**What is SRS**: This will detail your product features and expectations.

Minimum requirements:

1) Title

2) Version [Version 0 would be what you submit on Oct 10]

3) Personnel of the project and their roles (names and emails)

4) List of abbreviations and notations, naming conventions and definitions

**1-6, all: Archivist.**   
  
**8-9, 10: Requirements Writer.**  
  
**7, 10-12: Non-functional Requirements Writer.**   
  
**13-16: Project Coordinator.**  
  
**17-18, (contributes to 8-9, 10): Co-requirements Writer.**

5) Table of contents and contributions (Who contributed what part, in a table format)

6) The Purpose of the Project

**SRS (9-11 pages):**

## 7. Stakeholders

7.1 Primary Stakeholders

a. Client

* Dr. Phil Kollmeyer - Requesting for a website that lets users make accounts, submit battery SOC estimation algorithms, and test their submitted algorithm’s performance

b. Course Stakeholders

* Dr. Mehdi Moradi
* Amir Hossein Sabour

7.2 Secondary Stakeholders

c. Users of Batter SOC Estimation Tool

* Individuals who use the website that hosts the battery SOC estimation tool to make an account and test their submitted algorithm(s).

8) Project Constraints (high level choices you have made or that are mandated by your

supervisor) and relevant facts. Examples are app versus web-based or limiting your

solution to certain scenarios. Please consult your supervisor if you have one.

• The app will be web-based   
 • Users will be able to download the app on desktop to run tests on local machine in order to bypass submission limit  
 • The app will

9) Functional Requirements (formal list with priority ranking (P0-P3)). Include details about

what data is needed for building each function. This is the most important section and

needs to clearly state what you will build. Detail both backend and frontend features.

Please consult your supervisor if you have one.

**FRONTEND**

**MODEL UPLOADING:**

◦ The system will allow the user to name a new model and set it’s visibility (public or private). (P1)

◦ The system will require the user to upload a file when creating a new model. (P0)

◦ The system will support different model file types (Python, Matlab, etc). (P1)

◦ The system will have an interface for file upload that supports drag-and-drop as well as browsing the file system. (P2)

**MODEL VISUALIZING:**

◦ There will be a list consisting of all public models, their creators, upload dates, scores, and other relevant attributes, sorted by score in a “leaderboard” fashion by default. (P0)

◦ Users will be able to search by title/creator, as well as sort and filter by attributes such as model type, certain value ranges, etc. (P2)

◦ Users will be able to view model performance graphically (charts, etc). (P1)

◦ Users will be able to directly compare models, both in table and chart form(s). (P1)

◦ Users will be able to download/export/share models and results quickly, including code and documentation details. (P2)

◦ Users will be able to interact with one another in comments sections below models. (P3)

**GENERAL:**

◦ Users with an “Admin” role will see an admin page and will be able to delete comments, and models. (P2)

**BACKEND**

**USER ACCOUNTS:**

◦ Users should be able to sign up, log in, log out, and reset their password securely. (P0)

◦ The system must provide secure user authentication with industry-standard best practices, including password requirements. (P0)

◦ The system will have support for Single Sign-On. (P2)

◦ The system will assign roles to users, including the default “User” role and an “Admin” role. (P2)

◦ The system shall restrict access to CRUD operations to authenticated users. (P0)

**SYSTEM:**

◦ The system will enforce a limit on number of submissions by a user to preserve system resources. (P1)

◦ Uploaded models will join a queue to be processed if the server is busy testing models at the time of upload. (P0)

◦ Models will be tested in parallel to increase throughput. (P0)

◦ The app will have an uptime of >99.9% to ensure reliability. (P1)

◦ The app will have

**CREATE OPERATION:**

◦ The system shall allow users to create new model records (name, visibility, author, date created, files). (P0)

◦ The system shall store model records in a database.

◦ The system shall sanitize and validate values on creation of a model record

**READ OPERATION:**

◦ The system will allow users to retrieve and display a list of model records. (P0)

◦ The system shall allow users to search, sort, and filter records. (P0)

**UPDATE OPERATION:**

◦ The system shall allow users to update the visibility and name of a model. (P1)

◦ The system shall sanitize and validate updated values. (P0)

◦ The system will overwrite updated model in a database. (P0)

**DELETE OPERATION:**

◦

**In descending order of priority, under each category:**

---User Account---

Legacy Code bug-fixing

As a user, the first thing you do is upload a file, your own model, private or public.

Sort by model types, download the results fast

Password Restriction

---Hosting---

PARALLELIZED for simultaneous load.

CRUD functionality and authorization for admin access to change database features, change something from private to public. Delete a row, cull database, etc. Can be an ADMIN PAGE.

BACKEND - TIMER TO TIMEOUT SUBMISSIONS, PREVENT DDOS.

--Features--

MATLAB / PYTHON INTEGRATION

COMPARE 2 Models, add a compare page.

SHARE ALGORITHM, include documentation details.

