# Use Case Specification Document

<<Adjenda>> Development project

Team Members

| # | Name |
|---|---|
| 1 | Dan Imbimbo |
| 2 | Tristan Somcio |
| 3 | Jake Breitfeller |
| 4 | Kevin Sanchez |
| | |

# Contents

# 1. Stakeholders

[Instructions: Identify anyone and everyone who has interest in this system (users, managers, sponsors, etc.). Stakeholders should be humans or human organizations]

| **Stakeholder** | **Description (**how is the stakeholder impacted by the success or failure of this project?**)** |
|---|---|
| Instructors at university | The professor's ability to automate attendance relies on the ability of our system to provide said service as well as the reliability of our service to be operating at all times. The professor stands to gain better interactivity amongst students via creation of polls and attendance sheets which can be viewed and downloaded any time. |
| Students at university | The students are provided with collaborative resources in the form of shared schedules and class groups. By allowing students to compare schedules they can make decisions about class groups based upon shared free time that can be used for cooperative endeavors. |
| Software developers | Since the developers of this software are also students in University there are certain baseline insights and understandings that already exist pertaining to the relevancy of the project scope. The requirements of the finalized software are well understood by the developers and provide useful outcomes for the developers given they apply the software for their personal use upon completion. |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

# 2. Actors and Goals

[Instructions: In table given below, Identify the roles of people or devices that will directly interact with the system, their types (initiating vs. participating) and the goals of the initiating actors.]

| **Actor** | **Initiating / Participating** | **Description** |
|---|---|---|
| Instructor | Initiating & Participating | ● To ease the process of taking |

| | | |
|---|---|---|
| | | attendance through generated attendance codes<br>● To create courses and rosters for their students<br>● To create polls for their students to interact with<br>● To provide calendar events for their students in a given course |
| Student | Initiating & Participating | ● To work in collaborative groups with their classmates<br>● To log their attendance for individual classes<br>● To provide responses to their instructors' polls<br>● To keep track of their events via the calendar |
| Database | Participating | ● To retrieve stored data (viewing courses, class rosters, attendance logs, groups, polls, calendars, and poll results, as well as logging in)<br>● To insert new data (creating accounts, creating courses, adding to rosters, generating attendance codes, creating groups, adding to groups, creating polls, and adding calendar events)<br>● To update stored data (attendance logging, poll response totals, and changing passwords of accounts)<br>● To remove existing data (removing from roster and removing calendar events) |
| University Email Service | Participating | ● To validate newly created accounts |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

| | | |
|---|---|---|

# 3. Use Case Specifications

Each use case description should be provided under its own sub-section. Also, note that each use case should have a name (a verb phrase) and unique identifier (e.g., UC1, UC2, and so on).

## 3.1. Account Creation / UC-1

| **UC-1** | **Account Creation** |
|---|---|
| **Related Requirements:** | SRS-3.1.2.1, SRS-3.1.2.2, SRS-3.5.1 |
| **Initiating Actor:** | Any of: Student, Instructor |
| **Actor's Goal:** | To successfully sign up and create an account to utilize the System. |
| **Participating Actors:** | University Email Service, Database |
| **Preconditions:** | The **Student/Instructor** does not already have an account in the **Database** and is on the login/sign up page. |
| **Post-conditions:** | The **Student/Instructor** will have created an account to utilize the **System** and will be registered in the **Database**. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → The **Student/Instructor** selects the "Sign Up" button on the entry page of the Adjenda website. |
| 2 | ← The website takes the **Student/Instructor** to the registration page where they can provide a Username, Institution Email & Password (must contain at least 8 characters with 1 uppercase letter, 1 number & one symbol) (no fields may be left blank). |
| 3 | → **Student/Instructor** provides aforementioned credentials and selects "Create Account". |
| 4 | ← The **System** validates the username and password and checks if the Institutional Email utilizes the proper extension. |
| 5 | ← The **Database** checks if the email address belongs to an Instructor and assigns that account as a one, otherwise the account is defaulted to a Student account. |
| 6 | ← The **System** emails the address with a verification code for signup. |
| 7 | ← The **System** sends the **Student/Instructor** to the verification page where they enter the code that was received through the **University Email Service**. |
| 8 | → The **Student/Instructor** enters their verification code to complete account registration. |
| 9 | ← The **System** stores the account in the **Database**. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: **Student/Instructor** enters invalid/missing credentials during account creation<br><br>Flow of events for this scenario:<br>1. ← **System** detects username, password and email do not meet the specified criteria and blocks and signals to the user to enter valid inputs. |

| | |
|---|---|
| | 2. → **Student/Instructor** supplies valid inputs for all fields and the **System** continues to step 4 of the main success scenario. |
| 8a | Scenario Description: **Student/Instructor** enters an invalid verification code<br><br>Flow of events for this scenario:<br>1. ← **System** detects invalid verification key from user.<br>2. ← **System** alerts the user that the wrong verification key was entered and sends a new verification key to the provided email from step 4.<br>3. → **Student/Instructor** enters valid verification key and continues to step 9 of the main success scenario. |

