. They choose the times and dates of the availability they wish to plan the event in, give the event a name, then create the event. Then they are given the option to copy a shareable link to the event for meeting members to join and add their availability. When a member joins the event, they will be prompted to sign in to this event. Upon signing in they input their availability and that will update the Group Availability table. The section titled “Meeting Time Recommendations” is a section provided by us which will provide the event with times and dates of most availability. We also have an About TimeScape tab that allows users who are curious to understand more about TimeScape.

5.4 **Testing**

Given the time, we would have chosen to use **Unit Testing** on our project. We would use the JUnit Framework for testing since our code is written in Java. We would test all the classes and methods and assert that the output is what we expect. Just us developers would be doing the testing to ensure that the code is written correctly.

Once we know that our classes and methods are working correctly, we would move to Black Box testing—to ensure all of the components of the application works in conjunction, this would also involve integration testing as well. This would ensure that our system as a whole works expectedly. With preconditions and scenarios, we would be confident that our system flows smoothly with proper error handling, and no bugs.

Test ID	Description	Expected Results	Actual Results
CreateNewMeeting (Thomas) ~~Functional~~	Preconditions: - Signed in with valid account Steps: - Input timeline for event - Input date for event - Input name for event	Message shown—new meeting created with the details	n/a
SendParticipantInvite (Thomas) ~~Integration~~	Preconditions: - Signed in with valid account - Owner of the meeting Steps: - Input participant email	Pop-up shown—invite emailed to participant	n/a

Snippet of Black Box Testing Plan

6 **Deployment Plan**

Since the goal of TimeScape is to serve as a reliable and resilient platform—a substitute for traditional scheduling applications such as Google Calendar and Microsoft Outlook—we would employ the Blue-Green Deployment strategy. Doing this way would ensure that any new features could go to the blue environment (staging) for testing, while the stable green environment (production) could continue to serve clients. Once we ensure that the blue environment passes our tests, it would replace the green.

As such, while we would follow the principles of continuous integration, we wouldn’t necessarily have much focus on continuous delivery. That is, with the separation of blue-green environments, there is only one theoretically stable version to serve to clients. As such, new features, changes, and bug fixes would have to go through the entire process of testing (part of continuous integration) in the blue environment first, before being given the go ahead to replace the green.

With having the blue-green approach, there would be two separate branches within our code repository for both respectively. Continuous integration principles—having the staging environment, application build, (integration, chaos engineering, etc.) testing, and integration itself—would be present in the blue branch using DevOps tools such as GitLab and Jenkins. On the other hand, the application built

from the green branch would be used to handle the live application being served to customers. This would probably require running it on Docker containers managed by Kubernetes, and using a platform such as AWS to deliver the content across the internet. When we decide to release a new version (via the blue branch), the containers would simply be re-built from this code and served to clients—there would be a little bit of downtime as the process of rebuilding takes place.

Conveniently, maintainability can be done on the blue branch as well. In terms of standard practices:

- At least 2 people should perform code reviews on pull requests before merging
- Tests should be re-ran on the blue branch before a merge happens
- After each release, we would go through and refactor code as-needed in the floss refactoring style
- After multiple releases (3-5), we would go through and see if anything needs to be restructured
- About once a month during off-peak hours, take the application offline (built from the green branch) for roughly half an hour to re-run tests, discover any bugs, and review application statistics/logs

As such, our style for maintainability would be more geared towards corrective and preventive maintenance—to not focus on large, drastic improvements, but strive to maintain and deliver a stable, working system with minor improvements every now and then.

7 Discussion

While our foundation for the project is strong in terms of our understanding of the problems and roadmap, we didn't have enough time to begin implementing TimeScape as a web application. As such, we will discuss its current limitations as well as possible plans for future work.

7.1 Limitations

The following contains details about limitations with our current planning and vision for TimeScape.

Scheduling Constraints:

- May not be possible to find a time that suits everybody, the algorithm will focus on the majority, potentially leaving certain individuals out of consideration
- Must consider various complex constraints such as time zones, recurring events, availability preferences, and conflicts with existing appointments.

User Trust:

- Concerns about the algorithm's fairness, transparency, and reliability may hinder building and maintaining user trust
- May cause adoption challenges among users accustomed to traditional and familiar scheduling methods

Optimization:

- TimeScape's algorithm may prioritize certain aspects that do not align with the user's actual priority
- TimeScape's effectiveness relies heavily on the accuracy and reliability of input data. Inaccurate or outdated data could lead to inefficient scheduling decisions.

7.2 Future Work

Since we already have the design in place with the wireframe, as well as how our front-end/back-end would interact with each other in our sequence diagrams, the next step is to actually build out working prototypes of TimeScape.

Additional features could include:

- Linking TimeScape to existing calendar tools (Microsoft Outlook, Google Calendar, etc.) to make it easier for users to integrate it into their workflows
- Having built-in messaging and/or notes for users to elaborate upon their availability times if needed
- Allowing for archiving old meetings for book-keeping and reference

8 Conclusion

Scheduling group meetings is often a challenging task, leading to several key issues:

- Diverse schedules: Team members' varied responsibilities can lead to scheduling conflicts, resulting in missed meetings or inconvenient times.
- Lack of coordination: Organizers must gather availability information from multiple platforms, such as Slack, Discord, and email. This manual process can lead to confusion and miscommunication.
- Unproductive meetings: Meetings without key participants or with poor planning due to time constraints can lead to wasted time and project delays.

TimeScape aims to address these issues by offering a comprehensive solution:

- Centralized Platform: TimeScape reduces the need for cross-platform communication by providing a single space for organizers and participants to manage meeting scheduling.
- Visual Representation: The tool visually displays time slots that accommodate the most participants, minimizing manual effort and confusion, and making scheduling quicker and more efficient.
- Integration with Existing Tools: TimeScape aims to integrate seamlessly with tools like Google Calendar and Microsoft Outlook, allowing users to incorporate it into their existing workflows and reducing manual data entry.

Our project work completed consisted of our team designing and planning the architecture of TimeScape through wireframing, and use cases. We adopted an iterative approach, incorporating feedback through the Prototyping model to refine the solution. Our project management consisted of concepts such as Kanban and Scrum which were incorporated to manage workflow and track tasks. We also used Blue-Green deployment to ensure stability by testing new features in a staging environment before rolling them out to production.

Things we learned during the time of our project were the importance of using project management tools like Trello alongside processes to track tasks, progress, and status for all group members. In addition we developed a foundation for the product through visualization with sequence diagrams (paired with use cases) and wireframing to better understand action flows and interactions. Lastly we applied software engineering principles to break down and analyze our project in different components and perspectives.

In summary, TimeScape offers a streamlined solution for scheduling meetings, reducing communication breakdowns and improving team collaboration, ultimately enhancing productivity for software engineering teams and other groups.

REFERENCES

[1] “7 Best Group Scheduling Apps in 2024: Calendly.” Calendly.Com, calendly.com/blog/group-scheduling-app-guide. Accessed 16 Feb. 2024.

[2] Sathsara Thalawattha, Dushyanthi Vidanagama. 2021. A Survey on Web-based Meeting Scheduling Application. KDU Faculty of Computing Student Symposium. *KDU, Sri Lanka. 6 pages.* https://www.researchgate.net/publication/348211673_A_Survey_on_Web-based_Meeting_Scheduling_Application