CS/SE/CE 3354 Software Engineering

Final Project Deliverable 2

Time Management Application

Group Members:
Alex Countryman, Alex Tang, Aston Esquivel, Chris Cowan, Faizan-Ali Lalani,
Griffin Tamm, Lawrence Ho, Michael Zeller, Yousef Ali

I. Delegation of Tasks

By member

- Lawrence Ho: For Deliverable 1, I worked on sequence and case diagrams. For Deliverable 2, I worked on the cost, effort, and price estimation. I also worked on the presentation, adding diagrams and information from both deliverables.

- Faizan-Ali Lalani: I worked on the requirements needed for the software and the process in which the software functions. I worked on the non-functional requirements, added one of the comparisons to our project, and talked about our project and Monday.com had some similarities and differences. I started the powerpoint presentation for all team members to add their information to.

- Yousef Ali: I worked on the class diagrams as well as the architectural design needed for the framework of the project. I also worked on the cost, effort, and price estimation.

- Michael Zeller: I worked on sequence and case diagrams. I developed the Software Test Plan and developed the software that was tested. I worked on the conclusion with Alex Countryman.

- Alex Tang: I worked on the architectural design and class diagram. I worked on the cost, effort, and price estimation.

- Chris Cowan: I worked on brainstorming the software process model, brainstorming the software requirements and non-functional requirements, making the GitHub, making the project schedule, making the demo on the slide set and formatting the report.

- Alex Countryman: I worked on the class diagram and the architectural design diagram for the project. I also performed calculations for the cost, effort, and estimation model, as well as adding to the conclusion for this deliverable.

- Ashton Esquivel: I worked on brainstorming the software model we are utilizing, brainstorming functional requirements, and the comparison to other works as well as the slide for it.

- Griffin Tamm: I worked on sequence and case diagrams as well as the comparisons to other works.

By Category

- Requirements (Ashton Esquivel, Chris Cowan, Faizan-Ali Lalani)
- Software Process Model (Ashton Esquivel, Chris Cowan, Faizan-Ali Lalani)
- Case and Sequence Diagrams (Griffin Tamm, Lawrence Ho, Michael Zeller)
- Architectural Design (Alex Countryman, Alex Tang, Yousef Ali)
- Class Diagram (Alex Countryman, Alex Tang, Yousef Ali)
- Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing (Chris Cowan, Alex Countryman, Alex Tang, Lawrence Ho)
- Software Test Plan (Michael Zeller)
- Comparison To Other Works (Faizan-Ali Lalani, Ashton E., Griffin Tamm)

II. Project Deliverable 1 content

A. Motivation/Goals

Motivation

A few of our group members agreed that prioritizing tasks over others was difficult when they have a lot on their plate. So we came up with a way to prioritize tasks based on time remaining and complexity so that the user can be displayed with what's up next in importance. The app also combines functionality with activities (time blocked events) for even further urgency tracking.

"Oh I have class and work all day for the rest of the week and I have an urgent task due Saturday! I need to start ASAP!"

Goals

Calendar Functionality:
Display time blocks for specific activities and deadlines.
Allow users to add, edit, and delete time blocks.

For each time block, users can specify:
- Title
- Type of Activity
- Associated course/activity (if applicable)
- Duration

Deadline Management:
Enable users to add, edit, and delete task deadlines.

For each deadline, users will input:
- Task name
- Due date
- Task complexity

Task Prioritization:
Implement a sorting algorithm to prioritize tasks based on:
- Deadline proximity
- Task complexity
- Other relevant metrics (to be determined)

User Interface:
Provide an intuitive, easy-to-navigate calendar interface.
Ensure smooth transitions between month, week, and day views.

Data Persistence:
Store user data securely to maintain task and deadline information across sessions.

Notification Pushes:
Notifications pushed to user daily based on upcoming deadlines and activities

B. Project Feedback

We received “Well done.”

C. GitHub Repository

Our task manager GitHub repo is housed at the following link:
https://github.com/chrisCowan98/3354-TeamTaskManager

- GitHub Repo made by Chris Cowan
- README created by Lawrence Ho
- Project Scope created by Alex Countryman
- All members are included, as well as the TA.
- Testing and test methods made by Michael Z.

D. Software Process Model

Agile Scrum is being utilized due to the need for flexibility as requirements may evolve throughout the development process, especially at the end stages of the product due to frequent IOS and Android system changes. This framework emphasizes collaboration, continuous feedback, and iterative development, ensuring the product adapts to changing needs. Since the product is targeted to be a free application at launch, cost-efficiency is crucial, and Agile Scrum allows us to prioritize essential features, delivering value incrementally. The focus on collaboration and adaptability in Agile will help streamline development while maintaining a high-quality product tailored to user feedback.

E. Software Requirements

1. The system shall allow the user to create a “Time Block” that contains a title, description, duration, time, date, and shall optionally allow a user to add a course name/number.

2. The system shall allow a user to create a “Task” that contains a title, due date, time, and complexity, where complexity is an integer user input between 1 - 5; 1 being the lowest complexity, and 5 being highest complexity.

3. The system shall display both Tasks and Time Blocks on a calendar view with transitions between month, week, and daily views, where the user can switch between periods of dates.

