

Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Automatizálási és Alkalmazott Informatikai Tanszék

Ai Yoshida

Development of a Web Application for Cross-Timezone Meeting Scheduling

SUPERVISOR

Dávid Sik

BUDAPEST, 2025

Table of Contents

Introduction

Recently, remote work and international teams have become more common, especially after COVID-19. Therefore, one big problem arises: it is difficult to schedule meetings among team members who live in different time regions.

I have experienced this problem while I was working remotely as an intern at Japanese company that provides programming educational platform for kids. Usually, I needed to organize meetings among three countries, Hungary, Japan, and one more country like USA or India. Depending on the members who join the meeting, the time zones that I needed to consider were different. Each time I had to go through a troublesome process: firstly, search for each person’s time zone (knowing country name is not enough, since U.S. has several time zones), secondly find available time (usually between 8:00 and 22:00) for each time zone and find overlapping time slots among three or four different time zones, finally, choose several timeslots from the candidates. Especially around period when summertime starts or ends, because of these dynamic changes, organizing meetings

usually causes a lot of confusions and mistakes.

To solve this problem, I have searched several meeting scheduling tools, but most of them do not consider time differences, are made for large business teams or part of paid services and are not suitable for small teams or private use. This result gave me an idea of this project. I thought I can create a web application to solve this problem. Also, I was sure that there are some other people who face the same difficulties.

The purpose of this project is to develop a simple web application for cross time zone meeting scheduling. React is used for Frontend development, while Backend is implemented using Fast API, and SQLite for database. Main functions are to visualize users’ time differences, to provide intuitive timeslot selection implemented by FullCalendar, and sharing meeting via URL.

# 2 Related Work and Technologies

This project was developed using JavaScript, FastAPI, React, FullCalendar, SQLite. For the development environment Visual Studio Code was used. Also, moment.js library was used for time and time zone conversions. These technologies have enough support, widely used and appropriate to build modern single page application. React is suitable to implement dynamic frontend while FastAPI is good to develop simple and fast backend using Python. SQLite is the database, which is light, so it was ideal for small scale project like this, since this is local environment development.

Before start developing the project, I have done several research for existing tools which tried to solve the same problems. As a result, there are some web application to calculate time differences, but most of the time with restrictions.

For example, ⭐️World Clock Meeting Planner from timeranddate.com offers meeting scheduling functions with at most three different locations, but it requires user to set participants’ time zones manually. However, it is often troublesome for users since we cannot usually specify time zone information from country name. For instance, even if the user knows one participant lives in U.S, but it is not enough to schedule meeting since U.S has many time zones inside the country.

While UI of ⭐️”everytimezone.com” is very practical, but to meeting scheduling function is a paid service. Therefore, it is not likely to use by students or individual users.

⭐️Calendly, Google Calendar, and Zoom Scheduler are the popular meeting scheduling application, but they do not have time zone difference visualization functions. Most of the cases, users need to check the differences manually. Also, their main target is business use, so they assume that participants know each other’s schedule or users’ weekdays schedule in assigned for business.

To conclude, these existing meeting scheduling tools needs manual process to check the time differences, and not practical for small teams or individuals. Therefore, goal of my project is to develop simple, free, specializing for time difference adjustment web application meeting scheduling.

# Project Goals and Requirements.

## Project Objectives

The main goal of this project is to develop a local web application which two to three people (everyone lives in the different time zones) can easily schedule meeting.

### Functional Requirements

The application should have these requirements below.

* The user can create an account and can login/logout to the account.
* The unique ID number is automatically assigned to the user when they create an account.
* Each user can choose maximum 10 timeslots which length is 30 minutes fixed by clicking the calendar when they create new meeting scheduling.

.3.3 Non-Functional Requirements

4. System Design and architecture

4.1 Architectural Overview

4.2 Project Structure

5.Implementation

5.1 Interesting Source Code / Pseudocode

6. Evaluation and Results

6.1 Manual Testing

7. Engineering Experiences

8. Outlook and Implementation plan

# Reference

|  |  |
| --- | --- |
| [1] | P. Koopman, „How to Write an Abstract,” október 1997. [Online]. Available: https://users.ece.cmu.edu/~koopman/essays/abstract.html. [Hozzáférés dátuma: 20 október 2015]. |
| [2] | W3C, „HTML, The Web’s Core Language,” [Online]. Available: http://www.w3.org/html/. [Hozzáférés dátuma: 20 október 2015]. |
| [3] | K. Nahtkasztlija, „Az idegen szavak toldalékolása,” június 2009. [Online]. Available: http://www.pcguru.hu/blog/kredenc/az-idegen-szavak-toldalekolasa/5062. |

[**Appendix: Weekly Work Diary Table**](https://docs.google.com/document/d/1jo0eKjlSCmb-cOd95XduxGmvC6YpBYhl/edit#heading=h.chcc967awuwi)

Appendix: Weekly Work Daily Table