Software Requirements Specification for Software Engineering: SOCAlgoTestPlatform

Team 1, BANDwidth
Declan Young
Ben Dubois
Nathan Uy
Aidan Mariglia

February 13, 2025

Contents

1 Purpose of the Project **6**
- 1.1 User Business . . . 6
- 1.2 Goals of the Project . . . 6

2 Stakeholders **6**
- 2.1 Client . . . 6
- 2.2 Customer . . . 7
- 2.3 Other Stakeholders . . . 7
- 2.4 Hands-On Users of the Project . . . 8
- 2.5 Personas . . . 8
- 2.6 Priorities Assigned to Users . . . 8
- 2.7 User Participation . . . 9
- 2.8 Maintenance Users and Service Technicians . . . 9

3 Mandated Constraints **9**
- 3.1 Solution Constraints . . . 9
- 3.2 Implementation Environment of the Current System . . . 10
- 3.3 Partner or Collaborative Applications . . . 10
- 3.4 Off-the-Shelf Software . . . 10
- 3.5 Anticipated Workplace Environment . . . 10
- 3.6 Schedule Constraints . . . 11
- 3.7 Budget Constraints . . . 11
- 3.8 Enterprise Constraints . . . 11

4 Naming Conventions and Terminology **11**
- 4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project . . . 11

5 Relevant Facts And Assumptions **12**
- 5.1 Relevant Facts . . . 12
- 5.2 Business Rules . . . 12
- 5.3 Assumptions . . . 12

6 The Scope of the Work **13**
- 6.1 The Current Situation . . . 13
- 6.2 The Context of the Work . . . 14
- 6.3 Work Partitioning . . . 15

6.4 Specifying a Business Use Case (BUC) . . . 15

7 Business Data Model and Data Dictionary 17
7.1 Business Data Model . . . 17
7.2 Data Dictionary . . . 18

8 The Scope of the Product 19
8.1 Product Boundary . . . 19
8.2 Product Use Case Table . . . 20
8.3 Individual Product Use Cases (PUC's) . . . 20

9 Functional Requirements 21
9.1 Functional Requirements . . . 21

10 Look and Feel Requirements 24
10.1 Appearance Requirements . . . 24
10.2 ~~Style Requirements~~ . . . 24

11 Usability and Humanity Requirements 24
11.1 Ease of Use Requirements . . . 24
11.2 Personalization and Internationalization Requirements . . . 24
11.3 Learning Requirements . . . 25
11.4 ~~Understandability and Politeness Requirements~~ . . . 25
11.5 ~~Accessibility Requirements~~ . . . 26

12 Performance Requirements 26
12.1 Speed and Latency Requirements . . . 26
12.2 Safety-Critical Requirements . . . 27
12.3 ~~Precision or Accuracy Requirements~~ . . . 27
12.4 Robustness or Fault-Tolerance Requirements . . . 27
12.5 Capacity Requirements . . . 27
12.6 Scalability or Extensibility Requirements . . . 28
12.7 ~~Longevity Requirements~~ . . . 28

13 Operational and Environmental Requirements 28
13.1 Expected Physical Environment . . . 28
13.2 ~~Wider Environment Requirements~~ . . . 28
13.3 ~~Productization Requirements~~ . . . 29
13.4 Release Requirements . . . 29

14 Maintainability and Support Requirements **29**
14.1 ~~Maintenance Requirements~~ . . . 29
14.2 ~~Supportability Requirements~~ . . . 29
14.3 Adaptability Requirements . . . 29

15 Security Requirements **30**
15.1 Access Requirements . . . 30
15.2 Integrity Requirements . . . 31
15.3 ~~Privacy Requirements~~ . . . 31
15.4 Audit Requirements . . . 31
15.5 ~~Immunity Requirements~~ . . . 32

16 Cultural Requirements **32**
16.1 ~~Cultural Requirements~~ . . . 32

17 Compliance Requirements **32**
17.1 ~~Legal Requirements~~ . . . 32
17.2 ~~Standards Compliance Requirements~~ . . . 32

18 Open Issues **33**

19 Off-the-Shelf Solutions **33**
19.1 Ready-Made Products . . . 33
19.2 Reusable Components . . . 33
19.3 Products That Can Be Copied . . . 33

20 New Problems **33**
20.1 Effects on the Current Environment . . . 33
20.2 Effects on the Installed Systems . . . 33
20.3 Potential User Problems . . . 34
20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product . . . 34
20.5 Follow-Up Problems . . . 34

21 Tasks **34**
21.1 Project Planning . . . 34
21.2 Planning of the Development Phases . . . 35

22 Migration to the New Product **36**
22.1 Requirements for Migration to the New Product 36
22.2 Data That Has to be Modified or Translated for the New System 36

23 Costs **36**

24 User Documentation and Training **37**
24.1 User Documentation Requirements 37
24.2 Training Requirements 37

25 Waiting Room **37**

26 Ideas for Solution **37**

Revision History

Date	Version	Notes
2024-10-11	0.0	Revision 0
2024-10-19	0.1	Adding requirement identifiers
2024-10-19	0.2	Fix directories
2025-01-04	0.3	Added specific deliverables and deadlines to section 21.1
2025-01-04	0.4	Fixed non-verifiable requirements in section 11.4
2025-02-13	0.5	Prune SRS and add rationales to requirements

1 Purpose of the Project

1.1 User Business

Battery state of charge (SOC) estimation requires specialized algorithms. Standardized testing is needed to identify which SOC estimation approaches proposed yearly are the best. The current online SOC estimation algorithm testing tool is based on Matlab and receives submissions through a Google form, then the algorithms are tested in serial on a McMaster server. This is not scalable, as tests usually take an hour or more to complete. In addition, the software regularly crashes due to unhandled errors from the submitted algorithms. The final product will be scaled up to have several hundred active users and be used with multiple algorithm submissions at a time.