## 3.2. Account Login / UC-2

| **UC-2** | **Account Login** |
|---|---|
| **Related Requirements:** | SRS-3.1.1.1, SRS-3.3 |
| **Initiating Actor:** | Any of: Student, Instructor |
| **Actor's Goal:** | Provide access to the default landing page upon successful login. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Student/Instructor** has already produced an account (Account Creation / UC-1) and is on the login/sign up page. |
| **Post-conditions:** | The **Student/Instructor** is sent to their dashboard after providing login credentials. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student/Instructor** provides the username and password (not blank) for their account and clicks on the "Submit" button. |
| 2 | ← The **System** logs the **Student/Instructor** into their account by matching their credentials with an account in the **Database** and sends them to the dashboard page containing their classes. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 1a | Scenario Description: **Student/Instructor** leaves a field blank<br><br>Flow of events for this scenario:<br>1. ← **System** detects blank fields and blocks the **Student/Instructor** from submitting while blank fields are present.<br>2. → **Student/Instructor** enters credentials to both fields and clicks "Submit" button allowing the **System** to continue to step 2 of the main success scenario. |
| 2a | Scenario Description: **Student/Instructor** enters invalid credentials<br><br>Flow of events for this scenario:<br>1. ← **System** is unable to match username and password with any existing accounts in the **Database**.<br>2. ← **System** stays on the login page and alerts the **Student/Instructor** that the credentials |

| | |
|---|---|
| | they entered were invalid.<br>3. → **Student/Instructor** enters valid login credentials and continues to step 2 of the main success scenario. |

## 3.3. Course Creation / UC-3

| **UC-3** | **Course Creation** |
|---|---|
| **Related Requirements:** | SRS-3.1.9.1, SRS-3.1.9.2, SRS-3.1.3.2 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Produce a course so the Instructor and their Students can utilize Agenda's services. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Instructor** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2). |
| **Post-conditions:** | A course is created that includes the **Instructor** who has administrative privileges for the course and each **Student** who can use the course page for educational/collaborative services. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Instructor** selects the "Create New Course" option from their dashboard. |
| 2 | ← The **System** sends the **Instructor** to the course creation page where they are prompted to enter a course name, section, schedule (days of the week, start and end time of class, the semester it is in), and initial roster. |
| 3 | → After filling out all of the fields the **Instructor** selects the "Create Course" button at the end of the form. |
| 4 | ← The **System** produces a new course in the **Database** with the given information and adds the course to the dashboard of the **Instructor** and each **Student** who was entered into the roster. |
| 5 | ← The **System** sends the **Instructor** to the new course's page. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: **Instructor** leaves blank fields in the course creation page<br><br>Flow of events for this scenario:<br>1. ← **System** requires all fields to be filled in order to create a course, so the **Instructor** is blocked from submitting.<br>2. → **Instructor** fills every field, selects the "Create Course" button and continues to step 4 of the main success scenario. |

## 3.4. Add Student to Roster / UC-4

| **UC-4** | **Add Student to Roster** |
|---|---|
| **Related Requirements:** | SRS-3.1.3.3 |

| | |
|---|---|
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to add a Student to their course roster. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Instructor** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), and the course which the **Instructor** seeks to add the **Student** to already exists (Class Creation / UC-3). |
| **Post-conditions:** | The class roster is updated to be consistent with the course enrollment roster of the University for the given course. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Instructor** selects the course which they seek to modify from their dashboard. |
| 2 | → From the course page the **Instructor** selects the navigation option for the class roster. |
| 3 | → The **Instructor** selects the "Add Student" option. |
| 4 | ← The **Instructor** is prompted to enter a student email. |
| 5 | → The **Instructor** enters the email of the **Student** who enrolled. |
| 6 | ← The **System** adds the **Student** to the course (the course is added to the dashboard of the **Student** and the roster stored on the **Database** is updated using the username of the account which the student email is linked to). |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 5a | Scenario Description: **Instructor** enters an email which is not linked with a **Student** account<br><br>Flow of events for this scenario:<br>1. ← **System** is unable to match the email with any existing **Student** accounts in the **Database**.<br>2. ← **System** alerts the **Instructor** to enter a valid **Student** email.<br>3. → **Instructor** enters an email which is linked with a **Student** account and the **System** continues to step 6 of the main success scenario. |

