**Name:** Jacob Miller

**System Name:** Student Progress Tracker System

**Short project code-name:** STApp

**Requested Launch Date:** Late 2024-2025

**Department Affected By Project:** Education, Software Development

**Project’s Customers:** Students, Teachers, Parents

**Date Request Submitted:** June 6th 2024



# **Table of Contents:**

1. Executive Summary

2. Introduction

2.1. Problem Statement / Project Vision

2.2. System Capabilities

2.3. Non-functional Requirements and Design Constraints

2.4. System Evolution

2.5. Document Outline

3. Structural Model

3.1. Model Introduction

3.2. Class Diagrams

3.3. Metadata

4. Architecture Design

4.1. Architecture Overview

4.2. Infrastructure Model

4.2.1. Deployment Diagram 1 – Architecture Overview

4.2.2. Deployment Diagram 2 – Nodes and Artifacts

4.3. Hardware and Software Requirements

4.3.1. Hardware Components

4.3.2. Required Software Components

4.4. Security Plan

4.4.1. Security Overview

4.4.2. Security Plan

5. User-Interface

5.1. User-Interface Requirements and Constraints

5.2. Window/Screen Navigation Diagram

5.3. UI Wireframes

6. Appendices

6.1. Glossary

6.2. References / Bibliography

6.3. Supporting documentation

# **Executive Summary:**

Currently, the development team is composed of just one developer, myself. Despite this, arrangements are in place to augment the team by incorporating more developers. This project’s primary demographic consists of students actively seeking an effective software solution to robustly manage and track their academic progress. This software aims to enhance student empowerment by providing tools for performance monitoring, goal-setting, and identifying areas in need of improvement. Using data-driven insights, students can make informed decisions to optimize their learning experience and accomplish their academic goals.

The project has advanced through several stages thus far, which have included requirements gathering, UI wireframing, and the establishment of security controls. These initiatives have established the groundwork for creating a user-focused and secure application that fulfills the requirements of its intended users. The document presented here provides an extensive overview of the various components of the student progress tracking application. It presents a comprehensive outline of the proposed system, including its functionalities, user interface design, security measures, and future development roadmap. This document guides the development team, guaranteeing their adherence to project goals and objectives.

As we look forward, the forthcoming stages encompass the revision of the system design in response to stakeholder feedback and the completion of the implementation plan. Following this, the development team will commence the coding and testing stages to deliver an application of high quality, reliability, and user-friendliness.

# **Introduction (Refer to System Proposal Executive Summary/Section 4.0 for more details):**

Developed to aid students, the Student’s Progress Tracking App is a comprehensive tool that facilitates monitoring and enhancing their academic performance. The primary goal of this system is to provide students with an intuitive and user-friendly platform, enabling them to manage their grades, assignments, and academic progress efficiently. The application provides a customized dashboard, a grade tracking system, and an assignment calendar, all of which cater to the specific needs of students in a straightforward and captivating manner.

Notable features encompass a Personalized Dashboard, offering a real-time overview of the student’s academic progression, encompassing present grades, upcoming assignments, and GPA trends across different time intervals. Through the utilization of the Grade Tracker, students have the convenient ability to input their grades for assignments and exams while simultaneously receiving instant feedback via automated GPA calculation. The Assignment Calendar provides a visual representation of upcoming tasks and deadlines, along with reminders and notifications, to assist students in maintaining organization and effectively prioritizing their work. The user-friendly interface ensures easy navigation and data entry, helping students track their progress and manage their academic responsibilities. Incorporating achievement badges and motivational quotes into Gamified Elements aims to cultivate a heightened sense of engagement and motivation.

The System Proposal document thoroughly explains these features and the benefits they offer. The system enhances students’ capabilities by equipping them with the resources for academic success within an accessible and motivating platform.

## **Problem Statement / Project Vision:**

By addressing the notable challenges students encounter in effectively managing their academic responsibilities, the Student Progress Tracking App offers a solution to this issue. Conventional approaches to monitoring assignments, grades, and overall progress are frequently insufficient, resulting in disarray and diminished motivation. The app’s development focuses on providing a refined and user-friendly solution, enabling students to efficiently organize their academic endeavors, foster motivation, and attain superior outcomes.

The project aims to develop a comprehensive tool that revolutionizes how students track and improve their academic performance. The application enables students to manage their academic pursuits effectively and make well-informed educational choices by providing features such as a customized dashboard, grade monitoring, task administration, and motivational components. By offering valuable insights and promoting effective communication, the system aims to enhance the experience for students, educators, parents, and school administrators.

The stakeholders include students who require dependable progress monitoring, educators who provide insights into student performance, parents who desire to stay informed about their child’s academic journey, and school administrators who require efficient tools to foster academic success. Educational technology specialists, IT departments, curriculum developers, regulatory bodies, and third-party vendors uphold the system’s functionality, security, and compliance with educational standards. We expect the application to enhance academic performance, foster engagement, facilitate communication, and enable data-driven decision-making for all parties involved.

## **System Capabilities:**

**Functional Requirements** **(Refer to System Proposal Section 4.0 for more details):**

* User-Friendly Interface: The system must provide a straightforward and intuitive interface for students to navigate easily.
* Grade Tracking and GPA Calculation: Students must input and track their grades, and the system automatically calculates GPA.
* Assignment Prioritization: The system must prioritize assignments based on importance and due dates.
* Data Security and Privacy: Strict data security and privacy measures must be implemented to safeguard student information.
* Comprehensive Insights: The system should offer comprehensive insights into academic performance, including summaries and GPA trends.
* Assignment Calendar: An assignment calendar feature should organize tasks and deadlines, with reminders and notifications.
* Cross-Platform Accessibility: The system should be accessible across various devices and platforms.
* Gamification: Optional gamified elements like badges and motivational quotes can enhance student engagement.
* Social Media Integration: Social media integration for sharing academic progress or achievements will not be included.
* Third-Party Application Integration: Integration with third-party applications outside the academic scope will not be supported.