1.2 Goals of the Project

The final product can test multiple algorithm submissions in parallel - since each algorithm takes approximately an hour to complete, running them in parallel will help reduce the overall time required to run all algorithms.

The final product generates reports and compares algorithm performance - the reports will summarize the performance of the submitted algorithm and compare it with other submitted algorithms, helping users understand and visualize their results.

The final product can handle any error encountered throughout the algorithm's runtime - this will minimize the software's crashes and downtime and overall, it will increase the reliability of the software. It will make debugging much easier and as a result, improve the user experience.

The final product is secure and prevents malware attacks - secure software is necessary to protect users' sensitive data as well as keep the software's integrity and availability.

2 Stakeholders

2.1 Client

Dr. Phil Kollmeyer

Dr. Kollmeyer is the client for this project. He is an electrical engineering professor who specializes in battery engineering. An org chart is omitted as

there is no existing organization commissioning this product to be built, in this case, we can consider Dr. Kollmeyer as the entire organization. He is responsible for decisions such as the priority of features to be included in the final system and resource allocation; particularly hosting resources.

2.2 Customer

- Dr. Philip Kollmeyer
  - As this is in-house development Dr. Kollmeyer is both the Client and a customer. He is responsible for all of the decisions outlined in his client description.
- Battery Engineering professors
  - Battery Engineering professors may provide feedback but will not be responsible for any of the development decisions of the project
- Battery Engineering PhD/Masters students
  - Battery Engineering PhD/Masters students may provide feedback but will not be responsible for any of the development decisions of the project

2.3 Other Stakeholders

- Dr. Phil Kollmeyer
- Atjen von Liebenstein
- Professors and PhD/Master students at educational institutions
- Battery engineering research students, Students in battery engineering classes.
- Team 1 Developers, Dr. Smith, Yiding Li

2.4 Hands-On Users of the Project

- Battery engineering research students, Students in battery engineering classes.
  - These students are knowledgeable about the subject matter. They have experience using a similar technology and are technology literate. They are at least 18+ years old.

2.5 Personas

- Bill
  - Bill is a 3rd year Electrical Engineering student in Canada who is taking a battery engineering course. Bill is familiar with programming from an academic point of view, but doesn't have a lot of time to learn a new tool alongside learning about batteries for his course. Bill is a frequent internet user, and is comfortable using a computer and websites on the modern internet.
- Steve
  - Steve is an Electrical Engineering professor at McMaster University, who specializes in battery engineering. Steve wants to provide a fun competition to his students, who are learning about battery SOC algorithms. However, as he is a professor, Steve is very busy and does not have time to do manual work, administering the competition.

2.6 Priorities Assigned to Users

- Key Users:
  - Dr. Phil Kollmeyer
  - Atjen von Liebenstein
  - Battery engineering research students
- Secondary Users:
  - Students in battery engineering classes.

- Unimportant Users:
  - Software Testers/Maintainers

2.7 User Participation

The estimated user participation time from key users is 1 hour weekly. This is to discuss any use cases for the project and the functionalities of the software to be implemented. Knowledge of expected output for certain inputs is also needed for testing purposes.

2.8 Maintenance Users and Service Technicians

Dr. Phil or Atjen von Liebenstein will be responsible for maintaining the product

3 Mandated Constraints

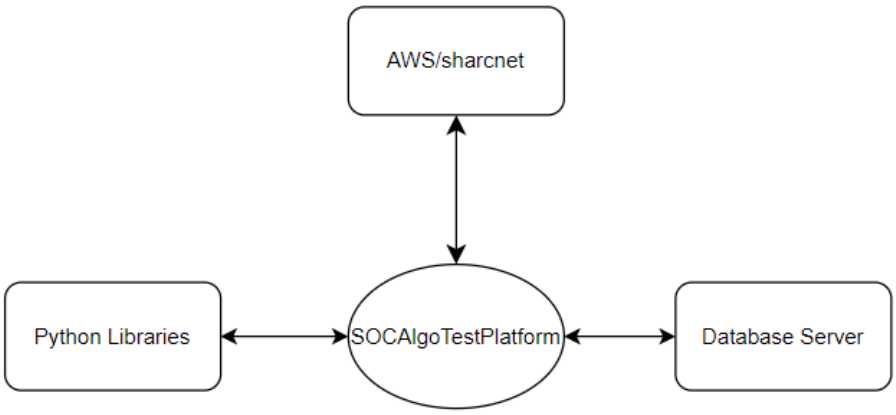
3.1 Solution Constraints

- 1.
  - Description: The product shall accept user code written in Matlab
  - Rationale: Matlab is the language of choice for developing battery SOC algorithms
  - Fit Criterion: Valid Matlab code should be accepted and executed by the system
- 2.
  - Description: The product shall be useable across multiple modern browsers (chrome, firefox, safari) on multiple modern operating systems (windows, Mac, Linux)
  - Rationale: The product will be accessed by various users via the web, and the user computers will not be standard
  - Fit Criterion: The product is tested and is operable on all combinations of the above-listed operating systems and browsers

3.2 Implementation Environment of the Current System

The current system exists as a Matlab script running on a McMaster server, listening for submissions via a Google form. Responses are sent to the user via email.

3.3 Partner or Collaborative Applications



3.4 Off-the-Shelf Software

Partner Application	Usage
Matlab	compile/execute user-provided algorithms
Database Server	Persist necessary data
Various python libraries	Open-source libraries will be used for providing standard functionality (i.e. an SQL driver, web server framework)

3.5 Anticipated Workplace Environment

As the product is a web application it will be used in any environment that has an internet connection.

