Space Scheduler - Planning sheet

A scheduling app by Gavin Ray

This project was created as a capstone project at the end of my Software Development program at WGU. It is a meeting scheduler that you can use to schedule a meeting with me. Once you have scheduled a meeting, the backend will automatically send you an email. There is also a secure dashboard that I can use to view meeting information, send automated emails, print reports, and add notes to meetings.

While this project uses Python (Django) for the backend, and JS/HTML/CSS for the frontend, I also have experience with other languages like Java, C++, and C#. Mostly what I hope you will learn about me is that I love to learn new tools.

This application is a work in progress, as all good projects are. Please continue to visit the live version of this site at https://gavin-ray-group.herokuapp.com/ often to see what I am adding. (Future additions are likely NOT to be included in the public repo of this project for security reasons.)

Table of Contents

Business Problem

The Customer:

The customer for this application is a working professional (myself) that must schedule meetings with lots of people from different organizations and institutions. These meetings are often one-on-one tutoring sessions that happen at different times each week or classes that follow a weekly schedule. But sometimes I have previous students who want help with something on the fly, and I'm generally pretty generous with my time when I can be.

To coordinate with others, I use a variety of platforms. For example, I run a tutoring business on the site Wyzant, communicate with other educators on Slack, and sign up for volunteer shifts at my son's preschool over signup.com. Traditionally I have chosen which third party sites to use based on the preference of my clients. Most of the time this would involve apps like Google Calendar, Slack, Email, or other lesser known sign-up sites. At any one time, I am sure to be using a variety of platforms, which can make coordinating between them difficult.

Business Case:

The proposed scheduling application will consolidate all my scheduling needs into one place, while automating some scheduling activities. It will consist of an online portal that people can use to schedule meeting times with me. This alone will save lots of time because I will no longer have to check multiple sites and apps to find out who I am meeting with and when. Instead, I will only have to log in to the online dashboard to see my upcoming appointments.

Many tasks will also be automated through this site, including verification and cancellation of meeting times. Recording notes and displaying the notes at later times will also be included in the site. The other major function of the site will be to allow me to set my availability so that clients do not overbook me.

Security will be a major factor in the site, which will require authentication to use many features. Because the site will record some user data, securely storing this information in a database will be vitally important.

Fulfillment:

The scheduling application will have two web pages: a Scheduler that can be used by anyone to schedule an appointment with me, and a Dashboard that requires a login and password to access.

The Scheduler will allow a person to input their name, email, and reason for scheduling a meeting with me. Next they will be able to select a duration, date, and start time based on my availability.

The Dashboard will have several sections, each providing functionality of a different sort. The first will be a reports section that allows access to meeting information in CSV format. The second will display, by week, the times that I have registered availability and allow me to add or delete times. The third section will allow me to search through all current meetings to quickly ascertain what my weekly obligations are. It will also allow me to confirm, cancel, and archive meetings with a single button press. Every meeting can have a note attached to it, and both adding and viewing these notes will be done in this section as well.

Security will be provided in several ways. CSRF tokens will be used to prevent cross site forgery and man-in-the-middle attacks. Django's built-in database management classes will provide protection against SQL injection, as well as provide User objects for password-required logins.

Existing Gaps:

The current system used leads to an unacceptable number of schedule mix-ups. Because I have to look for meeting times in several places, across several apps and sometimes in different time zones, I can often get confused about when I have a meeting. I have tried transferring every meeting to a Google Calendar, but this is time consuming and does not prevent errors.

There is currently no way for clients to find out my availability except by asking me directly. This takes time from both my client and myself. The client must ask either over email or while on a call with me. I then have to check my schedule while they wait, and give them a few options at a time while they check them against their own schedule.

I also need a way to record notes about my meetings. Right now, I mostly rely on the communications that I have with clients to remind me of our objectives. This often works, but when our previous communications are not sufficiently detailed, I can enter a meeting unsure and unprepared.

Existing program:

This program will be built from the ground up.

Methodology:

For this project, I will be practicing the RAD development methodology. This will be most appropriate given the limited time available.

