Software Design Document

Document Name: Team 1_D2.pdf

Rest Johnson (801379538)
Abi Olukeye (801177435)
Eric Betties (801208523)
Kenneth Fields (801405755)

1. Project Overview

PhD students often face the challenge of managing a complex array of academic and career milestones. These include exams, research, dissertation deliverables, publications, internships, and job search activities. The lack of centralized tools to organize this journey can lead to missed deadlines and opportunities, disorganized records, and limited self-awareness about progress. Student Advisors and administrators also need better visibility through dashboards, notifications, viewable and exportable reports, and secured portals into student progress to provide timely feedback and support.

1.1 Stakeholders

i. **PhD Students (Primary Users)**: Use the system to track academic progress, manage goals, receive reminders, append and examine relevant milestones.
ii. **Student Advisors**: Monitor the progress of students, inclusive of their milestones and other relevant academic requirements; Provide necessary feedback for students' progress, and recommend relevant resources.
iii. **Administrators**: Track students' progress to generate relevant reports and assess program performance.
iv. **Developers**: Design and implement a scalable, functional, user-friendly, and secure PhD Student Milestone Tracking system.

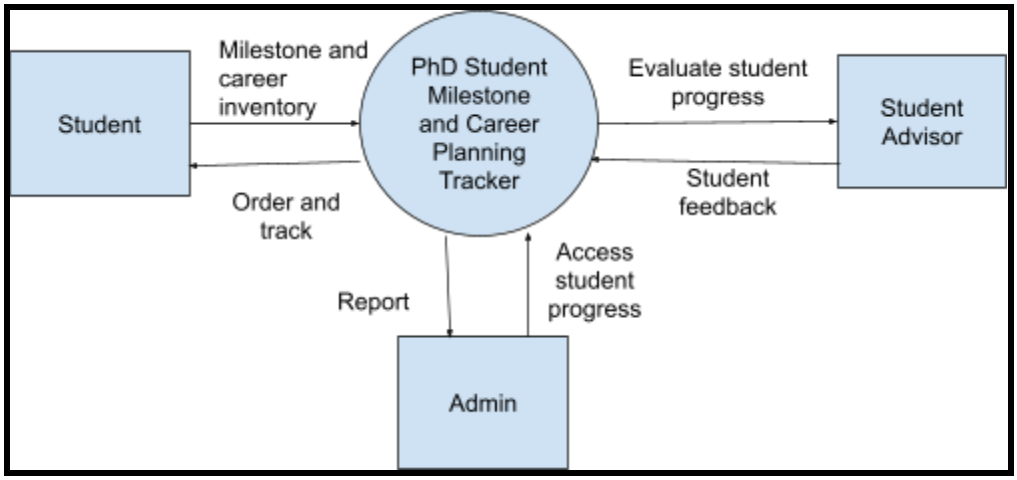
1.2. Solution Summary

The **PhD Milestone and Career Tracker (PMaCT)** is a web-based application designed to help doctoral students manage their academic and career development. The key features of this application include:

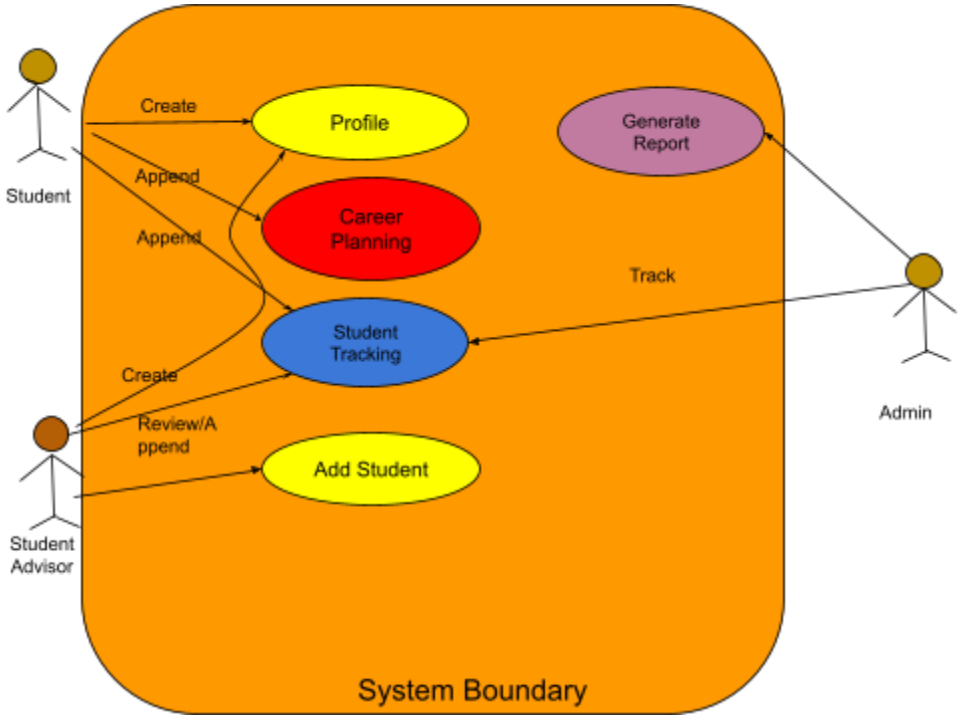
v. **Academic Milestone Tracking** (qualifying exams, proposal defense, dissertation defense, publications)
vi. **Career Planning and Goal Tracking**
vii. **Visual Dashboards** (Students, Student Advisors, Admins) for progress insights
viii. **Reminders and Notifications** (Milestones Creation, Updates, Completion) with configurable channels

1.3. General Model

ix. **Context Diagram**: The PMaCT system interacts with the students, advisors, and administrators to achieve a functional system



x. **Use Case Diagram**: The actors in this system are Students, Student Advisors, and Admins. The core Use Cases include creating and editing profiles, appending milestones, setting career plans, tracking progress dashboard, student advisor feedback, and student management.



xi. **User Stories:**

1.1.xi.1. **As a PhD student, I want to create a profile with my specific milestones (qualifying exam, proposal, defense) so that I can keep track of everything in one place.**

1.1.xi.2. **As a PhD student, I want to receive reminders or notifications for upcoming deadlines (proposal submission, conference dates, etc.) so that I don't miss any important tasks.**

1.1.xi.3. **As a PhD student, I want to see notes for Milestones from the admin so that I can respond to milestone notes**

1.1.xi.4. **As a PhD student, I want to update my career goals (target industries, desired job roles) so that I can track my progress and readiness for the job market.**

1.1.xi.5. **As a student advisor, I want to be able to track and verify PhD students' progress so that I can keep them on track.**

1.1.xi.6. **As a PhD student advisor, I want to create and update my secured user profile so that my personal information is up to date and accurate.**

1.1.xi.7. **As a student advisor, I want to see notifications of student milestones (papers, comp exams, research assignments, dissertation) deadlines and approvals and student advisor feedback so that I can complete all tasks on time.**