3.6 Schedule Constraints

- Problem Statement, POC Plan, Development Plan - Sep 16
- Requirements Document Revision 0 - Oct 11
- Hazard Analysis 0 -Oct 23
- V&V Plan Revision 0 - Nov 1
- POC Demo - Nov 11-22
- Design Document Revision 0 - Jan 15
- Revision 0 Demo - Feb 3
- V&V Report Revision 0 - Mar 7
- Final Demo Rev 1 - March 24
- EXPO Demo - April
- Final Doc - Apr 2

3.7 Budget Constraints

The budget to create the project is $750 coming from the team. No hard budget exists to run the system, but a strong emphasis is placed on having the lowest possible operational costs for the running system.

3.8 Enterprise Constraints

N/A. (As this is an academic project, there is no "Enterprise" sponsoring the project providing additional constraints.)

4 Naming Conventions and Terminology

4.1 Glossary of All Terms, Including Acronyms, Used by Stakeholders involved in the Project

- SOC - State of charge

- AWS - amazon web service
- Leaderboard - a table displaying the results of all algorithms submitted by all users with categories that can be filtered on/sorted.
- Algorithm - algorithms submitted by users to be tested for efficiency and performance.
- Performance - quantitative measurement of the algorithm's performance in different categories

5 Relevant Facts And Assumptions

5.1 Relevant Facts

- Running tests on an algorithm typically takes about an hour to complete.
- The existing application is only capable of executing tests in serial, no support exists for parallel test execution.
- The current application supports executing Matlab algorithms only.
- Responses from the current application are handled via email.

5.2 Business Rules

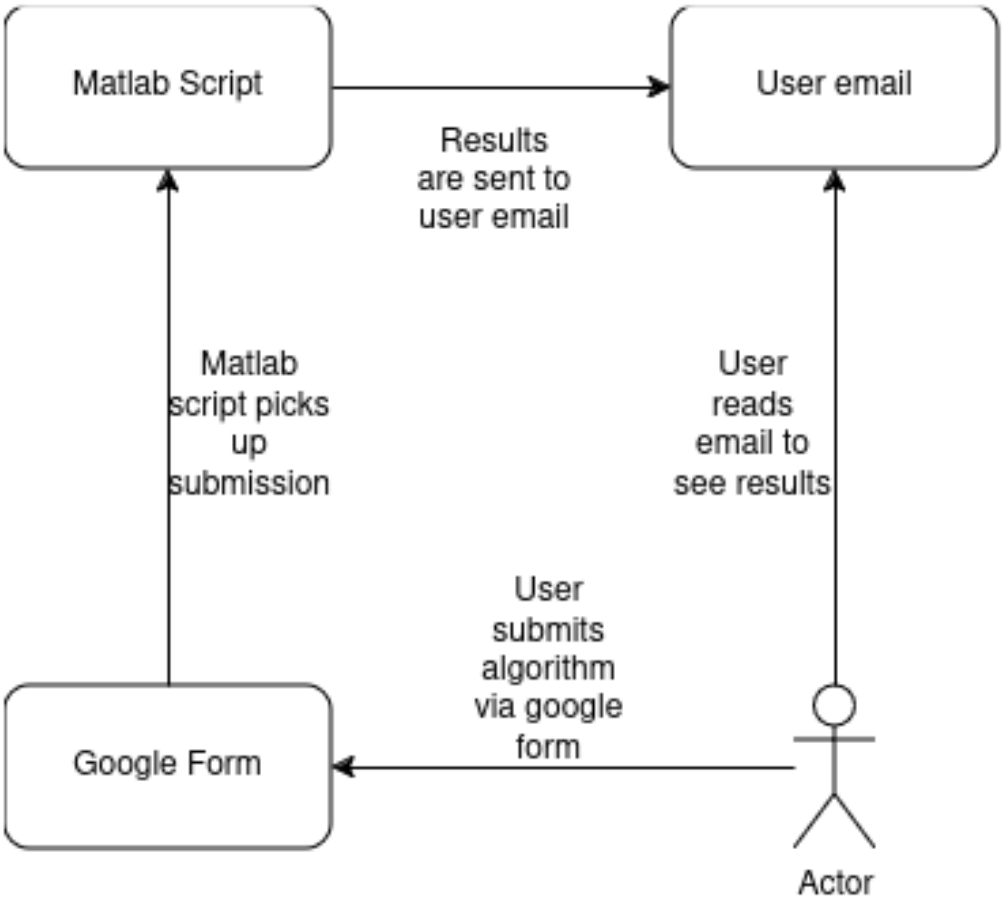
- The system must not have excessive hosting costs

5.3 Assumptions

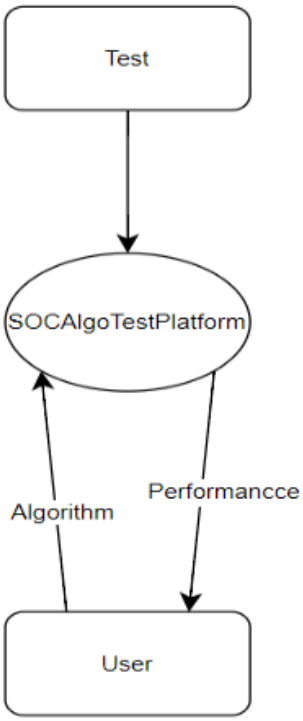
- Applicable Matlab licensing is available and affordable.
- Required Cloud resources are available and affordable.

