A logo with a black background

Description automatically generated

**Statement of Work (SOW)**

Baylor University

Software Engineering II Class Project

Spring 2024

Table of Contents

[1. Objective 3](#_Toc156558346)

[2. Expectations 4](#_Toc156558347)

[2.1 Students 4](#_Toc156558348)

[2.2 Baylor University 4](#_Toc156558349)

[2.3 Credera 4](#_Toc156558350)

[3. Project Details 5](#_Toc156558351)

[3.1 Project Assessment 5](#_Toc156558352)

[3.1.1 Problem Statement 5](#_Toc156558353)

[3.1.2 Milestones and Deliverables 5](#_Toc156558354)

[3.1.3 Schedule 9](#_Toc156558355)

[3.2 Scrums 10](#_Toc156558356)

[3.3 Group Member Roles 11](#_Toc156558357)

[3.4 Product Backlog 12](#_Toc156558358)

[4. Technology Stack Overview 13](#_Toc156558359)

[4.1 Java Spring (Boot) 13](#_Toc156558360)

[4.2 React 13](#_Toc156558361)

[4.3 REST 13](#_Toc156558362)

[4.4 Google Cloud Platform (GCP) 14](#_Toc156558363)

[4.5 MySQL 14](#_Toc156558364)

[5. Credera Mentors 15](#_Toc156558365)

# 1. Objective

Credera seeks to interact with students and faculty to develop deeper relationships through a semester-long project in Software Engineering II. Students will cooperate in teams with the professor and Credera mentors serving as guides to complete the project.

Credera will provide students with the opportunity to participate in a mock consulting project, where they will take the role of consultants to build a product for the Credera mentors, who will take the role of the clients and be the primary decision-makers regarding product requirements. Students will be expected to think and act like a consultant to build the best product they can, with their mentors offering guidance throughout their journey. In addition, just like with any consulting project, some requirements will be ambiguous or not as fleshed out as others. The specific implementation and execution of such requirements will be left to the teams’ discretion.

Credera mentors will also provide technical guidance to teams to assist them with unfamiliar technologies. They will provide students with a template project to get them started as well as answering questions in the dedicated communication channels between the students and the mentors.

# 2. Expectations

## 2.1 Students

Students should expect to face a real-world situation where they will be given a problem and are tasked with developing a solution that meets a set of requirements provided by the client. They will need to understand how to approach the problem, propose a solution to the client, and finally implement it. They will work in teams, just as commonly occurs on Credera projects, to solve the problem and present their findings, emulating actual work practices on Credera projects, such as regular deliverables and client input. This will not be an easy task. Students are expected to research and find solutions to their team’s issues, in addition to learning any unfamiliar technologies. Mentors will be available to assist and occasionally point in the right direction, but not to solve challenges. Students should come to their office hours/scrum meetings prepared with specific questions and deliverables complete.

## 2.2 Baylor University

Baylor University will partner with Credera to manage the project to better prepare their students for jobs following graduation. Professor(s) will work as a point of contact for Credera and the students on all project questions. Professor(s) will work with Credera to ensure project meets all class expectations.

## 2.3 Credera

Credera will provide 3-4 employees to participate in both on-campus activities and regular virtual meetings with the students. Through weekly Zoom calls with the students, Credera employees will be expected to mentor in best practices and problem-solving skills. Credera will be expected to work with professors to meet class expectations. Deliverable criteria will be given to students and mentors are expected to review and offer feedback to students.

# 3. Project Details

## 3.1 Project Assessment

### 3.1.1 Problem Statement

A new company, StudyBuddies, Inc., wants to create a custom experience for students and tutors alike to allow them to plan study sessions anywhere, on- and off-campus. You’ve been hired as a consultant to design and build a web application that allows connecting with other users to become “study buddies”, creating and scheduling meetups, recommending locations to study at, and more. The client wants your team to create a user-friendly web app for users to register under the appropriate classification, input their class(es) and/or area(s) of study, and find other students or tutors with similar profiles. In addition, your team can enhance the application with other features at your discretion.

### 3.1.2 Milestones and Deliverables

This semester will be broken out into four milestones for your technical implementation. Each team will be responsible for completing each milestone by the assigned week (see 3.1.3 Project Schedule for dates). At the end of each milestone, each team will present their current implementation to the mentors and professor during scrum. For the first three milestones, only an informal but thorough walkthrough of current complete functionality is expected. For the final milestone, a formal presentation is required.