1.1.xi.8. **As a student advisor, I want to be able to provide feedback to students so that the foundational knowledge, literature review, methodologies, and resources are communicated early on.**

1.1.xi.9. **As an admin, I want to be able to measure the progress of PhD students based on their required milestones so that I can outline progress and keep students on track.**

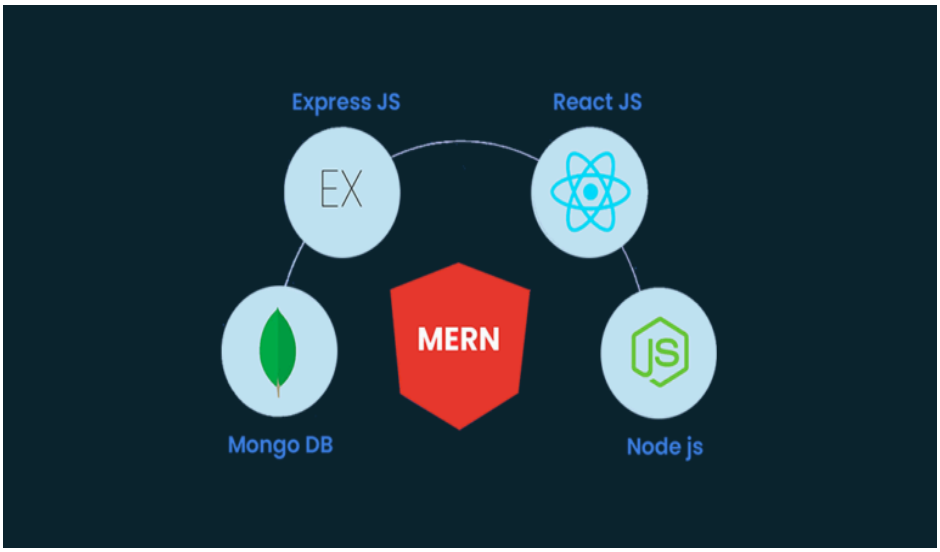
1.1.xi.10. **As an admin, I want to create and update my secured user profile so that my personal information is up to date and accurate.**

1.1.xi.11. **As an admin, I want to post notes of student milestones, so that the student has visibility and responds timely to my comments**

1.1.xi.12. **As an admin, I want to view student details (Major, Program, Assigned Student), so that I can generate and export reports for student information.**

1.1.xi.13. **As an admin, I want to view student progress so that I can generate and export reports for student progress.**

2. Architectural Overview



MERN stack architecture

This system follows a **MERN** stack architecture (MongoDB, Express.js, React, Node.js) to support a modular, scalable, and maintainable design. The **full-stack JavaScript ecosystem** provided by MERN allows consistency across the entire stack through the use of a homogeneous language from the front end to the backend and even database interactions via Node.js drivers. This helps the team to develop faster with shared tooling, coding standards, and reusable libraries. The **React** component-based structure provides an easy-to-work-with modular interface for profile sectioning, milestone formation, dashboard, and resource widgets. It also supports reactive, real-time updates that allow instant feedback as changes are made to the database or UI. It also presents an easy integration option for visualization to render milestone progress and tracking visually. **RESTful** APIs using Express are fast to build and also easy to maintain. They provide flexibility when it comes to the integration of external services. Our project involves the use of complex and varied data such as academic milestones, career goals, and possible reminders, all of which may evolve; **MongoDB**'s schema-less structure supports rapid prototyping and flexibility, making it easy to nest related data and reference collections, such as the relationship between actors. Other considered architectures included:

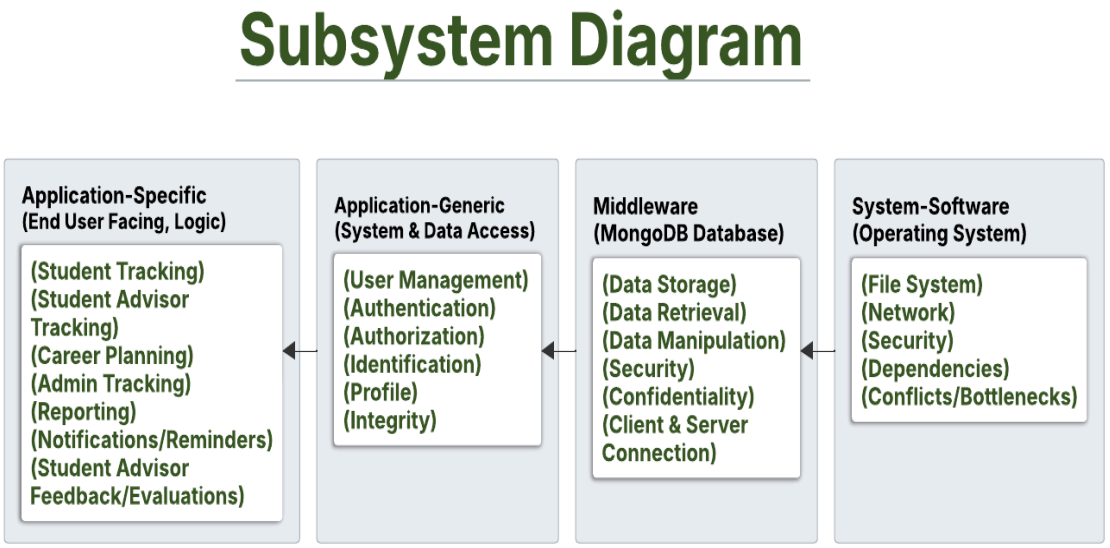
- **Firebase Architecture**: This provides a real-time database syncing, authentication, and cloud functionality. However, the MERN stack, through Mongoose, offers more control over data relationships and custom validation. Our project requires a flexible querying platform for milestone analytics and a dashboard; Firebase querying capabilities are limited compared to what MongoDB offers. Firebase is excellent for lightweight apps but MERN provides the power and control that is needed for the deep customization and scalability of our project.
- **Monolithic Architecture**: This combines all components, UI, business logic, and database into one single application. It is typically faster to prototype, but it introduces some form of significant technical debt in the long run. Our MERN architecture cleanly separates the frontend (React), backend Node/Express), and data (MongoDB), making it

easier to modify individual layers. As our software grows, updates can be effectively isolated to specific services and components.

- **Serverless Architecture**: This uses cloud functions, such as AWS Lambda, that run on demand. It removes infrastructure management and scales projects automatically. Implementing the functionalities of our project requires persistent user sessions and relevant updates to the user dashboard. Serverless functions are stateless, and managing these sessions requires extra complexity. Features like the dynamic dashboard or advisor-student data sync would be difficult to implement without a persistent backend. Although this architecture makes sense for atomic, stateless tasks, PMaCT's RESTful API design with user authentication, user-specific data, and nested resources aligns better with Express.js.

2.1. Subsystem Architecture

- We are using the four-layered subsystem architecture **application-specific, application-generic, middleware, and system-software**), with each layer accountable for and ensuring the PhD student milestone tracking application-driven functionalities and architectural styles in alignment with layered architecture for expandability, modular configuration, fortified horizontal and vertical structure, and flexible database design.