## 3.5. Remove Student from Roster / UC-5

| **UC-5** | **Remove Student from Roster** |
|---|---|
| **Related Requirements:** | SRS-3.1.3.3 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to drop a Student from their course roster. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Instructor** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), the course which the **Instructor** seeks to add the **Student** to already exists (Course Creation / UC-3), and the |

| | |
|---|---|
| | **Student** has already been added to its roster (Add Student to Roster / UC-4). The **Instructor** has also already selected the course from their dashboard. |
| **Post-conditions:** | The **Student** is removed from the roster, the course is removed from the **Student**'s dashboard, the events for the course are dropped from the **Student**'s calendar, and the **Student** is dropped from any groups within that course |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Instructor** navigates to the class roster page and selects the "Drop Student" button next to the **Student** listed in the roster. |
| 2 | ← **System** provides a pop up asking if the **Instructor** wants to drop the **Student** with options for "Yes" or "No". |
| 3 | → **Instructor** selects "Yes" from the pop up. |
| 4 | ← **System** removes the **Student** from the associated **Database** fields for the **Student**'s enrollment within that course (removes **Student** from roster, removes course from **Student**'s dashboard, removes events for course from **Student**'s calendar, removes **Student** from any groups within the course) |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: **Instructor** cancels option to "Drop Student"<br><br>Flow of events for this scenario:<br>1. → **Instructor** selects "No" from the pop up to remove the **Student** from the course<br>2. ← **System** does not remove the **Student** from the roster and returns **Instructor** to the roster page, ending the use case. |

## 3.6. Generate Attendance Code / UC-6

| **UC-6** | **Generate Attendance Code** |
|---|---|
| **Related Requirements:** | SRS-3.1.4.2, SRS-3.1.4.3, SRS-3.5.2 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to generate a room code to display to the class for an attendance check. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Instructor** has already created an account (Account Creation / UC-1), logged in to the **System** (Account Login / UC-2), created a course (Course Creation / UC-3), and populated the course with **Students** (Add Student to Roster / UC-4). The **Instructor** has also already selected the course from their dashboard. |
| **Post-conditions:** | The **System** will have generated a randomized code for attendance displayed on the page for the **Instructor** to copy onto a board for **Students** to then enter onto the **System** from their end. |

| **Flow of Events for Main Success Scenario:** | |
|---|---|
| 1 | → **Instructor** selects the navigation option for the class roster. |
| 2 | → **Instructor** clicks on the "Take Attendance" button on the roster page. |
| 3 | ← **System** generates a randomized room code, stores the code in the **Database**, displays it to the **Instructor** on the roster page, and also displays a checkbox to the side of every **Student** on the class's roster representing their marked attendance. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: Attendance code was already generated<br><br>Flow of events for this scenario:<br>1. ← **System** detects an existing code in the **Database** for the current attendance log.<br>2. ← **System** alerts the **Instructor** that there is already an existing room code and does not generate a new one, ending the use case. |

## 3.7. Log Attendance / UC-7

| **UC-7** | **Log Attendance** |
|---|---|
| **Related Requirements:** | SRS-3.1.4.1, SRS-3.1.4.3, SRS-3.5.2 |
| **Initiating Actor:** | Student |
| **Actor's Goal:** | Allow the Student to log their attendance via the Instructor's attendance code. |
| **Participating Actors:** | Database, Instructor |
| **Preconditions:** | The **Student** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), the course has already been created by the **Instructor** (Class Creation / UC-3), the **Student** has already been added to that course (Add Student to Roster / UC-4), and the attendance code has also already been generated by the **Instructor** (Generate Attendance Code / UC-6). The **Student** has also already selected the course from their dashboard. |
| **Post-conditions:** | The **Student** will have been marked as present in the attendance log. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student** selects the navigation option for the class roster. |
| 2 | → On the class roster page the **Student** selects the option to "Enter Attendance Code". |
| 3 | ← **System** prompts the **Student** to enter the attendance code provided by their **Instructor**. |
| 4 | → **Student** enters the attendance code in the presented field. |
| 5 | ← **System** marks the **Student** as present on the class roster and stores their attendance in the **Database**. |

| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
|---|---|
| 4a | Scenario Description: Student enters invalid attendance code<br><br>Flow of events for this scenario:<br>1. ← **System** detects that the entered attendance code does not match the one in the **Database**.<br>2. ← **System** keeps the attendance code prompt open and alerts the **Student** to enter the correct code.<br>3. → **Student** enters the valid attendance code allowing the **System** to proceed to step 5 of the main success scenario. |