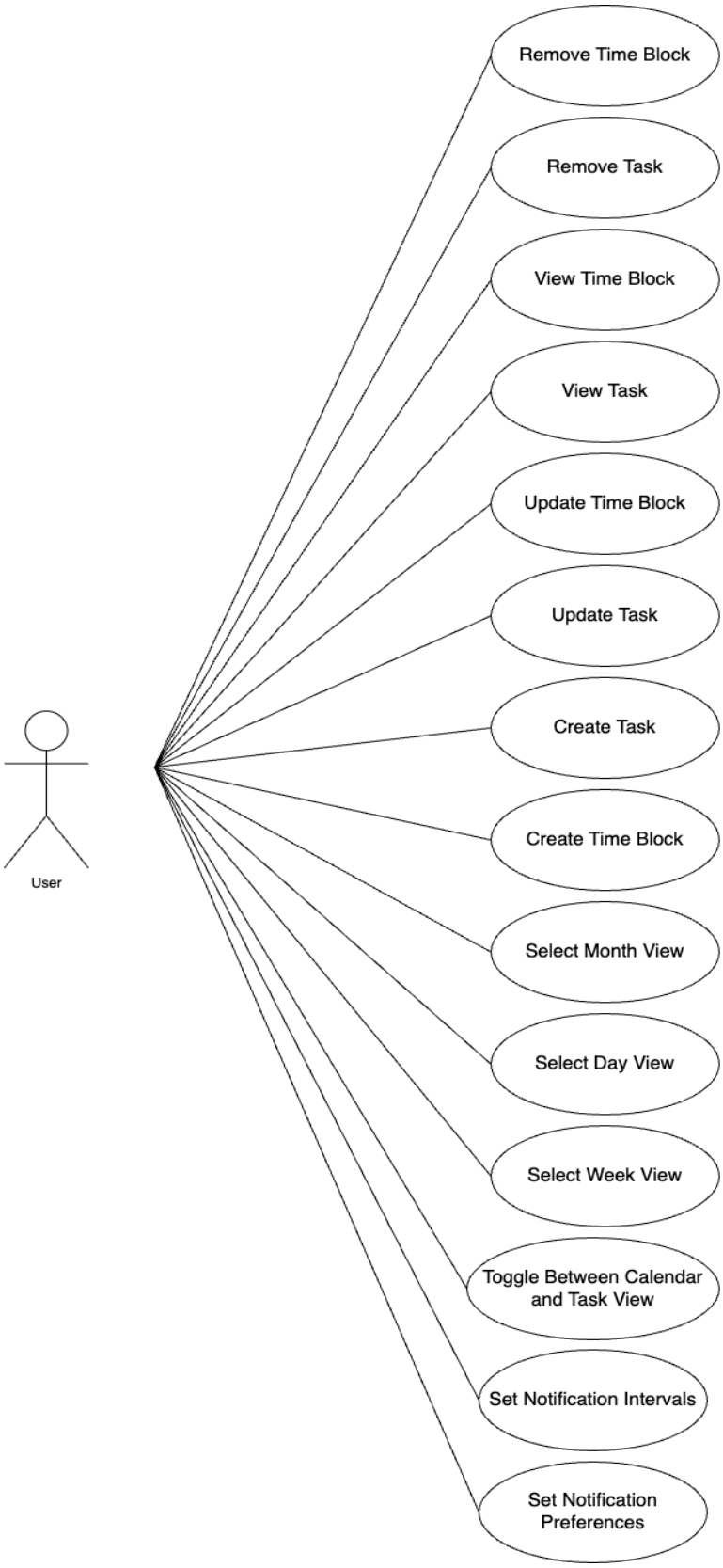
4. The system shall generate and show a priorities list view, where one list, “Upcoming”, is ranked based on the task/activity due date, and the other, “Urgent tasks”, ranked based on urgency, where urgency is a calculated value based on time remaining and complexity.

5. The system shall generate notifications for users daily based on tasks with high urgency value relative to the rest, or multiple if tasks have the same urgency value.