**Use Cases/Descriptions (Refer to System Proposal Section 5.0 for more details):**

* View Dashboard (UC-001): Students can access their personalized dashboard to view academic performance summaries and earned badges.
* Input Grades (UC-002): Students or instructors can input grades for various assignments and exams, with the system automatically calculating GPA.
* Calculate GPA (UC-003): The system automatically calculates students’ GPA based on the grades entered.
* View Assignment Calendar (UC-004): Students can view upcoming assignments and deadlines on a visual calendar.
* Set Reminders (UC-005): Students can set reminders for upcoming assignments and deadlines.
* Track Study Time (UC-006): Students can log and monitor the time spent studying.
* Earn Badges (UC-007): Students can earn achievement badges for reaching specific academic milestones.
* View Performance Summaries (UC-008): Students can view summaries of their academic performance over different time periods.
* Manage Users (UC-009): Administrators can add, edit, and delete user accounts within the system.
* Monitor System Performance (UC-010): Administrators can monitor the system’s performance and reliability.
* Ensure Data Security (UC-011): Administrators can manage security protocols to ensure the protection of user data.
* Update System Settings (UC-012): Administrators can update and configure system settings as needed.
* Input Course Data (UC-013): Instructors can add details about courses, including assignments and exams.
* Monitor Student Progress (UC-014): Instructors can monitor and review the academic progress of their students.

## **Non-functional Requirements and Design Constraints:**

**Constraints (Refer to System Proposal Section 1.0 for more details):**

* Time Constraints: Limited timeframes may restrict the development scope or lead to rushed implementation.
* Resource Limitations: Limited budget, staff, or technology may impede development and restrict system capabilities.
* Technical Constraints: Compatibility issues, platform restrictions, or legacy system dependencies may pose challenges during integration or deployment.
* Regulatory Compliance Requirements: Complying with data security and privacy regulations can influence system performance and user satisfaction.
* User Adoption Challenges: Resistance to change or usability issues may impede user acceptance of the system.

**Feasibility Assessment (Refer to System Proposal Section 3.0 for more details):**

* Technical Feasibility: The feasibility analysis will assess technical capabilities to determine if the project’s requirements can be effectively met.
* Resource Feasibility: Budget, staff, and technology availability will be evaluated to ensure sufficient resources for project execution.
* Scheduling Feasibility: Project timelines will be carefully considered mitigating any potential delays and ensure timely delivery.
* Organizational Feasibility: The readiness of the organization to implement and support the system will be assessed to address any potential challenges.
* Legal Feasibility: Legal implications, including data protection laws and regulations, will be scrutinized to ensure compliance.
* Contractual Feasibility: Existing contracts and agreements will be reviewed to identify any constraints or requirements affecting the project.

**Non-functional Requirements (Refer to System Proposal Section 4.0 for more details):**

* Performance: The system must efficiently respond to user commands, even during periods of high activity.
* Reliability: Users should encounter minimal disruptions or errors, ensuring consistent availability and functionality.
* Scalability: The system must expand alongside the user base and manage increased data volumes effectively.
* Usability: The user interface design should prioritize intuitiveness and user-friendliness.
* Accessibility: The system must adhere to accessibility standards to accommodate users with disabilities.
* Security: Robust security measures are necessary to protect user data from unauthorized access and cyber threats.
* Compatibility: The system must be compatible across various web browsers and operating systems.
* Maintainability: Maintenance and updates should be straightforward due to clear documentation and modular design.
* Compliance: The system must adhere to relevant laws, regulations, and industry standards, especially regarding data protection and privacy.
* Performance Monitoring: Regular monitoring and evaluation ensure consistent system functioning and identify areas needing improvement.

## **System Evolution:**

In Version 1, we will deliver the basic components of the Student Progress Tracking App, focusing on features like user-friendly interface, grade tracking, assignment prioritization, data security, and basic insights.

In Version 2, we will introduce the following additional functionalities:

**Advanced Reporting (Use Case ID: UC-015):**

Implementing this functionality will empower administrators and instructors to produce in-depth reports on student performance, assignment completion rates, and academic metrics. Valuable insights for data-driven decision-making can be derived from customizable reports that consider criteria such as date ranges, course codes, and individual student performance.

**Sophisticated Notification System (Use Case ID: UC-016):**

Students will receive notifications about vital updates, assignment deadlines, upcoming exams, and other critical academic events through multiple channels, including email, SMS, and in-app alerts. This feature aims to enhance communication, minimize missed deadlines, and improve overall academic performance.

**Integration with Learning Management Systems (LMS) (Use Case ID: UC-017):**

Subsequent iterations will effortlessly integrate with renowned Learning Management Systems (LMS) such as Canvas, Blackboard, and Moodle. This integration will seamlessly synchronize grades, assignments, and course data between the tracking app and the LMS. This will improve educators’ data management and a more enhanced user experience.

* + 1. **Version 3 and beyond Changes:**

In future versions 3 and beyond, we envision the following additional functional capabilities:

**Enhanced Mobile App Features (Use Case ID: UC-018):**

The upcoming enhancements to the mobile app will encompass offline access, push notifications, and a revamped user interface tailored for mobile devices. These enhancements will foster greater engagement and usability, enabling students to access the system effectively from any location.

**System Infrastructure Upgrades (Use Case ID: UC-019):**

With the expansion of the user base, it will be necessary to implement upgrades to the server infrastructure to accommodate the growing data storage and processing requirements. The potential solution might involve the integration of more powerful servers, expanding storage capacity, and implementing load-balancing strategies to enhance system reliability and performance.

**Continuous Improvement and Maintenance (Use Case ID: UC-020):**

We have planned regular software updates to introduce new functionalities, resolve bugs, and bolster system security. This will cause upgrading the fundamental database management system, implementing security patches, and enhancing system performance to guarantee continued efficacy and pertinence.

## **Document Outline:**

1. Executive Summary:

The following section summarizes the app designed to track student progress. It provides a comprehensive overview of goals, objectives, and projected outcomes. Prominent features are emphasized, such as progress monitoring, goal setting, and personalized recommendations. The intended demographic encompasses students searching for a user-friendly tool to track their progress. The present state of the project is managed by a sole developer who wants to expand the team in the future.

1. Introduction:

In the introduction, we explore the difficulties students encounter when monitoring their progress and identifying areas in need of improvement. The primary aim of the project vision is to create a platform that empowers students and is easy to use. The system can track grades, set goals, and visualize progress. Factors such as scalability, performance, and security are given priority in non-functional requirements. There are also discussions regarding potential enhancements and updates.

1. Structural Model:

The focus of this section is on the structural aspects of the application. The purpose of class diagrams is to visualize the associations between entities, such as users, courses, and assignments. Metadata encompasses attributes and properties connected to each class, allowing for a comprehensive data model view.