4 Layer Subsystem Structural Responsibility and Packages Details

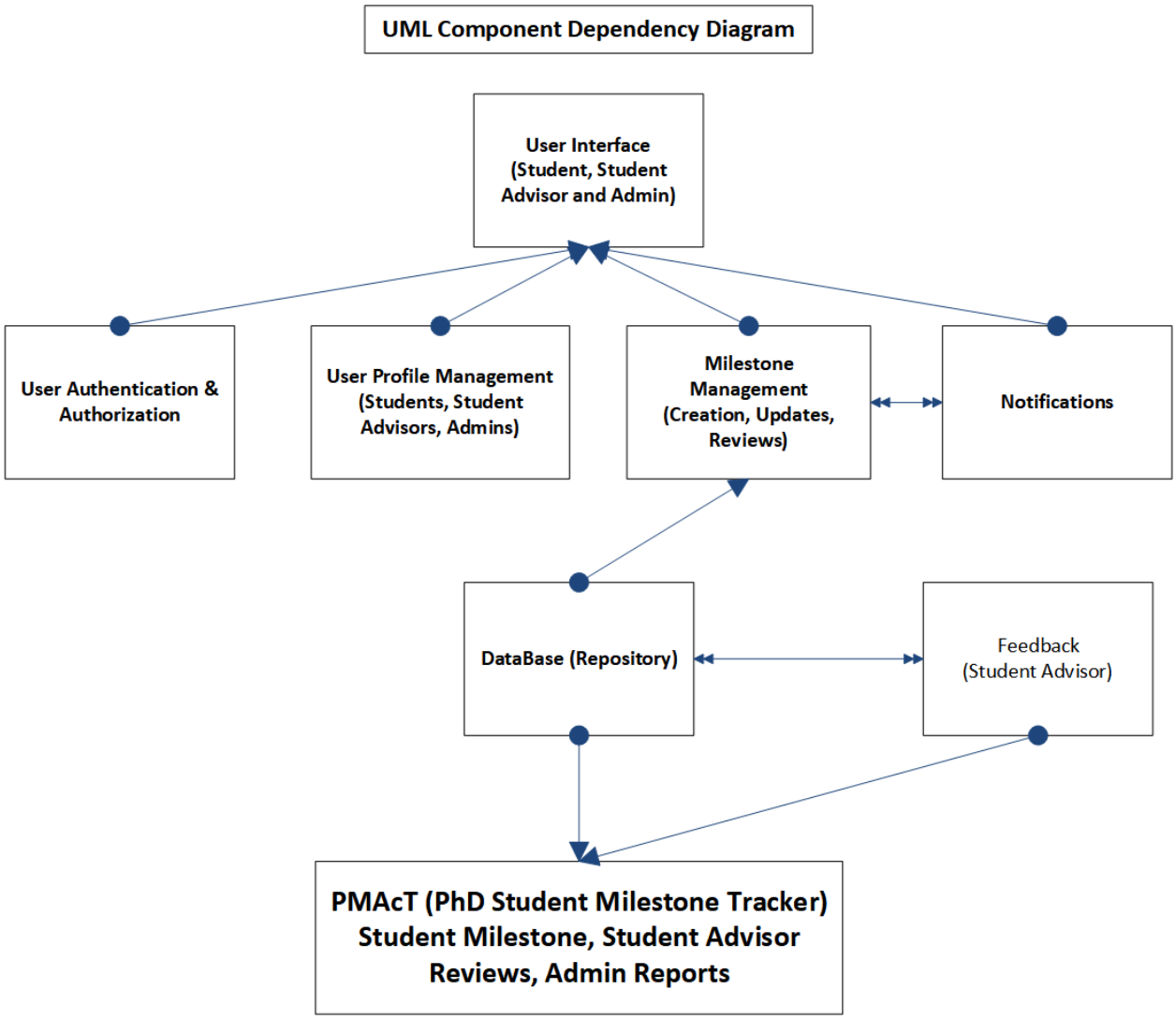
- **Application-Specific Layer:**
  - **Responsibility:** Manages system user access of user profiles and appropriate functional capabilities for PhD Student Milestone Tracking. Handles the specific functionalities of the PhD student tracking system, including the user facing interface, milestone tracking logic and procedures, application-specific data requirements.
  - **Packages:**
    - Student Tracking: Manages and tracks milestones, milestone reviews, profile, progression, assistance from student advisors and admin, research assignments, dissertation defense, written and oral comps, and dissertation throughout respective PhD program enrollment.
    - Student Advisor Tracking: Guides, ensures, and accounts for assigned student progression towards PhD. Keeps students and admin informed of

student progression and reviews and approves milestones and milestone reviews.
        - Admin Tracking: Provides reports, analytics, student feedback, milestones, milestone reviews, advisor feedback, and student progression.
        - Career Planning: Ensures students and the student advisor devise, approve, and deploy strategic academic career planning. Provides a blueprint for the student to follow in alignment with career goals, PhD program, and research requirements.
        - Student Advisor Feedback/Evaluation: Monitor the success, effectiveness, and results of the student-advisor relationship and activities. Ensure the overall relevance and output of the milestone tracking, career planning, and student progression.
        - Reporting: Admin produces, views, assesses, and distributes student progress, milestones, and deliverable reports.
        - Notifications/Reminders: Supports communication, collaboration, and confirmation of student, faculty advisor, and admin-assigned tasks.

- **Application-Generic Layer:**
    - **Responsibility:** Provides functionalities that are common across multiple applications, such as user management, profile management, authentication, identification, authorization, and data access.
    - **Packages: (System Does this Package)**
        - User Management: This handles user registration, login, and profile management.
        - Authentication: Ensures system user access checks based upon accurate and protected logon credentials.
        - Authorization: Manages user access to well-defined, approved, and applicable system resources.
        - Identification: Ensure that users are who they claim to be during system access requests.
        - Data Access: Ensures that users (Student, Admin, Advisor) view or access the data (i.e., info) in compliance with system user policy and procedures. Protects the data from bad actors or unauthorized users, views, and manipulations. Protects data integrity and confidentiality.
        - Milestone Management: Creates, updates, and tracks student responsibilities (i.e., Papers, research, written and oral comps, dissertation defense, dissertation).