## 3.8. View Class Attendance / UC-8

| **UC-8** | **View Class Attendance** |
|---|---|
| **Related Requirements:** | SRS-3.1.4.4 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to view a previous attendance log. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Instructor** has already created an account (Account Creation / UC-1), logged in to the **System** (Account Login / UC-2), created a course (Course Creation / UC-3), populated the course with **Students** (Add Student to Roster / UC-4), generated an attendance code (Generate Attendance Code / UC-6), and **Students** have logged their attendance (Log Attendance / UC-7). The **Instructor** has also already selected the course from their dashboard. |
| **Post-conditions:** | The **Instructor** will be presented with the specified attendance log. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Instructor** navigates to the class roster page and clicks on the "View Attendance Logs" button. |
| 2 | ← **System** directs the **Instructor** to a new page with a prompt saying "Select Date" accompanied by a dropdown menu. |
| 3 | → **Instructor** clicks on the dropdown menu. |
| 4 | ← **System** displays dropdown menu of all attendance dates listed in chronological order. |
| 5 | → **Instructor** selects a date. |
| 6 | ← **System** directs **Instructor** to a new page displaying the selected date's attendance log with all the registered students who were present for that class from the **Database**. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 6a | Scenario Description: Attendance has not yet been logged<br><br>Flow of events for this scenario:<br>1. ← **System** detects that attendance has not yet been logged for the selected date. |

| | |
|---|---|
| | 2. ← **System** alerts the **Instructor** of this and the **Instructor** is returned back to step 2 of the main success scenario. |

## 3.9. Download Class Attendance / UC-9

| **UC-9** | **Download Class Attendance** |
|---|---|
| **Related Requirements:** | SRS-3.1.4.5 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to download the selected attendance log. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Instructor** has already created an account (Account Creation / UC-1), logged in to the **System** (Account Login / UC-2), created a course (Course Creation / UC-3), populated the course with **Students** (Add Student to Roster / UC-4), generated an attendance code (Generate Attendance Code / UC-6), and **Students** have logged their attendance (Log Attendance / UC-7). The **Instructor** has also already selected the course from their dashboard and is currently viewing a selected attendance log (View Class Attendance / UC-8). |
| **Post-conditions:** | The **Instructor** will have a downloaded copy of the selected attendance log on their machine. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Instructor** clicks on the "Download Log" button on the page of the previously selected log that was retrieved from the **Database**. |
| 2 | ← **System** provides a pop up asking if the **Instructor** wants to download the selected attendance log with the options of "Yes" or "No" |
| 3 | → **Instructor** selects the "Yes" option from the pop up. |
| 4 | ← **System** downloads the attendance log onto the **Instructor**'s machine as a pdf file. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: Cancel attendance log download<br><br>Flow of events for this scenario:<br>1. → **Instructor** selects the "No" option from the pop up.<br>2. ← **System** returns to the selected date's attendance log page, ending the use case. |

## 3.10. Group Creation / UC-10

| **UC-10** | **Group Creation** |
|---|---|
| **Related Requirements:** | SRS-3.1.5.1, SRS-3.5.1 |
| **Initiating Actor:** | Student |

| | |
|---|---|
| **Actor's Goal:** | Allow the Student to create a group to facilitate collaboration with peers. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Student** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), the course has already been created by the **Instructor** (Class Creation / UC-3), and the **Student** has already been added to that course (Add Student to Roster / UC-4). The **Student** has also already selected the course from their dashboard. |
| **Post-conditions:** | The **Student** will have created a group for peers to join. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student** selects the navigation option for groups. |
| 2 | → From the groups page the **Student** clicks on the "Create Group" button. |
| 3 | ← **System** provides a pop up with a "Group Name" field and a scrollable list containing the names of each registered **Student** in the class roster who can be invited with checkboxes next to each. |
| 4 | → **Student** fills in the name field, selects at least one other **Student** to join the group, and clicks the "Create" button. |
| 5 | ← **System** inserts the newly created group into the **Database**, sends each selected **Student** an invitation to access the group, and lists the group on the course's groups page. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 4a | Scenario Description: **Student** does not enter a name for the group<br><br>Flow of events for this scenario:<br>1. ← **System** detects the "Group Name" field is blank, blocks the **Student** from submitting, and alerts the **Student** to enter a name in the field.<br>2. → **Student** enters a group name and clicks the "Create" button, allowing the **System** to continue to step 5 of the main success scenario. |
| 4b | Scenario Description: **Student** does not select another group member<br><br>Flow of events for this scenario:<br>1. ← **System** detects no other **Student** was selected from the list, blocks the initiating **Student** from submitting, and informs the initiating **Student** to select at least one group member to invite.<br>2. → **Student** selects another **Student** from the list and clicks the "Create" button, allowing the **System** to continue to step 5 of the main success scenario. |