F. Non-Functional Requirements

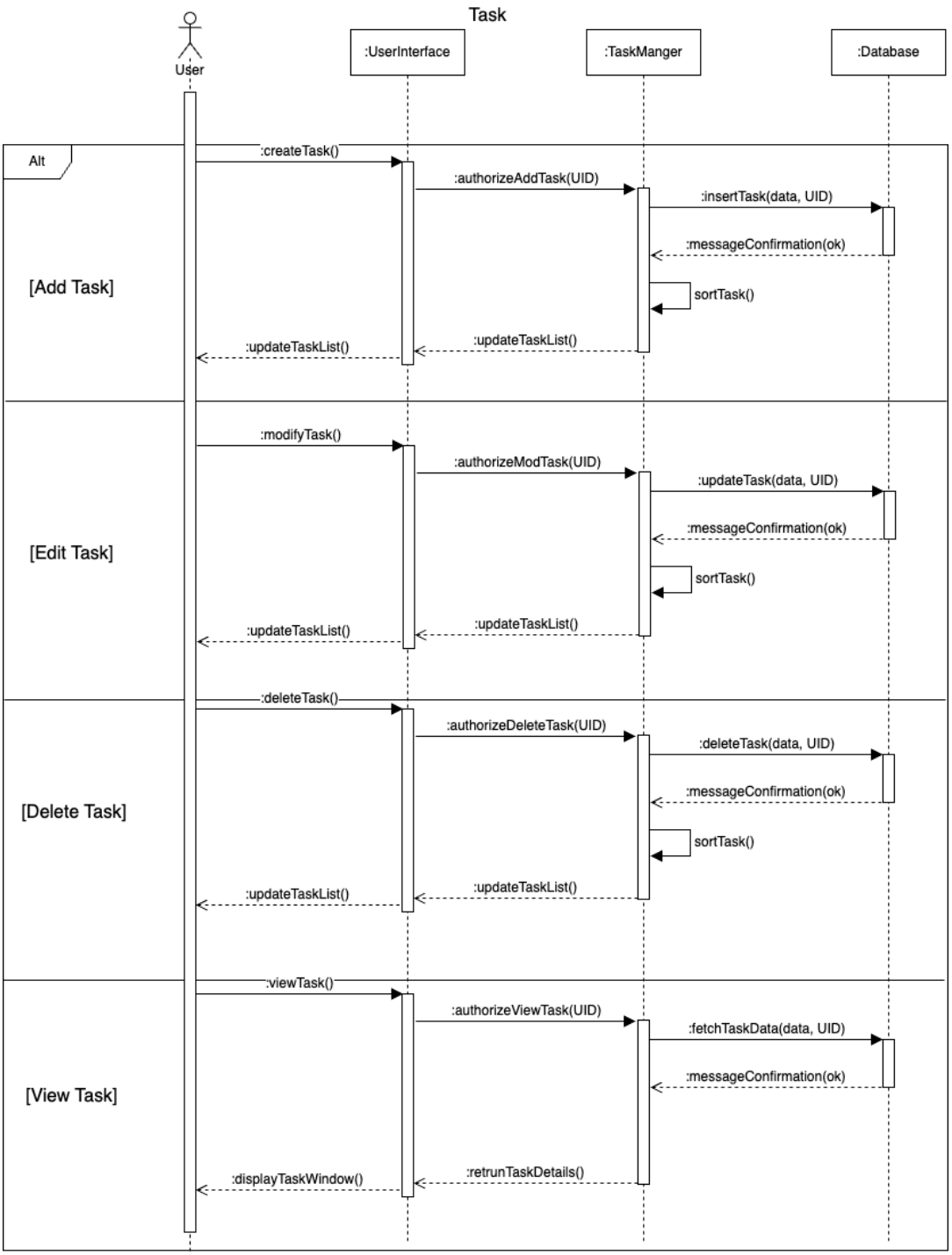
- Usability requirements
    - Must be a mobile based app distributed publicly by Google Play Store and Apple App Store.
    - Must be offered on most mainstream devices (IPhone, Galaxy, etc) in the last 5 years.
- Performance Requirements
    - Must have less than half a second load time into application.
- Space Requirements
    - Must take less than 400Kb of storage on the user's device.
    - Must take less than 200Kb of ram on the user's device.
- Dependability Requirements
    - Must crash or terminate less than .1% of application startups.
- Environmental Requirements
    - Must only send notifications between 10:00AM - 7:00PM user time.
- Operational Requirements
    - Must operate in any user operations as long as the application is open.
    - Must push notifications while the user device isn't displaying the application.
- Development Requirements
    - Must predominantly use either Java or Swift.
    - Must produce a short description for every file created in documentation.
- Accounting Requirements
    - Application must be free in every country the application is available.
    - Ads must be kept to the minimum threshold to meet the cost of development within 5
      years.
- Safety/security requirements
    - User data must be backed up to a server with every application use.
    - Any data saved must be encrypted using industry standards.
- Ethical requirements
    - The App must get the consent of the user to collect all data and provide clear instructions on how that data will be used.
    - Any and all user data collected must not be used for profit.
- Regulatory Requirements
    - Must follow applicable laws to every country the app is available to.

