DoDate Project — Iteration 1

User Stories

Description

DoDate is a personal planning application designed to help a single person keep track of tasks, deadlines, and scheduled events in one place. It combines a to-do list with date-based organization so users can see what needs to be done and when it needs to happen.

The app supports creating and organizing tasks, updating their status as work progresses, and managing events that occur on specific dates or times. Users should be able to review all tasks and events together, see what is coming up, and quickly identify items that are overdue or incomplete.

DoDate also helps users think about the size of the work they are taking on. Some tasks are quick and lightweight, while others take more time and energy. The app supports distinguishing between small and large tasks so users can plan their time realistically and avoid overloading a single day.

Instructions

In this iteration, you will explore the **functionality** (what the system can do) and **scope** (the boundary that defines what belongs in the system and what does not) of the system by writing user stories. These stories define what the application should support and help establish priorities for development. The emphasis is on identifying meaningful behavior and determining which features are essential versus optional.

- Write a set of user stories that describe the behavior of the system.
- Each user story must follow the format below:
  - Title: As a <user role>, I want <some goal> so that <some reason>.
- Organize your user stories using MoSCoW prioritization:
  - Must — required for a usable MVP
  - Should — important, but not required for MVP
  - Could — optional enhancements
  - Won't — out of scope or not a priority

Your user stories should describe a complete, usable core system; once those stories clearly define how the application functions at a basic level, you likely have enough. There is no required number of user stories, but your submission should reflect a thoughtful and well-rounded set of functionality rather than a minimal or superficial feature list. If you are unsure whether your scope is sufficient, ask the instructor or a TA for feedback.

This is an individual assignment, though you are encouraged to discuss ideas with classmates.

Submission

1. Create a single PDF that contains your prioritized user stories (Must, Should, Could, Won't), then **place that PDF inside your project repository folder**.

2. Open a terminal and **make sure you are inside the repository directory** before running any Git commands.

3. Add the PDF file to Git by running the following command:

```
git add your_file_name.pdf
```

4. Commit the file with a clear message describing what you added:

```
git commit -m "Add Iteration 1 user stories"
```

5. Push your commit to GitHub:

```
git push -u origin main
```

If your repository uses a different branch name (such as master), replace main with that branch name.

After the push completes, **open your repository on GitHub in a web browser and confirm that the PDF file appears in the repository**. If you do not see the file on GitHub, your work has not been successfully pushed. Reach out to a TA or the Instructor for help.

Grading Rubric (15 points)

Category	**Poosible Points**	**Description**
Formatting	2	All stories follow the required format and include titles
Story Quality	4	Stories are clear, meaningful, and user-focused
Robustness	4	Covers a wide and appropriate range of system functionality
Prioritization	3	All stories are prioritized and categorized thoughtfully and consistently
Scope Control	2	Scope is realistic and appropriate for the described system