6 The Scope of the Work

6.1 The Current Situation



6.2 The Context of the Work



6.3 Work Partitioning

Business Event Number	Business Event Name	Input and Output	Summary of BUC
1	User registration/Login	User email and password (in)	Process user registration
2	Submission of algorithm	File with an algorithm (in)	Process algorithm submission and initiate testing
3	Completion of algorithm testing	Results of algorithm performance (out)	Generate and store the results of algorithm performance
4	Leaderboard Update	Results of algorithm performance (in) Update leaderboard (out)	Update the leaderboard to allow performance comparison

6.4 Specifying a Business Use Case (BUC)

Scenario 1: Process user registration/Login
Actor: user
Precondition: N/A
Trigger: user clicks sign up
Steps:
User accesses the registration page and submit his email, name, institution and password System does some validation and create the account
User can now login using his/her credentials
Postcondition: New user is added to DB

Scenario 2: Submission of algorithm
Actor: user
Precondition: User is logged into his account
Trigger: user submits an algorithm
Steps:
User accesses the algorithm submission page and upload his/her algorithm file.
System validates submission and display success/failure to upload

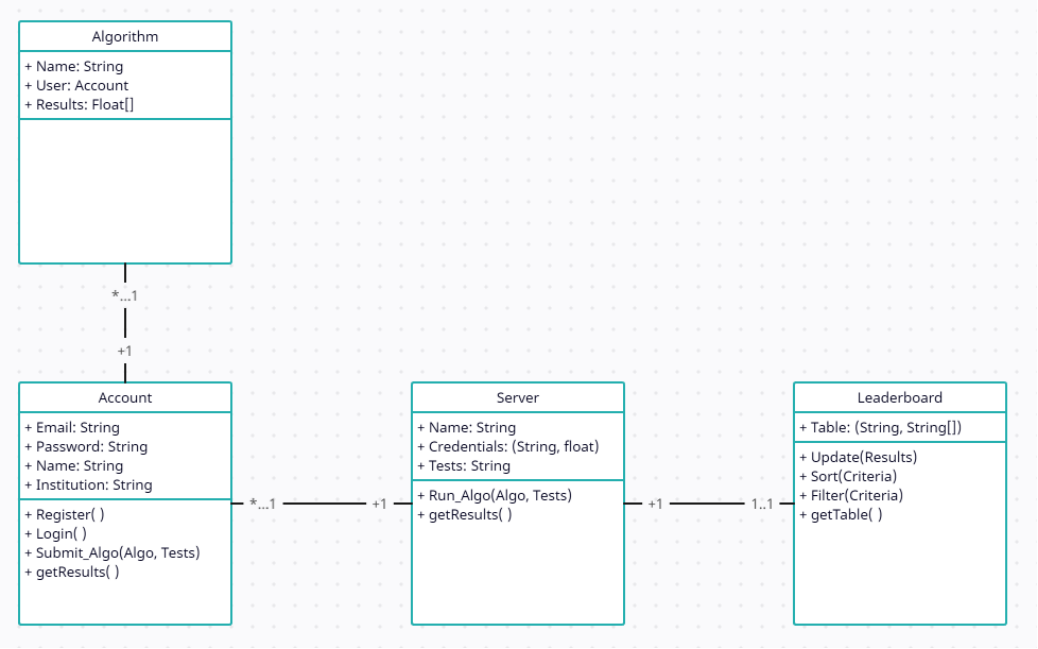
System displays the status of the test
Postcondition: Algorithm is being tested

Scenario 3: Completion of algorithm testing
Actor: system
Precondition: An algorithm was submitted and is being tested
Trigger: algorithm testing is complete
Steps:
System calculates and reports the performance of the algorithm
System notifies the user if the testing is done/if any errors was encountered
Postcondition: performance results are stored in db that can be accessed by user

Scenario 4: Leaderboard update
Actor: system
Precondition: An algorithm was tested successfully
Trigger: algorithm testing is complete
Steps:
System receives the new algorithm results and updates the leaderboard table in the database.
Postcondition: Leaderboard is up to date