1. Architecture Design:

This section summarizes the app’s architecture. A concise summary outlines the major components and interactions at a higher level. Deployment diagrams and other infrastructure models showcase both physical and logical deployments. The hardware and software requirements specify the infrastructure and software dependencies. The security plan outlines various measures designed to protect user data.

1. User-Interface:

Our primary focus lies in the design and usability of the application’s interface. Requirements and constraints define principles and guidelines for the design process. The purpose of navigation diagrams is to represent the flow and organization of screens and functionalities visually. The user interface wireframes visually represent the application’s interface, highlighting its features and interactions.

1. Appendices:

This section will find supplementary materials that further support the main document. The purpose of a glossary is to provide definitions for technical terms. Including references and a bibliography is necessary to cite the sources that were consulted during the project properly. Additional supporting materials may comprise diagrams, charts, mockups, prototypes, and testing reports.

# **Structural Model**

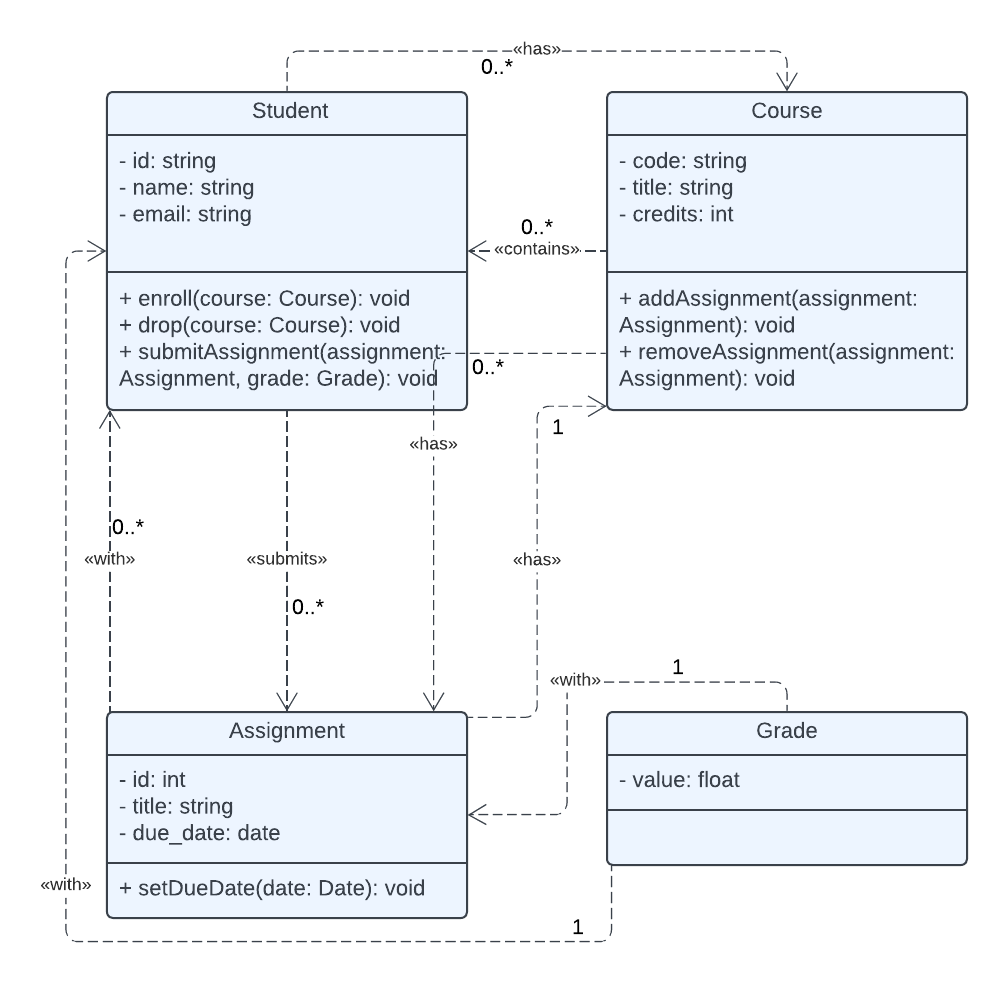
## **Model Introduction**

The structural model of our app for tracking student progress comprises two essential elements, namely class diagrams and metadata.

The Class Diagrams visually represent the different classes and their interconnections within the system. Their overview delves into the system’s structure at a high level, highlighting entities such as users, courses, assignments, grades, and more. All class diagrams visually represent the attributes and methods linked to the corresponding classes, facilitating comprehension of the system’s structure and operations.

The Metadata section provides a more comprehensive examination of the characteristics and qualities of the system’s components. This thoroughly explains the classes, attributes, and relationships illustrated in the diagrams. Metadata can encompass details such as data types, constraints, cardinality, and other relevant information crucial for developers to comprehend the intricacies of the system’s architecture.

## **Class Diagrams**



URL: <https://lucid.app/lucidchart/235a057f-4f1c-4ff5-b3f7-82392e8a7a9e/edit?viewport_loc=-837%2C131%2C2994%2C1538%2CHWEp-vi-RSFO&invitationId=inv_11a5bb2e-ed0d-4cbb-9161-e4637097d7b7>

## **Metadata**

1. **Assignment Class Diagram:**

Class Description:

The Assignment class represents an assignment in the system, encapsulating attributes such as ID, title, and due date. It provides a method to set the due date.

Attributes:

* ID: String
  + Visibility: Private (-)
  + Data Type: String
* Title: String
  + Visibility: Private (-)
  + Data Type: String
* DueDate: Date
  + Visibility: Private (-)
  + Data Type: Date

Methods:

* + setDueDate(date: Date): void
  + Visibility: Public (+)
  + Return Type: void
  + Parameters: date: Date
  + Description: Sets the due date for the assignment.

Relationships:

* Belongs to one course (Association with Course, Multiplicity: 1)
* Submitted by multiple students (Association with Student, Multiplicity: 0..\*)

1. **Course Class Diagram:**

Class Description:

The Course class represents a course in the system, encapsulating attributes such as code and title. It provides methods for adding and removing assignments.

Attributes:

* Code: String
  + Visibility: Private (-)
  + Data Type: String
* Title: String
  + Visibility: Private (-)
  + Data Type: String

Methods:

* + addAssignment(assignment: Assignment): void
  + Visibility: Public (+)
  + Return Type: void
  + Parameters: assignment: Assignment
  + Description: Adds the specified assignment to the course.
* + removeAssignment(assignment: Assignment): void
  + Visibility: Public (+)
  + Return Type: void
  + Parameters: assignment: Assignment
  + Description: Removes the specified assignment from the course.

