

Planner Webapp

Goals - Background

- Calendar: events happening during a given time
- Todo list: tasks to be completed before a given time

- Planner: bridges these 2 data sources for a more centralized experience in planning out a day

Goals - Approach

- Time blocks: scheduled block of time that's associated with the task to be done during that time
- Direct reference to the actual task, so any details and resources associated with the task are available directly in context
- And likewise with events

Screenshots - Login

Welcome to the Planner App

The main screen will show you your events and tasks for the day.

You can press the + button to dedicate a block of time for a certain task.

Clicking on an event or task will give you details about it.

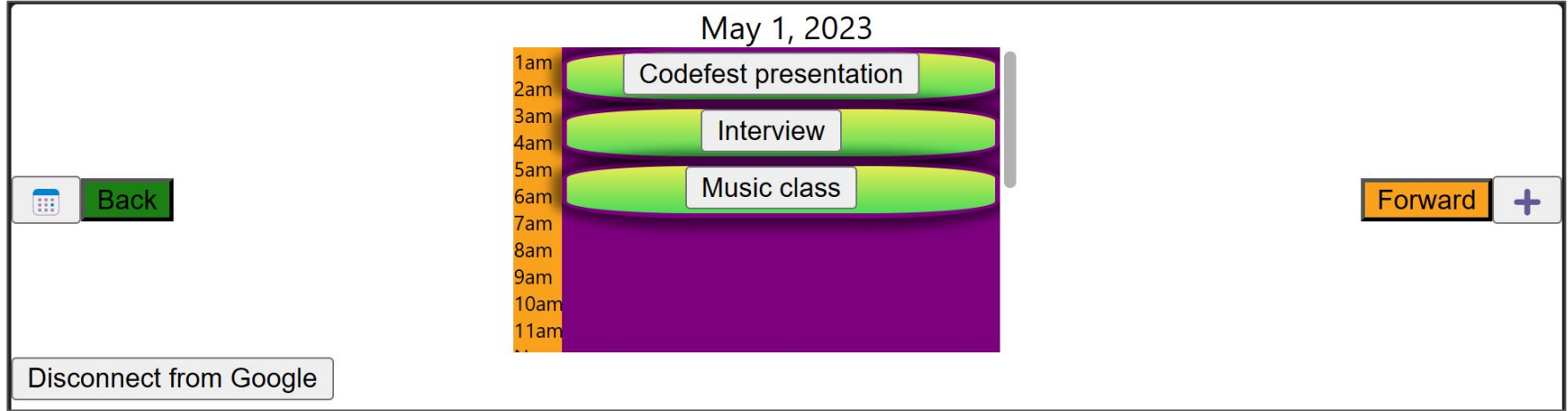
This planner uses your Google Calendar and Tasks to identify your events and tasks. Your data never leaves your neither your browser nor your Google account.

[Connect to Google](#)

Screenshots - Date selection

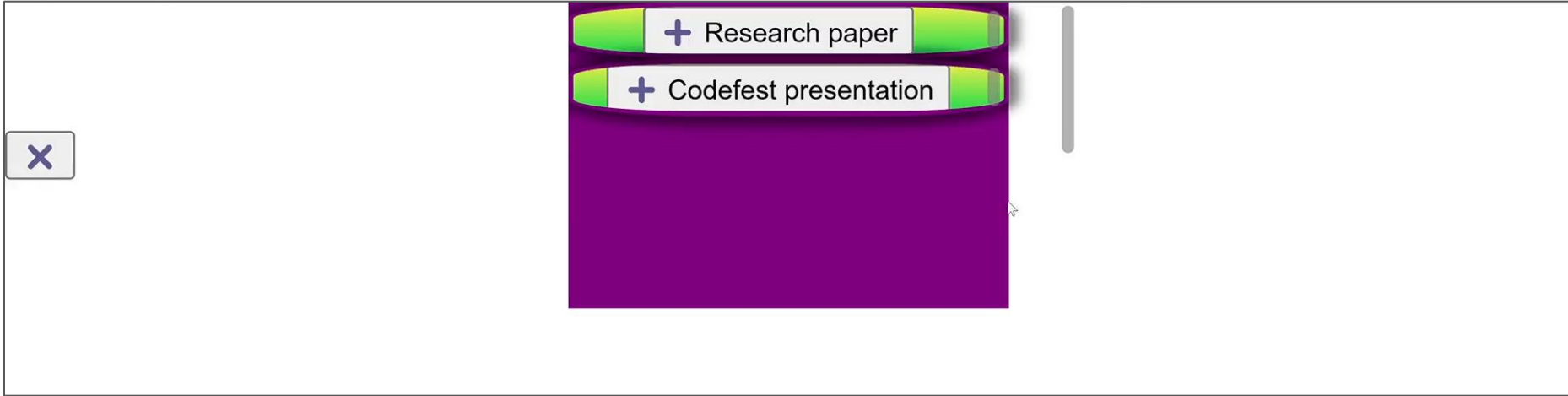


Screenshots - Planner view



The main screen, listing what Events and Timeblocks are scheduled for the day, coming from Google Calendar