G. Case Diagram

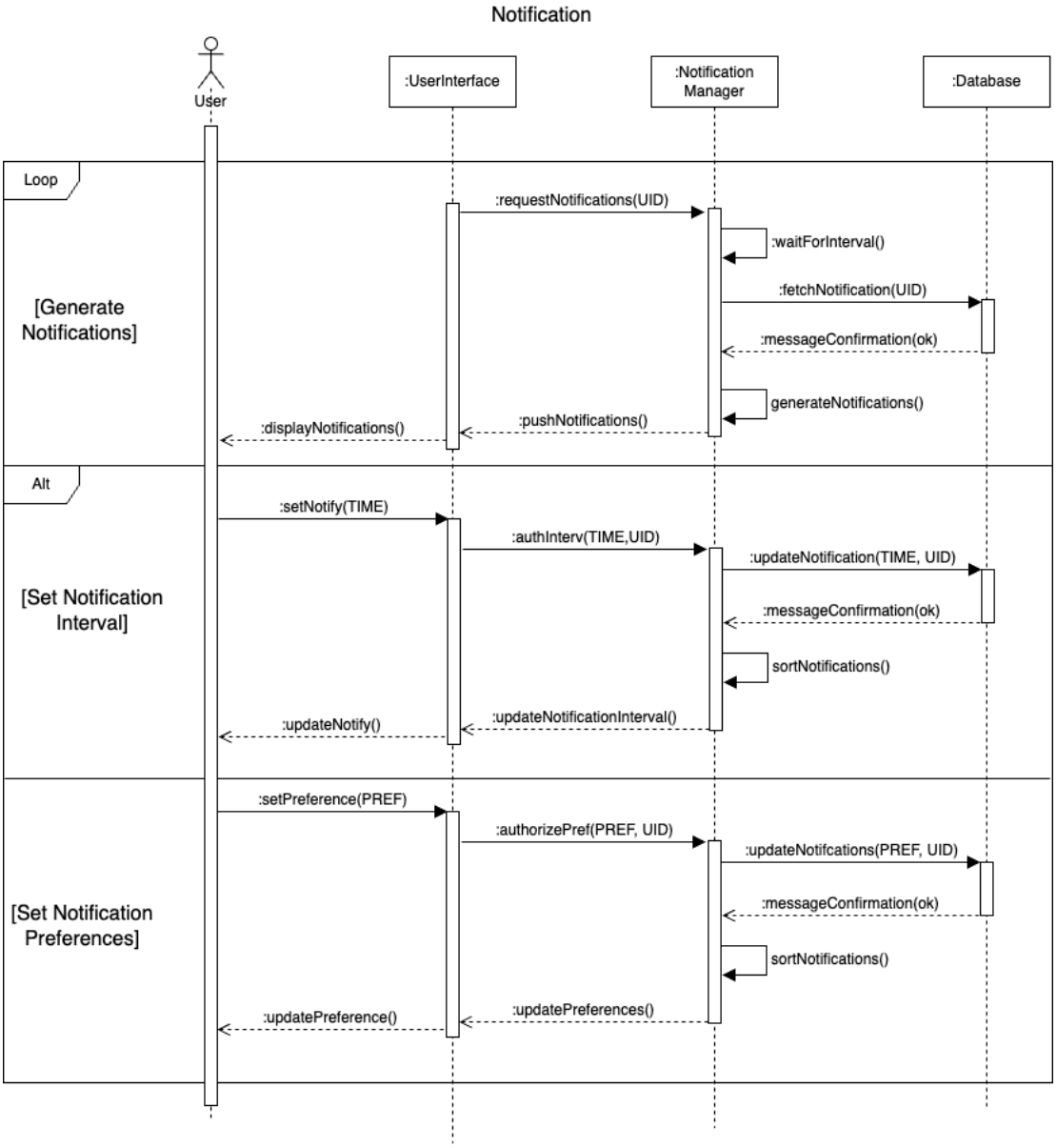


H. Sequence Diagrams

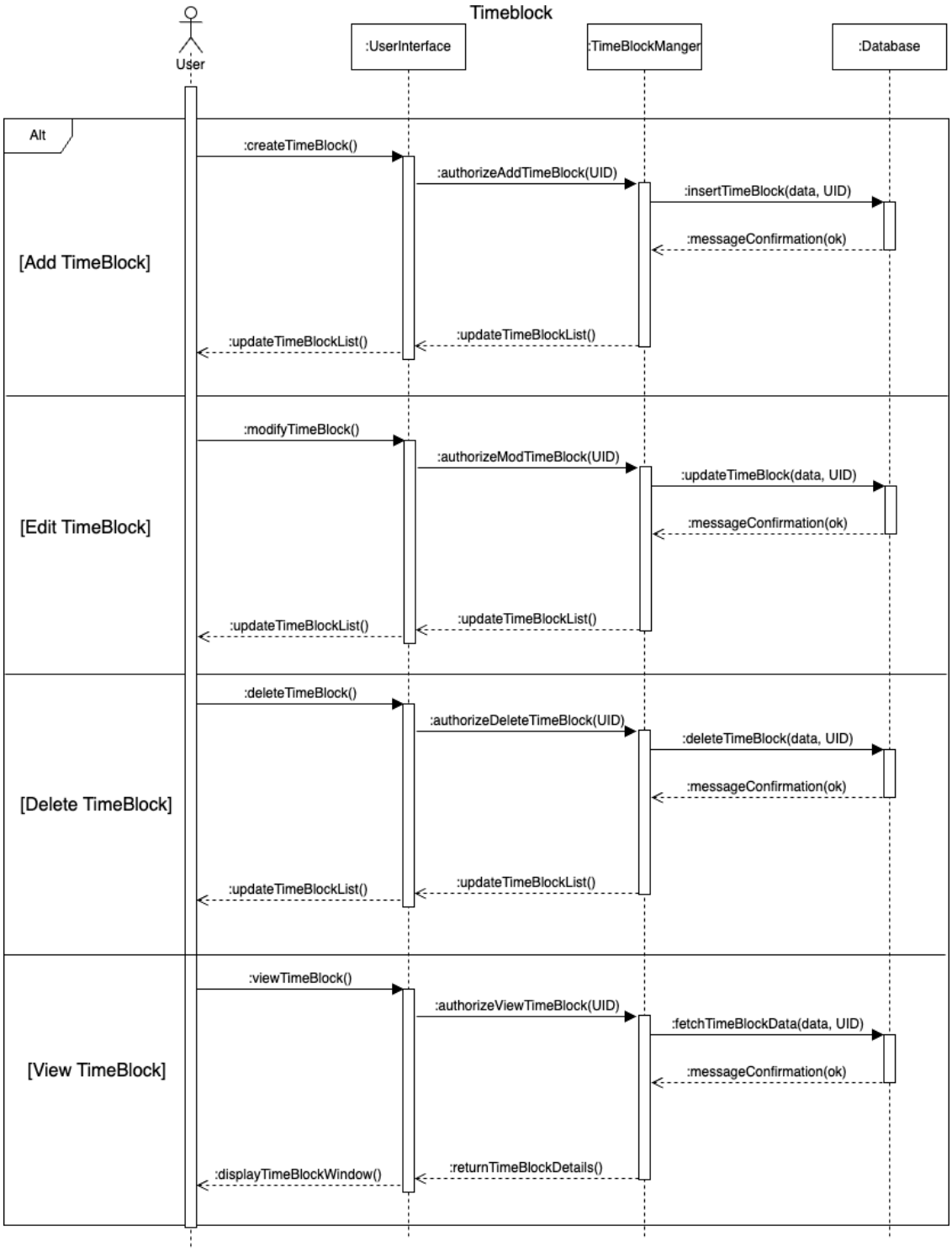
Sequence diagrams for all task actions/uses:



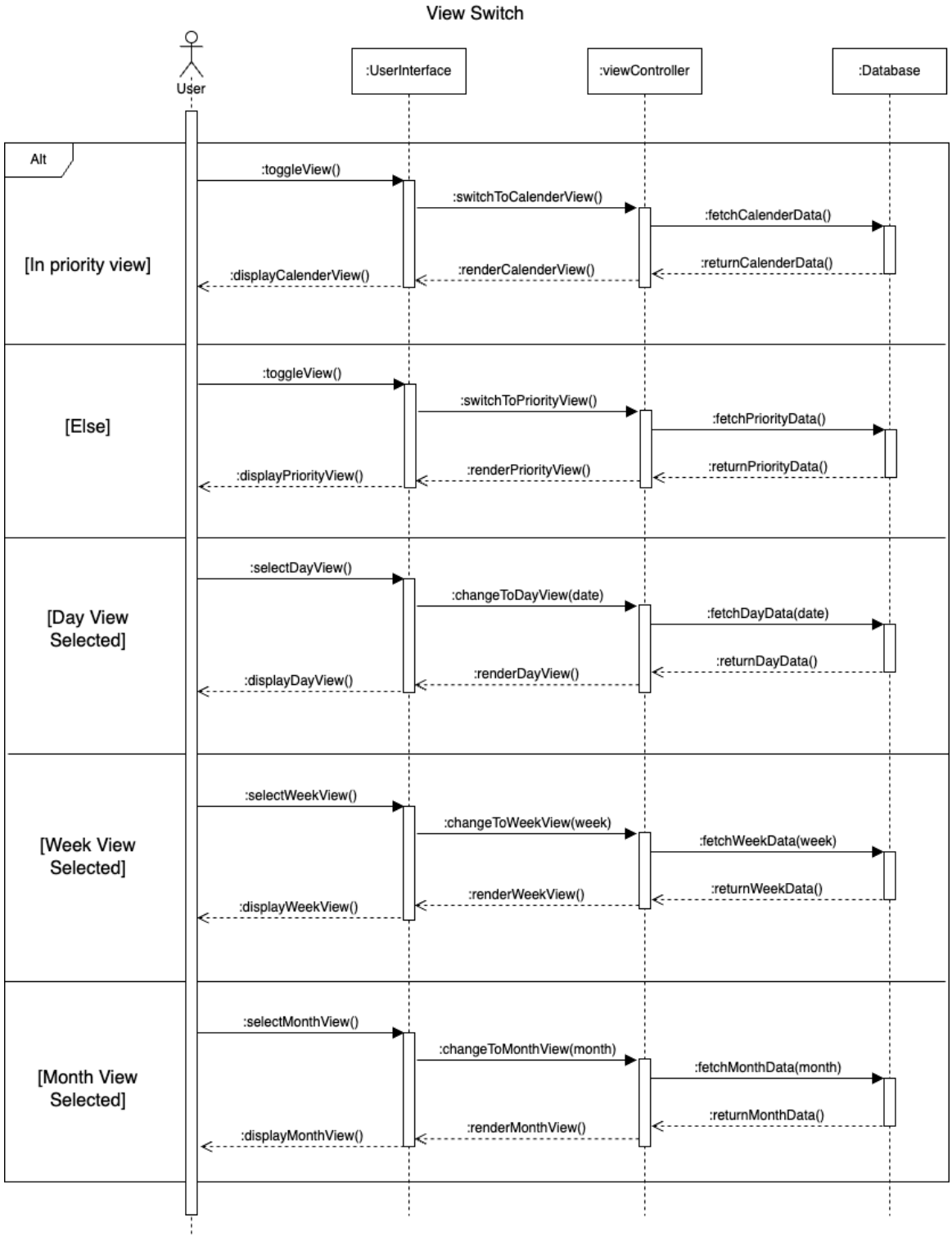
Sequence diagrams for all notification actions/uses:



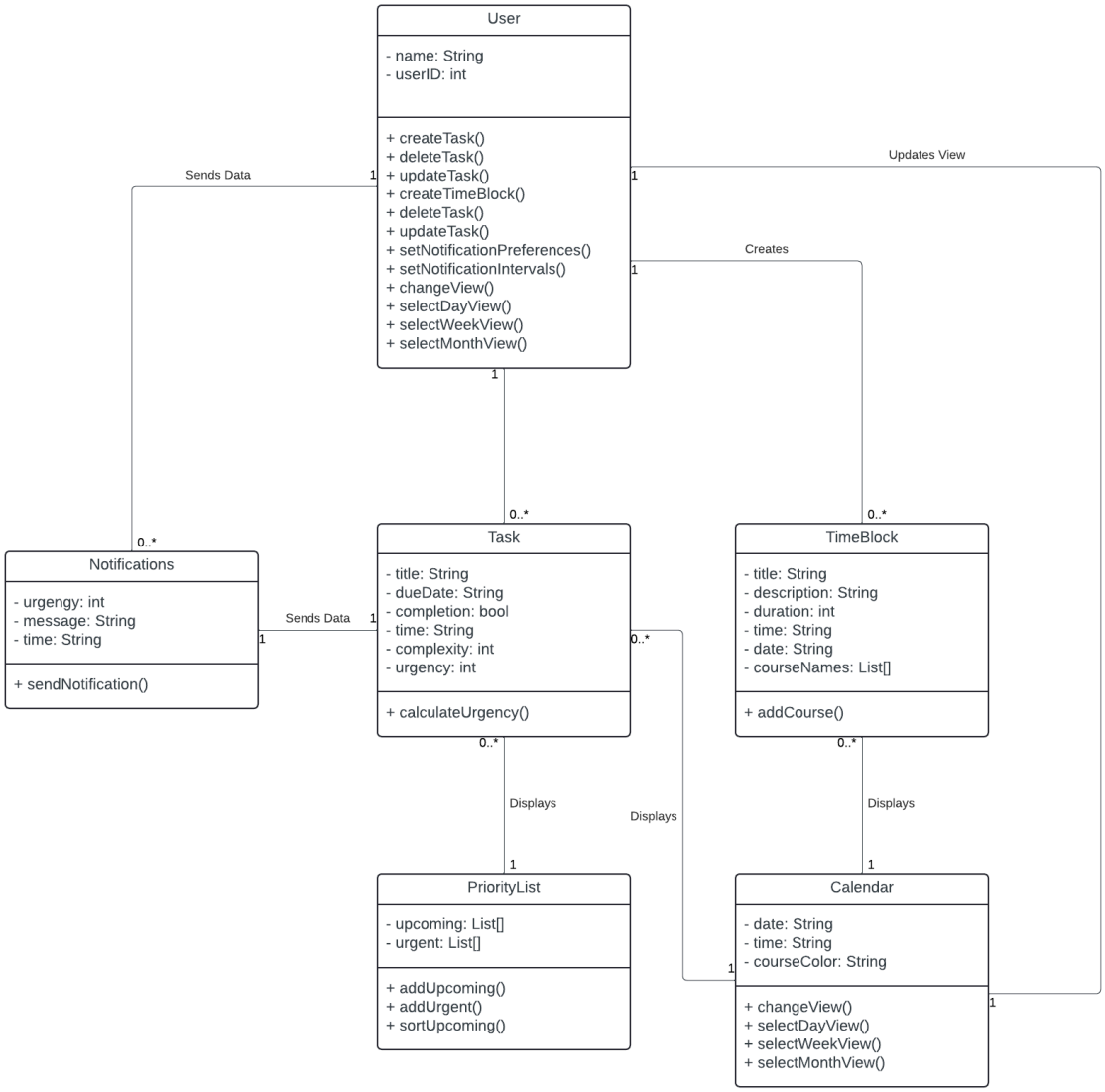
Sequence diagrams for all time-block actions/uses:



Sequence diagrams for all view actions/uses:

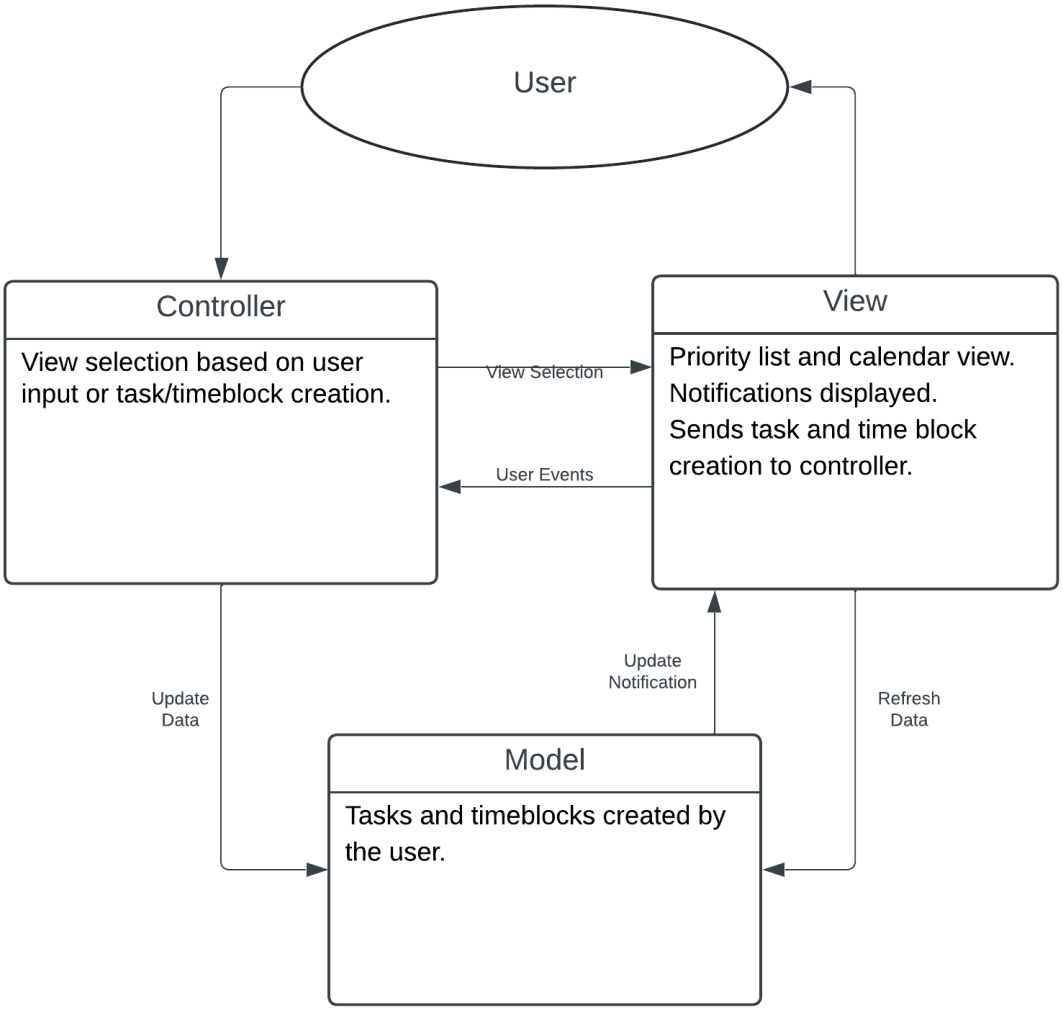


I. Class Diagram



J. Architectural Design

Model-View-Controller (MVC)



III. Project Scheduling, Cost, Effort and Pricing Estimation, Project duration and staffing

Project Scheduling Assumptions

The following assumptions shall be made for the product development:

1. The team will be composed of 5 developers (2 junior Full Stack SWE, 3 Senior Full Stack SWE), 1 scrum master, and 1 product owner.
2. The team is fully remote, works at the same time, and earns the national average for their role
3. The timeline will end at the product launch (Anything after is maintenance)
4. Standard work week (No weekends)
5. Typical 8 Hour work days (40 hour work weeks)

Schedule Breakdown

- Requirement Gathering and Planning (2 Weeks)
    - The team will work to define the scope of the project, features, due dates, and deliverables. Estimated to be about 2 weeks.
- Design Planning (2 Weeks)
    - UI design, architectural planning, database planning, API endpoint planning. Estimated to be another 2 weeks.
- Development and Testing (5 Weeks, 5 Sprints)
    - Sprint 1 will target the “Tasks”.
    - Sprint 2 will target the calendar.
    - Sprint 3 will target categorizing tasks and displaying them in a task view.
    - Sprint 4 will target user notifications.
    - Sprint 5 will target testing, tech-debt, and quality assurance. This is done during the other sprints as well, but Sprint 5 is allocated purely for cleaning things up.
- Deployment and Final Adjustments (2 Weeks)
    - Final touch-ups, setting up final infrastructure, deploying servers. Estimated to be 2 weeks.

The start date will be **March 1, 2025**, and the end date will be **May 17, 2025**.

