

*[Provide a short overview of your project to entice readers to read your project portfolio]*

*[Team Name]*

Project Portfolio

*[Portfolio Due Date]*

[Introduction <Milestone 1: Proposal> 2](#_Toc125988975)

[The [Team Name] Team < Milestone 1: Proposal > 3](#_Toc125988976)

[System Requirements < Milestone 1: Proposal > 4](#_Toc125988977)

[Requirements [optional] 4](#_Toc125988978)

[Epics [optional] 4](#_Toc125988979)

[Epic #1 4](#_Toc125988980)

[User Stories 4](#_Toc125988981)

[User Story #1 4](#_Toc125988982)

[Project Management 5](#_Toc125988983)

[Continuity of Operations Plan (COOP) < Milestone 1: Proposal> 5](#_Toc125988984)

[Project Plan 0](#_Toc125988985)

[System Architecture Design and Development < Milestone 1: Proposal & Milestone 2: Architecture> 0](#_Toc125988986)

[System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation> 0](#_Toc125988987)

[Project Postmortem <Postmortem> 0](#_Toc125988988)

[Project Wins 0](#_Toc125988989)

[Root Cause Analysis 0](#_Toc125988990)

[Lessons Learned 0](#_Toc125988991)

[System Design <Milestone 2: System Architecture> 1](#_Toc125988992)

[System Architecture <Milestone 2: System Architecture> 1](#_Toc125988993)

[Component Design 1](#_Toc125988994)

[Data Flow 1](#_Toc125988995)

[System Components <Milestone 3: System Implementation> 1](#_Toc125988996)

[Component [Component Name 1] 1](#_Toc125988997)

[Component [Component Name 2] 1](#_Toc125988998)

[Component [Component Name n] 1](#_Toc125988999)

[Design Pattern <Milestone 3: System Implementation> 1](#_Toc125989000)

[Design Pattern <Milestone 3: System Implementation> 2](#_Toc125989001)

[System Implementation <Milestone 3: System Implementation> 3](#_Toc125989002)

[Project portfolio template directives and placeholders (delineated by “[ ]” or “< >” and/or highlighted or optional sections not included) should be removed from the document prior to submission. Empty sections for inclusion in later submissions may remain in the document for early submissions.]

[IMPORTANT: All diagrams developed using Enterprise Architectures must include the following acknowledgement: “Thanks to SPARX Systems for LSU student and faculty use of Enterprise Architect for academic purposes”.]

# Introduction <Milestone 1: Proposal>

[*Provide a 1-2 paragraph description of the problem and proposed solution. You will want to include the technologies that are incorporated within your project design and implementation plan*.]

In daily rush, people forget essentials like keys or water bottles. A task reminder app with dynamic checklists based on the time of day helps users remember to pack before leaving. It prompts users to ask questions to professors and supports routines like drinking water regularly. With real-time access, users can set reminders for events, like bringing an umbrella based on weather forecasts. The app simplifies daily routines by offering straightforward reminders tailored to users' needs.

[*Include a figure that includes the logos of the technologies that your project employs*. *IMPORTANT: The core of the system architecture must be implemented in an object-oriented programming language (i.e., Java, C#, or C++)* ]

Core Features:

* Time-Based Checklists
* Planner Functionality
  + Upload or set a class schedule
  + Add tasks for each day (add deadline for tasks)
* Dashboard
  + Displays tasks organized based on when their set deadlines are

Viable Features:

* Weather Warnings
* Health-Conscious Reminders
  + Water Goals
  + Step Goals
  + Physical Activity Suggestion

Stretch Features [optional]

* Location based Task-Option (user wants to be reminded before/after they leave somewhere)
* Real time weather updates?