## 3.11. Group Invitation / UC-11

| **UC-11** | **Group Invitation** |
|---|---|
| **Related Requirements:** | SRS-3.1.5.2 |
| **Initiating Actor:** | Student |
| **Actor's Goal:** | Allow the Student to invite more members to their created group. |

| | |
|---|---|
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Student** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), the course has already been created by the **Instructor** (Class Creation / UC-3), the **Student** has already been added to that course (Add Student to Roster / UC-4). The **Student** has also already selected the course from their dashboard, created a peer group for the course (Group Creation / UC-10), and is currently on the course's groups page. |
| **Post-conditions:** | The **Student** will have invited additional peers to join their group. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student** chooses their group from the existing list on the course's groups page. |
| 2 | ← **System** verifies with the **Database** that the **Student** has access privileges to the group. |
| 3 | → **Student** selects the "Invite Students" option within their group. |
| 4 | ← **System** provides a pop up with a scrollable list containing the names of each **Student** in the class roster who is not currently in the group with checkboxes next to each. |
| 5 | → **Student** ticks the checkboxes of each **Student** they want to invite to the group and then clicks the "Invite" button. |
| 6 | ← **System** notifies the initiating **Student** that the invite has been sent and sends each selected **Student** an invitation to access the group. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 5a | Scenario Description: **Student** selects no one to invite<br><br>Flow of events for this scenario:<br>1. ← **System** detects the no **Student** was selected from the list, blocks the initiating **Student** from submitting, and alerts the initiating **Student** that no **Student** on the list was selected.<br>2. → **Student** selects another **Student** on the list to invite to the group and clicks the "Invite" button, allowing the **System** to continue to step 6 of the main success scenario. |
| 5b | Scenario Description: **Student** cancels invitation<br><br>Flow of events for this scenario:<br>1. → **Student** clicks the "Cancel" button in the pop up.<br>2. ← **System** returns to the **Student**'s group page, ending the use case. |

## 3.12. Group Request / UC-12

| **UC-12** | **Group Request** |
|---|---|
| **Related Requirements:** | SRS-3.1.5.3, SRS-3.1.5.4 |
| **Initiating Actor:** | Student |

| | |
|---|---|
| **Actor's Goal:** | Allow the Student to request access to an existing group. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Student** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), the course has already been created by the **Instructor** (Class Creation / UC-3), and the **Student** has already been added to that course (Add Student to Roster / UC-4). The **Student** has also already selected the course from their dashboard, is currently on the course's groups page, and there are existing groups created (Group Creation / UC-10). |
| **Post-conditions:** | The **Student** will have requested to join a specified group. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student** selects a group from the existing list on the course's groups page. |
| 2 | ← **System** checks the **Database** and detects that the **Student** does not have access privileges to the group. |
| 3 | ← **System** provides a pop up asking if the **Student** would like to request to join the group with the options of "Yes" or "No". |
| 4 | → **Student** clicks the "Yes" button. |
| 5 | ← **System** notifies the **Student** that the request to access the group has been sent and sends the join request to the owner of the group. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 4a | Scenario Description: **Student** does not send group request<br><br>Flow of events for this scenario:<br>1. → **Student** clicks the "No" button in the pop up.<br>2. ← **System** returns to the **Student** the course's groups page, ending the use case. |

## 3.13. Poll Creation / UC-13

| **UC-13** | **Poll Creation** |
|---|---|
| **Related Requirements:** | SRS-3.1.7.1 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to create a poll for their Students to respond to. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Instructor** has already produced an account (Account Creation / UC-1), logged in to the **System** (Account Login / UC-2), created a course (Course Creation / UC-3), and populated the course with **Students** (Add Student to Roster / UC-4). The **Instructor** has also already selected the course from their dashboard. |

| | |
|---|---|
| **Post-conditions:** | The **Instructor** will have created a poll for all of their **Students** in the specified course. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Instructor** selects the navigation option for polls. |
| 2 | → From the polls page the **Instructor** clicks on the "Create Poll" button. |
| 3 | ← **System** provides a pop up with a field for the poll's question and multiple fields for its chosen responses. |
| 4 | → **Instructor** fills in the "Poll Question" field and more than one "Response" field, and then clicks the "Create" button. |
| 5 | ← **System** inserts the newly created poll into the **Database**, notifies each **Student** in the course to participate in the poll, and lists the poll on the course's polls page. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 4a | Scenario Description: **Instructor** does not provide a question<br><br>Flow of events for this scenario:<br>1. ← **System** detects the "Poll Question" field is blank, blocks the **Instructor** from submitting, and notifies the **Instructor** to enter a question in the field.<br>2. → **Instructor** enters a poll question and clicks the "Create" button, allowing the **System** to continue to step 5 of the main success scenario. |
| 4b | Scenario Description: **Instructor** does not provide more than one response<br><br>Flow of events for this scenario:<br>1. ← **System** detects more than one "Response" field was not filled, blocks the **Instructor** from submitting, and instructs the **Instructor** to enter at least two responses to the poll.<br>2. → **Instructor** enters more than one response and clicks the "Create" button, allowing the **System** to continue to step 5 of the main success scenario. |