Cost, Effort and Pricing Estimation

The following is done with function point estimation:

	Function Category	Count	Complexity			Count x Complexity
			Simple	Average	Complex	
1	Number of user input	29	3	4	6	87
2	Number of user output	11	4	5	7	55
3	Number of user queries	6	3	4	6	18
4	Number of data files and relational tables	8	7	10	15	56
5	Number of external interfaces	4	5	7	10	28
					GFP	**244**

PCA = 0.65 + 0.01 (Total PC) = 0.65 + 0.01 (34) = 0.99

FP = GFP * PCA = 244* 0.99 = 241.56 FP

E = FP / productivity = 241.56 / 4.392 = 55 person-weeks

D = E / team size = 55 / 5 = 11 weeks

PC	Question	Score
1	Does the system require reliable backup and recovery?	2
2	Are data communications required?	3
3	Are there distributed processing functions?	0
4	Is performance critical?	2
5	Will the system run in an existing, heavily utilized operational environment?	2
6	Does the system require online data entry?	5
7	Does the online data entry require the input transaction to be built over multiple screens or operations?	4
8	Are the master files updated online?	3
9	Are the inputs, outputs, files, or inquiries complex?	2
10	Is the internal processing complex?	2

11	Is the code designed to be reusable?	3
12	Are conversion and installation included in the design?	3
13	Is the system designed for multiple installations in different organizations?	0
14	Is the application designed to facilitate change and ease of use by the user?	3
	Total PC	**34**

We used function point estimation to determine the size, complexity, and effort required for our application. We began with counting the number of each function category that we had in our application and then chose the complexity that best fit each category in regards to our application. Then, we calculated the complexity x count and added them all up to get our gross functional points (GFP). We then answered the 14 questions for processing complexity (PC), so that we could calculate the processing complexity adjustment (PCA).

Using the GFP and the PCA, we were then able to calculate our functional points. Assuming 4.392 functional points per person-week as the productivity for the developer team, we calculated the estimated effort as FP / productivity (241.56 / 4.392) = 55 person-weeks. Given this, we were also able to estimate the estimated project duration which is D = E (estimated effort) / team size (developers) = 55 / 5 = 11 weeks.

<u>Estimated cost of Hardware Products</u>

- Web server/Data-Base
    - Since we will be using a cloud-based service such as AWS, this cost will be based on DB instances, which for AWS can be $.011 per DB instance hour [1]. This amount could be anywhere from $100 to thousands depending on the popularity of the app.
- Workstation
    - 5 developers each needing high-end systems (Let's say 5 MacBook Pros because this is an IOS app) at $1,738 each after sales tax [2]. So ~ $8,700.
- Internet
    - Since all developers are remote, this cost goes to them.

<u>Estimated cost of Software Products</u>

- Version Control and Repository
    - Git/GitHub. Totally free.
- IDE
    - 5 Copies of XCode. Totally free.
- Project Management
    - Jira. 3 Months of development at $7.53/Month ~ $23 [3].

Estimated cost of personnel

- Average pay of a junior SWE = $77,904 a year or $37.45 per hour [4]
- Average pay of a senior SWE = $155,141 a year or $74.59 per hour [5]
- Average pay of a scrum master = $108,033 a year or $51.94 per hour [6]
- Average pay of a product owner = $107,511 a year or $51.69 per hour [7]

Total number of weeks = 11

Total number of hours = 11 * 40 = 440

Total personnel cost = 2 ($37.45 * 440) + 3 ($74.59 * 440) + ($51.94 * 440) + ($51.69 * 440)

= 2 ($16,478) + 3 ($32,819.60) + $22,853.60 + $22,743.60

= $32,956 + $98,458.80 + $22,853.60 + $22,743.60

= $177,012 + $10,000*

= $187,012 Total Personnel Costs

We estimate an additional 10,000 for additional employee training* [8].

IV. Software Test Plan

Test Plan Objective: CalculateUrgency() Method

The objective of this test plan is to verify that the calculateUrgency method calculates the urgency accurately based on the task's complexity and due date. The range for urgency is from 0 to 100, which is used as a metric to determine the priority of the specified task.

This test plan utilizes the automated testing tool JUnit 4, a Java-based testing framework, that automates the test cases and verifies the correctness of each result.

The test checks the following:

- Boundary Testing (Edge cases)
    - Testing for maximum urgency cap (100).
    - Testing for minimum urgency (0).
- Functional Testing
    - evaluating various combinations of complexity and due date to verify the correct functionality of the calculateUrgency algorithm.

Test Case Summary: CalculateUrgency() Method

1. **Future Due Date (Moderate Complexity)**: Verifies calculated urgency for average complexity and future due date.
2. **High Urgency Capped at 100**: Confirms urgency caps at 100 when complexity and proximity to due date are high.
3. **High Urgency Without Exceeding**: Ensures high urgency calculation remains below 100.
4. **Low Urgency**: Validates urgency calculation for low complexity and distant due date.
5. **Exact Zero Urgency**: Checks that urgency is zero when complexity is zero, irrespective of due date.

Inputs and Results: CalculateUrgency() Method

Test Case	Complexity	Days Until Due Date	Expected Urgency	Result
Future Due Date (Moderate)	5	8	56	Pass
High Urgency Upper Bound	12	7	100	Pass
High Urgency	8	8	89	Pass
Low Urgency	3	14	20	Pass
Low Urgency Lower Bound	0	5	0	Pass

Test Plan for TaskManager Priority Queue

The objective of this test plan is to verify that the TaskManager class accurately prioritizes tasks based on urgency and due date using a priority queue. In the case of two or more Tasks with equal urgency, the queue will have two tie-breakers. The first tie-breaker is the due date, then in lexicographical order by title in case the due date is equal.

This test plan utilizes the automated testing tool JUnit 4, a Java-based testing framework, that automates the test cases and verifies the correctness of each result.