# The Group 4 Team < Milestone 1: Proposal >

[*Provide the team structure. This should include the team member name, role(s), and responsibilities. If team members have different roles/responsibilities for different project milestones, these should be listed by milestone*. *This section should be provided during the Proposal phase, but it should be updated as roles change for different milestones. The team GitHub link should be included in this section.*]

Group 4: Task App

GitHub link: <https://github.com/hkaiserteaching/csc3380-spring-2024-project-group-4>

Andrew Nguyen - Frontend/Backend

Justin Nguyen – Front End/Backend

Nathan Hoang - Backend

Amin Taher – Front End

Caitlynn Sengchiam – Front End (UI/UX Design)

Duc Vu - Project Manager / Group Leader

Web app approach:

* Front End Roles (UI/UX Design)
  + React or JavaFX or Qt dev
    - Make components in one of these to render the information from the backend
  + Team Members
    - Andrew
    - Justin
    - Amin
    - Caitlynn
    - Duc
* Back End Roles
  + Database Setup
  + Server/Endpoint creation
  + Team Members
    - Duc
    - Andrew
    - Nathan

# System Requirements < Milestone 1: Proposal >

## Requirements [optional]

[*A list of system requirements. This should include, at a minimum, the requirements imposed by the class project*.]

## Epics [optional]

[*A list of system epics. Epics are similar to user stories, but they are more broad; epics cannot be completed in a single sprint. Epics follow the same format as user stories.*]

### Epic #1

[*Epic Statement, using the following format:*

*As a \_\_\_\_\_\_\_\_\_\_, I want to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, so I can \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(optional).*]

## User Stories

[*A list of 1-2 user stories; additional user stories are optional.* ]

### User Story #1

[*User Story Statement, using the following format:*

*As a \_\_\_\_\_\_\_\_\_\_, I want to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, so I can \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(optional).*]

As a student, I want to create time-based tasks, so I can remember to grab my things before and after class.

As a person with a busy routine, I want to be reminded of when to complete certain tasks, so I can focus on other things, like work.

# Project Management

## Continuity of Operations Plan (COOP) < Milestone 1: Proposal>

[*The team should discuss how they plan on communicating and coordinating their efforts. This should include a contingency plan in case one or more team member is unable to meet in-person (e.g., COVID-19 quarantine) or suddenly becomes unavailable (temporarily, such as illness or injury, or permanently, such as dropping the class). The Continuity of Operations Plan is a paragraph or two capturing this discussion.*]

The Group 4 team plans on communicating through frequent Discord voice calls and in person meetings. The Project leader, Duc, will coordinate these meetings. These meetings will begin with team members providing progress reports on their current task and allow for team input and feedback on their respective implementations. The discord chat will also be utilized to provide updates in case a team member is unable to meet, or if not voice call or in person meeting is held in the first place.

In case of a team member being unable to meet in person, discord is always an option for communication. If they become temporarily or permanently unavailable due to unforeseen circumstances, other team members can pick up the work needed to replace the leaving member. Many members are well-versed in all areas of the project and can divide the work between available/active members.

## Project Plan

### System Architecture Design and Development < Milestone 1: Proposal & Milestone 2: Architecture>

[Milestone 1 (Proposal): The Project Plan WBS provides a list of activities/tasks to be undertaken to complete Milestone 2 (Architecture). The WBS activity chart should include task dependencies, estimated level of effort, and expected start and completion dates.

Milestone 2 (Architecture): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| 1 | Setup Backend Architecture: Choose Framework and Basic Implementation | 1 | Low |  | 02/09 | 02/21 |  |  |
|  | Backend: Implement Database | 1 | Medium |  | 02/15 | 02/21 |  |  |
|  | Backend: Configure Server / Create Necessary Endpoints | 1 | Medium |  | 02/15 | 02/21 |  |  |
|  | Backend: More endpoints to implement viable features | 1 | Low |  | 02/15 | n/a |  |  |
| 2 | Setup Frontend Architecture: Choose Framework and basic implementation | 1 | Low |  | 03/01 | 02/21 |  |  |
|  | Frontend: add routing logic | 2 | Medium |  | 03/01 | 02/21 |  |  |
|  | Develop user interface to create api/backend calls | 2 | High |  | 02/09 | n/a |  |  |
|  | App Testing | Last | High |  | 02/09 | 04/20 |  |  |

### 

### System Implementation <Milestone 2: Architecture & Milestone 3: System Implementation>

[Milestone 2 (Architecture): The Project Plan WBS provides a list of activities/tasks to be undertaken to complete Milestone 3 (System Implementation). The WBS activity chart should include task dependencies, estimated level of effort, and expected start and completion dates.

Milestone 3 (System Implementation): The WBS activity chart for the milestone should be updated to include actual level of effort and start and completion dates.]

