Henok Eyayalem

Henok3095@gmail.com

Software requirement SPECIFICATION (srs document)

May 25, 2025

## Table of Contents

1. **Introduction**   
   1.1 Purpose   
   1.2 Document Conventions   
   1.3 Intended Audience and Reading Suggestions   
   1.4 Product Scope   
   1.5 References
2. **Overall Description**   
   2.1 Product Perspective   
   2.2 Product Functions   
   2.3 User Classes and Characteristics   
   2.4 Operating Environment   
   2.5 Design and Implementation Constraints   
   2.6 User Documentation   
   2.7 Assumptions and Dependencies
3. **External Interface Requirements**   
   3.1 User Interfaces   
   3.2 Hardware Interfaces  
   3.3 Software Interfaces   
   3.4 Communications Interfaces
4. **System Features**   
   4.1 System Feature: Role-Based Access Control   
   4.2 System Feature: Kanban Task Management   
   4.3 System Feature: Document Versioning   
   4.4 System Feature: Comment Threads   
   4.5 System Feature: Admin/Mentor Dashboard   
   4.6 System Feature: Progress Overview and Reporting
5. **Other Nonfunctional Requirements**   
   5.1 Performance Requirements   
   5.2 Safety Requirements   
   5.3 Security Requirements   
   5.4 Software Quality Attributes   
   5.5 Business Rules
6. **Other Requirements**   
   Appendix A: Glossary   
   Appendix B: Analysis Models   
   Appendix C: To Be Determined List

# Software Requirements Specification(SRS)

## 1. Introduction

### 1.1 Purpose

The **Custor Portal System** is a web-based platform designed to streamline internal team collaboration at Custor Computing by providing centralized tools for project tracking, task management, document sharing, and performance visibility. This Software Requirements Specification (SRS) defines the functional and non-functional requirements for the initial release, aligning with Custor Computing’s goal of enhancing workflow efficiency, reducing dependency on scattered tools, and improving transparency across departments, as outlined in the Project Charter (2024).

### 1.2 Document Conventions

* Requirements are prioritized as **High (H)**, **Medium (M)**, or **Low (L)**.
* **Bold text** denotes key terms defined in **Appendix A**.
* **TBD** indicates items to be refined during development.
* Each requirement has a unique priority; sub-requirements do not inherit priorities unless specified.
* Error messages are displayed in red text with clear instructions in the UI.

### 1.3 Intended Audience and Reading Suggestions

This SRS is intended for:

**Developers**: Implement features per Sections 3–4 (Betel Yemanebirhan, Eden Getachew, Henok Eyayalem).

**Testers**: Verify requirements per Sections 4–5 (team members and mentor Mr. Yabssra).

**Stakeholders**: Review Sections 1–2 for alignment with business goals (Software Development Department, HR & Project Management Team, Company Executives, Internship Program Coordinators, Intern Mentorship Leads).

Readers should start with Sections 1–2 for an overview of the project scope and functions, then proceed to Sections 3–5 for technical details. **Appendix A** provides definitions for technical terms.

### 1.4 Product Scope

The **Custor Portal System** centralizes project-based collaboration for Custor Computing’s internship program and internal teams, enabling:

* **Role-based access control** for **Interns**, **Mentors**, and **Admins**.
* Task management via **Kanban** boards with drag-and-drop functionality.
* Document uploads with version tracking.
* Lightweight communication via comment threads.
* Dashboards for **Mentors** and **Admins** to monitor progress.
* Progress overview and reporting for project tracking.

**Out of Scope**: Full-featured chat, mobile app, third-party integrations (e.g., email/calendar).

The portal supports Custor Computing’s objective to improve collaboration efficiency and transparency, as defined in the Project Charter (2024).

### 1.5 References

* Project Charter, Custor Computing, Version 1.0, 2024, available at Custor’s internal document repository.
* Angular Documentation, Version 16, angular.io, accessed May 2025.
* ASP.NET Core Documentation, Version 6.0, docs.microsoft.com, accessed May 2025.
* Business Requirements Document (BRD), Eden Getachew, Custor Computing, 2024.

## 2. Overall Description

### 2.1 Product Perspective

The **Custor Portal System** is a standalone web application replacing manual workflows (e.g., spreadsheets, email, Telegram) for project management at Custor Computing. It integrates with Custor’s internal authentication system via **JWT** for secure access. The portal operates independently but coexists with Custor’s HR and email systems without direct integration. See **Appendix B** for a system architecture diagram.