COMMENTING, USER INTERACTION UPGRADES

## 10. Data and Metrics

10) Data and Metrics (This is particularly important for ML/AI projects, but metric could be

important for any project with quantitatively measurable outcomes): - Ellen

a. For each feature, explain what data, if any, you will use to train/build.

b. Links to dataset or a clear plan to obtain or simulate data

c. One or more performance metrics (accuracy, precision, recall, area under ROC

curve, area under precision-recall curve). Explain why you think the metric is

relevant. Also set expectations (Goal for the metric).

## 11. Non-Functional Requirements

a. Look and Feel Requirements

* Must have a main landing page that introduces the battery SOC estimator tool, general information about how to use the tool, and examples of submissions. This informs visiting users of what the estimator tool is and how a user can get started
* Must be able to display a graph that visualizes the performance of the user's submitted SOC algorithm.
* Must display information about a user’s submitted algorithms in an organized and informative manner.
* The overall look of the user interface should be consistent throughout the different displays, while also being informative, clean, and easy-to-use.

b. Usability and Humanity Requirements

* The sign-up process should be quick and easy for the user by having a straightforward and understandable process.
* Submitting an algorithm for a user should be an easy process, where the user can upload their algorithm and provide additional needed information in a way that will not cause confusion. The user should also have access to a user guide/documentation that they can refer to that outlines the details of the submission process as well as what the user should expect after. The submission process should also be easy to remember, since users are expected to make one or more algorithm submissions.
* The results returned after an algorithm is submitted and processed by the battery SOC estimation tool should be presented in a clear, organized, and readable format. Graphs should be easy to interpret, and numerical or textual outputs should be logically grouped to enhance readability. An organized display also helps users recall, interpret, and compare results across multiple submissions. In addition, users should have access to documentation that provides detailed guidance on interpreting and understanding the results.
* All website layouts should maintain consistency and clarity to help users navigate easily and avoid confusion.
* Error messages should be easy for users to understand, without overwhelming them with unnecessary details that could confuse the user.

c. Performance Requirements

* The system should be able to handle a large number of users accessing the website at a time, so that users won’t experience any latency.
* The system should be able to store a large number of user accounts along with their user data.
* The algorithm submission system should include safeguards to ensure that user-submitted algorithms are not too computationally intensive. To achieve this, the system should evaluate the runtime demands of each submission.
* - needs to be able to run multiple submissions at the same time (multiple accounts) -

d. Operational and Environmental Requirements

e. Maintainability and Support Requirements

f. Security and Privacy

g. Legal Requirements

12) Risks and issues predicted. - Ellen

**Project Development Plan (4-6 Pages):**

What is project development plan: Briefly, this is supposed to describe: “Who does what, when, with what tool” and your “what” should cover the requirements from the SRS, along with your workflow and communication.

**13) Team Meeting and Communication Plan** (including how you share documents and work on them together). Use of program management tools is optional but strongly encouraged.

The team, of 5 members, will work under the AGILE methodology in direct collaboration with Dr. Kollmeyer to develop the project, and much of group communication will also involve direct communication with Dr. Kollmeyer in bi-weekly (with the exception of especially intensive weeks, such as midterm season) meetings in which ALL members are expected to attend. These meetings are important collaborative sessions, and group members are expected to engage in huddle afterwards where we touch base and describe our progress on assigned tasks.

Aside from these regular meetings, team members will communicate and share progress via **Discord**, where we have established a group chat. We will share temporary and working files (unfinished code, documents that we want to quickly share) via a shared documents folder in Microsoft [OneDrive](https://tinyurl.com/yeypyhfk) online, and we will commit finished code and documentation on our group’s [Github Repo](https://github.com/Dason-IsopodOverseer/Battery-Algorithm-Standardized-Testing-Tool).

To plainly lay out these rules, our team operates under the following rules, for which each member has consented to and will admit to the penalty associated with each violation:

**Team Meeting and Communication Agreement:**

1. Every team member agrees to work on a *“three strikes and you’re out”* model.   
   For each member, we record a tally of misconducts, defined below. Each member has leeway of **3 penalties,** and upon accruing a multiple 3 penalties results in the specific disciplinary actions defined below:
   1. For 3 penalties, the offending team member will be asked to **compensate for the missing effort/work** that has occurred as a direct result of the penalty. Members will agree upon a suitable modified work schedule for adequate compensation.
   2. For 6 penalties, the inadequacy of the offending team member will be **reported to the instructor and TAs** by **ALL** other members. Whatever consequences may follow will be the responsibility of the offending team member alone.
   3. For 9 penalties and above, the team will make a**ll efforts to eject the offending group member from the group**, and all members will actively lobby for this with the professor and TAs.

Any of these disciplinary actions or penalties can be waived via proof of a filed MSAF, type A or type B, for which the team will afford leniency on the basis of evidenced extenuating circumstance.

1. The team will agree to communicate via **Discord,** share temporary files and working code, documents, or media on our **Microsoft Shared Drive,** and do versioning and commits for implemented code on **GitHub.**
   1. Attempts to communicate, file share, and commit changes on any platform other than these three, without explicitly informing other group members, will result in a **penalty**.
   2. Failure to interact with direct communications within a week of the time the communication was made, for example, ignoring multiple direct ***@mention*** in the Discord chat, will result in a **penalty**. Responses in a timely manner are always appreciated, but this team also acknowledges extenuating circumstances
2. **On a bi-weekly basis,** the team will host a 30-minute **meeting on (insert date), starting with (start date)** with Dr. Kollmeyer to present a **progress report.** After the progress report, the team (without Dr. Kollmeyer) will gather in a **huddle to assign new milestones and delegate new tasks,** which will take 30 minutes or more.
   1. **ALL MEMBERS** are required to participate in this meeting and the huddle afterwards. Members must provide a forewarning **1 hour before the meeting; failure to do so will result in a penalty** for missing this crucial meeting.
   2. **ALL MEMBERS** are required to present the progress of their assigned tasks during this meeting for Dr. Kollmeyer. Failure to present anything at all will result in a **penalty.**
   3. Members who clearly present unfinished work, upon a majority vote of agreement, will suffer a **penalty** for missing work. This penalty will be applied **regardless of explanations or excuses, with the SOLE EXCEPTION of a valid MSAF.**
   4. During the huddle, each member will be assigned new tasks derived from the direct feedback obtained from Dr. Kollmeyer.
3. Every member agrees to treat each other with respect and dignity in all communications. It is the right of any group member to raise violations of conduct to the instructor or the TA. Should the instructor or TA condemn the behavior, then the offender will suffer a **penalty.**

**14) Team Member Roles**

The initial team member job assignments were done with the goal of completing this SRS document. Specifically, the roles assigned are as follows, labelled by the SRS portions assigned:

1-6, all: **Archivist**. This person handles the administrative details, and also proof-reading and grammar, making the final document up to spec, and rewriting all sections.  
   
8-9, 10: **Requirements Writer**. This person works on the functional requirements, basically the entirety of the proposal that is functional. Will need to communicate with all other members. The 'hard' part of SRS. Most important will be the description of features, which will be shared responsibility with Co-requirements Writer.  
   
7, 10-12: **Non-functional Requirements Writer**. This person works on user requirements. The 'soft' part of SRS.  
   
13-16: **Project Coordinator**. This person will basically do all the workflow and planning based on what Requirements Writer writes.  
   
17-18, (contributes to 8-9, 10): **Co-requirements Writer**. Complements the requirements writer by helping with half the stuff, focusing on technological implementation. Also does the Gantt chart.

For the remainder of the project, members will proceed along roles directly related to the initial assignments. For example, the Project Coordinator might do more administrative tasks; the non-functional requirements writer might do more UX and UI design.

The following assignments are understood as **suggestions,** rather than hard-set rules. This is because of the AGILE nature of this project; *we expect numerous reassignments and distribution of roles between team members over the life cycle of our software.*

15) Workflow Plan

a. How will you be using GitLab or GitHub, including branches, pull request, issue management?

b. Using agile methods is encouraged. You can use scrum and sprint planning in Jira. State if you will do this. We will not monitor your sprints.

c. Where do you store your data (especially if you are doing machine learning)?

d. Where do you run compute heavy tasks like training models.

e. What tool/method is used to achieve each of the requirements and achieve the performance metrics that were proposed in your SRS?

16) Proof of Concept Demonstration Plan

a. What will you demonstrate during your proof-of-concept demonstration to convince yourself that you will be able to overcome this risk? For example, one approach would be to have a mock website for a service, with limited backend. There must be code. You cannot just plan to show slides. Seek advice from your TA/instructor if you are unsure.

17) Technology

a. Specific programming language (front end and backend), coding environment. State if will use unit testing framework, why or why not. If your project is primarily software development (as opposed to research), we expect you to follow software engineering best practices including unit testing.

Front end: Javascript will be used as our frontend programming language within the React framework. Styled-components will be our styling library for the UI. React-router-dom will navigate between pages.   
  
Backend: We will be using an AWS backend to support our project. The will be a Lambda and a Gateway created for the front-end to connect to the backend. We want users and admin to to be able to perform CRUD operations. Amazon Cognito will be used in order to authenticate users and add roles to admin user where   
Regular users → CRUD on their own data   
Admins → Full CRUD on all data.   
Axios will be used to call for the API. Amazon DynamoDB Stores users, metadata, scores, and results. We wil use Matplotlib within the AWS Lambda function in order to convert the MATLAB code to python for visualization. This will help in avoiding any future MATLAB licensing issues that occurred with the last group.  
  
React Frontend

↓ (POST data or file)

AWS Lambda / EC2 (Python Flask or FastAPI backend)

↓

NumPy / SciPy / Matplotlib

↓

Computation + Visualization

↓

Save result → S3 or return JSON to frontend

b. ML libraries (if relevant)

c. Will you use GPU? Any other relevant technology aspects.

18) Project Scheduling: Include a Gantt chart