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Activity** | **Pre #** | **Estimated**  **Effort** | **Actual**  **Effort** | **Estimated**  **Start Date** | **Estimated**  **Finish Date** | **Actual**  **Start Date** | **Actual**  **Finish Date** |
| 1 | Setup Backend Architecture: Choose Framework and Basic Implementation | 1 | Low | Low | 02/09 | 02/21 | 2/10 | 2/20 |
|  | Backend: More endpoints to implement viable features | 1 | Low | Low | 02/15 | n/a | 2/15 | 2/20 |
|  | Setup Frontend Architecture: Choose Framework and basic implementation | 1 | Low | Low | 03/01 | 02/21 | 2/15 | 2/20 |
| 2 | Frontend: add routing logic | 2 | Medium | Medium | 03/01 | 02/21 | 04/01 | 04/15 |
|  | Develop user interface to create api/backend calls | 2 | High | High | 02/09 | n/a | 04/14 | 04/15 |
|  | App Testing | Last | High | Medium | 02/09 | 04/20 | 04/20 | 04/22 |
|  | Backend: Implement Database | 1 | Medium | High | 02/15 | 02/21 | 03/01 | 03/25 |
|  | Backend: Configure Server / Create Necessary Endpoints | 1 | Medium | Medium | 02/15 | 02/21 | 03/01 | 03/15 |

## Project Postmortem <Postmortem>

### Project Wins

[Provide a bulleted list of at least 3 positive aspects of the project.]

### Root Cause Analysis

[Provide a bulleted list of at least 3 negative aspects of the project. For each negative, provide the answer to the three successive “Why” questions. ]

### Lessons Learned

[For each negative aspect identified in the Root Cause Analysis, provide a mitigation strategy (i.e., what process should be introduced) to ensure that the problem is not repeated in subsequent projects.]

# System Design <Milestone 2: System Architecture>

[*Include a short (1-2 sentences) statement about the system design*.]

Regarding the system design, our group is planning on doing a web app approach which utilizes React for the frontend, providing a dynamic and responsive user interface. On the backend, it employs WebToolKit C++ for efficient processing and communicates with an SQL database, allowing for data management for the planner app's functionalities.

## System Architecture <Milestone 2: System Architecture>

[*A short description of the system architecture.*]

**System Architecture Main Components:**

The system architecture for our project consists of 2 major components:

The frontend, developed with React, will handle inputs and outputs given by the user, ensuring an interactive user experience.

The backend, implemented with WebToolKit C++, will manage data processing and communication with the SQL database, which will serve as an external data storage for the system.

**Frontend Overview:**

The React frontend communicates with the backend by sending HTTP requests to the specified endpoints. For example:

* When a user logs in,
* To create a new task,
* Updating a task involves .
* Deleting a task

**Backend Overview**

The WebToolKit C++ backend exposes several endpoints to handle various operations:

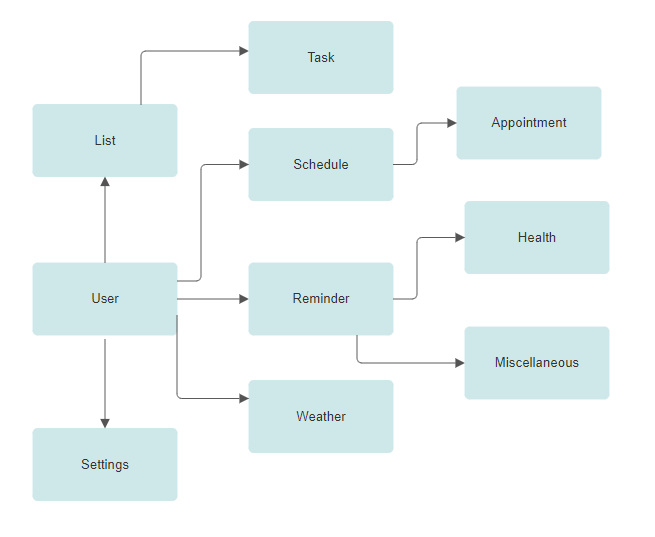
* /api/login: POST request for user authentication, accepting username and password.
* /api/tasks: GET request to retrieve tasks for a specific user.
* /api/tasks/create: POST request to create a new task, accepting task details.
* /api/tasks/update/:id: PUT request to update an existing task identified by its ID.
* /api/tasks/delete/:id: DELETE request to delete a task by its ID.