### 2.2 Product Functions

The portal provides the following high-level functions:

1. User authentication with **role-based access control** (**Intern**, **Mentor**, **Admin**).
2. Project and task creation/assignment with **Kanban**-style management.
3. Document uploads with version tracking.
4. Lightweight communication via comment threads with mentions and notifications.
5. **Mentor** and **Admin** dashboards for real-time progress monitoring.
6. Progress overview and reporting (task completion, deadlines, reports).

Task management and document uploads feed into dashboards for real-time insights. See **Appendix B** for a data flow diagram.

### 2.3 User Classes and Characteristics

|  |  |
| --- | --- |
| **Role** | **Characteristics** |
| **Intern** | Primary users; create/join project teams, manage tasks, upload files; basic web application experience assumed. |
| **Mentor** | Secondary users; monitor multiple teams, approve tasks/documents, provide feedback; moderate technical expertise. |
| **Admin** | Manage users, roles, teams, and system settings; advanced technical expertise required. |

**Interns** are the primary focus, requiring intuitive interfaces due to limited expertise.

### 2.4 Operating Environment

**Frontend**: Angular (v16+), supported on modern browsers (Chrome, Firefox, Edge).

**Backend**: ASP.NET Core 6.0, SQL Server for data storage.

**Hosting**: Custor’s internal servers (Windows Server).

### 2.5 Design and Implementation Constraints

* Must use Custor’s approved tech stack (Angular, ASP.NET Core, SQL Server).
* Development timeline: ~3–4 weeks (per Project Charter).
* Team consists of interns with a learning curve in full-stack development.
* Must comply with Custor’s internal data security policies.

### 2.6 User Documentation

* Online help system with tutorials for task management, file uploads, and comment threads, accessible via a help button in the UI.
* User manual for **Admins**, detailing user, role, and team management.
* Training/demo slides for **Interns** and **Mentors**

### 2.7 Assumptions and Dependencies

* **Assumptions**:

**Mentors** provide weekly feedback via sync-up meetings.

**Interns** have basic training in Angular and web application usage.

Stakeholders (e.g., Software Development Department)

* **Dependencies**:

Custor’s internal servers must be available for hosting.

SQL Server must be configured for data storage.

Internet access for development tools (e.g., GitHub).

## 3. External Interface Requirements

### 3.1 User Interfaces

**Login Screen**: Email and password input with **JWT** authentication; error messages in red for invalid credentials (e.g., “Invalid email or password”).

**Board**: Drag-and-drop tasks across “To Do,” “In Progress,” “In Review,” “Done” columns.

**File Upload**: Modal dialog with version history; supports drag-and-drop uploads and file type tagging.

**Comment Threads**: Inline comment section for tasks/documents with @username mentions and notifications.

**Dashboard**: Charts showing project progress (e.g., task completion percentages); includes a help button.

**Standards**: Keyboard shortcuts (e.g., Ctrl+S for saving tasks) and consistent error messaging.

### 3.2 Hardware Interfaces

The portal operates on standard desktop hardware via modern web browsers (Chrome, Firefox, Edge). No specific hardware interactions are required.

### 3.3 Software Interfaces

* **Backend API**: RESTful endpoints handle:
  + User authentication (**JWT** tokens, JSON payloads).
  + Task updates (JSON payloads for task status, assignees).
  + File uploads (multipart form data with metadata).
  + Comment threads (JSON payloads for comments, notifications).

**Database**: SQL Server stores user profiles, project/task metadata, document versions, and comment threads.

**Operating System**: Compatible with Windows Server

**Development Tools**: GitHub for version control and collaboration.

### 3.4 Communications Interfaces

The portal uses HTTPS for secure communication between the frontend and backend. Notifications for comments/mentions are handled internally (no email integration). No external communication protocols are required for the initial release.

## 4. System Features

This section outlines the core functionalities of the **Custor Portal System**, organized by system features as defined in the Project Charter. Each feature includes an in-depth description, priority assessment, detailed stimulus/response sequences, and functional requirements to provide a clear, verifiable specification for developers, testers, and stakeholders (Software Development Department, HR & Project Management Team, Company Executives, Internship Program Coordinators, Intern Mentorship Leads). The features are designed to support Custor Computing’s objectives of enhancing workflow efficiency, reducing reliance on scattered tools (e.g., spreadsheets, Telegram), and improving transparency across internship and internal teams. Requirements are concise, verifiable, and aligned with the 3–4 week development timeline and intern-led team constraints.

### 4.1 System Feature: Role-Based Access Control

#### 4.1.1 Description and Priority

The **Role-Based Access Control (RBAC)** feature is the cornerstone of the **Custor Portal System**’s security architecture, ensuring that users—categorized as **Interns**, **Mentors**, or **Admins**—access only the resources and functionalities permitted by their assigned roles. This feature is critical for safeguarding sensitive project data, such as task assignments and document uploads, while enabling structured collaboration within Custor Computing’s internship program. By leveraging **JSON Web Tokens (JWT)** for authentication, the system provides secure, scalable access control that aligns with Custor’s internal data security policies. For example, **Interns** are restricted to their assigned projects to prevent unauthorized modifications, while **Mentors** have oversight over multiple teams, and **Admins** manage system-wide settings, such as user roles and team configurations. This feature supports the Project Charter’s goal of accountability by logging all access attempts and ensuring traceability. The implementation must be robust to handle concurrent user sessions and prevent security vulnerabilities, such as unauthorized access or session hijacking, within the constrained 3–4 week development timeline.

**Priority**: High. This feature is essential for ensuring data security and role-specific functionality, with a benefit rating of 9/9 due to its role in protecting project integrity and enabling collaboration. The penalty for non-implementation is 8/9, as lack of access control could lead to data breaches, unauthorized task modifications, and misalignment with Custor’s security policies, undermining stakeholder trust.

#### 4.1.2 Stimulus/Response Sequences

* **User Login Process**:

**Stimulus**: An **Intern**, **Mentor**, or **Admin** navigates to the login screen and enters their email and password, then clicks “Login.”

**Response**: The system validates the credentials against the SQL Server database. If valid, it issues a **JWT** token, stores it in the browser’s session storage, and redirects the user to their role-specific homepage (e.g., **Intern** to their project’s **Kanban** board, **Mentor** to the dashboard). If invalid, the system displays a red error message (e.g., “Invalid email or password”) and increments a failed attempt counter.

* **Admin Role or Team Assignment**:

**Stimulus**: An **Admin** accesses the user management UI, selects a user (e.g., an **Intern**), assigns them the **Mentor** role, and adds them to a project team.

**Response**: The system updates the user’s role and team assignment in the database, refreshes their permissions, and displays a confirmation message (e.g., “User role updated successfully”) within 1 second. The affected user is notified via the portal’s notification panel on their next login.

* **Unauthorized Action Attempt**:

**Stimulus**: An **Intern** attempts to access another team’s **Kanban** board or edit a restricted task.

**Response**: The system checks the user’s **JWT** token permissions, denies access, displays an error message in red (e.g., “Access denied: Insufficient permissions”), and logs the attempt with the user’s ID and timestamp for auditing purposes.

* **Session Timeout Handling**:

**Stimulus**: A user’s **JWT** token expires after 24 hours of inactivity or at the end of the token’s validity period.

**Response**: The system invalidates the token, logs out the user, clears session storage, and redirects to the login screen with a message (e.g., “Session expired, please log in again”).

* **Password Reset Workflow**:

**Stimulus**: A user clicks “Forgot Password” on the login screen, enters their email, and requests a reset link.

**Response**: The system verifies the email, generates a secure, time-limited (30-minute) reset link, sends it to the user’s registered email, and displays a confirmation (e.g., “Password reset link sent to your email”). If the email is invalid, an error message (e.g., “Email not found”) is shown.

#### 4.1.3 Functional Requirements

**REQ-1**: The system shall authenticate users by validating email and password against the SQL Server database, issuing a **JWT** token upon successful login, verifiable via API response (status 200 with token payload).

**REQ-2**: The system shall restrict **Interns** to viewing and editing only their assigned projects and tasks, verifiable via UI access controls and server-side permission checks.

**REQ-3**: **Admins** shall manage user roles (e.g., promote to **Mentor**) and team assignments through the admin UI, updating the database within 1 second, verifiable via admin UI and database logs.

**REQ-4**: The system shall display an error message in red (e.g., “Invalid email or password”) for invalid login attempts, limiting to 5 attempts before a 15-minute lockout, verifiable via UI and audit logs.

### 4.2 System Feature: Kanban Task Management

#### 4.2.1 Description and Priority

The **Kanban Task Management** feature provides a visual, intuitive interface for project teams to create, assign, and track tasks using **Kanban** boards, which are central to the **Custor Portal System**’s goal of streamlining project workflows. Each project has a dedicated **Kanban** board with four columns—“To Do,” “In Progress,” “In Review,” and “Done”—allowing **Interns** to manage their tasks and **Mentors** to monitor progress. Tasks include metadata such as titles, descriptions, deadlines, priorities (High, Medium, Low), and assignees, enabling structured collaboration and accountability. The drag-and-drop functionality allows users to update task statuses seamlessly, while filtering options (by assignee, status, or priority) help teams focus on relevant tasks. This feature reduces dependency on manual tools like spreadsheets, as highlighted in the Project Charter, by providing a centralized, real-time view of project activities. It must be user-friendly for **Interns** with limited technical expertise and robust enough to handle concurrent updates by multiple team members, aligning with the 3–4 week development timeline and intern-led development constraints.

**Priority**: High. This feature is critical for project tracking and collaboration, with a benefit rating of 9/9 due to its role in improving workflow efficiency and transparency. The penalty for non-implementation is 7/9, as manual task tracking would persist, reducing productivity and stakeholder satisfaction.

#### 4.2.2 Stimulus/Response Sequences

* **Task Creation Workflow**:

**Stimulus**: An **Intern** or **Mentor** navigates to a project’s **Kanban** board, clicks “Add Task,” and fills in a form with title, description, deadline, priority, and assignee.

**Response**: The system validates the input (e.g., non-empty title, valid deadline), adds the task to the “To Do” column, saves it to the database, and sends a notification to the assignee via the portal’s notification panel. A confirmation message (e.g., “Task created successfully”) is displayed.

* **Task Status Update**:

**Stimulus**: A user drags a task from one **Kanban** column (e.g., “To Do”) to another (e.g., “In Progress”).

**Response**: The system verifies the user’s permissions, updates the task status in the database, refreshes the **Kanban** board within 1 second, and notifies the assignee and **Mentor** of the change.

* **Task Filtering**:

**Stimulus**: A user selects a filter option (e.g., “Show tasks assigned to [username]” or “High priority only”) from the **Kanban** board UI.

**Response**: The system queries the database, filters tasks based on the selected criteria, and updates the **Kanban** board to display only matching tasks, preserving column organization.

* **Unauthorized Task Modification**:

**Stimulus**: An **Intern** attempts to edit or move a task assigned to another team or user.

**Response**: The system checks permissions via the **JWT** token, denies the action, displays an error message in red (e.g., “Permission denied: Unauthorized action”), and logs the attempt for auditing.

* **Task Deletion**:

**Stimulus**: A **Mentor** or **Admin** selects a task and clicks “Delete.”

**Response**: The system archives the task in the database with a deletion timestamp, removes it from the **Kanban** board, and displays a confirmation (e.g., “Task deleted successfully”). The action is logged for auditing.

#### 4.2.3 Functional Requirements

**REQ-5**: Tasks shall have four states (“To Do,” “In Progress,” “In Review,” “Done”), verifiable via UI inspection of the **Kanban** board.

**REQ-6**: The system shall update task status within 1 second of a drag-and-drop action, verifiable via database logs and UI refresh.

**REQ-7**: The system shall allow task creation with mandatory fields (title, description, deadline, priority, assignee), verifiable via task creation UI.

**REQ-8**: The system shall display an error message (e.g., “Permission denied: Unauthorized action”) for unauthorized task modifications, verifiable via UI and audit logs.

### 4.3 System Feature: Document Versioning

#### 4.3.1 Description and Priority

The **Document Versioning** feature enables users to upload, manage, and track project-related documents (e.g., reports, code archives, presentations) with automatic version control, ensuring traceability and preventing accidental overwrites. This feature is essential for maintaining a clear history of document changes, as specified in the Project Charter, allowing teams to collaborate on shared files without confusion. Users can tag files by type (e.g., report, code, presentation) for easy categorization, view version history with metadata (e.g., timestamps, uploader names), and revert to previous versions if needed. The system must handle concurrent uploads efficiently and enforce file size limits to maintain performance, particularly given the intern-led development team’s learning curve and the 3–4 week timeline. This feature supports the Project Charter’s goal of work transparency by ensuring all team members work with the latest approved documents, reducing errors in collaborative workflows.

**Priority**: Medium. This feature enhances collaboration and document management, with a benefit rating of 6/9 due to its role in reducing file confusion. The penalty for non-implementation is 4/9, as teams could revert to manual document sharing, though with reduced efficiency.

#### 4.3.2 Stimulus/Response Sequences

* **File Upload Process**:
  + **Stimulus**: An **Intern** or **Mentor** opens the document upload modal, selects a file (e.g., a PDF report), chooses a file type (e.g., “Report”), and clicks “Upload.”
  + **Response**: The system validates the file (size, type), stores it in the database, creates a new version if the filename exists, tags it with the selected type, and updates the version history. A confirmation message (e.g., “File uploaded successfully”) is shown, and relevant team members are notified.
* **Version History Access**:
  + **Stimulus**: A user clicks “View Version History” for a document in the project UI.
  + **Response**: The system retrieves all versions from the database and displays a list with metadata (e.g., version number, timestamp, uploader name, file size) in a modal, allowing the user to download or preview specific versions.
* **File Size Limit Exceeded**:
  + **Stimulus**: A user attempts to upload a file exceeding the maximum size (**TBD**).
  + **Response**: The system rejects the upload, displays a red error message (e.g., “File size exceeds limit of [TBD] MB”), and logs the attempt for auditing.
* **Version Reversion**:
  + **Stimulus**: A **Mentor** selects a previous version of a document and clicks “Revert.”
  + **Response**: The system restores the selected version as the current file, archives the replaced version with a timestamp, and notifies team members of the change.
* **File Deletion Request**:
  + **Stimulus**: An authorized user (e.g., **Mentor** or **Admin**) deletes a document from the project UI.
  + **Response**: The system archives the document in the database with a deletion timestamp, removes it from the UI, and displays a confirmation (e.g., “Document deleted successfully”).

#### 4.3.3 Functional Requirements

**REQ-9**: The system shall create a new version when a file with the same name is uploaded, storing metadata (timestamp, uploader, file size), verifiable via version history UI.

**REQ-10**: The system shall support file type tagging (e.g., report, code, presentation, **TBD** additional types), verifiable via upload UI dropdown.

**REQ-11**: The system shall reject files exceeding the maximum size (**TBD**) with an error message in red (e.g., “File size exceeds limit”), verifiable via UI.

### 4.4 System Feature: Comment Threads

#### 4.4.1 Description and Priority

The **Comment Threads** feature provides a lightweight, context-specific communication mechanism, allowing users to post threaded comments on tasks and documents within the **Custor Portal System**. This feature eliminates the need for external tools like Telegram, as outlined in the Project Charter, by keeping discussions directly linked to project elements (e.g., a specific task or document). Users can use @username mentions to notify team members, fostering collaboration and ensuring timely responses. The system supports threaded replies to maintain conversation clarity and sends notifications for mentions, enhancing team coordination. This feature is designed to be intuitive for **Interns** with limited technical expertise and must handle concurrent comment updates efficiently. It supports the Project Charter’s goal of consolidating communication within a single platform, reducing reliance on fragmented tools and improving traceability of feedback.

**Priority**: Medium. This feature enhances collaboration by centralizing communication, with a benefit rating of 6/9. The penalty for non-implementation is 4/9, as teams could use external tools, though with reduced context and efficiency.

#### 4.4.2 Stimulus/Response Sequences

* **Comment Posting**:

**Stimulus**: An **Intern** or **Mentor** opens a task or document, enters a comment with an @username mention (e.g., “@Eden Please review this report”), and submits it.

**Response**: The system saves the comment to the database, appends it to the relevant thread, sends a notification to the mentioned user (e.g., Eden) via the notification panel, and displays a confirmation (e.g., “Comment posted”).

* **Comment Thread Viewing**:

**Stimulus**: A user clicks to view the comment thread for a task or document.

**Response**: The system retrieves all comments from the database and displays them in chronological order, showing user names, timestamps, and mention highlights in the UI.

* **Empty Comment Submission**:

**Stimulus**: A user submits a comment with no text.

**Response**: The system rejects the submission, displays a red error message (e.g., “Comment cannot be empty”), and prompts the user to enter text.

* **Comment Notification Management**:

**Stimulus**: A user opens their notification panel to view mentions or comment updates.

**Response**: The system displays unread notifications with links to the relevant task or document, marking them as read upon viewing.

* **Comment Moderation**:

**Stimulus**: A **Mentor** or **Admin** deletes an inappropriate comment from a thread.

**Response**: The system archives the comment with a deletion timestamp, removes it from the UI, and displays a confirmation (e.g., “Comment deleted”).

#### 4.4.3 Functional Requirements

**REQ-12**: The system shall support threaded comments on tasks and documents with @username mentions, verifiable via notification logs and UI.

**REQ-13**: The system shall send notifications for mentions to the user’s notification panel, verifiable via UI and logs.

**REQ-14**: The system shall reject empty comments with an error message in red (e.g., “Comment cannot be empty”), verifiable via UI.

### 4.5 System Feature: Admin/Mentor Dashboard

#### 4.5.1 Description and Priority

The **Admin/Mentor Dashboard** feature provides a centralized, role-specific interface for **Mentors** and **Admins** to oversee project progress and manage system configurations, aligning with the Project Charter’s emphasis on mentor visibility and administrative control. For **Mentors**, the dashboard displays real-time **Kanban** boards, task completion percentages, overdue deadlines, and file submissions for their assigned teams, enabling them to approve or reject tasks/documents and provide feedback. **Admins** use the dashboard to manage users, roles, and team assignments, ensuring system integrity and scalability. The dashboard includes visual charts (e.g., task completion, milestone timelines) to facilitate quick decision-making and is designed to be intuitive for **Mentors** with moderate technical expertise. This feature supports the Project Charter’s goal of enhanced oversight without micromanagement, ensuring timely guidance and alignment with Custor’s internship program objectives. It must handle multiple team views efficiently within the 3–4 week timeline.

**Priority**: High. This feature is critical for mentor oversight and system administration, with a benefit rating of 8/9. The penalty for non-implementation is 7/9, as lack of oversight would hinder mentor feedback and stakeholder visibility.

#### 4.5.2 Stimulus/Response Sequences

* **Dashboard Access**:

**Stimulus**: A **Mentor** or **Admin** navigates to the dashboard from the main menu.

**Response**: The system verifies the user’s role via **JWT** token, retrieves relevant data (e.g., team **Kanban** boards, task statuses, file submissions), and displays charts and tables summarizing progress, overdue tasks, and pending approvals.

* **Task/Document Approval**:

**Stimulus**: A **Mentor** reviews a task in the “In Review” column and clicks “Approve” or “Reject,” adding optional feedback.

**Response**: The system updates the task or document status (e.g., to “Done” or back to “In Progress”), saves feedback in the comment thread, notifies the assignee, and logs the action within 1 second.

* **User/Team Management**:

**Stimulus**: An **Admin** accesses the user management section, modifies a user’s role (e.g., promotes an **Intern** to **Mentor**), or reassigns a team.

**Response**: The system updates the database, refreshes permissions, and displays a confirmation (e.g., “User role updated”) within 1 second. Affected users are notified on their next login.

* **Unauthorized Dashboard Access**:

**Stimulus**: An **Intern** attempts to access the dashboard URL directly.

**Response**: The system denies access, displays a red error message (e.g., “Access denied: Mentor or Admin role required”), and logs the attempt.

* **Feedback Submission**:

**Stimulus**: A **Mentor** enters feedback on a task or document via the dashboard.

**Response**: The system saves the feedback to the comment thread, notifies relevant team members, and displays a confirmation (e.g., “Feedback submitted”).

#### 4.5.3 Functional Requirements

**REQ-15**: The system shall display task completion percentages and overdue deadlines for **Mentors**, verifiable via dashboard UI.

**REQ-16**: **Mentors** shall approve or reject tasks and documents, updating the database within 1 second and notifying assignees, verifiable via UI and logs.

**REQ-17**: **Admins** shall manage system settings (e.g., user roles, team assignments) through the admin UI, verifiable via UI and database logs.

### 4.6 System Feature: Progress Overview and Reporting

#### 4.6.1 Description and Priority

The **Progress Overview and Reporting** feature provides stakeholders with visual and textual insights into project status, enabling **Mentors**, **Admins**, and other stakeholders (e.g., Company Executives, Internship Program Coordinators) to monitor task completion, milestone timelines, and team activity. This feature generates real-time progress overviews with charts (e.g., task completion percentages, overdue tasks) and allows **Mentors** to produce detailed reports summarizing team performance, submission history, and deadlines. It supports the Project Charter’s goal of enhancing visibility by offering a clear, centralized view of project health, helping identify bottlenecks and ensure timely interventions. The feature is designed to be accessible and intuitive, catering to stakeholders with varying technical expertise, and must generate reports efficiently to meet the 3–4 week development timeline. Reports can be viewed in the UI or exported as PDFs for presentations, aligning with the Project Charter’s deliverable of summary reports.

