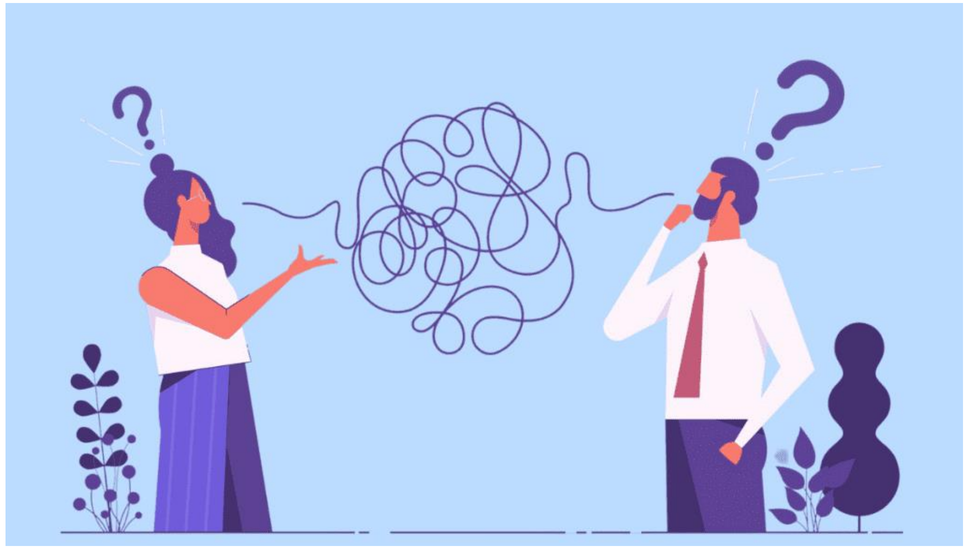
TaskFlow

CodeFellas: Andrew Tang, Daniel Abaye, Dylan Lau,

Mazin Abdelrahman

Problem

- Software engineering teams waste time sending emails concerning tasks
- Teams have trouble communicating about tasks
- May include
    - Questions about task requirements
    - Clarifying unclear task descriptions
    - Have trouble prioritizing assigned tasks
- There is no application designed specifically for communicating about tasks



Solution

- TaskFlow is a specialized communication application designed to make communication within teams about tasks more efficient
- Developers enjoy a centralized application for TODO lists
  - Check off completed tasks
  - Direct message project managers about concerns or other developers tasked to the same assignment
- Project managers will be able to send out interactive TODO lists
  - Prioritize tasks on the TODO list
  - Direct message individual developers or teams about the lists
  - Track tasks completed within teams and within the organization
- Avoids needing two applications for messaging and task tracking

Related Work

Similar existing software are:

Communication Software:

- Microsoft Teams
- Slack
- Zoom

List Software:

- Trello
- Notion

Sprint Meetings

- Overall improved planning
- Retrospectives helped in identifying needed improvements between PM assignments
- Review incoming work and distribute tasks among team members accordingly
- Estimated when work could be accomplished and how long it would take to do so
- Identified potential risks and how we would mitigate them

Sprint Planning Checklist

Preparation	Meeting	Follow Up
☒ Look at the slides for testing from the class slides ☒ Review PM2 and PM3 for ideas on test cases	☒ Assign each member a number of test cases to write	☐ Review the created test cases

Sprint Team Members 12/2/2024

Name	Role
Andrew Tang	Project Manager
Daniel Abaye	Developer
Dylan Lau	Developer
Mazin Abdelrahman	Developer

Agenda

Review and close previous sprint.

1. Review deliverables created from PM3 as they influence some of the tests created for PM4.

2. Review what a black box test plan looks like.

Previous Sprint Summary

Sprint theme	Requirements elicitation, analysis, and specification
Story points	70 points
Summary	The deliverables for PM3 were straightforward, however they were more time consuming that previously estimated (70 points)

Details

Start date	12/2/2024
End date	12/6/2024
Sprint theme	Create black box test plan

Velocity Tracking

This assignment should be similar in complexity to PM2. Creating the black box test plan will require a lot of cross referencing with PM2 and PM3 in order to create accurate tests. So, it may be slightly more tedious. This assignment will require less overall complexity than PM3 and will be more similar to PM2. The work will be split between two people writing 2 tests and two people writing 3 tests. Each test should take between 10-20 minutes to complete. 5 days is an ample amount of time to complete all 10 tests with the work being distributed.

Capacity Planning

	Current Sprint	Previous Sprint
Total days	5 days	7 days
Team Capacity	100%	100%
Projected Capacity	75 points	70 points
Individual Capacity	Andrew – 25% Daniel – 25% Dylan – 25% Mazin – 25%	Andrew – 20% Daniel – 26% Dylan – 26% Mazin – 26%

Potential Risks

Risk	Mitigation
Finals coming up	Spread out writing tests, schedule time to write tests, or get them done now because it shouldn't take long
Work due for other classes	Spread out writing tests, schedule time to write tests, or get them done now because it shouldn't take long