**Database Overview:**

The SQL database employs a basic schema tailored to the needs of the planner app. This schema might include tables such as:

* Users: storing user information like username, password hash, and email.
* Tasks: containing details about tasks such as task ID, description, deadline, and completion status.
* Categories: holding information about task categories or tags for organizing tasks.
* UserTasks: a join table linking users with their respective tasks.

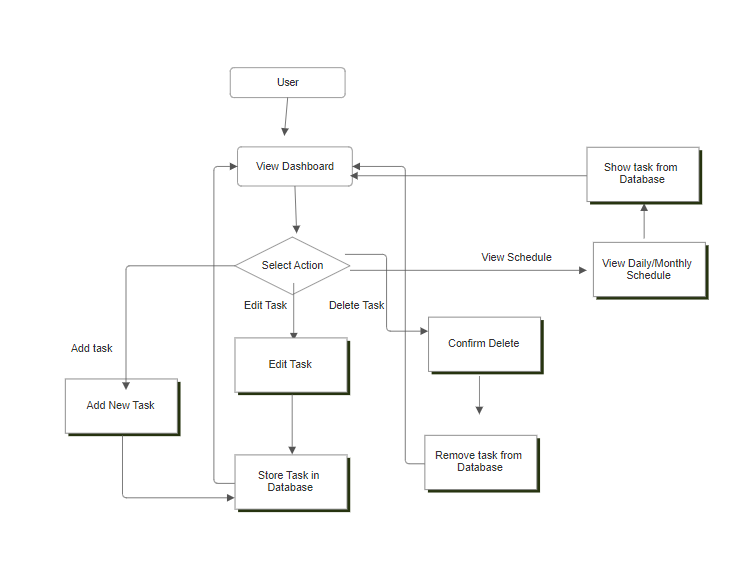
### Component Design

[*Insert image of system architecture component diagram. Include the name of the team member that created the diagram in EA.*] Duc Vu  


[*Architecture overview, to include user I/O, external data sources, and major system components.* ]

### Data Flow

[*Insert image of system architecture data flow diagram. Include the name of the team member that created the diagram in EA.*] Creator: Justin Nguyen