In addition to these four milestones, teams will be provided with a list of (usually 5-6) weekly deliverables designed to keep them on track for milestone completion. As the weekly deliverables may change over the course of the semester, they are included in a separate document. The four milestones, however, are described below.

**Milestone 1**

* Team repository is set up (clone of template project)
* Pages are scaffolded out with routing to access them all (pages can be mostly empty, like the home page just saying “Home”, etc.)
* Call example Java endpoint (/ping) from client and display the results in the frontend (ask mentors for more details about this in our scrum meetings)

***Demo***

* Demonstrate that all pages exist by directly hitting their URLs (/home, /profile, etc.)
* Display the result of the test endpoint service call

**Milestone 2**

* Every page is implemented – sign-in/registration, profile page, home page, meetup creation/editing (at a minimum)
* In the service layer, be hitting a MySQL database and be able to create/edit account, add/edit classes/areas of study, add/edit “study buddies”, add/edit profile details, create/edit meetups, review tutors, etc.
* Project is deployed to GCP
* Continuous integration is set up (a Git commit triggers a build, which triggers a GCP deployment)
* In the frontend, an individual should be able to:
  + Register and log in
  + Select user type (student vs tutor)
  + Add and edit classes/areas of study
  + Search for and mark other users as “study buddies”
  + Create and edit study meetups
* Add unit tests to the Java (API) layer and add a testing step to the CI/CD pipeline (teams should maintain, update, and expand these tests going forward)

***Demo***

* Show the app running live on GCP
* Change some text on the home page and commit the change, then show that the CI/CD pipeline is working and that the message changed on the live site
* Demonstrate creating a new account and logging in
* Demonstrate creating a new study meetup and providing its details (date, time, location, subject, etc.)
* Show the deployment pipeline passing in GitHub Actions, as well as showing that unit tests were run and are passing

**Milestone 3**

* Recommendations: A critical component of this project is a sophisticated system for connecting users based on shared interests. Typically, teams achieve this by allowing a user to enter classes/areas of study they wish to study, select whether they are searching for students or tutors, and set preferences for study methods. Then, the system can recommend a list of other users that the user can become buddies with or schedule meetups with. You can take a different approach to this one and even add components like suggesting study playlists or locations, but your recommendations engine needs to be capable of finding relevant students to create a meetup with and support a large user base.
* Notifications: Your system should provide notifications to its users and alert them when they have unread notifications. At a minimum, users should receive notifications when other users become buddies with them, when users join one of their meetups, and when a meetup that they have joined is about to start.
* In the service layer, be able to:
  + Suggest other users to become buddies with
  + Suggest other students and tutors to add to a meetup/create a meetup with
  + Schedule, edit, and cancel meetups
  + Send notifications
  + Submit ratings/reviews for tutors
  + Update existing and add new unit tests in the Java layer
* In the frontend, an individual should be able to:
  + Review recommended students and tutors
  + Become buddies with and join/schedule meetups with recommended students and tutors
  + Easily allow one’s study buddies to join a meetup they own
  + Get notifications about study buddy and meetup statuses
  + Edit or cancel a meetup
  + Rate the performance of a tutor after a tutoring session

***Demo***

* Demonstrate suggestions for students and tutors to connect with, related to study preferences and subject
* Demonstrate that the system sends a reminder notification before a meetup that the user has joined
* Demonstrate rating a tutor after a meetup

**Milestone 4** (due at time of final presentation)

* Security features
  + Input validation
  + User authentication (do unauthenticated users receive a different view of the site? etc.)
  + User authorization (for example, a validated user can only edit their own classes/areas of study, but can see others’ in a read-only format)
  + Password encryption
* Final UI/UX design
* Bonus features (see below for some examples)

***Demo***

* *This demo will be a full-product demo. Treat it as if nobody has seen your product before and you are showing off every part of it. However, also highlight the following Milestone 4 features:*
* Demonstrate security
* Demonstrate bonus features
* The product and UI should be in a complete state

**Example Bonus Features**

* Socializing/gamifying the app
* In-app tutorial
* OAuth sign-in (sign in with Google/Facebook/X/etc.)
  + There is an approval process with some platforms for OAuth integration, if you choose to do this then make sure to start early
* Social media integrations
* Calendar integrations
* Planning/notetaking app integrations
* Mobile/push (out-of-app) notifications
* Playlist generator/recommendations
* Location recommendations (e.g., places with public Wi-Fi, student discounts, etc.)
* Password reset
* Mobile version of the app
* Ensure your application is compliant with the [OWASP Top 10](https://owasp.org/Top10/)

### 3.1.3 Schedule

This schedule is subject to change as the semester progresses. Check Canvas and Slack announcements regularly to stay informed if it does.

|  |  |  |
| --- | --- | --- |
| **Week of** | **Week #** | **Credera Assignment** |
| 1/14 | 0 | *Semester starts* |
| 1/21 | 1 | **Project Kickoff**   * Class introduction * Project kick-off presentation * Role assignments and requirements clarifications |
| 1/28 | 2 | * First scrum meeting * Continued requirements clarification for Milestone 1 * Begin Milestone 1 development * **Deliverable 0 (Project Setup) due** |
| 2/4 | 3 | * Milestone 1 development * **Deliverable 1 (ERD) due** |
| 2/11 | 4 | **Milestone 1 Due + Demo**   * Continued requirements clarification for Milestone 2 * Begin Milestone 2 development |
| 2/18 | 5 | * Milestone 2 development * **Deliverable 2 (Account Creation & Login) due** |
| 2/25 | 6 | * Milestone 2 development * **Deliverable 3 (E2E Service Demo) due** |
| 3/3 | 7 | *Spring break –* ***no scrums*** |
| 3/10 | 8 | **Milestone 2 Due + Demo**   * Continued requirements clarification for Milestone 3 * Begin Milestone 3 development |
| 3/17 | 9 | * Milestone 3 development |
| 3/24 | 10 | * Milestone 3 development * **Deliverable 4 (Recommendations) due** |
| 3/31 | 11 | * Milestone 3 development * **Deliverable 5 (Notifications) due** |
| 4/7 | 12 | **Milestone 3 Due + Demo**   * Continued Requirements clarification for Milestone 4 * Begin Milestone 4 development |
| 4/14 | 13 | * Milestone 4 development * **Deliverable 6 (UI Hardening) due** |
| 4/21 | 14 | * Milestone 4 development |
| 4/28 | 15 | **Milestone 4 Due + Final Presentations** |

## 3.2 Scrums

Scrums will occur weekly. Each team will have a 20-minute time slot determined by the students in the evening. At least two Credera mentors will be present for every scrum, but an individual mentor will not be assigned to a specific team; rather, all mentors will be available for all teams. Every team member is responsible to stating what they accomplished last week, what they plan to accomplish the following week, and any roadblocks they have. The remainder of the time will be used on some or all of the following:

* Each team should have questions ready (i.e., requirement questions, technical questions, etc.) to discuss during the meeting for further understanding
  + Feel free to send the mentors questions ahead of time so that we can better utilize scrum time (and possibly get a quicker response)
  + Any code needed for questions should be available in source control so the mentors can look at it directly and run it locally
* Review weekly deliverable and discuss the upcoming deliverable
* Provide a brief demo of new functionality

Questions submitted asynchronously will be answered in a timely manner, but please note that project mentors will most likely be confined to business hours, meaning questions may not be answered until 24 hours layer. Please plan questions outside of scrum meetings accordingly.

## 3.3 Group Member Roles

The minimum prescribed roles in each group are as follows:

* The **Project Manager** will be responsible for coordinating the activities of the group and to be the liaison with the professor to resolve issues affecting the group from time to time. The project manager should be: professional and responsible, a good organizer, and an effective communicator. Their functions include:
  + Coordinating group activities
  + Overseeing and managing activities
  + Allocating roles, activities, responsibilities, coordinating and managing documentation activities
  + Also refer to: [What a team leader does/does not](https://cs.baylor.edu/~song/courses/3372-13f/project/3372-TeamLeaders.htm)
* The **Requirements Engineer** leads the requirements classification and organization efforts
* The **Design Engineer** leads the UI and technical design efforts
* The **Quality Assurance Engineer** oversees test case design, validation of requirements, consistent design, etc.
* The **Project Librarian** keeps all meeting logs, makes all design artifacts available for the team, and ensures that the team submits all artifacts on time

The initial role assignment will be decided within the group and can change during the semester, if needed, after consulting with the professor/mentors. Each student is also required to summarize the activities they carry out and give the times spent on each. Regardless of group member role, each team member is expected to make contributions to the codebase.

It is important to recognize these roles as “effort leaders” rather than “designated contributors”. For example, the project manager shouldn’t be the only one leading and encouraging the team, the requirements engineer shouldn’t be the only one keeping track of requirements, the design engineer shouldn’t be the only one working on the frontend, the quality assurance engineer shouldn’t be the only one writing unit tests, and the project librarian shouldn’t be the only one responsible for project artifacts. This is a team project, and thus all team members are expected to both contribute equally to the final product but also contribute to each individual component of the project.

## 3.4 Product Backlog

In addition to technical deliverables, this product backlog includes a list of non-technical artifacts that must be completed this semester. The following list must be included in each team’s final deliverable.

|  |
| --- |
| **Product Backlog** |
| **Build activity diagrams** - Initial use cases outlining, in general, core user interactions with the new feature |
| **Build use case documents** - Brief listing of requirements generated from use case diagrams |
| **User interface mockups (if applicable)** - UI mocks based on the use cases to help showcase how the feature will be implemented |
| **Create requirements class diagrams** - Class diagrams be built off the requirements & use case document. Students need to keep in mind that requirements class diagrams are meant to show how the domain is related, not how it is implemented, and the ER diagrams show the actual database tables; correlating them may take some work |
| **Develop architecture diagram** - An initial architecture diagram that outlines major elements of the feature and how they relate |
| **Build sequence diagrams** - Initial sequence diagrams showcasing major flows in design class diagram |
| **Document project plan** – The project plan outlines deliverables. The students should indicate how much of the feature they can complete and what parts of the feature will be accomplished by each deliverable in detail |
| **Create test plan** – The test plan details how features are tested in each milestone. Testing should include both functional and unit testing, and a milestone should be considered complete when the tests are satisfied |
| **Document design patterns used** – Documentation showcasing which patterns you have selected and how you are implementing |
| **Create User Manual –** Guide should detail operations and maintenance of your project |
| **Final project documentation** - Document organizing all prior deliverables |
| **Final project source code** |
| **Final presentation** - PowerPoint presentation or otherwise that demos your project, you should talk about the software development process and any risks/setbacks/difficulties encountered along the way |

# 4. Technology Stack Overview

The following is a brief synopsis of the technologies that are pre-installed in the template project. See the README files in the template project for more information about each of these technologies.

## 4.1 Java Spring (Boot)

Spring/Spring Boot is an open-source framework and inversion of control container for Java that was created to address the complexity of developing enterprise applications. Spring has layered architecture that allows developers to leverage certain components while not utilizing others that they may not care about. Spring’s biggest features include:

1. Transaction management – Spring allows for pluggable transaction managers to deal with transactions
2. Inversion of control container – Spring allows for dependency injection, which helps with the configuration and management of Java objects
3. Data access – Easy integration with Hibernate and JDBC
4. Model-View-Controller (MVC) – Spring provides a framework for extending and customizing web applications
5. Messaging – Spring is able to leverage existing technologies, such as Java Messaging Service (JMS) for sending messages

## 4.2 React

React is a JavaScript-based open-source web application framework for the application’s front-end that allows you to create dynamic views for your single-page application. It is a widely used platform and is thus many developers go-to when creating a web application. React allows you to declare re-usable, composable components to quickly and easily create interactive webpages written in a way that looks almost like normal HTML.

A popular React UI component/styling library called Material UI is included and pre-configured in the template project but is not required to be used.

## 4.3 REST

Representational state transfer (REST) is an architecture approach (not a tool or language) that consists of a coordinated set of constraints that applies to components, connectors, and data elements. The REST architecture has been applied to developing web services and is commonly used as an alternative to SOAP. REST allows for data to be quickly obtained via point-to-point communications with clients. For this project, REST will be used to obtain data from the service layer.

## 4.4 Google Cloud Platform (GCP)

GCP is Google’s cloud platform. You will use GCP to host your application so that it will be publicly accessible on the web. You will be able to control how the application deploys, runs, and connects with your database and any other resources.

## 4.5 MySQL

MySQL is a popular open-source relational database management system. It is reliable and easy to use and provides official Docker images maintained by the MySQL team for Linux platforms. MySQL is a leading choice for SQL databases and is used in a wide variety of applications.

# 5. Credera Mentors

A person and person with text

Description automatically generatedA close-up of a person's face

Description automatically generated