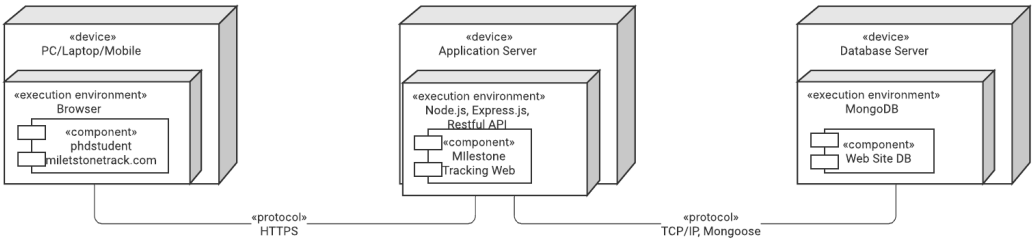
- **Middleware Layer:**
    - **Responsibility:** Handles communication, authentication, security and data exchange between subsystem architectural layers including database (MongoDB), node.js, RestAPIs, and applications (specific and generic).
    - **Packages: (System Does this Package)**
        - Data Storage: Gatekeeper for data security, storage, and retrieval of data.
        - Data Retrieval: Retrieves data from the database in response to users request from system, students, advisors and administrators.
        - Data Manipulation: Enables additions, changes and deletions of data in accordance with system and user needs. Maintains authorized and compliant data alteration in sync with user access and security policies.
        - Security: Ensures and supports the functionality of the system and business users with data security, integrity, and confidentiality.
        - Client and Server Connection: Monitors, establishes, and integrates the connectivity of the client (user interface) and server.

- **System-Software Layer:**
    - **Responsibility:** Maintains the system software (node.js, JavaScript, React interface) dependencies and relationships entailing operating system, database (MongoDB) system, network, bottlenecks, conflicts, and security.
    - **Packages: (System Does this Package)**
        - File System: Manages file storage and access.
        - Network: This handles network communication.
        - Security: Provides security services, such as encryption and access control.
        - Bottlenecks/Conflicts: Resolves and eliminates disruptions and denial of services attributes by virus, malicious actions, and subsystem layer bottlenecks, conflicts, functional contentions, and database contentions.

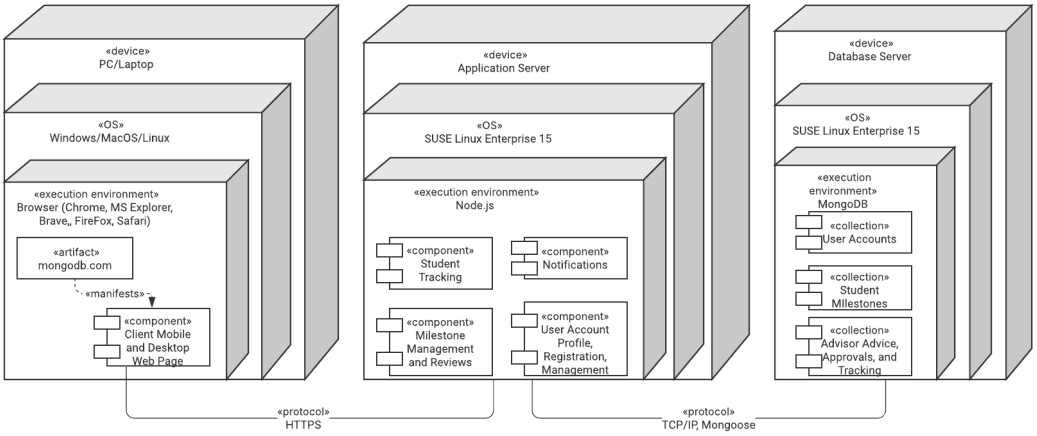

UML Component Dependency Diagram
User Interface (Student, Student Advisor and Admin)
User Authentication & Authorization
User Profile Management (Students, Student Advisors, Admins)
Milestone Management (Creation, Updates, Reviews)
Notifications
DataBase (Repository)
Feedback (Student Advisor)
PMAcT (PhD Student Milestone Tracker) Student Milestone, Student Advisor Reviews, Admin Reports

2.2. Deployment Architecture

High Level UML Deployment Diagram



UML Low Level Deployment Diagram



2.3 Persistent Data Storage

The PMaCT system stores long-term data for users, milestones, progress tracking, advisor interactions, and admin review outcomes. To support this, we will use **MONGO.DB**, a non-relational database, due to its innate ability to model complex relationships, enforce data integrity, and efficiently query structured data.

Storage Approach

We chose a non relational database for persistent storage because the system involves multiple roles (students, advisors, admins), interrelated entities (milestones, approvals, comments, reports), and workflows that require structured querying and atomic updates. This model supports role-based access, filtering, and reporting functions effectively.

Database Schema

Below is the current schema reflecting all core entities and system requirements:

1. Users Table

Stores core user profiles for all roles.

Column Name	Data Type	Description
id	UUID	Primary key
first_name	TEXT	User’s first name
last_name	TEXT	User’s last name
email	TEXT	Unique login email
password_hash	TEXT	Secure hashed password
role	TEXT	`student`, `advisor`, or `admin`
program	TEXT	Student’s academic program
department	TEXT	Affiliated department
created_at	TIMESTAMP	Account creation date

2. Milestones Table

Represents academic milestones created and tracked by students.

Column Name	Data Type	Description
id	UUID	Primary key
student_id	UUID	Foreign key → Users(id), only students
title	TEXT	Milestone name (e.g., "Thesis Proposal")
description	TEXT	Milestone description
due_date	DATE	Target completion date
status	TEXT	`not-started`, `in-progress`, `completed`
created_at	TIMESTAMP	Date milestone was created

3. MilestoneApprovals Table

Captures advisor and admin interactions during milestone review.

Column Name	Data Type	Description
id	UUID	Primary key
milestone_id	UUID	Foreign key → Milestones(id)
reviewer_id	UUID	Foreign key → Users(id), advisor or admin
role	TEXT	`advisor` or `admin`
status	TEXT	`approved`, `changes-requested`, `pending`
comments	TEXT	Reviewer comments or notes
reviewed_at	TIMESTAMP	Timestamp of milestone review

4. Tasks Table (Subtasks)

Subtasks allow detailed progress tracking for each milestone.

Column Name	Data Type	Description
id	UUID	Primary key
milestone_id	UUID	Foreign key → Milestones(id)
title	TEXT	Task description
is_done	BOOLEAN	Completion status

5. Notes Table (Admin → Student Notes)

Stores notes added by admins on a student's profile.

Column Name	Data Type	Description
id	UUID	Primary key
student_id	UUID	Foreign key → Users(id), role: student
admin_id	UUID	Foreign key → Users(id), role: admin
note	TEXT	Admin's feedback or internal note
created_at	TIMESTAMP	Timestamp of note creation

6. Reports Table (Generated by Admin)

Stores milestone status reports per student, generated by the admin.

Column Name	Data Type	Description
id	**UUID**	**Primary key**
student_id	**UUID**	**Foreign key → Users(id)**
generated_by	**UUID**	**Admin user who generated the report**
content	**JSONB**	**Snapshot of milestone status and progress**
created_at	**TIMESTAMP**	**Time the report was generated**

Entity Relationships Summary

- A student user can have many milestones.
- Each milestone can have multiple tasks and reviews.
- Student Advisors and admins review milestones via MilestoneApprovals.
- Admins can leave notes and generate reports for any student.
- All user actions and relationships are bound by the role attribute in the Users table.