The test checks the following:

- Functional Testing
    - Testing the priority queue given tasks with different urgencies.
    - Testing the priority queue functionality given Tasks with the same urgency

but different due dates.
    - Testing the priority queue given tasks with the same urgency and due date.

Test Case Summary: Priority Queue

1. **Different Urgencies:** Verifies that tasks with different urgencies are ordered correctly in the priority queue.
2. **Same Urgency and Different Due Date**: Verifies that tasks with identical urgency and different due dates are correctly ordered by their due date.
3. **Same Urgency and Due Date**: Verifies that tasks with identical urgency and due date are ordered alphabetically by title as a tiebreaker.

Test Results: Priority Queue

Test Case	Urgency	Due Date	Expected Order	Result
Different Urgencies	75, 50, 30	6 days, 15 days, 7 days	Tasks ordered by urgency first	Pass
Same Urgency, Different Due Date	50	5 days, 3 days, 7 days	Task with earliest due date first	Pass
Same Urgency and Due Date	50	2 days	Tasks are ordered alphabetically by title	Pass

V. Comparison To Other Works

Todoist

Todoist, an online task manager and to-do list app, has a design that closely resembles our work. It shares several features with our product, including calendar functionality, deadline management, and a similar user interface[9]. However, Todoist also includes additional features that we currently do not support[9]. Firstly, Todoist allows users to set recurring task due dates, making it more practical than adding individual tasks for each day or week[9]. It also offers feature sections and subtabs, enabling users to break down large tasks into smaller, more manageable parts[9].

Additionally, Todoist provides labels that let users categorize and group similar tasks for easier organization[9]. Despite these differences, we do share some feature similarities. For example, both apps offer prioritization options that allow users to rank tasks based on importance, from high to moderate priority[9]. Additionally, Todoist offers a bird's-eye view of the user's schedule with the flexibility to switch between upcoming tasks and today's schedule[9]. In summary, our app and Todoist share a solid foundation as task management tools, though Todoist provides more advanced features that offer users greater options for task filtering and organization.

Monday.com

Monday.com is another online task manager that's very similar to our work with multiple different uses. It can be a task manager, project manager, tasks & to-do’s, time tracking, goals & strategy, and business operations [10]. Monday.com shares multiple features with our task manager, it shares the task manager, calendar, and deadline management. However, Monday.com has some advanced features that we don't have in our project.

Monday.com provides a customizable and collaboration-friendly environment with advanced scheduling tools, detailed task information, and integration capabilities[10]. Our project prioritizes user-friendliness over complex functionality. In conclusion, our project and Monday.com share the fundamental functions of task managers, but Monday.com is more complex, offers more advanced features, and is collaboration-friendly.

Google Calendar

Google Calendar is a similar app that allows the user to create a calendar with tasks allotted times and days [11]. This shares many features with our software, including the ability to manage tasks, calendars, and deadlines[11]. Google Calendar is integrated with Google's other services, which allows users to quickly set up tasks from emails, or see scheduled meetings automatically appear on the calendar[11].

VI. Conclusion

Our team set out to develop a task management application that allows users to efficiently schedule tasks, and improve their time management. The app features a variety of views that allow users to organize tasks in multiple ways, as well as notification and task prioritization systems to ensure users complete tasks efficiently.

One view that we had added was the calendar view, which we initially hadn't considered essential. This feature was introduced to help users better visualize tasks in real-time with daily, weekly, and monthly views of their schedules. These different views make it simple for users to see all their tasks and when they are due.

Another view that we included was a priority list. Each task has a certain priority number, with some having higher priority than others. This list sorts tasks by their urgency, allowing users to address those that are most important first. In addition, daily notifications linked to both the calendar and priority list assist users with staying on top of each task.

In developing the Software Test Plan, we focused on verifying that all functionalities performed as expected. We provided examples for our task priority method and task priority queue data structure, using JUnit 4 to test functionality across various scenarios. This approach ensured that our application functioned correctly under different conditions.

Our project achieved its main objectives by creating a simple yet effective time management application with intuitive task, priority, and calendar features. Moving forward, we could add more customization options for users, such as implementing custom scheduling routines, as well as support for integration with third-party calendar apps. The simplicity and reusability of our program combined with our agile development methodologies makes our project highly scalable for future growth.

VII. References

[1] I. Kurkina, “Average Cloud Computing Costs for Small Businesses,” Academy SMART, Aug. 11, 2023. https://academysmart.com/insights/average-cloud-computing-costs-for-small-businesses/

[2] Apple, “MacBook Pro,” Apple, 2023. https://www.apple.com/macbook-pro/

[3] Atlassian, “Jira Pricing - Monthly and Annual” https://www.synthesia.io/learn/training-videos/training-cost

[4] “Junior Software Engineer salary (November 2024),” Zippia, https://www.zippia.com/salaries/junior-software-engineer/ (accessed Nov. 8, 2024).

[5] “2022 Senior Software Engineer Salary in US (Updated Daily) | Built In,” *builtin.com*. https://builtin.com/salaries/dev-engineer/senior-software-engineer

[6] “2022 Scrum Master Salary in US (Updated Daily) | Built In,” *builtin.com*. https://builtin.com/salaries/project-mgmt/scrum-master

[7] “2023 Product Owner Salary in US (Updated Daily) | Built In,” *builtin.com*. https://builtin.com/salaries/product/product-owner

[8] K. Alster, “How much does employee training cost? (2024 Guide),” www.synthesia.io, 2024. https://www.synthesia.io/learn/training-videos/training-cost

[9] “Features,” Todoist, https://todoist.com/features

[10] Monday.com, “monday - team management software | monday.com,” *monday.com*, 2023. https://monday.com/

[11] “What can you do with Calendar? - G Suite Learning Center,” support.google.com. https://support.google.com/a/users/answer/9302892?hl=en