Software Requirements Specification (SRS)

Project Title: Capstone Management System

Author: Basant Hamada.

Date: July 2025

Version: 1.0

# 1. Introduction

## 1.1 Purpose

**The purpose of the system is to facilitate the management of the Capstone project within the school through a digital platform that helps the administration, supervisors, and students organize and submit tasks, and track team progress.**

## 1.2 Scope

The system provides comprehensive management for Capstone projects within the school. The system’s structure is organized as follows:

* The school is divided into multiple **Grades**.
* Each Grade includes a number of **Classes**.
* Each Class contains several **Teams**, and each Team consists of:
* A **Team Leader**, who is responsible for submitting tasks.
* **Students**, who can view all tasks assigned within the project’s **Phases**.
* Only the Team Leader is authorized to submit tasks, while all team members can view them.
* The system includes multiple user roles with specific permissions:
* **Super Admin**:  
  Has full access to view all system data but cannot make modifications. The Super Admin is responsible for managing student, teacher, and engineer accounts.
* **Admin**:  
  Responsible for creating **Phases** and assigning **Tasks** to the appropriate Teams.
* **Teachers and Engineers**:  
  They monitor students’ and teams’ progress by reviewing submitted tasks and tracking which members submitted each task. This supports effective evaluation and supervision.

This structure enables clear task management, role-based permissions, and transparency across the entire Capstone workflow.

## 1.3 Definitions

1. **Capstone**: A graduation project implemented by student teams within the school and managed through multiple phases, from analysis to delivery.
2. **Phase**: A stage in the Capstone project such as system analysis, design, development, or testing. Created by the Admin, with tasks assigned to teams.
3. **Task**: A specific assignment given to a team within a phase. Only the Team Leader can submit it, while other team members can view it.
4. **Grade**: The educational level of students (e.g., Grade 1, Grade 2, etc.), and each Grade contains multiple Classes.
5. **Class**: A group of students within a specific Grade, organized into teams.
6. **Team**: A group of students in one Class working together on a Capstone project.
7. **Team Leader**: A student appointed as the team leader, responsible for submitting tasks on behalf of the team.
8. **Student**: A member of the Capstone team who can view tasks but is not allowed to submit them.
9. **Admin**: The project manager who creates phases and tasks, and monitors team progress.
10. **Super Admin**: The school administration who can view all system data but cannot edit. Manages user accounts such as students, teachers, and engineers.
11. **Teacher / Engineer**: A supervisor who monitors team progress and reviews task submissions within the system.
12. **Class Model**: A base model that defines the shared attributes among all system users (e.g., name, email, password, role), and is inherited by all account types.

# · 2. Overall Description

## 2.1 Product Perspective

The system is a standalone application used within a technical school to manage students’ Capstone projects. It does not depend on any external systems and provides all the necessary tools and features to handle the entire Capstone process from one unified platform.

The application is used by multiple types of users, including:

* **Super Admin**: Has full access to view system data but cannot modify it. Responsible for managing user accounts (students, teachers, engineers, etc.).
* **Admin**: Manages project phases and tasks, and assigns them to teams.
* **Teacher / Engineer**: Monitors submitted tasks and can see who submitted each one.
* **Team Leader**: A student who leads the team and is solely responsible for task submissions.
* **Students**: Can only view tasks within each phase but are not allowed to submit them.
* The structural organization of the system follows:
* **Grades** → contain **Classes** → each Class contains **Teams** → each Team has a **Team Leader** and multiple **Students**.

All user roles inherit shared properties from a common **Class Model**, which defines the core attributes of every account in the system.

## 2.2 Product Functions

The system provides a set of core functions that enable each user role to fulfill its responsibilities within the Capstone management workflow. These functions include:

* Creating **project phases (Phases)** by the Admin and assigning relevant **tasks (Tasks)** to each phase.
* Full management of **teams** within each Class, including the assignment of a **Team Leader** for every team.
* Displaying a comprehensive **Dashboard** for the Super Admin to monitor the entire system without editing permissions.
* Allowing **Teachers and Engineers** to review submitted tasks and identify which students submitted them.
* Granting **Team Leaders** the exclusive ability to **submit tasks** on behalf of their teams.
* Allowing all **Students** to **view tasks** assigned within each project phase, without submission privileges.
* Enabling the **Super Admin** to manage user accounts (students, teachers, engineers, etc.).

## 2.3 User Classes and Characteristics

|  |  |  |
| --- | --- | --- |
| User | Description | Permissions |
| Super Admin | School administration | View all system data (read-only); manage user accounts (students, teachers, engineers)  View all system data (read-only); manage user accounts (students, teachers, engineers). |
| Admin | Capstone project coordinator. | Create phases, assign tasks, and manage teams. |
| Team Leader | Team representative. | Submit tasks on behalf of the team. |
| Student | Capstone team member. | View tasks within assigned phases only. |
| Engineer | Technical/project supervisor. | Same as Teacher, with focus on technical task review and progress tracking. |
| Teacher | Academic supervisor | Review submitted tasks and track which students submitted them. |

## 2.4 Operating Environment

The system will be a web-based application using:  
 · **Frontend**: HTML, CSS, JavaScript with React.js.

· **Backend**: ASP.NET Core Web API.

· **Database**: SQL Server.

## 2.5 Design and Implementation Constraints

· All user roles inherit from a **base model called "Account"**, which includes shared attributes.

· The user interface should be **simple and easy to use**, especially for school students.