2.4. Global Control Flow

The PMaCT system adopts a primarily event-driven control model, supported by some procedural sequences during onboarding and specific workflows (e.g., milestone review). It is a responsive, user-driven application with no hard real-time constraints and limited concurrency concerns typical of modern web applications.

Event-Driven or Procedural Flows

The system is fundamentally event-driven. User actions—such as logging in, submitting a milestone, approving progress, or navigating to dashboards—trigger event handlers that manage application logic, state changes, and backend API interactions. This model allows different users (students, advisors, admins) to interact with the system based on their roles, in a non-linear order.

For example:

• Students can create and update milestones in any order and track their progress interactively.

• Student Advisors may review milestones submitted by any student assigned to them and decide to approve or request changes.

• Admins can generate milestone reports at any time, initiate status review requests, and provide final feedback to students.

However, certain flows are procedural. For example:

• The student onboarding process follows a sequence: registration → login → complete profile → access dashboard.

• The milestone review process by admins follows: report generation → request advisor input → submit final feedback to the student.

These procedural workflows are embedded within the broader event-driven framework of the system.

Time Dependency

The system does not operate in real time and has no strict time constraints. It is an event-response system, meaning that it reacts to user inputs or backend responses rather than operating on a timed schedule. However, some time-sensitive features exist:

• Milestone deadlines and reminders: These may trigger notifications to students, advisors, or admins when a due date is approaching or missed.

• Admin-triggered review cycles: Admins may initiate a time-bound review request (e.g., give student advisor 7 days to respond), though enforcement is business logic-driven, not real-time.

These features can be supported using background tasks or scheduled jobs, such as cron-based functions, to check dates and send automated reminders. Still, they are asynchronous and tolerant of delay, so the system is not real-time.

Concurrency

Concurrency is handled in the backend through asynchronous request processing, not via manual thread management:

• Frontend (React) is single-threaded. It uses event-based logic and React hooks to manage state updates asynchronously.

• Backend (Node.js/Express API routes) handles concurrent HTTP requests via an event loop, supporting multiple user actions simultaneously.

• Database (MongoDB) supports concurrent reads/writes and uses transactions where needed to prevent data conflicts.

Although no custom multithreading is introduced, backend components like milestone approval workflows or report generation may trigger concurrent operations, such as:

• Processing multiple milestone updates from different students.

• Advisors reviewing several milestones at once.

• Admin is running batch report generation while other users interact with the system.

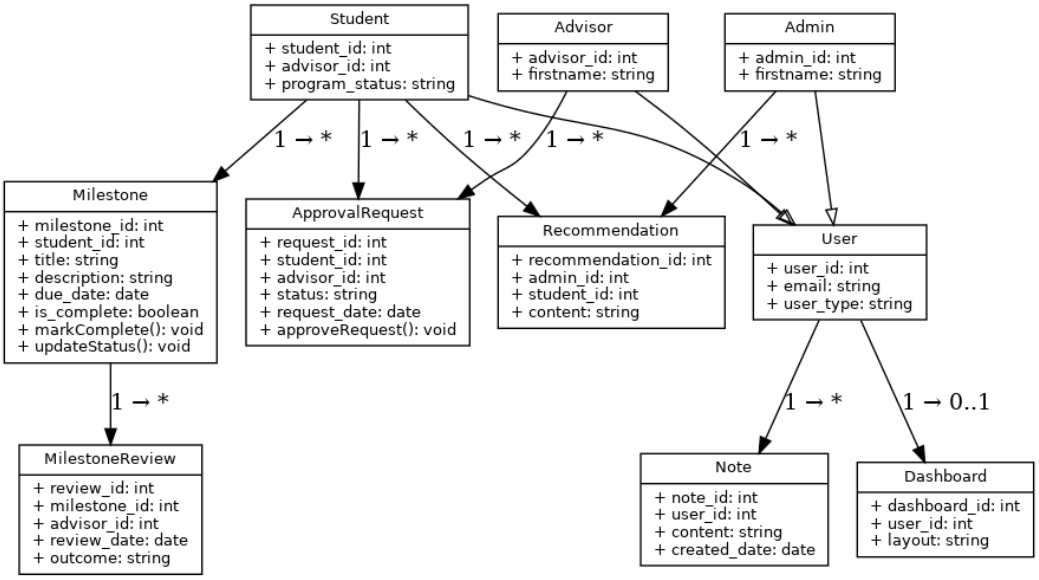
These operations are isolated and can be handled concurrently via asynchronous processing or background jobs.

3 Detailed System Design

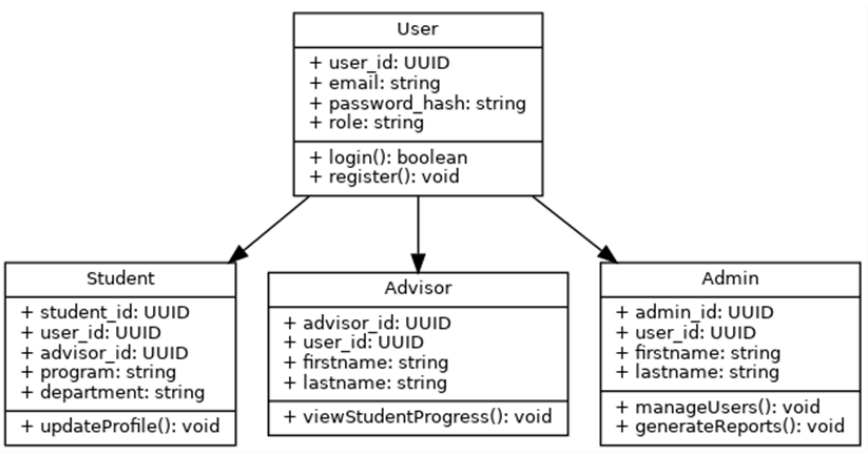
3.1 Static view

This section describes the static design of the PMaCT system. It comprises well-defined modules that support the architecture. Unified Modeling Language (UML) diagrams are used to represent the internal structure and relationships of the components. The justifications and applied design patterns are provided for each module.

System Overview - UML Diagram



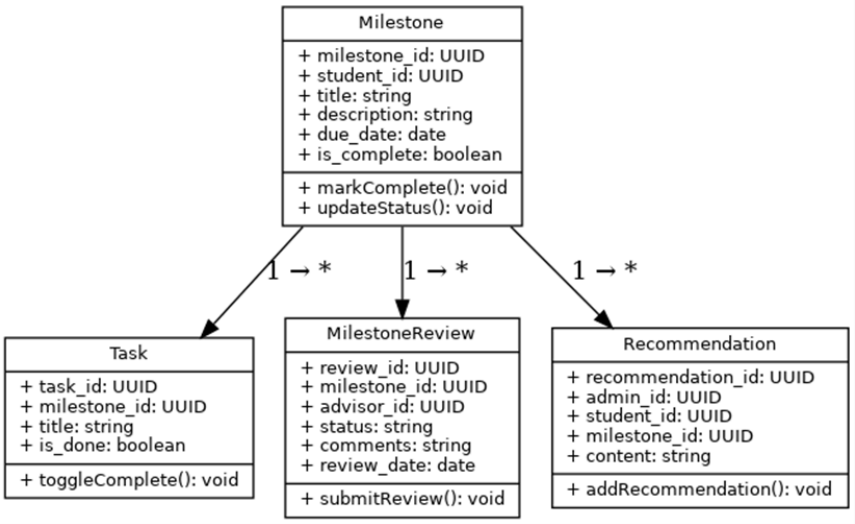
User Management Module



Design Justification: This module defines the hierarchy of the user and their inheritance. The superclass :User" provides common operations and fields and the logic of roles are in"Student ","Advisor ", and "Admin".