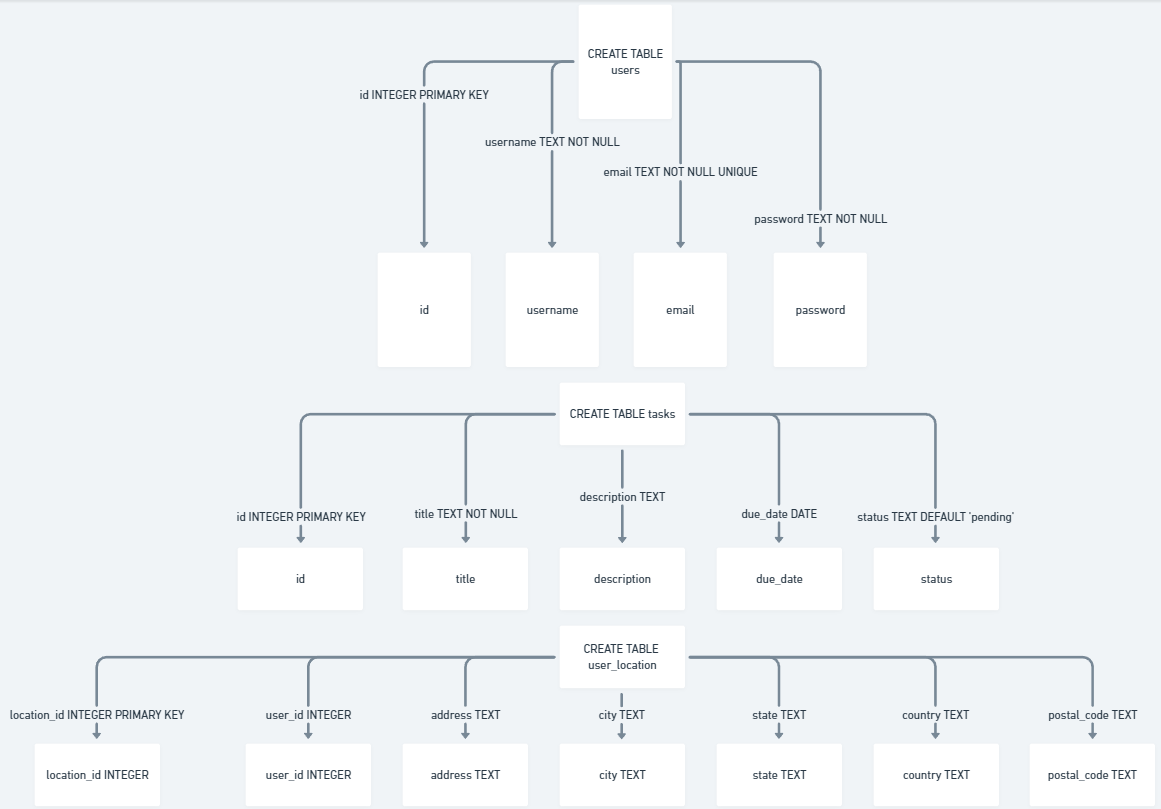


[*Architecture data flow discussion: a high-level description of the data between both internal major components and external data sources.*]

## System Components <Milestone 3: System Implementation>

[*Include a component sub-section for each component in the architecture diagram. Each component subsection will include a class diagram*]

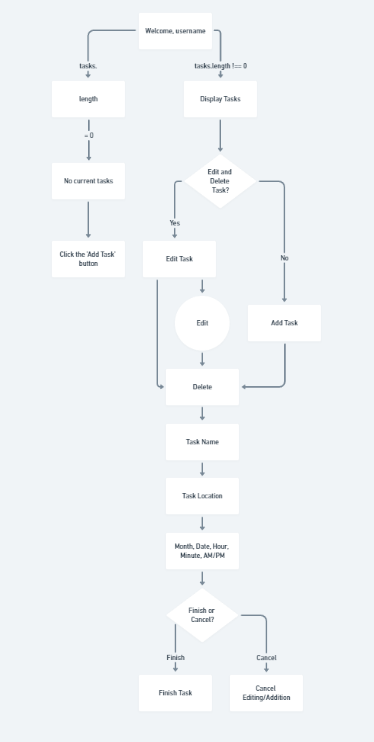
### Component [Database Query 1] - Duc Vu

I've created the backend infrastructure and database setup. This includes SQL queries along with dummy data for initial testing. Additionally, I've designed a UML diagram to provide a visual representation of the database structure. The user query contains fields for user information such as id, username, password, and email. This query facilitates the addition of user data to the database. The task query includes fields such as id, title, description, due date, and user id. It serves to manage tasks and records the tasks completed by users. The user location query comprises fields like location id, user id, address, city, state, country, and postal code. This query is responsible for storing user location information within the database. 

[*An EA class diagram of the component that includes method parameters. Include the name of the team member that created the diagram in EA.*]

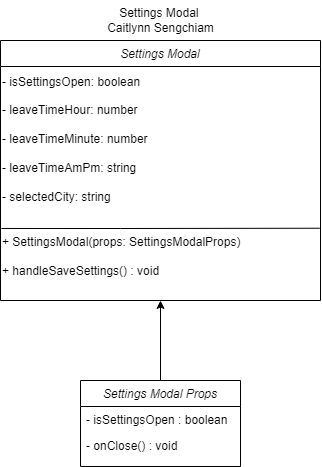
Component [Sorting Tasks/Delete Button/Upcoming Tasks Popup] - Justin Nguyen

Implemented a feature that sorts tasks by their scheduled date and time, ensuring the most imminent task appears at the top of the list. As well as a delete button enabling users to remove tasks.



