Git Push Master Ltd.

Help developers keep track of LeetCode problems they’ve solved

User Personas

- Cody is an Undergraduate Computer Science student who is trying to balance school and practicing for coding interviews
- However he struggles to dedicate time consistently due to a lack of focus and other priorities.
- He doesn't have set times to do these things, and isn't good at gauging the time he should be spending on them
    - He'll try to practice for coding interviews, but spend 2 hours on a problem instead of 30 minutes and then moving on.
    - His lack of efficiency is costing him positions at potential jobs.

-

Brian is a graduate student trying to balance his time between classes and devote at least 30 minutes a day to solving Leetcode problems. I have a goal of solving 3 LeetCode problems a week, but is taking too much time to solve each problem. I also want to be able to easily revisit problems that I've already solved, see their difficulty level etc is really helpful for my learning.

When working on LC problems, we can lose track of time and get stuck on a problem for 1-2 hours, which is rather inefficient because we probably should be looking at solutions before that. We want to have a timer that reminds us when we spent too much time on ourselves already.

When I do LC problems, I often browse through and find several problems I want to solve before actually working on them, but it is a little annoying to keep all those tabs open. I want to have a list of problems that I want to "work on later."

I want to keep a list of problems I want to revisit, so that I could further polish my skills by redoing them.

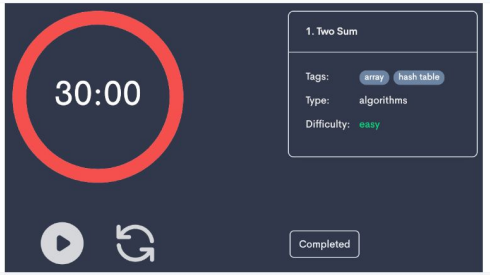
A feature to keep track of lowest time solved and maintain a record. I.e. "You solved Problem X 2 min faster"

User Stories

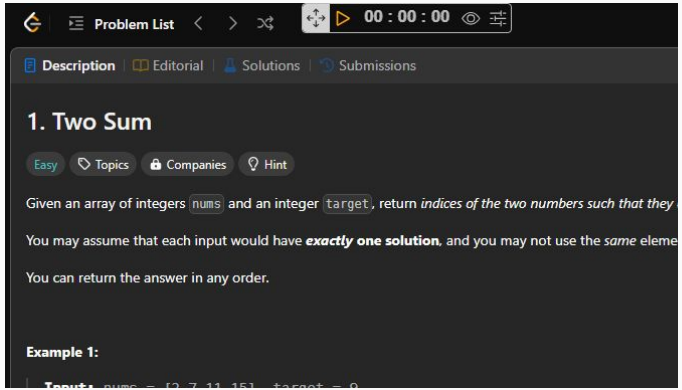
- As a software developer interview candidate, I want to set a timer for problems of different difficulty levels, so that I can practice the problems in a interview-like setting, when there is a time limit.
- As a Computer Science student getting ready to interview, I want to avoid spending too much time practicing a single question so that I may get a larger breadth of knowledge to succeed in my interviews.
- As a student I want to make sure I am using my available time efficiently and am solving enough LeetCode problems within my limited free time , so that I can remain competitive in the job market.
- As someone who wants to review leetcode problems without going through all my submissions, I want to keep a list of problems I want to revisit, so that I could master those problems by redoing them.
- As a Computer Science student working on leetcode problems in free time, I want to keep a list of leetcode problems that I want to work on later, so that I don’t need to keep all the tabs open at the same time.

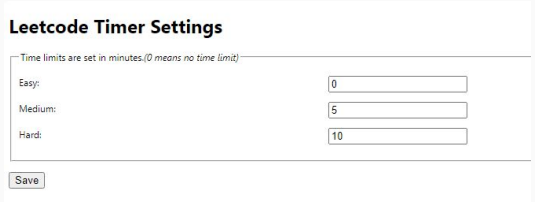
Similar Products

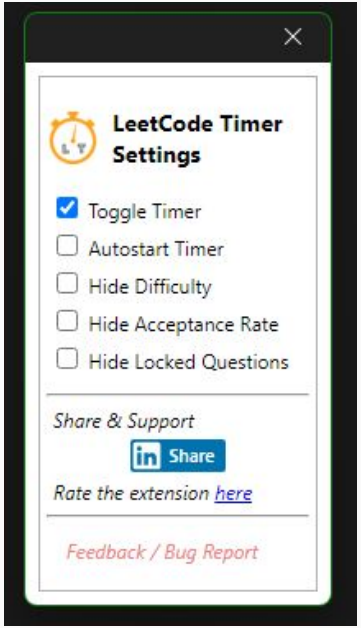
- DittoCode.io
  - Made by a developer who wanted to track LC problems and integrate pomodoro timer