Relationships:

* Enrolled by multiple students (Association with Student, Multiplicity: 0..\*)
* Contains multiple assignments (Association with Assignment, Multiplicity: 0..\*)

1. **Grade Class Diagram:**

Class Description:

The Grade class represents a grade assigned to an assignment, encapsulating the value of the grade.

Attributes:

* Value: float
  + Visibility: Private (-)
  + Data Type: float

Relationships:

* Associated with one assignment (Association with Assignment, Multiplicity: 1)
* Associated with one student (Association with Student, Multiplicity: 1)

1. **Student Class Diagram:**

Class Description:

The Student class represents a student in the system, encapsulating attributes such as ID, name, and email. It provides methods for enrolling in courses, dropping courses, and submitting assignments.

Attributes:

* ID: String
  + Visibility: Private (-)
  + Data Type: String
* Name: String
  + Visibility: Private (-)
  + Data Type: String
* Email: String
  + Visibility: Private (-)
  + Data Type: String

Methods:

* + enroll(course: Course): void
  + Visibility: Public (+)
  + Return Type: void
  + Parameters: course: Course
  + Description: Enrolls the student in the specified course.
* + drop(course: Course): void
  + Visibility: Public (+)
  + Return Type: void
  + Parameters: course: Course
  + Description: Drops the specified course for the student.
* + submitAssignment(assignment: Assignment, grade: Grade): void
  + Visibility: Public (+)
  + Return Type: void
  + Parameters: assignment: Assignment, grade: Grade
  + Description: Submits the specified assignment with the given grade for the student.

Relationships:

* Enrolls in multiple courses (Association with Course, Multiplicity: 0..\*)
* Submits multiple assignments (Association with Assignment, Multiplicity: 0..\*)

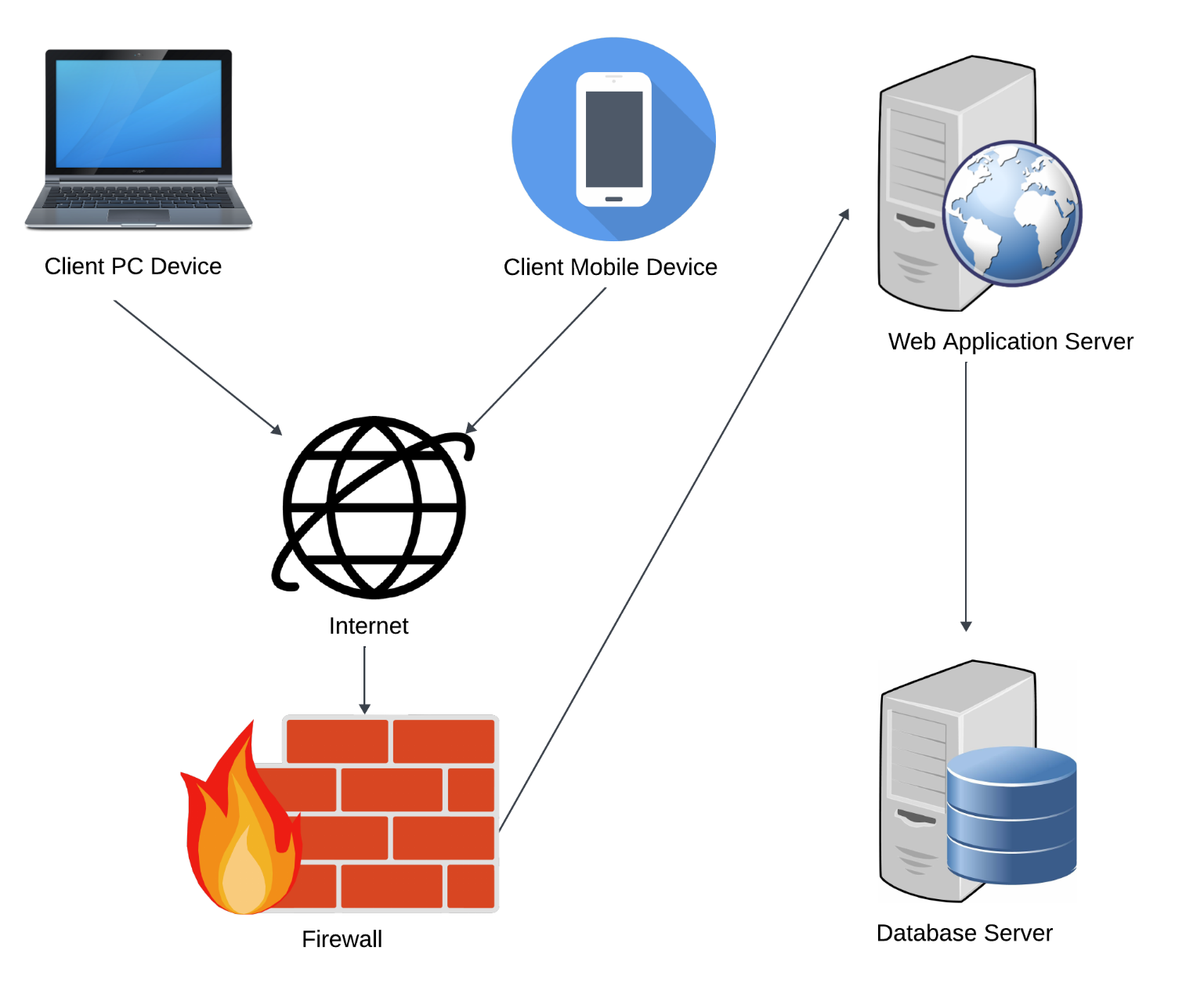
# **Architecture Design**

## **Architecture Overview**

This section offers a comprehensive summary and architectural overview of the Student Progress Tracking System. The deployment diagrams will present an overview of the system architecture, including the crucial components, their interactions, and the chosen deployment strategy. Under the architecture recommendations, a three-tier client-server system should be deployed using client devices (including mobile and PC/laptop), a web application server, and a database server. This architectural design ensures scalability, security, and efficient data management. The system will gain a web application server and a database server to accommodate the application and handle data storage. The following sections will present detailed descriptions of the physical system architecture, specifically deployment diagrams. These diagrams can effectively illustrate the main components and their connections, thus promoting consistency and clarity in the system’s representation infrastructure.

## **Infrastructure Model**

* + 1. **Deployment Diagram 1 – Architecture Overview**

Link: <https://lucid.app/lucidspark/0eca0d39-0216-4971-bdc7-47a803d37b5c/edit?viewport_loc=-4368%2C-2856%2C7680%2C3972%2C0_0&invitationId=inv_77adee07-1337-49bc-b448-d64916f4076c>

* + 1. **Deployment Diagram 2 – Nodes and Artifacts**

A diagram of several boxes

Description automatically generatedLink: <https://lucid.app/lucidchart/e62d12a9-4a71-4244-8fcb-dc95547f4aa0/edit?viewport_loc=-546%2C84%2C3328%2C1710%2CSD~btuP.wggj&invitationId=inv_56aa29b7-d208-4f86-83d4-15ce077651fb>

## **Hardware and Software Requirements**

* + 1. **Hardware Components**

**Required Hardware Components:**

1. Client Devices

* PC/Laptop:
  + Purpose: To access the Student Progress Tracking System via a web browser.
  + Reused Hardware: Users can use their current PCs or laptops; no new purchases are necessary.
* Mobile Devices:
  + Purpose: To access the system via mobile applications.
  + Reused Hardware: Users can use their current smartphones or tablets; no new purchases are necessary.

1. Web Application Server

* Specification:
  + Processor: Quad-core or higher
  + RAM: 16 GB or higher
  + Storage: SSD, 500 GB or higher
  + Network: High-speed Ethernet connection
* Purpose: To host the Student Progress Tracking web application, handle requests, and manage data processing.
* Recommendation: Purchase a new server to ensure reliability and performance.

1. Database Server

* Specification:
  + Processor: Quad-core or higher
  + RAM: 32 GB or higher
  + Storage: SSD, 1 TB or higher
  + Network: High-speed Ethernet connection
* Purpose: To store and manage the database for the Student Progress Tracking System.
* Recommendation: Purchase a new server to ensure data integrity and performance.
  + 1. **Required Software Components**

**Required Software Components:**

1. Operating Systems:

* Web Application Server:
  + Linux-based OS (Ubuntu Server or CentOS)
  + Windows Server (minimum version: Windows Server 2016)
* Database Server:
  + Linux-based OS (Ubuntu Server or CentOS)
  + Windows Server (minimum version: Windows Server 2016)

1. Database Management System (DBMS):

* MongoDB (minimum version: MongoDB 4.0)
  + For storing and managing the system’s data.
  + Available for both Linux and Windows environments.

1. Web Server Software:

* Node.js (minimum version: Node.js 14.x)
  + For hosting the web application server.
  + Cross-platform runtime environment.

1. Web Browser:

* Google Chrome (minimum version: Chrome 86)
  + For users to access the web-based interface of the Student Progress Tracking System.

1. Mobile Application Development:

* Android Studio (minimum version: Android Studio 4.1)
  + For developing Android mobile applications.
* Xcode (minimum version: Xcode 12)
  + For developing iOS mobile applications.

1. Version Control:

* Git (minimum version: Git 2.30)
  + For version control and collaboration during software development.

1. Integrated Development Environment (IDE):

* Visual Studio Code (minimum version: VS Code 1.52)
  + For coding, debugging, and testing the web application’s frontend and backend components.

1. Text Editors:

* Sublime Text (minimum version: Sublime Text 3.2)
* Atom (minimum version: Atom 1.54)
  + Optional text editors for developers’ preferences.

1. Package Managers:

* npm (Node Package Manager) (minimum version: npm 6.14)
* yarn (minimum version: yarn 1.22)
  + For managing dependencies and packages required by the Node.js environment.

1. Security Software:

* Firewall Software (Windows Defender Firewall)
  + To protect the servers from unauthorized access and malicious attacks.

1. Networking Tools:

* Secure Shell (SSH) Client (OpenSSH)
  + For secure remote access and administration of servers.