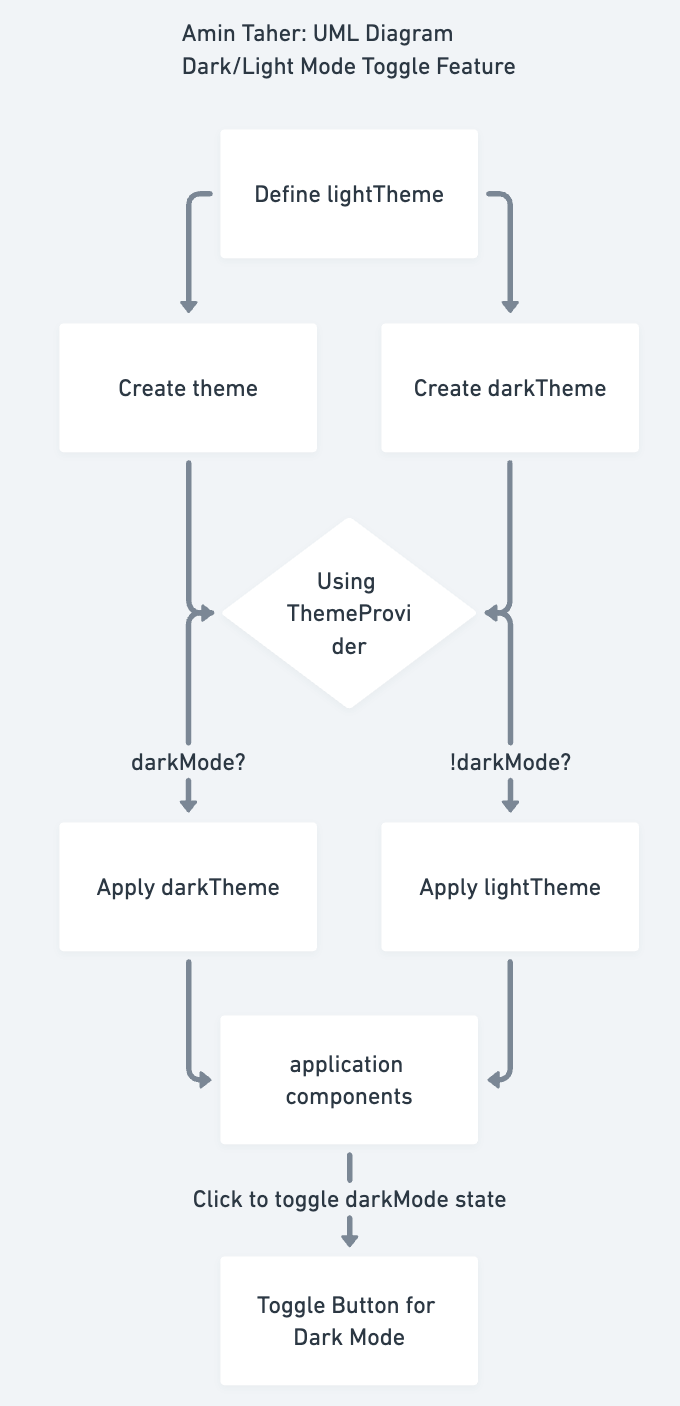
### Component [Settings Modal] - Caitlynn Sengchiam

The settings modal is a type of pop-up window that is accessed from the user by clicking the settings button. In this window, the user inputs the time they plan to leave for each day as well as a location.



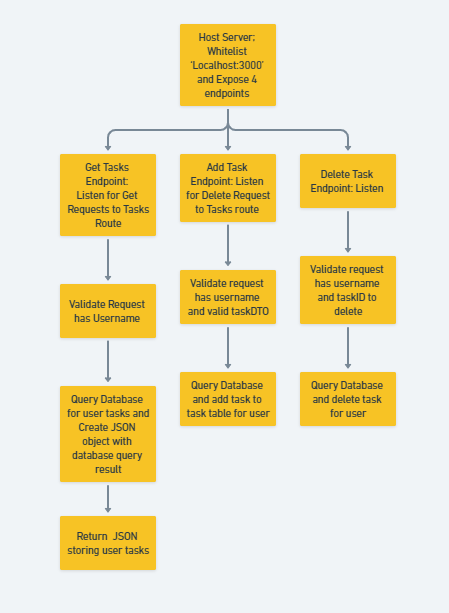
### Component [Dark/Light Mode] - Amin Taher

This component is a feature of the dashboard within the app. It is a simple toggle button that changes the user interface between dark and light mode. Created by Group Member Amin Taher