## 3.14. Poll Response / UC-14

| **UC-14** | **Poll Response** |
|---|---|
| **Related Requirements:** | SRS-3.1.7.2, SRS-3.1.7.3 |
| **Initiating Actor:** | Student |
| **Actor's Goal:** | Allow the Student to submit their response to the Instructor's poll. |
| **Participating Actors:** | Database, Instructor |

| | |
|---|---|
| **Preconditions:** | The **Student** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2), the course has already been created by the **Instructor** (Class Creation / UC-3), the **Student** has already been added to that course (Add Student to Roster / UC-4), and a poll has been created by the **Instructor** (Poll Creation / UC-13). The **Student** has also already selected the course from their dashboard and is currently on the course's polls page. |
| **Post-conditions:** | The **Student**'s response to the poll will have been recorded. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → On the course's polls page the **Student** selects a poll from the existing list. |
| 2 | ← **System** provides a pop up containing the **Instructor**'s question and the potential responses that can be selected with checkboxes next to each. |
| 3 | → **Student** ticks the checkbox next to their chosen response and then clicks on the "Submit" button. |
| 4 | ← **System** updates the totals for each response of the poll in the **Database** and notifies the **Instructor** that a **Student** responded to the poll. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: No response was selected<br><br>Flow of events for this scenario:<br>1. ← **System** detects no response was selected from the list, blocks the **Student** from submitting, and alerts the **Student** to choose a response.<br>2. → **Student** selects a response from the list and clicks the "Submit" button, allowing the **System** to continue to step 4 of the main success scenario. |

## 3.15. View Poll Results / UC-15

| **UC-15** | **View Poll Results** |
|---|---|
| **Related Requirements:** | SRS-3.1.7.4 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to view the overall Student response to their poll. |
| **Participating Actors:** | Database, Student |
| **Preconditions:** | The **Instructor** has already created an account (Account Creation / UC-1), logged in to the **System** (Account Login / UC-2), created a course (Course Creation / UC-3), populated the course with **Students** (Add Student to Roster / UC-4), created a poll (Poll Creation / UC-13), and **Students** have submitted a response to the poll (Poll Response / UC-14). The **Instructor** has also already selected the course from their dashboard and is currently viewing the course's polls page. |
| **Post-conditions:** | The **Instructor** will be presented with the results to the specified poll. |

| | |
|---|---|
| **Flow of Events for Main Success Scenario:** | |
| 1 | → On the course's polls page the **Instructor** selects the "View Results" button next to one of their existing polls. |
| 2 | ← **System** provides a pop up displaying a bar graph of the overall selected **Student** totals stored in the **Database** for each response to the poll. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 2a | Scenario Description: No responses to the poll<br><br>Flow of events for this scenario:<br>1. ← **System** detects no responses have yet been recorded in the **Database**, alerts the **Instructor** of this, and does not display a bar graph of the results in the pop up, ending the use case. |

## 3.16. Download Poll Results / UC-16

| **UC-16** | **Download Poll Results** |
|---|---|
| **Related Requirements:** | SRS-3.1.7.5 |
| **Initiating Actor:** | Instructor |
| **Actor's Goal:** | Allow the Instructor to download the selected poll's results. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Instructor** has already created an account (Account Creation / UC-1), logged in to the **System** (Account Login / UC-2), created a course (Course Creation / UC-3), populated the course with **Students** (Add Student to Roster / UC-4), created a poll (Poll Creation / UC-13), and **Students** have submitted a response to the poll (Poll Response / UC-14). The **Instructor** has also already selected the course from their dashboard and is currently viewing the course's polls page. |
| **Post-conditions:** | The **Instructor** will have a downloaded copy of the selected poll on their machine. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → On the course's polls page the **Instructor** selects the "View Results" button next to one of the existing polls. |
| 2 | → From the results page for the given poll, the **Instructor** selects the option to download the results of the poll. |
| 3 | ← The **System** produces a pdf of the poll results which is downloaded on the browser of the **Instructor**'s machine. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: No students have submitted results to the poll |

| | |
|---|---|
| | Flow of events for this scenario:<br>1 ← The **System** checks if any students in the class have submitted results to the **Database**.<br>2 ← If no results exist, the **System** prompts the **Instructor** that the poll is empty and cancels the download request, ending the use case. |

## 3.17. Add Calendar Event / UC-17

| **UC-17** | **Add Calendar Event** |
|---|---|
| **Related Requirements:** | SRS-3.1.6.2 |
| **Initiating Actor:** | Any of: Student, Instructor |
| **Actor's Goal:** | Allow Student/Instructor to add an event to their calendar. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Student/Instructor** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2). |
| **Post-conditions:** | The **Student/Instructor** will have a new event on their calendar. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student/Instructor** selects the "Calendar" link on the navigation bar. |
| 2 | → On the calendar page the **Student/Instructor** selects the "Add Event" option. |
| 3 | ← **System** provides a pop up with individual dropdown menus to select the "Month," "Day," "Start time," and "End time" of the event. |
| 4 | → **Student/Instructor** selects one of the provided options for each dropdown menu and clicks the "Add" button. |
| 5 | ← **System** updates the **Student/Instructor**'s stored events in the **Database** and displays the new event within their calendar on the calendar page. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 1 | Scenario Description: Conflicting event times<br><br>Flow of events for this scenario:<br>1. ← **System** detects that the **Student/Instructor** already has an event stored in the **Database** at the selected month, day, and start time.<br>2. ← **System** alerts the **Student/Instructor** to this conflict and does not add the new event to their calendar, ending the use case. |