Screenshots - Tasks view




Lists of Tasks that the user can schedule a Timeblock for, coming from Google Tasks

Screenshots - Event details



Clicking on an Event on the Planner will present the details associated with the Event, drawn from Google Calendar


Screenshots - Task details




Codefest presentation

- * Should be around 5 to 7 minutes long.
- * Include a demo of the project.
- * Also include a discussion of the technical stack, why that specific thing was chosen, one or two technical challenges, and one or two challenges in the decision-making process

From

05/01/2023 04:00 AM 

To

05/01/2023 05:00 AM 

Update

Delete

Clicking on a Timeblock on the Planner will present details associated with the Task, drawn from Google Tasks, as well as the scheduling controls of the Timeblock, from Google Calendar

Critical user journeys

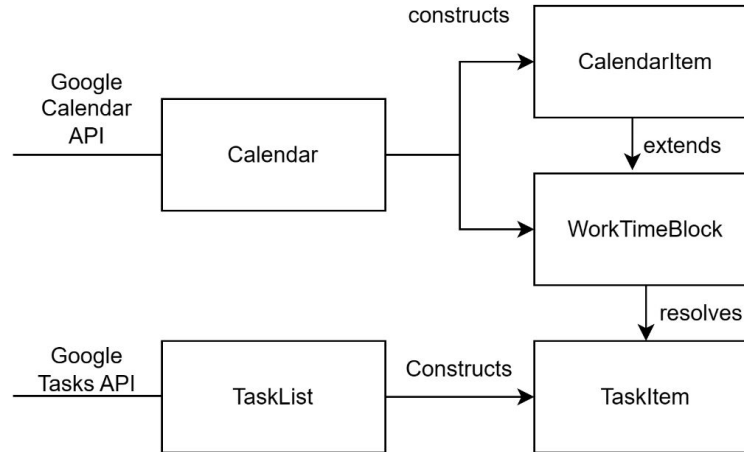
User goal	View a certain day's plans	Schedule a timeblock	View event details	View task details
User action	Open up the planner to the specified date	From the planner, click the + button to open up the task list, then click + on the desired task	From the planner, user clicks on an item that is an Event	From the planner, user clicks on an item that is a Timeblock
High-level implementation	<ul style="list-style-type: none">- Get all events for that date from Google Calendar API- Determine if event is an Event or Timeblock based on metadata	<ul style="list-style-type: none">- Get all unfinished tasks from Google Tasks API- When a task has been selected to add: create an event on Google Calendar, put custom JSON metadata in the description, containing corresponding task ID, to pair the timeblock to the task	<ul style="list-style-type: none">- Get all event details for the given event ID from Google Calendar	<ul style="list-style-type: none">- Get event details for the given event ID from Google Calendar- The event description has an ID for the corresponding task. Resolve the details of that corresponding task from Google Tasks

High-level Architecture

Back
Google Identity Services (GIS) for
authentication, and Google API Client
Library (GAPI) for API calls

Middle
Our own wrapper library using
GAPI calls to create the constructs
of tangible objects for calendars,
lists, events, tasks

Front
React.js single page app



event detail page

task detail page

Planner page

Task adder

Decisions Made - API Coverage

- Calendar and Tasks offer so much functionality: series, attachments, subtasks, links, etc
- Fully developing our wrapper library would take long enough to be its own project
- For initial version, cut scope to just the essentials for demoing the direction of our goal

Challenge / Decisions Made - Single Page App

- During development process, noticed we lose authentication on every page load or transition
- Research turns up that Google Identity Services doesn't allow client-side only webapps to persist authentication for security reasons
- Use of a server: requires finding infrastructure and some security research
- For time and resources purposes, went with single page app
- Learning opportunity for un/mounting React components to transition different pages, and passing states between them as arguments

Challenges / Lessons - Testing methodology

- Our client library was mainly tested in console
- But API involves async code
- Integrating front and back ends turned up bugs about data not being defined until after needed
- Async quirks not captured in console testing, as human typing is relatively slower than code execution
- Testing system kicked off by events (button clicks) would've helped