Phase 1, Requirements Planning:

This phase was completed in part 1 of my capstone, and involved evaluating the project's scope. There are many functions that I would benefit from including in this project, but the time and resources required would not make them feasible. Instead, I built a prototype of the Scheduler and decided what data would be needed to accomplish all of the site's goals.

Phase 2, User Design:

Rapid Application Development allows us to jump right to development much earlier than other methodologies. In phase 2, I will turn the prototype from phase one into a scheduling app. I will also build the dashboard, all backend logic, and database models. Because these components will not yet be working together, unit testing will be used to ensure the code works.

Phase 3, Rapid Construction:

During this phase, I will put all the pieces together and see how well it works. The main goal in phase 3 will be to ensure that the front end, backend, and database are communicating properly and securely. Iterative development will allow me to make changes to any part of the app throughout this phase.

Phase 4, Cutover:

The final phase will involve launching the application into a production environment. Heroku is a cloud based PaaS that will be used to host the application. Before this can happen, various changes will need to be made to the application to prepare it to go live from the local development environment.

Deliverables:

The deliverables for this project is a full stack web application with two web pages, a database, and an API that connects the two. One of the web pages will be a public facing portal for scheduling appointments. The other will be a private Admin page for managing appointments. Documentation of the application and instructions for installation will also be provided.

I will now break down the features provided by each component.

Scheduler:

- 2 forms, one for adding name, email, and objective. The other for submitting date, time, and duration.
- Displays only times during periods of availability.
- Easy to use on the first visit.

Dashboard:

- Secure: requires username and password to access.
- Uses Ajax with CSRF protection to give connect with the site's api.
- Displays information about meetings organized in an efficient gui.
- Allows updating of meeting information.
- Allows access to both archived and current meeting data in csv format.

API:

- Easy to understand endpoints.
- Accurately directs data to the appropriate place.
- Updates database models.
- Processes data appropriately before sending to the frontend.
- Checks for user permission before allowing access to confidential data.

DB Models:

- Relates to the necessary Sql tables.
- Compatible with both MySql (for development) and Postgresql (for production).
- Helper function added to models will cut down on the amount of code needed in the api.

Implementation:

Source code will be stored on GitHub in a private repo. The website will be hosted on Heroku, a cloud based service. Uploading to Heroku will be done with Git. Security features will be added through Heroku's site. For the duration of this capstone, I will use student dynos. These are free, and I already have a year's worth of them. When the site goes live, I will upgrade the dynos for faster loading times.

Verification:

Unit testing will be done for the API to ensure the site is utilizing the database correctly. The writing of unit tests have been written into the schedule, to be done after the API is built.

The private side of the site will be evaluated by me, as I will be the only one using it. It will not look pretty, which is intentional.

Environment and Costs:

Development will be done using VSCode, Windows 10, GitHub.

Backend development will be written in Python, using the Django framework. Frontend development will be written in JS, HTML, CSS.

The site will be hosted on Heroku. As I mentioned earlier, development will be done using free student dynos.

This project will require 172 human hours. 25 for planning, 86 for development, 30 for testing, and 25 for writing documentation.

Timeline:

	task	Planned start date	Planned completion date	Resources	Dependencies
1	Submit task one	12/8	12/9	Topic Approval Form, Waiver, Course instructor	-
2	Build Front end prototype	12/10	12/11	Coding Environment with JS, HTML, CSS.	1
Milestone 1- End of phase 1.					
3	Build database models	12/11	12/11	Coding Environment, with Django, postgreSQL	1
4	Build frontend.	12/15	12/20	No new resources.	2
5	Build API and test scripts.	12/20	12/23	Pytest framework.	1
Milestone 2- End of phase 2.					
6	Connect front and back ends.	12/25	12/27	No new resources.	3,4,5
7	System test, iteratively squash bugs.	12/27	12/29	No new resources.	6
Milestone 3- End of Secondary Build phase					
8	Launch to Heroku	12/29	1/1	Heroku app / cli	7
9	Write documentation	1/1	1/2	Word processing software.	7
10	Submit to WGU.	1/5	1/5		8,9