**Priority**: Medium. This feature supports oversight and decision-making, with a benefit rating of 6/9. The penalty for non-implementation is 4/9, as manual reporting could suffice but would reduce efficiency and stakeholder engagement.

#### 4.6.2 Stimulus/Response Sequences

* **Progress Overview Access**:

**Stimulus**: A **Mentor** or **Admin** navigates to the progress overview section for a project.

**Response**: The system retrieves project data from the database, displays task completion percentages, milestone timelines, and overdue tasks in charts and tables, and highlights critical issues (e.g., overdue deadlines in red).

* **Report Generation Request**:

**Stimulus**: A **Mentor** selects a team and clicks “Generate Report,” choosing a format (web or PDF) and optional date range.

**Response**: The system queries the database, compiles a summary of task completions, deadlines, and file submissions, and presents the report in the chosen format. A confirmation (e.g., “Report generated successfully”) is shown, and the report is downloadable if PDF.

* **Report Filtering**:

**Stimulus**: A user applies a filter (e.g., “Last 7 days” or “Team A”) to the progress overview or report.

**Response**: The system updates the UI to display filtered data, maintaining chart and table accuracy.

* **Report Generation Failure**:

**Stimulus**: The system encounters an error during report generation (e.g., database timeout).

**Response**: The system displays a red error message (e.g., “Report generation failed, please try again”) and logs the error for debugging.

#### 4.6.3 Functional Requirements

**REQ-18**: The system shall display task completion percentages and milestone timelines for each project, verifiable via progress UI.

**REQ-19**: The system shall generate summary reports for team activity (e.g., tasks completed, overdue deadlines), available in PDF or web format, verifiable via report output.

## 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

* The system shall load the **Kanban** board in <2 seconds for up to 100 tasks, verifiable via browser performance tools.
* File uploads shall complete in <5 seconds for files up to the maximum size (**TBD**), verifiable via API response time.
* Dashboard charts shall render in <3 seconds, verifiable via UI performance testing.

### 5.2 Safety Requirements

No safety-critical requirements apply, as the portal does not interact with physical systems or sensitive data beyond project documentation.

### 5.3 Security Requirements

* Passwords shall be stored hashed using bcrypt, verifiable via database inspection.
* **JWT** tokens shall expire after 24 hours, verifiable via API authentication logs.
* The portal shall adhere to Custor’s internal data security policy for applications.
* System shall log all user actions (e.g., logins, task updates) for auditing, verifiable via logs.

### 5.4 Software Quality Attributes

**Usability**: **Interns** shall learn the UI in <30 minutes, verifiable via user testing.

**Reliability**: The system shall maintain 99% uptime during business hours, verifiable via server logs.

**Maintainability**: Code shall follow Angular/.NET best practices for future updates, verifiable via code review.

**Responsiveness**: UI shall adapt to different screen sizes, verifiable via browser testing.

### 5.5 Business Rules

* Only **Mentors** can approve task completion or document submissions.
* **Admins** can modify any user’s role or project/team assignments.
* **Interns** can only view/edit their assigned projects/tasks.

## 6. Other Requirements

* The database schema shall support future scalability for additional project types and teams.
* The system shall integrate with GitHub for code collaboration, verifiable via repository access.

## Appendix A: Glossary

**JWT**: JSON Web Token for secure authentication.

**Kanban**: Visual task management system with columns for task states.

**Role-Based Access Control**: Restricts system access based on user roles (**Intern**, **Mentor**, **Admin**).

**Kanban Board**: UI component for drag-and-drop task management.

**Comment Thread**: Inline communication feature for tasks and documents.

**Dashboard**: UI for **Mentors** and **Admins** to monitor progress.

## Appendix B: Analysis Models

**System Architecture Diagram**: To be included, showing frontend (Angular), backend (ASP.NET Core), database (SQL Server), and **JWT** authentication flow.

**Data Flow Diagram**: To be included, illustrating task management, document uploads, and dashboard data flows.

## Appendix C: To Be Determined List

1. Maximum file upload size.
2. Performance thresholds for dashboard rendering.
3. Specific file types supported for uploads.