7 Business Data Model and Data Dictionary

7.1 Business Data Model

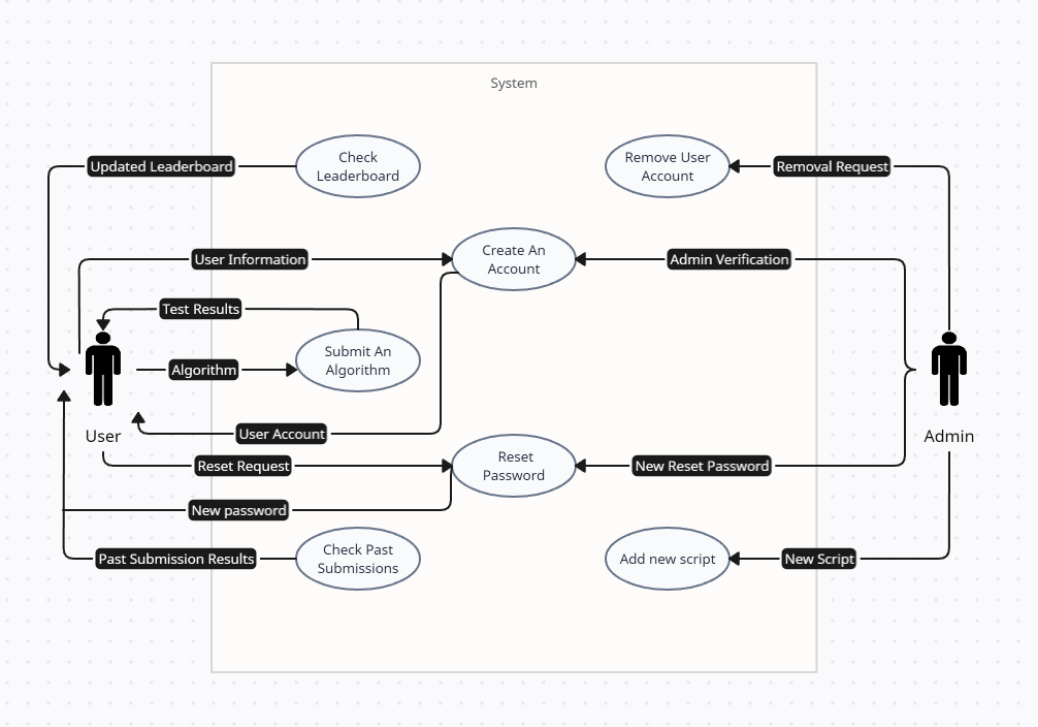


7.2 Data Dictionary

Name	Content	Type
Account	Email + Name + Institution + Password	Class
Leaderboard	Compilation of Performances	Class
Algorithm	Results and files	Class
Email	“”@””.””	Class
Name	String	Attribute/Element
Institution	String	Attribute/Element
Password	String	Attribute/Element
Algorithm Name	String	Attribute/Element
User	Account	Attribute/Element
Results	Float[]	Attribute/Element
Server Name	String	Attribute/Element
Credentials	(String, Float)	Attribute/Element
Tests	String (Name of the script containing tests)	Attribute/Element

8 The Scope of the Product

8.1 Product Boundary



8.2 Product Use Case Table

PUC No	PUC Name	Actor/s	Inputs & Outputs
1	Submit An Algorithm	User	Algorithm (In), Test results (Out)
2	Check Leaderboard	User	Updated Leaderboard (Out)
3	Check Past Submissions	User	Past submissions (Out)
4	Create An Account	User, Administrator	User information (In), Admin Verification (In), User Account (Out)
5	Reset Password	User, Administrator	Reset Request (In), Reset Password (In), New password (Out)
6	Remove User Account	Administrator	Removal Request (In)
7	Add New Script	Administrator	New script (In)

8.3 Individual Product Use Cases (PUC's)

1. A user creates a battery testing algorithm and wants it to be tested to see how efficient it is. They submit the algorithm to the system and after the system has completed the tests, it outputs the results back to the user.

2. A user navigates to the leaderboard screen to check what the current best algorithm for testing batteries is and which user created it. The system outputs the updated leaderboard results to the user.

3. A user navigates to their account page and wants to view the results of all algorithms they have submitted previously. The system will display these previous submissions to the user.

4. A user navigates to the main page of the website and inputs their information. The system inputs this system and requests an administrator reviews the information for correctness. Once the administrator has verified the information, the system will create the account for the user.

5. A user forgets their password and sends a request to the system to reset it. An administrator creates a new temporary password for the user

and sets the users current password to this temporary one. The system notifies the user of this new password.

6. An administrator wants to delete a user account for violating terms of service. They submit a request to the system to delete this user account and the system ensures it is deleted.

7. An administrator discovers new tests that they want to include within the script that the system runs against user algorithms. The administrator uploads this new script within the system and the system now runs the new script.

9 Functional Requirements

9.1 Functional Requirements

FR-1

- The system shall allow users to submit algorithms to be tested.
  Fit criterion: Users should be able to upload an algorithm to the system and receive confirmation of their upload.
  Rationale: Ensures core functionality by allowing users to submit algorithms for testing. Confirmation provides feedback, reducing uncertainty and preventing duplicate uploads.

FR-2

- The system shall report the results of submitted algorithms when their testing is complete.
  Fit criterion: Users should be able to view the result of their algorithm somewhere in the interface of the system after it has been completed.
  Depends on: FR-1
  Rationale: Provides users with feedback on their algorithm's performance.

FR-3

- The system shall allow for multiple submissions from users to run in parallel.
  Fit criterion: Users should be able to submit more than one algorithm

for testing and see that the execution of all submissions is underway.
Depends on: FR-1
Rationale: Enables efficient use of system resources and improves user experience by allowing multiple tests to run simultaneously.

FR-4

- The system shall allow users to make a new account.
  Fit criterion: Users should be able to provide a username and password and in return gain access to a new account corresponding to those values.
  Rationale: Enables user authentication and personalization, allowing access to submission history and results.

FR-5

- The system shall have a user login page.
  Fit criterion: Users should be able to navigate to a page in the system where they can provide a username and password to authenticate under an account.
  Rationale: Ensures secure access to user accounts, protecting submissions and results.

FR-6

- The system shall segregate data based on the logged users.
  Fit criterion: Users should not be able to see the private data of users and vice versa.
  Depends on: FR-4 and FR-5
  Rationale: Protects user privacy and ensures data security by restricting access to personal submissions and results.

FR-7

- The system shall save algorithm results in individual user accounts.
  Fit criterion: Users should be able to view their previous submissions anytime in the future.
  Depends on: FR-4
  Rationale: Ensures accessibility to past submissions, allowing users to track progress and review past results.

FR-8

- The system shall have a leaderboard feature that is accessible to all users.
  Fit criterion: Users should be able to view a list of all submissions from other users.
  Rationale: Encourages competition and engagement by allowing users to compare performance across different users' submissions.

FR-9

- The system shall have categorization in its leaderboard based on certain criteria.
  Fit criterion: Users should be able to select the type of algorithm they are interested in and only view results of that type.
  Depends on: FR-8
  Rationale: Enhances user experience making it easier to find results based on specific algorithm categories.

FR-10

- The system shall allow users to sort through the leaderboards based on criteria.
  Fit criterion: Users should be able to view results in order of lowest run time to highest or other criteria.
  Depends on: FR-8
  Rationale: Provides flexibility in viewing results, enabling users to focus on specific performance metrics.

FR-11

- The system shall display the progress of running algorithms to users.
  Fit criterion: Users should be able to view if a submission is accepted, pending, executed, or finished.
  Rationale: Keeps users informed about the status of their submissions, reducing uncertainty and enhancing user experience.