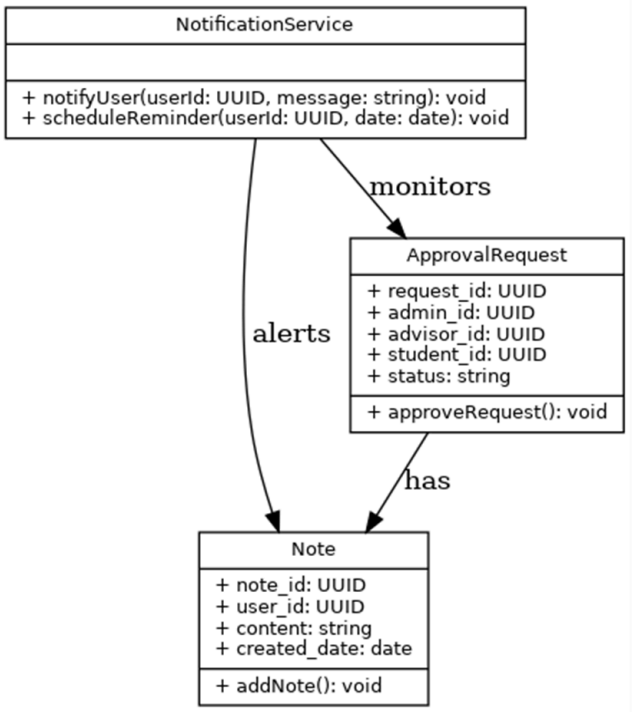
Design Pattern: Strategy (role-specific behaviors)
Alternative: is to use a role attribute in one class, which would make a more complicated design.

Milestone Tracking Module



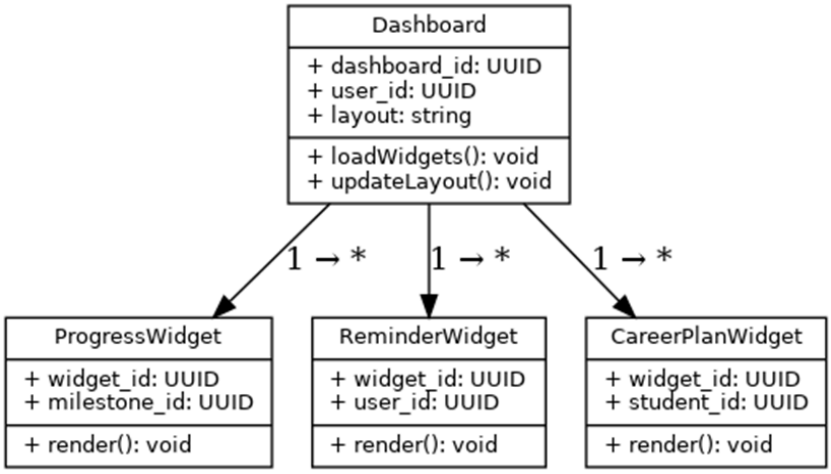
Design Justification: Handles creation and management of academic milestones. The "Milestone" class tracks academic elements/progress. Subtasks are broken down using "Task". "MilestoneReview" allows evaluation by the advisor. "Recommendation" allows admin to give feedback.
Design Pattern: Composite (nested tasks)
Alternative: Merging tasks and reviews into "Milestone" reduces flexibility.

Notifications and Feedback Module



Design Justification: This component is responsible for communication and feedback.
“ApprovalRequest” records sanctions needing review.
“Note” allows admins to comment, and “Notivican Service” sends out alerts.
Design Strategy: Observer(reminder tigger), Command (scheduling of future notifications)
Alternative: Use of external notification platforms, which limit flexibility and control.

Dashboard Module

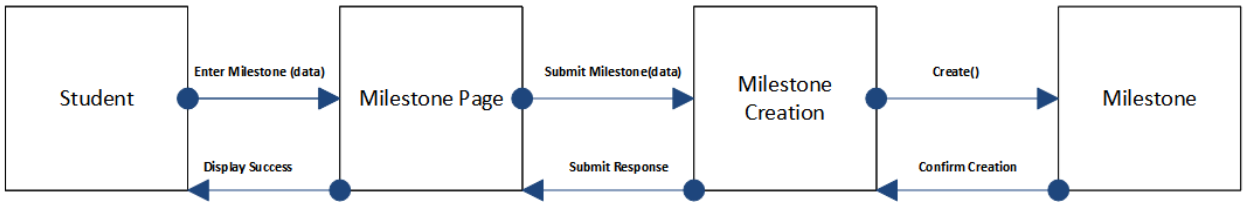


Design Justification: Differentiation of Dashboard functions and unique inputs
Design Strategy: Visual and clear illustration of student, admin, and student advisor dashboard sources and functionality
Alternative: Complex Consolidated depiction of dashboard functions (Progress, Reminder, and Career Planning)

3.2. Dynamic View

This section demonstrates the system's behaviour through a sequence of diagrams. Each diagram shows the interaction between classes to complete actions performed by students, advisors, and/or administrators. These models align with the static structures shown above in section 3.1.

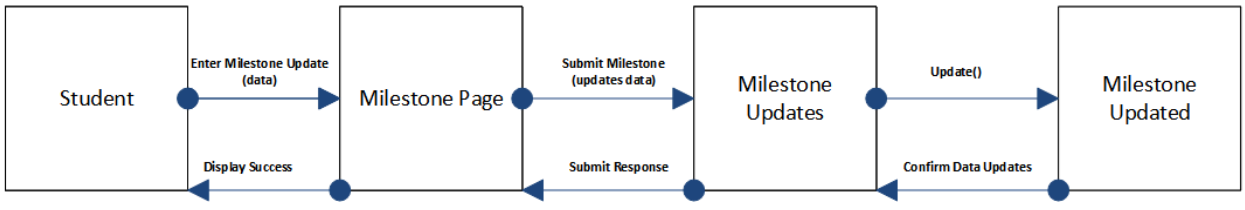
Student Creates a Milestone



Scenario Description:

A student uses the user interface to create and submit a milestone. The request is processed by the MilestoneCreation and stored in the Milestone class.