- LeetCode Timer Extension







All Planned Features

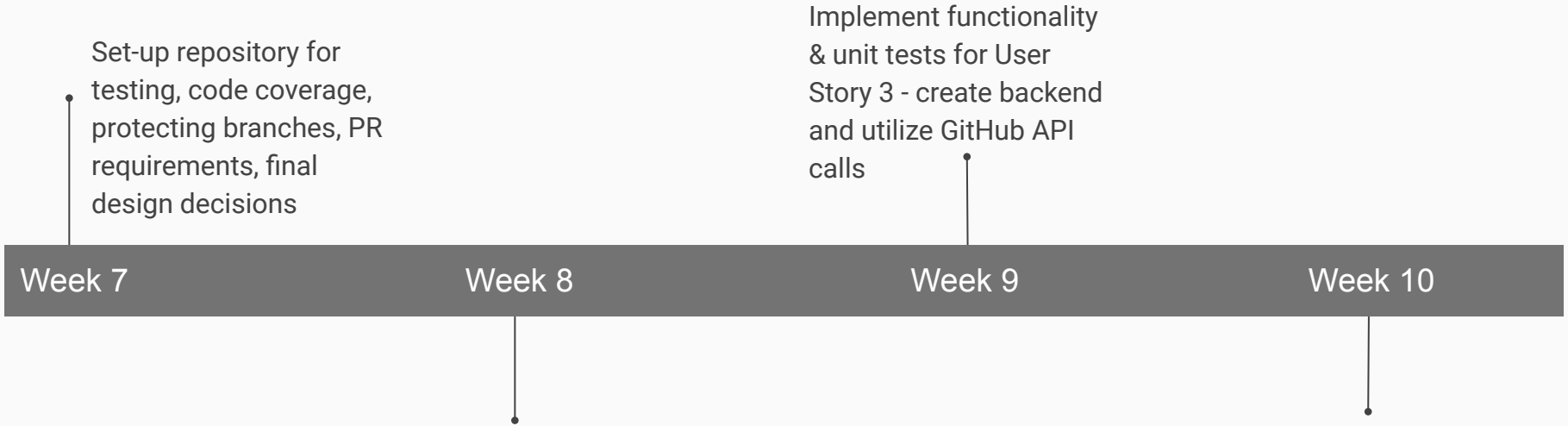
Core functionality

- Timer based on difficulty of problem
  - Retrieve problem's difficulty based on url
- Keep a to-do list of LC problems
- Keep list of LC problems that the user "wants to revisit"