## 3.18. Remove Calendar Event / UC-18

| **UC-18** | **Remove Calendar Event** |
|---|---|
| **Related Requirements:** | SRS-3.1.6.2 |
| **Initiating Actor:** | Any of: Student, Instructor |

| | |
|---|---|
| **Actor's Goal:** | Allow Student/Instructor to remove an event from their calendar. |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Student/Instructor** has already produced an account (Account Creation / UC-1), logged into the **System** (Account Login / UC-2), and has added an event to their calendar (Add Calendar Event / UC-18). The **Student/Instructor** is also already on the calendar page. |
| **Post-conditions:** | The **Student/Instructor** will have removed an event from their calendar. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → On the calendar page the **Student/Instructor** selects the "Remove Event" option. |
| 2 | ← **System** provides a pop up with a dropdown menu containing all of the **Student/Instructor**'s existing events. |
| 3 | → **Student/Instructor** selects the event they want to remove from the dropdown menu and then clicks the "Remove" button. |
| 4 | ← **System** removes the event from the **Student/Instructor**'s calendar by deleting its record from the **Database**. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 4a | Scenario Description: Calendar event was created by another user<br><br>Flow of events for this scenario:<br>1. ← **System** detects that the **Student/Instructor** is not the user who added the selected event.<br>2. ← **System** does not remove the event and alerts the **Student/Instructor** that they do not have permission to do so, ending the use case. |

## 3.19. Change Account Password / UC-19

| **UC-19** | **Change Account Password** |
|---|---|
| **Related Requirements:** | SRS-3.1.8.5, SRS-3.2.2, SRS-3.5.3 |
| **Initiating Actor:** | Any of: Student, Instructor |
| **Actor's Goal:** | Allow the Student/Instructor to change the password to their account |
| **Participating Actors:** | Database |
| **Preconditions:** | The **Student/Instructor** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2). |
| **Post-conditions:** | The password to the **Student/Instructor**'s account will be changed to the password they provided. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student/Instructor** selects the "Settings" link on the navigation bar |

| | |
|---|---|
| 2 | → On settings page the **Student/Instructor** selects the "Change password" option |
| 3 | ← **System** prompts the **Student/Instructor** to enter a new password (must contain at least 8 characters with 1 uppercase letter, 1 number & one symbol) and then enter it again for verification. |
| 4 | → **Student/Instructor** enters their new password in the two provided fields |
| 5 | ← **System** notifies the **Student/Instructor** that their password has been changed, updates the password associated with their account in the **Database**, and returns them to the settings page. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 4a | Scenario Description: **Student/Instructor** enters invalid password<br><br>Flow of events for this scenario:<br>1. ← **System** detects the password does not meet the specified criteria, blocks the **Student/Instructor** from submitting, and notifies them to enter a valid password.<br>2. → **Student/Instructor** supplies a valid password and the **System** continues to step 5 of the main success scenario. |
| 4b | Scenario Description: Entered passwords don't match<br><br>Flow of events for this scenario:<br>1. ← **System** detects that the password entered in the "Verification" field does not match the password entered in the "New Password" field.<br>2. ← **System** notifies the **Student/Instructor** to make sure that the two entered passwords match.<br>3. → **Student/Instructor** supplies matching passwords in both of the specified fields and the **System** continues to step 5 of the main success scenario. |