FR-12

- The system shall notify users in case of an exception in their algorithm.
  Fit criterion: Users should receive a notification if their algorithm fails to execute successfully.
  Rationale: Ensures transparency by informing users of issues with their submissions, enabling them to make necessary corrections.

10 Look and Feel Requirements

10.1 Appearance Requirements

LFR-1

- The system's appearance should look similar to a website called Kaggle.
  Fit criterion: Users should recognize similarities between the system and Kaggle.
  Rationale: Design preference of primary stakeholder.

10.2 ~~Style Requirements~~

~~**SR-1**~~

- ~~The system should use the same styles i.e. colour, background colour, user-interface structure, etc. as the Kaggle website.~~
  ~~Fit criterion: Users should see the same styles as the Kaggle website when using the system.~~
  ~~Rationale: Design preference of primary stakeholder.~~

11 Usability and Humanity Requirements

11.1 Ease of Use Requirements

EUR-1

- All tasks that can be completed with the system should have a similar workflow.
  Fit criterion: Users should be able to follow the same workflow when viewing a submission in progress or one that is already completed.
  Rationale: Promotes consistency and simplicity in the user experience, making the system intuitive and easy to navigate.

11.2 Personalization and Internationalization Requirements

N/A - No personalization or internationalization was mentioned by the client.

11.3 Learning Requirements

LR-1

- The system shall have detailed documentation for all core functionality.
  Fit criterion: Users should be able to easily find documentation which walks them through the steps involved in submitting an algorithm and viewing the results.
  Rationale: Ensures users can easily understand and use the system, reducing confusion and improving overall usability.

LR-2

- The system shall display hints for new/unused features for a specific user.
  Fit criterion: The user should be able to see a hint indicating how to use a new feature.
  Rationale: Enhances user experience by guiding users through new features, promoting exploration and efficient use of the platform.

11.4 ~~Understandability and Politeness Requirements~~

~~**UPR-1**~~

- ~~The system shall use universally used symbols when appropriate to ease understanding.~~
  ~~Fit criterion: At least 90% of users should be able to correctly identify the meaning of all symbols without explanation.~~

~~**UPR-2**~~

- ~~The system shall use an intuitive user interface and use consistent terminology.~~
  ~~Fit criterion: The terminology used on each page should be consistent with each other and at least 80% of users should be able to navigate through each stage of the system without external assistance.~~

~~**UPR-3**~~

- ~~The system shall provide visual feedback to indicate whether certain actions were successful or not.~~

~~UPR-4~~

- ~~The system shall provide walkthroughs for any complex use cases.~~

11.5 ~~Accessibility Requirements~~

~~AR-1~~

- ~~The system shall have a colour-blind mode~~

12 Performance Requirements

12.1 Speed and Latency Requirements

SLR-1

- The product shall display the results of algorithms in less than 2 seconds after it is requested
  Fit Criteria: Users should see a result after selecting an algorithm to view in under 2 seconds
  Rationale: This response time is on the slower side of what is typically observed on the modern web. It should be achievable while leaving room to perform the action faster if possible.

SLR-2

- The product shall display the Leaderboards table in less than 3 seconds after it is requested
  Fit Criteria: Users should see the leaderboards after selecting that view in under 3 seconds
  Rationale: This response time is on the slower side of what is typically observed on the modern web. It should be achievable while leaving room to perform the action faster if possible.

SLR-3

- Sorting or filtering the leaderboard should not take more than 3 seconds
  Fit Criteria: Users should see leaderboard content change in under 3 seconds after making a change to filtering criteria

Rationale: This response time is on the slower side of what is typically observed on the modern web. It should be achievable while leaving room to perform the action faster if possible.

12.2 Safety-Critical Requirements

N/A - we are building a web application without any interaction with the physical world

12.3 ~~Precision or Accuracy Requirements~~

~~**PAR-1**~~

- ~~Events in the system will be tracked using standard UTC timestamp~~

12.4 Robustness or Fault-Tolerance Requirements

RR-1

- The system should be able to handle any types of errors during runtime.
  Fit criterion: The system should catch and display error messages for any runtime exceptions, allowing users to understand and address issues without system failure.
  Rationale: Ensures system stability and reliability by gracefully managing unexpected issues, preventing crashes and maintaining user trust.

RR-2

- Errors arising from one users interactions with the system should be isolated from other users interactions.
  Fit criterion: Errors caused by one user's actions should not impact the functionality or data of other users, maintaining system integrity.
  Rationale: Prevents one user's errors from affecting others, ensuring a stable and reliable experience for all users.

12.5 Capacity Requirements

CR-1

- The system shall support concurrent requests from a few hundred users
  Fit criterion: Users should not see a difference in performance when there is one current user or one hundred current users.
  Rationale: Typical university lecture sizes at McMaster for engineering range from 100-300 people. Supporting this user capacity is a good starting point and would support live in class useage of the platform.

12.6 Scalability or Extensibility Requirements

SR-1

- The system shall automatically scale its execution back-end corresponding to the number of user requests.
  Fit criterion: The system should dynamically allocate resources to scale up during high traffic and scale down during low traffic, maintaining consistent performance without manual intervention.
  Rationale: Allows the system to efficiently handle different loads, ensuring optimal performance regardless of the number of simultaneous user submissions.

12.7 ~~Longevity Requirements~~

~~**LR-1**~~

- ~~The system shall be free of memory leaks and uncapped caches so that restarts will not be required~~

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

N/A - The product shall work in any environment with an internet connection.

13.2 ~~Wider Environment Requirements~~

~~**WER-1**~~

- ~~The system shall adhere to the best practices in data privacy to protect user information~~
  ~~Fit criterion: The system shall have secure login methods and keep any data from being accessed externally.~~

13.3 ~~Productization Requirements~~

~~**PR-1**~~

- ~~The product shall be deployable on AWS and accessible to the users through the web.~~

13.4 Release Requirements

N/A - there will not be regular releases for this project, and maintenance will be done by an external team.

14 Maintainability and Support Requirements

14.1 ~~Maintenance Requirements~~

~~**MR-1**~~

- ~~If a likely change is made to the finished software, it will take at most 10% of the original development time, assuming the same development resources are available (Example from a lecture that Dr. Smith said we can use if we do not think of anything else)~~

14.2 ~~Supportability Requirements~~

~~**SUR-1** The product must be self-supporting with all the necessary information provided in the user documentation and/or user guides.~~

14.3 Adaptability Requirements

ADR-1

- The product shall work under all major operating systems such as macOS, Linux and Windows.
  Fit criterion: Users should be able to access the system and its functionality on any of the above-listed operating systems.
  Rationale: Ensures broad accessibility by supporting all major operating systems.

ADR-2

- The product shall be compatible with all major browsers like Chrome, Safari, Microsoft Edge, Opera, and Firefox.
  Fit criterion: Users should be able to access the system and its functionality on any of the above browsers.
  Rationale: Ensures a consistent user experience across different browsers, allowing accessibility for all users.

15 Security Requirements

15.1 Access Requirements

ACR-1

- The system shall verify the credentials passed onto the login screen before giving access to the system.
  Fit criterion: A valid email and password combination should be passed in by the user before the user can access the system.
  Rationale: Ensures secure authentication, preventing unauthorized access and protecting user data.

ACR-2

- The system shall allow only verified users to view information about other accounts.
  Fit criterion: Non-logged-in users should not be able to view information about other users.
  Rationale: Protects user privacy by restricting access to account-related information only to authorized users.

ACR-3

- The system shall allow only administrators to see the emails of users.
  Fit criterion: Non-admin users should not have access to the emails of other users.
  Rationale: Maintains user privacy and security by restricting sensitive information to authorized administrators only.

15.2 Integrity Requirements

IR-1

- The database shall prevent incorrect data from being introduced.
  Fit criterion: The database should enforce constraints, validations, and data integrity checks to reject invalid or inconsistent data.
  Rationale: Ensures data accuracy and reliability, preventing data corruption.

IR-2

- The database shall be regularly backed up in case of any failure
  Fit criterion: The system should automatically create and store database backups at regular intervals to enable data recovery in case of failure.
  Rationale: Prevents data loss by allowing restoration in case of unexpected failures.

15.3 ~~Privacy Requirements~~

~~**PRR-1**~~

- ~~The product shall collect, store and use the user's email address for the login verification process only.~~

~~**PRR-2**~~

- ~~The product shall not share its users' email addresses with third parties under any circumstance.~~

15.4 Audit Requirements

N/A - The use of the product will not involve any sensitive data. It will also not be subject to any government regulations so auditing requirements are not needed.

15.5 ~~Immunity Requirements~~

~~IR-1~~

- ~~The product shall ensure that only verified users are allowed to submit algorithms~~

~~IR-2~~

- ~~The product shall ensure that user-submitted algorithms are executed in a secure and isolated environment and prevent malicious actions.~~

16 Cultural Requirements

16.1 ~~Cultural Requirements~~

~~CUR-1~~

- ~~The product shall not be offensive to any group of people.~~

17 Compliance Requirements

~~LGR-1~~

17.1 ~~Legal Requirements~~

- ~~In compliance with the Matlab End-user license agreement.~~
  ~~Fit criterion: The system must adhere to the licensing terms and conditions in the Matlab EULA.~~

17.2 ~~Standards Compliance Requirements~~

~~SCR-1~~

- ~~General Data Protection Regulation compliance should be the target for user data.~~
  ~~Fit criteria: The system must ensure only necessary data are collected from users and that the data are processed and stored per GDPR.~~

18 Open Issues

- Investigate Matlab licensing and the integration of Matlab submissions with our system.
- Explore different AWS pricing

19 Off-the-Shelf Solutions

19.1 Ready-Made Products

N/A - This system is a rebuild of an existing application (Borealis), for which there are no existing solutions on the market.

19.2 Reusable Components

- REACT
- AWS

19.3 Products That Can Be Copied

- Kaggle: The format of Kaggle fits our use case well, with the difference being the type of algorithms being evaluated. Kaggle can be used as a guide for how the front end of the service should function.

20 New Problems

20.1 Effects on the Current Environment

Since this is a completely new system, users will have to adapt and learn how to use it and potentially change their workflow.

20.2 Effects on the Installed Systems

- In this case, while some software will be reused from the existing system, the new system is intended to replace the existing system, so they

will not interact. The old system will be decommissioned after the new system is deployed.

20.3 Potential User Problems

It is possible that performance and scaling issues could arise while using the system which could cause users to get frustrated with the system. It is also possible that the submitted algorithms might not work with the system due to different versioning/ if the system is implemented in GnuOctave instead of Matlab and certain libraries or methods fails to work. Other potential issues could be problems with usability and security. To mitigate these concerns, regular feedback from stakeholders should be taken. Also, sufficient performance and security testing will be needed.