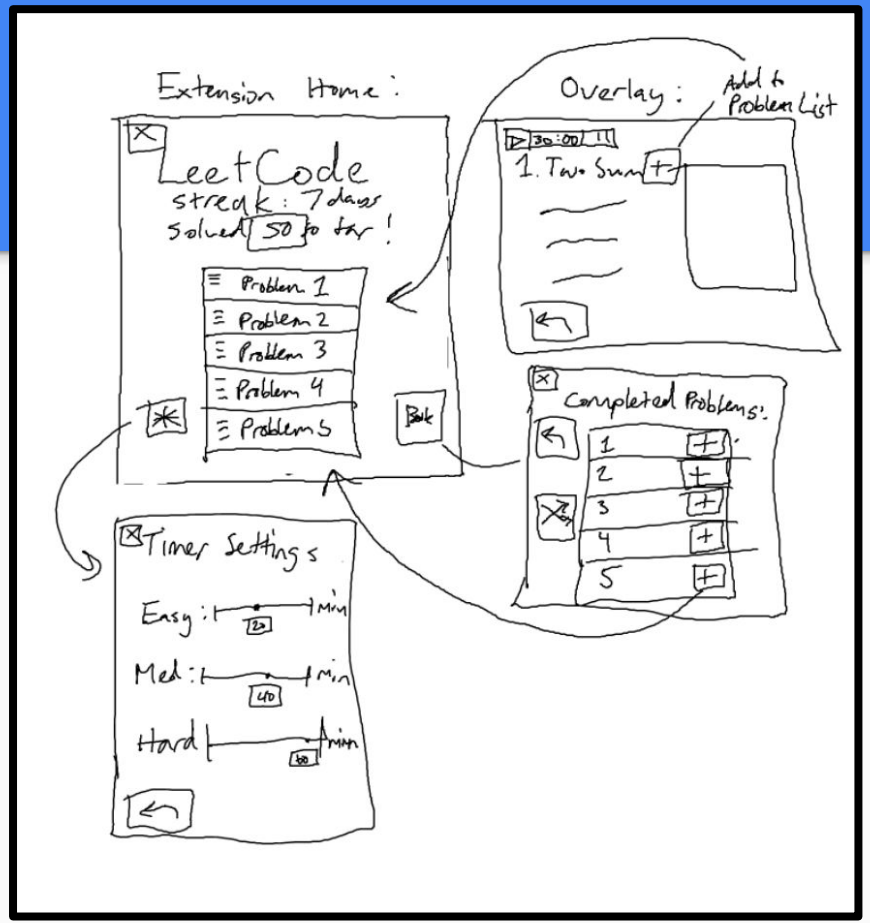
Optional Functionality

- Keep streaks of problems solved daily
- Sort lists in different ways
- Suggestions for what problems I can solve within the free time I have (e.g. I input 1 hr, I get one easy and one medium problem)
- Fetch user's data from session token

Project timeline



Sketches


Extension Home:
LeetCode
streak: 7 days
Solved 50 so far!
Problem 1
Problem 2
Problem 3
Problem 4
Problem 5
Bulk
Overlay:
Add to Problem List
30:00
1. Two Sum
Completed Problems:
1
2
3
4
5
Timer Settings
Easy:
Min
Med:
Min
Hard
Min

Potential Risks & Rabbit holes:

- LC API to retrieve information we need
- Tech stack for building chrome extension

Code to retrieve question difficulty

```python
import requests

# Leetcode API base URL for problems
BASE_URL = "https://leetcode.com/api/problems/all/"

# Function to fetch problems filtered by difficulty
def fetch_problems_by_difficulty():
    # Fetch all problems
    response = requests.get(BASE_URL)

    if response.status_code == 200:
        problems_data = response.json()
        problems = problems_data['stat_status_pairs']
    for problem in problems:
        print(f"- {problem['stat']['question__title']} (Difficulty: {problem['difficulty']['level']})")
fetch_problems_by_difficulty()
```

```
- Maximum Sized Array (Difficulty: 2)
- Second Highest Salary II (Difficulty: 2)
- Find Cities in Each State II (Difficulty: 2)
- Premier League Table Ranking III (Difficulty: 2)
- Count Number of Balanced Permutations (Difficulty: 3)
- Check Balanced String (Difficulty: 1)
- Total Characters in String After Transformations II (Difficulty: 3)
- Total Characters in String After Transformations I (Difficulty: 2)
- Find Minimum Time to Reach Last Room II (Difficulty: 2)
- Find Minimum Time to Reach Last Room I (Difficulty: 2)
```

Tech Stack

We are planning to stick to basic web technologies

- Plain Javascript, CSS, HTML
- Small Node/Python backend for Github API calls
  - Node might help unify our project under a single language (Javascript) but much of the team is unfamiliar with Node
  - Team is more familiar with python, may or may not be easier to get up and running depending on hosting solution
- Use local JSON file as a data storage solution until we outgrow it
  - Reasoning
    - Data is potentially personal, and should be stored locally instead of hosted remotely
    - We should avoid making the decision to add additional dependencies until we're sure that it's necessary

Potential Risks & Rabbit holes:

- Github API could be hard to implement
- Managing large sets of issues/tasks could be difficult
- Team has varying degrees of web development experience, would have to consider frameworks and technologies that are most efficient while still maintaining a relatively simple tech stack to begin with
- Potential “Rabbit hole”: How many features to implement? What to differentiate us from competitors? → Need to start with a minimum viable product to maintain realistic goals for completion by end of quarter, then add more later if time permits

Meeting Specific Requirements

- Availability
  a. We still need to find a solution that supports the easy deployment of backend and frontend code.
- Performance/RAIL Targets
  a. Because we're sticking to plain HTML, CSS, and JS, loading content under 5s should be easy if we keep our code compact
    i. This applies to user input as well
  b. We would like to include animations as a stretch goal (if user is not using reduce motion) and these will be implemented in plain CSS so they're quick and produce a frame under 10ms
  c. Will use integration tests to make sure load times are accurate (via selenium, cypress, etc)
- Accessibility
  a. Use appropriate HTML document design to allow multiple modalities (Screen readers, keyboard only)
  b. Verify foreground/background contrast color meets minimum requirements
- Usability
  a. We have three users that we know in different teams who are willing to test our product.
- Utility
  a. User stories will be implemented as Github Issues with description as definition
    i. Tasks related to user stories are sub-issues that are closed by PRs
  b. ADRs will be accompanied by design documents on the final architectural decision
    i. Design documents will be Markdown files with descriptions/figma links