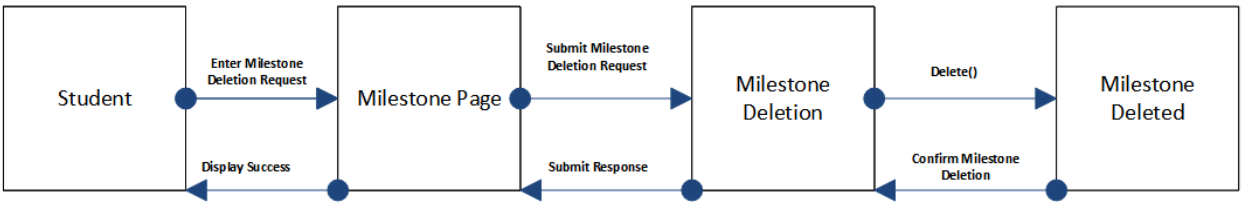
Student Updates a Milestone



Scenario Description:

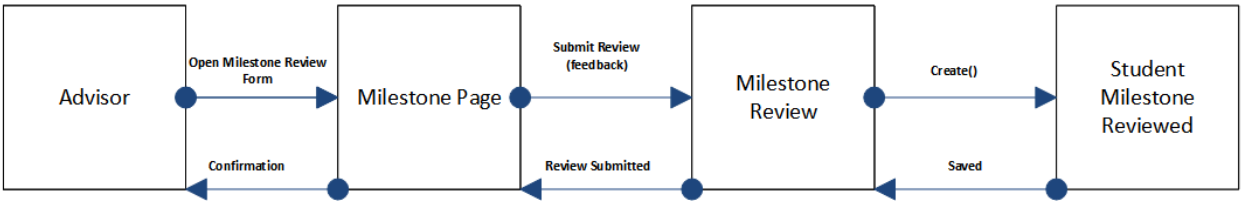
A student uses the user interface to update or make changes to an existing milestone. The request is processed by the MilestoneUpdates and stored in the Milestone class.

Student Deletes a Milestone



A student uses the user interface to delete or remove one or more milestones. The request is processed by the MilestoneDeletion and stored in the Milestone class.

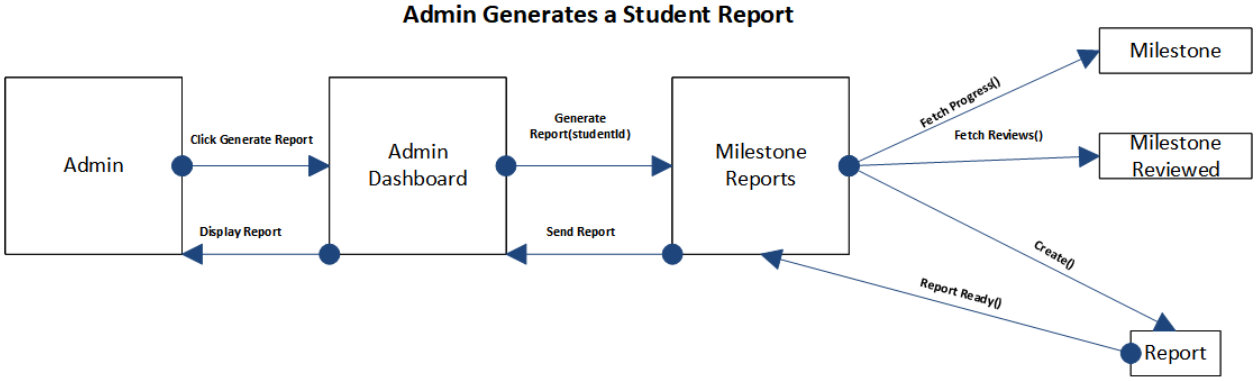
Advisor Reviews a Milestone



Scenario Description:

For any assigned student(s), an advisor submits a milestone review form. The Milestone Review processes this and stores a record.in MilestoneReview class.

Admin Generates a Student Report



Scenario Description:
An Admin selects “Generate Report”. The ReportService compiles the data from the milestones and reviews and renders/displays it for the admin.

PMaCT (All Inclusive) Project Sprint Schedule

Sprint 1: Project Planning & Setup (March 12-26, 2025)

Focus: Planning, organization, and foundation setup

Tasks:

- Research and select MERN-based to-do/organizer open-source applications as a foundation
- Set up the application in Git for team-based collaboration
- Create a comprehensive list of program functionalities (features)
- Develop a database ER diagram showing relationships between entities
- Create records for database entities
- Document the technical architecture and development approach

Deliverables:

- Git repository with properly configured collaboration structure
- Feature list document for sprint planning
- ER diagram showing database relationships
- Selected baseline MERN application to build upon

Sprint 2: Core User Functionality (March 27-April 9, 2025)

Focus: Basic user system and milestone tracking

Tasks:

- Implement a simple user login and registration
- Create a basic milestone creation and listing
- Build a minimal dashboard view
- Set up database connections

Deliverables:

- Basic user authentication
- Simple milestone tracking interface
- Working with database integration

Sprint 3: Essential Features (April 10-23, 2025)

Focus: Minimal viable product completion

Tasks:

- Add the student advisor's view of student milestones
- Implement basic resource listing
- Create simple email notifications
- Basic UI improvements

Deliverables:

- Minimal viable product with essential features
- Basic notification functionality
- Improved user interface

Final Sprint: Documentation (April 24-May 2, 2025)

Focus: Testing, Training, and Documentation

Tasks:

- Test application
- Create a basic user guide
- Prepare a simple demonstration
- Document known limitations
- End User Training

Deliverables:

- Bug-free application and basic project documentation
- Simple presentation materials
- Demonstration of core functionality
- Inclusive user training of functionality

GP_Sprint1_ Review (Page 1 of 2)

ID	Theme	Role	Story	Notes	Status	Effort	Priority	Sprint	Dates
1	Base Application	Framework	As a developer, I want to have a MERN-based to-do/organizer application, so that I can build upon a solid foundation for milestone tracking.		Done	8	High	1	March 12-26, 2025
2	Git Setup	Collaboration	As a developer, I want to use Git for version control, so that our team can collaborate effectively.		Done	3	Medium	1	March 12-26, 2025
3	Functionalities	Design and Development	As a developer, I want to have a clear list of program functionalities, so that we can organize our sprints efficiently.		Done	2	Medium	1	March 12-26, 2025
4	Database	Framework	As a developer, I want to have a well-designed database structure, so that data can be stored and retrieved efficiently.		Done	5	Medium	1	March 12-26, 2025
5	Prototype	Design and Development	As a developer, I want to see a high-fidelity prototype, so that I can visualize the final application.		Done	3	Medium	2	March 27-April 9, 2025
5	User Authentication	Backend Development	As a user, I want to register/login so that I can securely access my milestones.		Done	5	Medium	2	March 27-April 9, 2025
6	Milestone Tracking	Backend & Frontend	As a student, I want to create, view, and update milestones, so that I can track my progress through the program.	Depends on User Authentication	Partial	3	High	2	March 27-April 9, 2025
7	Dashboard	Frontend Development	As a user, I want to have a dashboard, so that I can quickly see my key milestones and progress.	Depends on Milestone Tracking	Done	5	Medium	2	March 27-April 9, 2025
8	Advisor View	Frontend Development	As a advisor, I want to view student milestones, so that I can monitor and support student progress.	Depends on User Authentication and Dashboard	Not Started	3	Medium	3	April 10-23, 2025
9	Admin View	Frontend Development	As an admin, I want to view advisor reviews, so that I can generate reports for student progress			5	Medium	3	April 10-23, 2025

GP_Sprint1_ Review (Page 2 of 2)

ID	Theme	Role	Story	Notes	Status	Effort	Priority	Sprint	Dates
10	Notification	Backend development	As a Backend developer, I want to receive email notifications for upcoming deadlines, so that I can stay on track with my milestones.	Depends on User Authentication	Not Started	5	Medium	3	April 10-23, 2025
11	System Settings	Backend & Frontend	As a user, I want to make changes profile and system settings, so that I can be concurrent with my needs		Not Started	5	High	3	April 10-23, 2025
12	UI Enhancement	Frontend Development	As a developer, I want to use an improved user interface, so that I can navigate the application more easily.		Not Started	3	Medium	3	April 10-23, 2025
13	Testing	Quality Assurance	As a developer, I want to have thorough testing, so that the application is reliable and bug-free.		Not Started	8	High	3	April 10-23, 2025
15	Documentation	Documentation Team	As a developer, I want to have comprehensive documentation, so that I can understand how to use the application.		Not Started	8	High	4	April 24 - May 2, 2025
16	Deployment	DevOps	As a developer, I want to use a deployed application, so that I can access the system from anywhere.		Not Started	13	High	4	April 24 - May 2, 2025

GP_Sprint2_ Review (Page 1 of 3)

ID	Theme	Role	Story	Notes	Status	Effort	Priority	Sprint	Dates
1	Base Application	Framework	As a developer, I want to have a MERN-based to-do/organizer application, so that I can build upon a solid foundation for milestone tracking.		Done	8	High	1	March 12-26, 2025
2	Functionalities	Design and Development	As a developer, I want to have a clear list of program functionalities, so that we can organize our sprints efficiently.		Done	2	Medium	1	March 12-26, 2025
3	Database	Framework	As a developer, I want to have a well-designed database structure, so that data can be stored and retrieved efficiently.		Done	5	Medium	1	March 12-26, 2025
4	User Authentication	Backend Development	As a user, I want to register/login so that I can securely access my milestones.		Done	5	Medium	2	March 27-April 9, 2025
5	Milestone Tracking	Backend & Frontend	As a student, I want to create, view, update and delete milestones, so that I can track my progress through the program.	Depends on User Authentication	Done	3	High	2	March 27-April 9, 2025
6	Dashboard	Frontend Development	As a student, I want to have a dashboard, so that I can quickly see my key milestones (Completed, In Progress, Await Approval from my assigned Student Advisor) and progress.	Depends on Milestone Tracking	Done	5	High	2	April 24 - May 4, 2025
7	Dashboard	Frontend Development	As a student advisor, I want to have a dashboard, so that I can quickly see milestones (Completed, In Progress, Await Approval) and progress of my assigned PhD students.	Depends on Milestone Tracking	Done	5	High	2	April 24 - May 4, 2025
8	Dashboard	Frontend Development	As an admin, I want to have a dashboard, so that I can quickly see student milestones (Completed, In Progress, Await Approval), student progress and student advisor feedback.	Depends on Milestone Tracking	Done	5	High	2	April 24 - May 4, 2025

GP_Sprint2_ Review (Page 2 of 3)

ID	Theme	Role	Story	Notes	Status	Effort	Priority	Sprint	Dates
9	Advisor View	Frontend Development	As a advisor, I want to view student milestones, so that I can monitor and support student progress.	Depends on User Authentication and Dashboard	Done	3	Medium	3	April 24 - May 4, 2025
10	Admin View	Frontend Development	As an admin, I want to view student progress, so that I can generate and export reports for student progress.		Done	5	Medium	3	April 24 - May 4, 2025
11	Admin View	Frontend Development	As an admin, I want to view student advisor feedback, so that I can generate and export reports for student advisor feedback		Done	5	Medium	3	April 24 - May 4, 2025
12	Notification	Backend development	As a Backend developer, I want to receive email notifications for upcoming deadlines, so that I can stay on track with my milestones.	Depends on User Authentication	Done	5	Medium	3	April 10-23, 2025
13	System Settings	Backend & Frontend	As a user, I want to make changes profile and system settings, so that I can be concurrent with my needs		Done	5	High	3	April 10-23, 2025
14	UI Enhancement	Frontend Development	As a developer, I want to use an improved user interface, so that I can navigate the application more easily.		Done	3	Medium	3	April 10-23, 2025
15	Testing	Quality Assurance	As a developer, I want to have thorough testing, so that the application is reliable and bug-free.		Done	8	High	3	April 10-23, 2025
16	Documentation	Documentation Team	As a developer, I want to have comprehensive documentation, so that I can understand how to use the application.		Done	8	High	4	April 24 - May 2, 2025
17	Deployment	DevOps	As a developer, I want to use a deployed application, so that I can access the system from anywhere.		Done	13	High	4	April 24 - May 2, 2025
18	Student View	Frontend Development	As a PHD student, I want to securely register and login so that I can track and celebrate my personal milestones.		Done	13	High	4	April 24 - May 4, 2025
19	Student User Profile	Frontend Development	As a PhD student, I want to create and update my secured user profile so that my personal information is up to date and accurate.	Depends on User Authentication	Done	12	High	4	April 24 - May 4, 2025

GP_Sprint2_ Review (Page 3 of 3)

ID	Theme	Role	Story	Notes	Status	Effort	Priority	Sprint	Dates
22	Student Dashboard	Frontend Development	As a PhD student, I want to be see notifications of milestone (papers, comp exams, research assignments, dissertation) deadlines and approvals so that I can complete all tasks on time.		Done	10	High	4	April 24 - May 4, 2025
23	Student Advisor Dashboard	Frontend Development	As a student advisor, I want to be see notifications of student milestone (papers, comp exams, research assignments, dissertation) deadlines and approvals and student advisor feedback so that I can complete all tasks on time.		Done	10	High	4	April 24 - May 4, 2025
24	Admin Dashboard	Frontend Development	As an admin, I want to view student details (Major, Program, Assigned Student), so that I can generate and export reports for student information.		Done	9	Medium	4	April 24 - May 4, 2025
25	Admin Dashboard	Frontend Development	As an admin, I want to post notes of student milestones, so that the student has visibility and responds timely to my comments.		Done	9	Medium	4	April 24 - May 4, 2025
26	Student Dashboard	Frontend Development	As a PhD student, I want to see notes for Milestones from the admin so that I can respond to milestone admin notes.		Done	9	Medium	4	April 24 - May 4, 2025