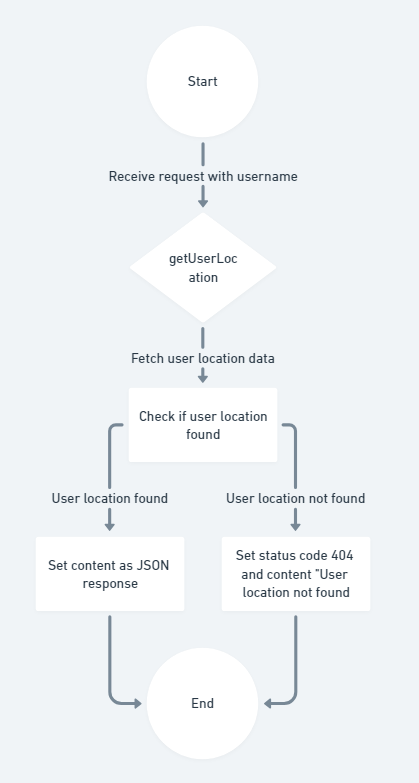


### Component [Planner Server] - Andrew Nguyen

Setup C++ Web-server created using httplib and rapidJSON libraries. Exposes endpoints to Get tasks, Create task, Delete Task by performing Database Queries for Each Endpoint and return any needed data as Json using rapidJSON library to format

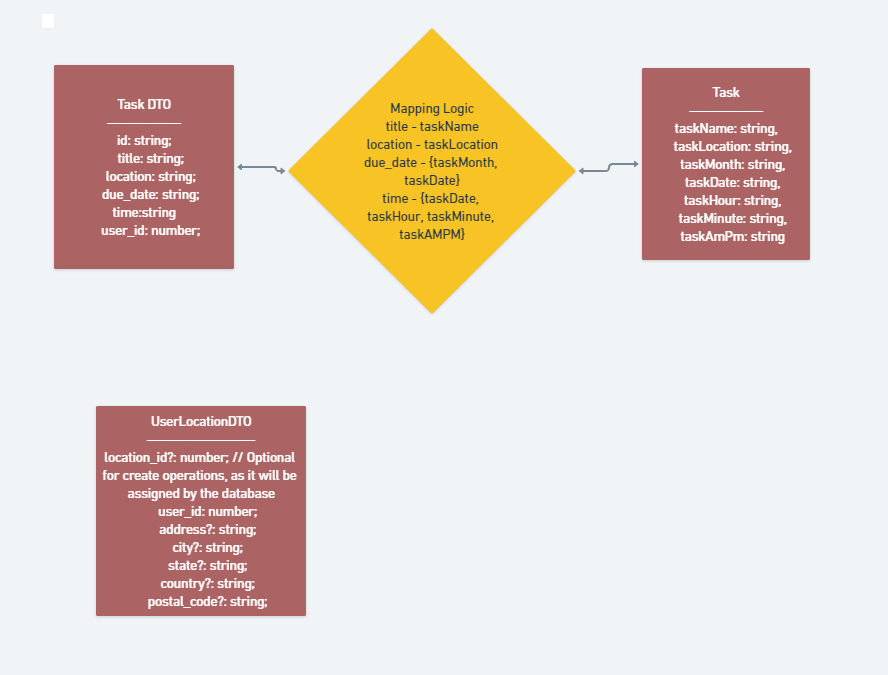


### Component [User Location Endpoint]- Nathan Hoang

date

Creates an endpoint for the user location to the webserver to obtain from the SQL database from the query called username. It then connects to the database and fetches the user location data based on the provided username. It then constructs a JSON response that contains the username and location retrieved from the database. If the user location isn’t found in the database, then it will give an error to where it's not located in the database.

## Design Pattern <Milestone 3: System Implementation>

Andrew Nguyen

## Design Pattern <Milestone 3: System Implementation>

[*Class diagram of design pattern incorporated into the project. Pattern must be specific to the project and not a general design pattern class diagram. Include the name of the team member that created the diagram in EA. A second design pattern may be included for bonus points.*]

# System Implementation <Milestone 3: System Implementation>

[*In the table below, include a row for each component in your System Architecture diagram. In the second column, list the programming language(s) used to implement the component and the what % of that programming language is used in the implementation. In the third column, list the team member(s) that implement the component and what % of that implementation was completed by that team member. IMPORTANT NOTE: All architectural components must be implemented by an object-oriented programming language: Java, C++, or C#.*]

|  |  |  |
| --- | --- | --- |
| **Architectural Component** | **Programming Language(s) %** | **Team Member(s) %** |
| *Planner Server – C++ webserver* | *C++ - 100%* | *Andrew Nguyen: 33%*  *Nathan Hoang: 33%*  *Duc Vu 33%:* |