## **Security Plan**

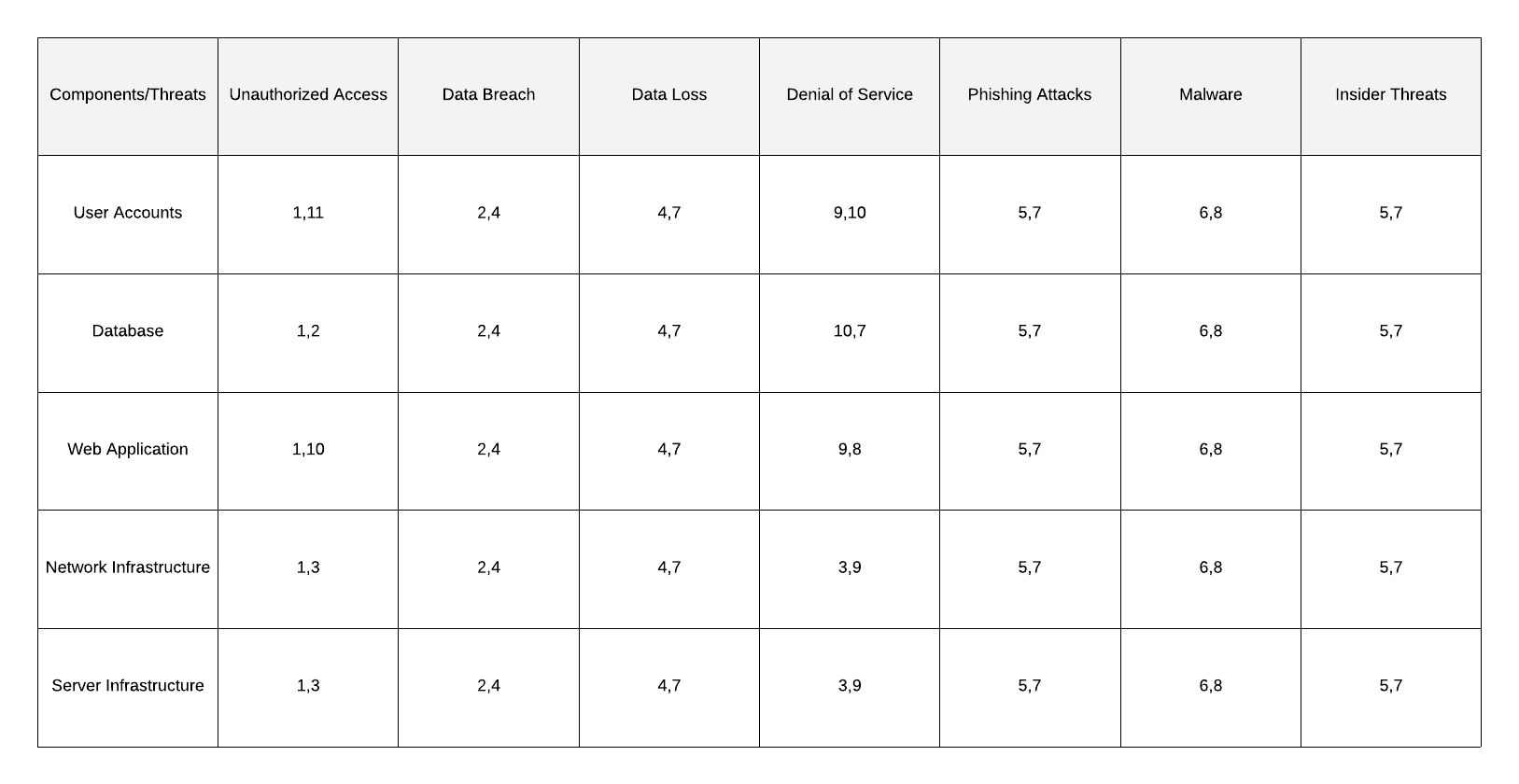
* + 1. **Security Overview**

**Potential Security Threats:**

1. Unauthorized Access: Unauthorized users attempting to gain access to student data or system resources.
2. Data Breaches: Accidental or intentional exposure of sensitive student information leading to data breaches.
3. Cyber Attacks: Malicious attacks such as DDoS attacks, SQL injection, cross-site scripting, or phishing attempts targeting the system.
4. Insider Threats: Unauthorized actions or data breaches by individuals with legitimate access to the system.
5. Data Loss: Data loss due to hardware failures, software errors, or natural disasters.

**Security Software:**

1. Firewall Software: Windows Defender Firewall (for Windows-based servers) or equivalent firewall software for Linux-based servers.
2. Data Encryption Tools: OpenSSL for data encryption at rest and secure transmission over HTTPS.
3. Authentication Mechanisms: Integration with authentication providers supporting MFA, such as Google Authenticator or Authy.
4. Intrusion Detection System: Snort or Suricata for real-time monitoring and detection of suspicious network activities.
5. Security Information and Event Management: ELK Stack (Elasticsearch, Logstash, Kibana) for centralized logging, monitoring, and analysis of security events.
   * 1. **Security Plan(TODO)**



**Controls:**

1. Implementing comprehensive password policies and the establishment of role-based access controls.

2. The encryption of sensitive data while it is stored and while it is being transferred.

3. The deployment of firewalls and IDS to monitor and block unauthorized network traffic.

4. Regular backups of student data have been scheduled to mitigate the risk of data loss.

5. Mandatory security training is provided to users to educate them on the best security practices.

6. Software and systems undergo regular updates and patches to effectively address vulnerabilities.

7. The ongoing monitoring of system activity and the logging of events are necessary for auditing purposes.

8. The development of a thorough and efficient response plan to promptly address security incidents.

9. Implementing rate-limiting measures is essential to mitigate the impact of DoS attacks and prevent excessive login attempts.

10. Implementing endpoint protection software on user devices aims to detect and prevent malware infections.

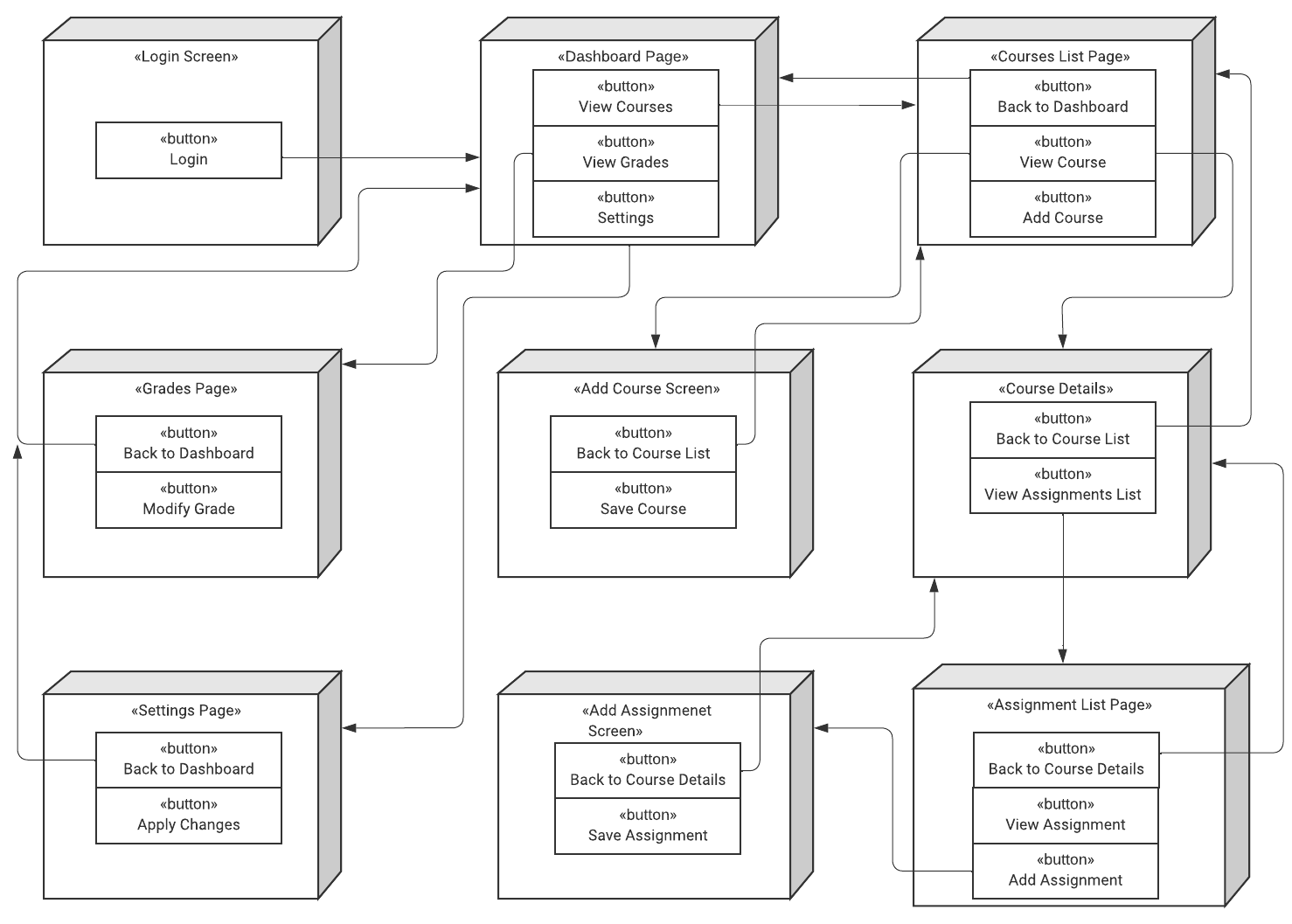
11. Limiting user privileges to the bare minimum required for their respective roles mitigates the potential for insider threats.

# **User-Interface**

## **User-Interface Requirements and Constraints**

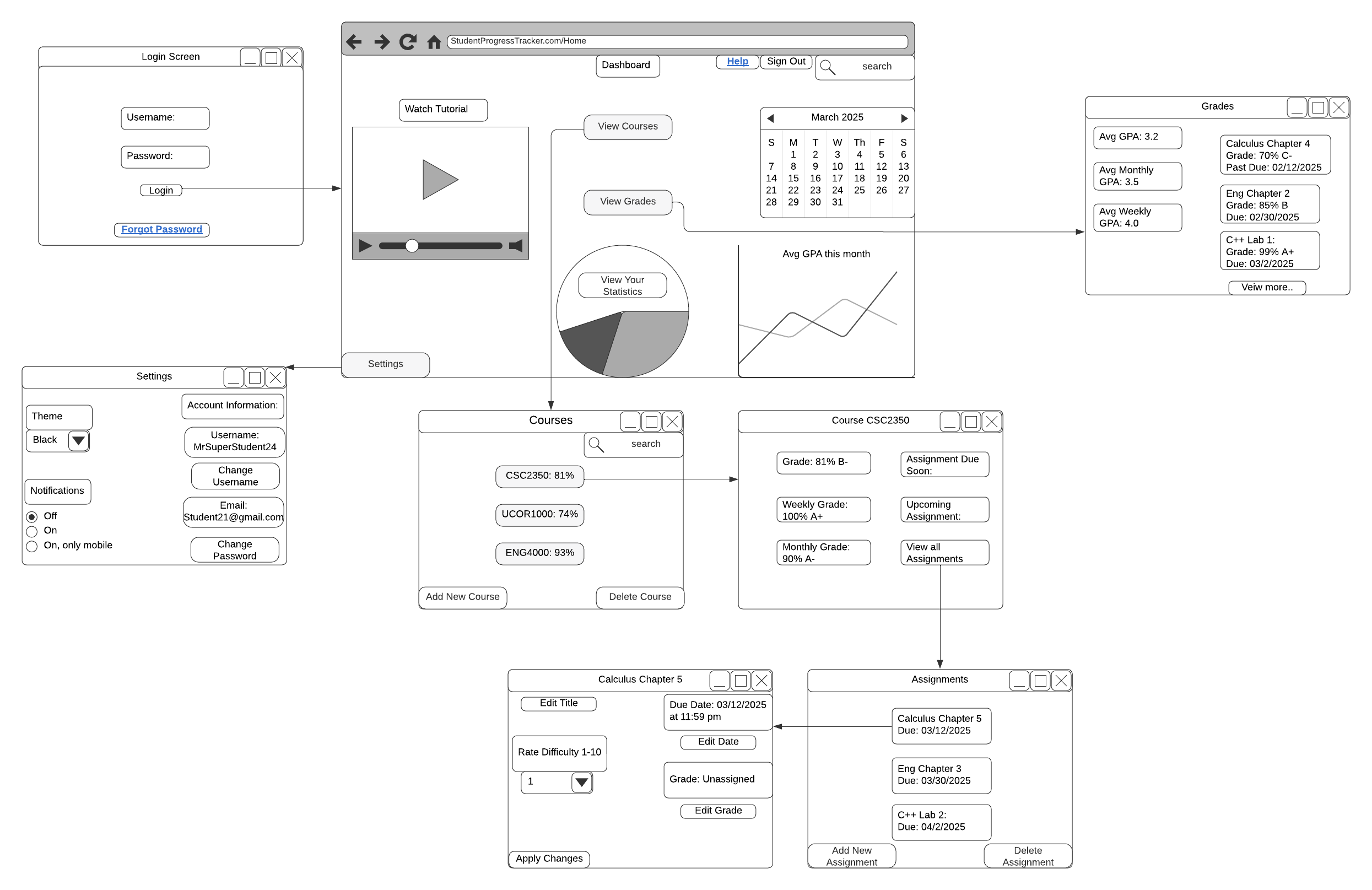
The user interface of our Student Progress Tracking Application has been purposefully designed to prioritize intuition, accessibility, and efficiency. The user interface will facilitate effortless navigation, well-organized layouts, and adherence to accessibility standards, guaranteeing usability across various devices and user capabilities. The design will cater to individuals with varying technological expertise while prioritizing secure authentication and data protection. The Window/Screen Navigation Diagram presents the primary navigation pathways, starting from the Login Screen and leading to the dashboard, where users can conveniently access courses, grades, and settings. From the dashboard, users can access various screens, including the Course List, Course Details, Assignment List, Assignment Details, Add Course, Add Assignment, Grades, and Settings. The user interface wireframe serves as a comprehensive visual reference for developers, guaranteeing the incorporation of essential components. The primary screens encompass the Login Screen, which encompasses input fields for the username and password. The Dashboard Screen contains buttons enabling access to courses and grades, and the Course and Assignment lists offer detailed views. These designs provide a user-friendly experience, allowing for efficient tracking of students’ academic progress.

## **Window/Screen Navigation Diagram**



Link: <https://lucid.app/lucidchart/b1143866-d13a-4e8d-83f6-d17b3be8d5f5/edit?viewport_loc=-811%2C-246%2C3330%2C1711%2CSD~btuP.wggj&invitationId=inv_3ad7c805-cab4-40a4-bea2-e1aa7fb7259c>

## **UI Wireframes**



Link: <https://lucid.app/lucidchart/3a25dad6-d84b-4b60-afd2-eac9d9395f23/edit?viewport_loc=-718%2C-207%2C3113%2C1901%2C0_0&invitationId=inv_47fb74a2-d7bc-4aac-895f-55e8b7adbf9d>

# **Appendices**

## **Glossary**

**Client-Server System**

The proposed network architecture involves client devices requesting services and receiving them from a centralized server.

**Database Server**

A server that uses a database application to offer database services to other computer programs or computers.

**Deployment Diagram**

A particular UML diagram depicts the physical configuration of hardware and software within a system.

**Execution Environment**

The framework is the environment in which a program or application operates, such as an operating system or browser.

**HTTPS**

An extensively employed computer network protocol designed to facilitate secure communication, widely adopted on the Internet.

**MongoDB**

This database program falls under the NoSQL category and uses JSON-like documents, which can have optional schemas.

**Node.js**

This JavaScript runtime environment is open-source, cross-platform, and capable of executing JavaScript code outside of a web browser.

**UML**

Using a standardized modeling language enables the creation of diagrams that accurately depict the design and structure of a software system.

**UI**

User interaction modalities with a computer system encompass screens, menus, and buttons.

**Wireframe**

A visual representation that highlights the underlying architecture of a user interface in an application.

**Web Application Server**

The server is purpose built for the execution of web applications, enabling business logic integration and seamless access to backend services, including databases.

**Web Browser**

The global network known as the World Wide Web allows users to access data using a software application.

## **References / Bibliography**

(You should always have some references, even if it’s textbook sections/chapters, class lectures, and your notes!) Don’t forget to cite any and all documents and websites you used properly. Use the APA v7 citation format.

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and

Iterative by Craig Larman. Third Edition; Prentice-Hall.

Class Lectures

Class Notes

Class Power points

Class Panopto video

Systems engineering for its - design and specifications. (n.d.). https://ops.fhwa.dot.gov/seits/sections/section3/3\_3\_7.html

Creating models and diagrams. to describe a system or solution architecture. (n.d.). https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=models-creating-diagrams

## **Supporting documentation**

Lucid visual collaboration suite: (n.d.). <https://lucid.app/>

Node.js V22.2.0 documentation. Index | Node.js v22.2.0 Documentation. (n.d.). <https://nodejs.org/docs/latest/api/>

MongoDB documentation.css-MongoDB Documentation. (n.d.). <https://www.mongodb.com/docs/>

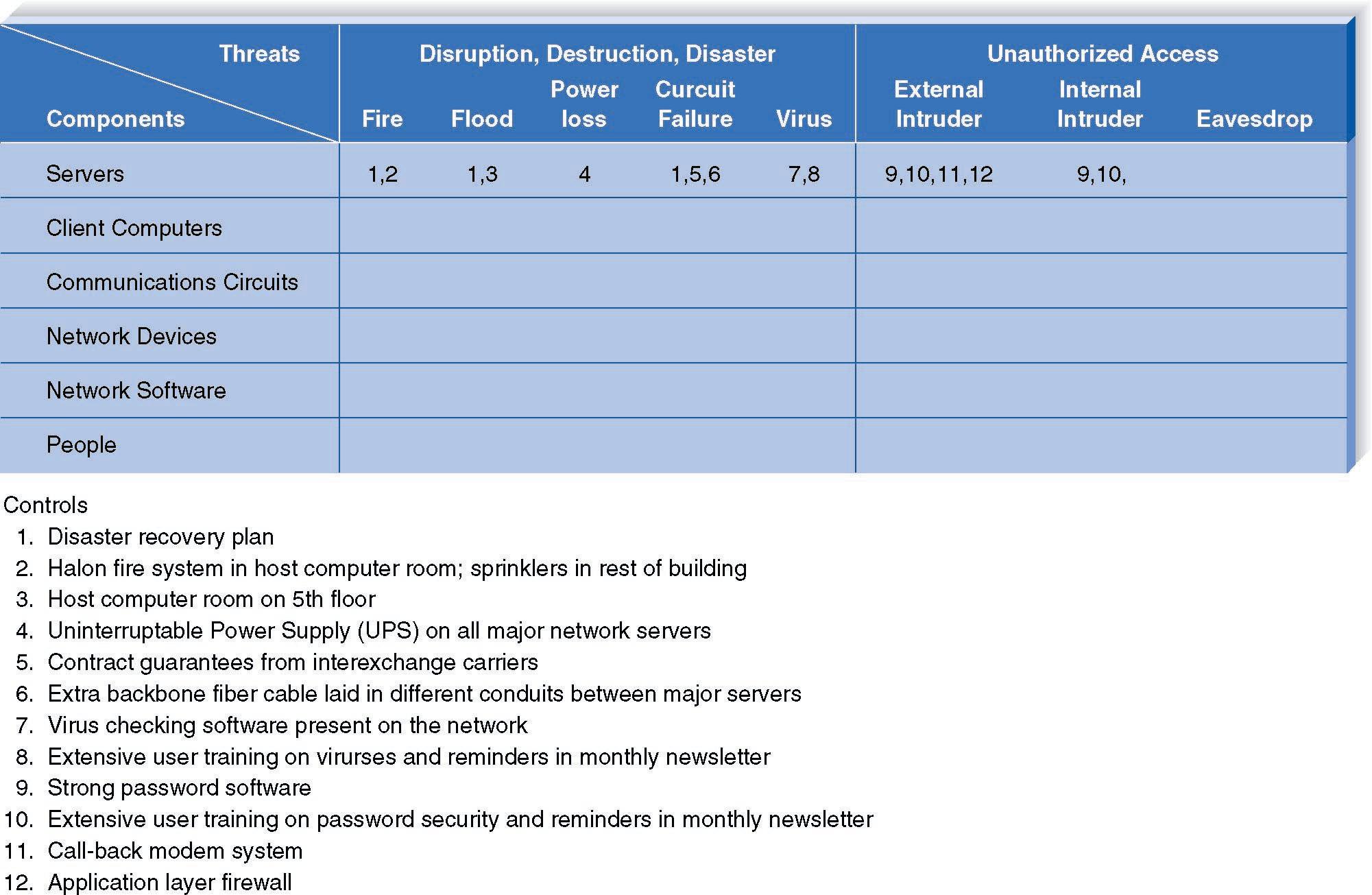
IBM documentation. IBM. (n.d.-b). https://www.ibm.com/docs/en

## 

**Layout Guide for Security Table**

See also sample documents for content ideas.

Remove this sample and replace it with your Security Plan in Section 4.4.2 above.

****

­­­­­­­­­­­­