20.4 Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

- Use of Matlab; we need to either run many concurrent instances of a Matlab interpreter or efficiently compile Matlab code repeatedly to execute user submissions. We may encounter issues with either licensing or product performance here
- Cloud Costs: Using AWS for hosting the execution backend may be prohibitively expensive

20.5 Follow-Up Problems

- In the event the product achieves widespread use, the cost of computing may be too high to support the product with a large number of users (1000s of concurrent users)

21 Tasks

21.1 Project Planning

Task List:

- Problem Description (Required by September 23, 2024)
- SRS document (Required by October 9, 2024)
- Hazard Analysis (Required by October 23, 2024)
- VnV plan (Required by November 1, 2024)
- Create Django Framework (Required by November 11, 2024)
- Configure Submissions Within Django Framework (Required by November 11, 2024)
- Configure Leaderboard Feature Within Django Framework (Required by November 11, 2024)
- Configure User Accounts and Registration Within Django Framework (Required by November 11, 2024)
- POC (Required by November 11, 2024)
- Design Document (Required by January 15, 2025)
- Design to Look Similar to Kaggle (Required by February 3, 2025)
- Product Revision 0 (Required by February 3, 2025)
- VnV Report (Required by March 7, 2025)
- Create Tutorial Videos (Required by April 2, 2025)
- Final Revision (Required by April 2, 2025)

21.2 Planning of the Development Phases

- Proof of Concept stage: This stage aims to provide the basic functionality of the new system, accept an algorithm submission, run it against the existing test script, and report the results back to the user. This stage must be operational by November 11th, 2024. The operational environment components involved are the hosting provider used for the front end, as well as the hosting provider used for the back end (these may not necessarily be the same). The function requirements involved at this stage are

  - The system shall allow users to submit algorithms to be tested
  - The system shall report the results of submitted algorithms when their testing is complete
  - The system shall allow for multiple submissions from users to run in parallel.

  None of the non-functional requirements are in scope at this stage

- Revision 0 stage: This stage aims to provide the first working version of the system with complete functionality. This stage must be complete by February 3rd, 2025. All operational environment components are involved, as are all functional and non-functional requirements
- Revision 1 stage: This stage aims to provide an improved complete version of the system, based on the learnings from the development and testing of revision 0. This stage must be completed by March 17th, 2024. Once again all operational environment components are involved, as are all requirements. Additional requirements not discovered before the revision 0 stage may enter the scope here and be included in this development phase.

22 Migration to the New Product

22.1 Requirements for Migration to the New Product

N/A - The existing system had no accounts/history or storage of results since results were emailed directly to users.

22.2 Data That Has to be Modified or Translated for the New System

N/A - Existing results of previously submitted algorithms can be discarded once users switch to the new system.

23 Costs

- During the development and testing phase, the application will not be under heavy load so cloud hosting costs should be less than $100

24 User Documentation and Training

24.1 User Documentation Requirements

As part of user documentation, the software will come with a user guide as well as walkthroughs that are built into the application, similar to how GitHub presents its tutorials.

24.2 Training Requirements

- Hints will be provided for new functionality to help board new users
- Video explanations can be added for complex components of the application

25 Waiting Room

- Submit algorithms in a variety of other programming languages (not exclusively accepting matlab algorithms)

26 Ideas for Solution

- SHARCNET
- AWS ECS
- AWS MATLAB prebuilt machine image

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain-specific knowledge from the domain of your application, software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, team management, etc. You should look to identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?

1. **Nathan Uy**

   - Being a team player is crucial to completing this project. Effective communication skills are necessary to manage conflicts and convey priorities and expectations from one another. Planning effectively, adapting to change, and problem-solving will also be needed throughout the project development since there will be changes in the requirements, tight deadlines and other unexpected changes that the team should be prepared for.

   Declan Young

   - Developing knowledge of best practices for both frontend and backend development to ensure that the two work together seamlessly is another critical aspect of the success of this capstone project. Since there are multiple components that must work together in order for the application being developed to function efficiently and effectively, it will be important to ensure that each performs as expected individually, but also when iterating with each other.

Benjamin Dubois

- Knowledge of configuring and maintaining a server will be crucial for our success in this capstone project. For this project, we will be developing a server that will allow for submissions to be tested in parallel and the majority of our project depends on this being successful as this is the number one requirement of our stakeholders.

Aidan Mariglia

- Success for this project hinges on our ability to manage many concurrently executing backend jobs, ensuring that throughput is high enough to provide a smooth experience to our users, while not producing high hosting costs. Managing a many concurrent job is a frequent task in backend engineering, so the skill most important to the success of this project is backend design.

2. **Team Working skills**

- Experience working in a team will help us acquire great communication skills, project management skills and conflict management skills. Participating in team-building activities or any hands-on team activities can also strengthen these skills. We will ensure to continually engage within our group to enhance our communication skills and project management skills and do retrospectives on how we could improve next time.

Best practices for both frontend and backend

- The first approach that can be used is referencing documentation, textbooks, research or other references on best practices to ensure that both the frontend and backend are developed properly. Additionally, using this information will allow our group to acquire a significant amount of knowledge and skills regarding these topics, which will allow us to make more informed decisions on less standard topics and issues related to these fields that we are encountering throughout the duration of the project.

- Another approach is experimenting with different technologies to see how different frontend and backend frameworks and technologies interact with each other. This will give us examples of different approaches to frontend and backend development, which we can apply to each iteration of the project.

Knowledge of configuring and maintaining a server

- The first approach to gaining this knowledge can be working on a test system and experimenting with features to ensure we know what works well and what we need to improve on. This can be considered the trial and error approach as if we make any changes on this test server that cause major issues, it is not a problem.
- Another approach that can be used is referencing online tutorials and walkthroughs from experts in this field. There are millions of videos available that can be very helpful in gaining this knowledge. This is the approach that our team is going to pursue as it is more efficient than trial and error and does not require the setup of a test server.

Backend Design

- The first approach to gaining this knowledge is studying theory. Many resources exist which discuss design patterns and other techniques which aid in efficient backend design. Studying these techniques provides a developer with the necessary background knowledge to make the correct design decisions.
- The second approach to gaining backend design knowledge is experience. Through designing many systems, and making mistakes, an intuitive knowledge is gained for backend design methodology, making future decisions easier and previous mistakes avoidable. This is the approach which will be used by the team, in order to leverage our existing co-op work experience.