Requirements Analysis, Specification, and Design

Goals

- **Usability (human factors, help documentation, etc.)**
  - Choice of colors in application must contrast well enough to ensure all text is readable while maintaining aesthetics.
- **Reliability (failure frequency and recoverability):**
  - The system should not fail or have a very low failure frequency. If there is a failure, the app should attempt to create an error log to explain the issue. It should also try to save any of the user's data to be acquired in the next boot up, such as any message drafts they were writing that were not sent.
  - A message in the app should not fail to send, and there should not be a failure in receiving messages.
- **Performance (response times, throughput, availability, resource usage, etc.)**
  - The performance of this application should handle a large number of users (from different companies) which would mean that the application should not have a centralized server but a distributed system which would greatly increase throughput. Another solution is to implement multithreading or concurrency (only 1 because both can dramatically increase code complexity and potential bugs)
- **Supportability (adaptability, maintainability, internationalization, configurability)**
  - The application should be accessible across a wide range of devices to support collaboration among team members using different platforms, such as Android, iOS, Windows, and macOS.
- **Implementation/Constraints (resource limitations, languages, tools, hardware etc.)**
  - Messages sent should not be more than 1000 words. Messages should be limited to text messages, images, and files.
  - The app should support displaying Unicode characters.
  - The application should be cross platform, supporting computers and mobile devices.

Functional Requirements:

- Provide an example of five hypothetical functional requirements for your system.
  - Ability to assign priorities to tasks on TODO lists.
  - Ability to filter by task type, priority, name, date, completion date, etc.
  - Managers should be able to synchronize by interacting with a list with another employee user. A manager should be able to leave comments on the list.
  - Messages must have the option to attach images or other files.
  - The ability to send messages to groups, or individually.

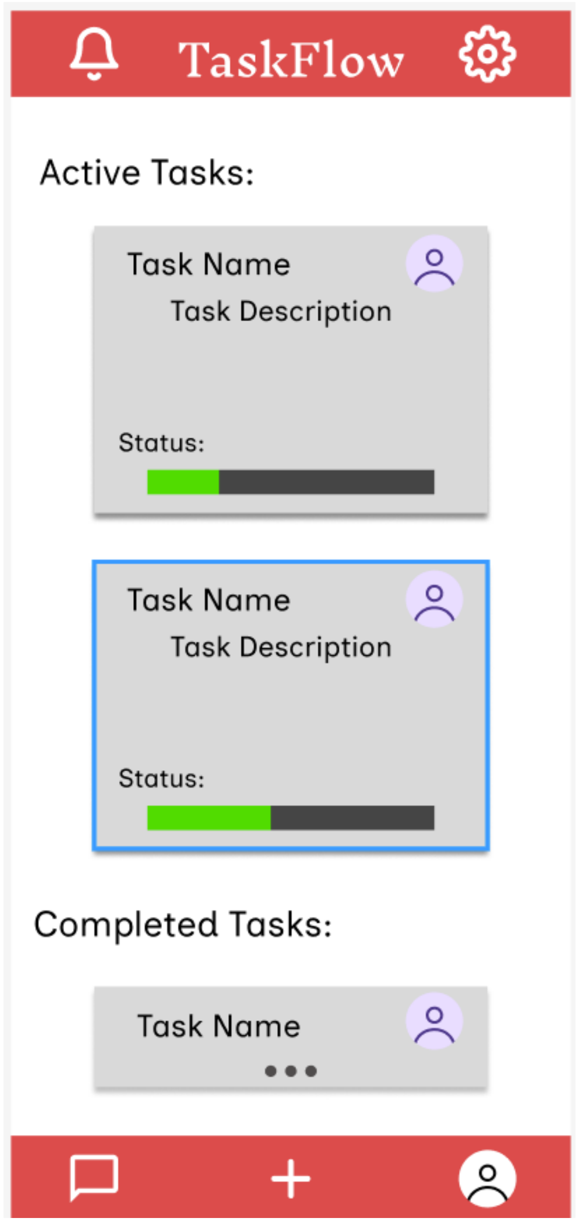
Specifications:

Use Cases

- Use case for assigning priorities to tasks on TODO lists.
    - □ Precondition: TODO list is populated with at least one task
    - □ Main flow:
        - User selects a task and selects edit
        - Select the “Prioritize” button
        - Select which level of priority the selected task is
    - □ Sub flow:
        - The edit button among other options will show when a task is highlighted by clicking on it, this will bring up a window with more details about the task
        - The “Prioritize” button is located somewhere near the bottom of the window
        - A dropdown menu will appear with the different priority levels that the user can select
    - □ Postconditions:
        - The tasks will be ordered hierarchically within the TODO list. Higher priority tasks will show at the top of the list with lower priority tasks following after it

Processes and tools used

- **Processes:**
  - Adopted an Agile methodology, utilizing sprint meetings to break work into manageable iterations and ensure regular progress updates.
  - Conducted requirements elicitation through team discussions and user story creation to identify and refine project needs.
- **Tools:**
  - **Outlook:** Used for scheduling meetings and team communication.
  - **Microsoft Word:** Documented requirements, meeting notes, and other deliverables.
  - **PowerPoint:** Created presentations for project milestones and final delivery.
  - **Figma:** Designed the application's interface.



Lessons Learned

- **Team Collaboration:**
  - Learned to effectively communicate and coordinate tasks within the team.
  - Developed strategies for resolving differences and maintaining team cohesion.
- **Technical Skills:**
  - Gained experience with Agile processes, such as sprint planning and retrospectives.
  - Improved proficiency in design tools.

Future Work

- VOICE AND VIDEO CALLS
- ADDITIONAL FILE UPLOAD FORMATS
- EMOJI SUPPORT
- IMPORT/EXPORT OF TODO LISTS
- EMAIL SHARING
- MEETING SCHEDULER