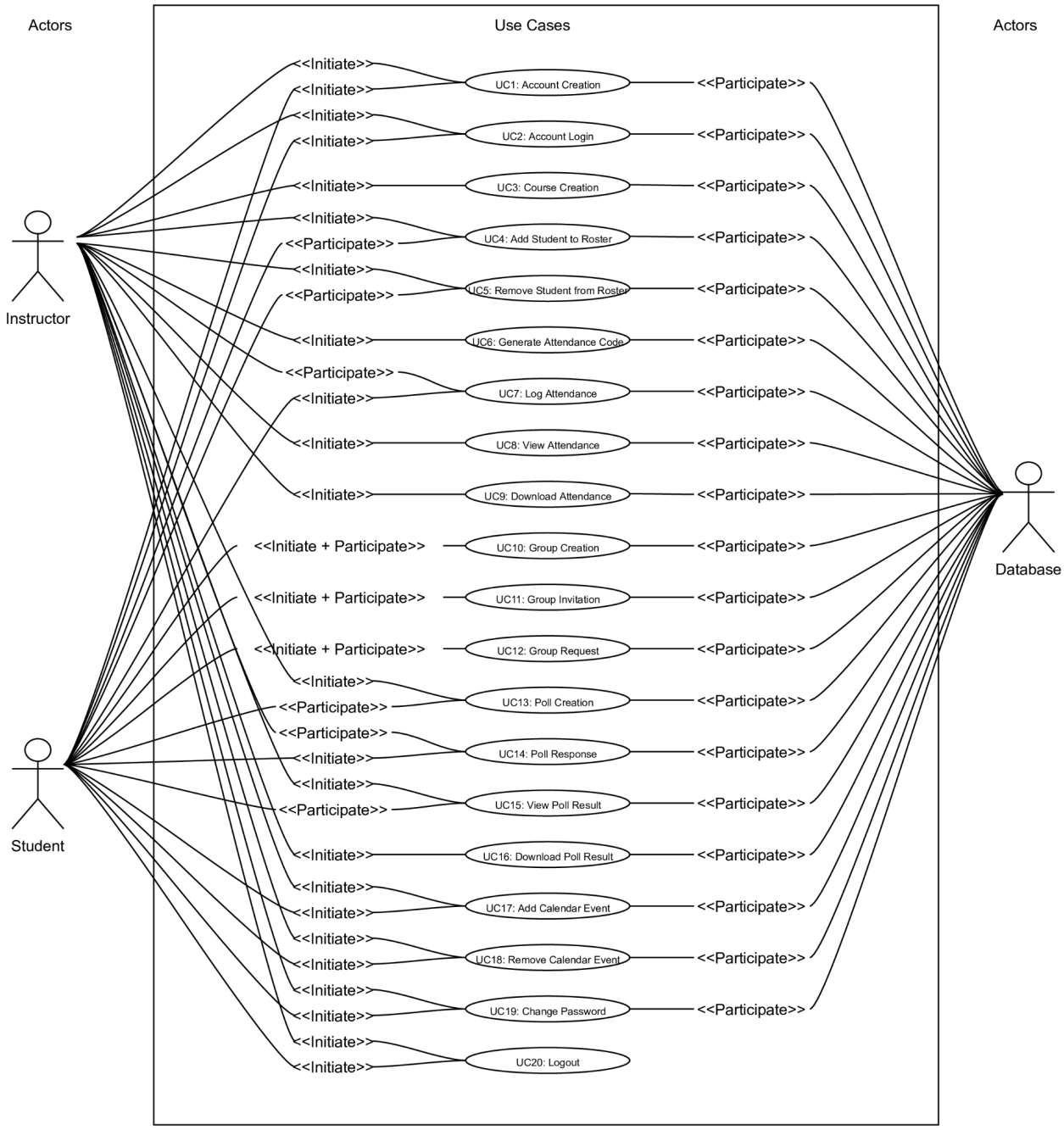
## 3.20. Account Logout / UC-20

| **UC-20** | **Account Logout** |
|---|---|
| **Related Requirements:** | SRS-3.1.8.6 |
| **Initiating Actor:** | Any of: Student, Instructor |
| **Actor's Goal:** | Allow the Student/Instructor to end their current session on the System |
| **Participating Actors:** | None |
| **Preconditions:** | The **Student/Instructor** has already produced an account (Account Creation / UC-1) and logged in (Account Login / UC-2). |
| **Post-conditions:** | The **Student/Instructor** will be logged out of their account. |
| **Flow of Events for Main Success Scenario:** | |
| 1 | → **Student/Instructor** selects the "Log Out" option on the navigation bar |
| 2 | ← **System** provides a pop up asking if the **Student/Instructor** sure they want to log out with options for "Yes" or "No". |
| 3 | → **Student/Instructor** selects "Yes" from the pop up. |

| | |
|---|---|
| 4 | ← **System** ends the **Student/Instructor**'s session, logging them out of their account and returning them to the initial login/signup page. |
| **Flow of Events for Extensions (Alternate Scenarios):**<br>**What could go wrong? List the exceptions to the routine and describe how they are handled** | |
| 3a | Scenario Description: Cancel logout<br><br>Flow of events for this scenario:<br>1. → **Student/Instructor** selects "No" from the pop up.<br>2. ← **System** does not log out the **Student/Instructor** and instead returns them to the page they were on beforehand, ending the use case. |

# 4. Use Case Diagram

[Instructions: In the space given below, provide the **use case diagram for the system-to-be**. See an example use case diagram below. Please note that the diagram shown below only represents the system-to-be partially**. For your system, you will have about twenty (20) use cases**]




# 5. Definitions and Acronyms

| Term | Definition |
|---|---|
| API | Application Programming Interface |

| | |
|---|---|
| SQL | The language being used to query, access, and modify our database. |
| HTML | The language being used for the website's frontend. |
| … | |
| … | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |