1. Each team member (without consulting with other team members) will create their own SRS document containing 15 functional requirements and 3 non-functional requirements.

2. Submit these documents WITHOUT NAMES to the scrum master assigned to this sprint.

3. The scrum master will divide the team into two groups and assign half of the documents to each section.

4. Each sub-team will discuss the documents they've been given and "merge" them into one document. (The merged documents should still have at least 15 functional requirements and 3 non-functional requirements in the end but can have more if you find the submitted requirements distinct and valuable enough apart from each other. Don't just copy and paste one into another, though -- you'll need to discuss how to add/combine/rewrite the contents of both files to create the new one so it is internally consistent without repeats, redundancy, or contradiction between all the requirements.)

5. Each sub-team submits its merged document back to the scrum master.

6. As a team, meet and examine the resulting documents, and create one final SRS document based on your group's discussion and feelings about the results of the previous exercises. (The final document should have at least the same 15/3 content as before after any merging/rewrites, but it can have more. As before, make sure your final document is consistent after the merge with no repeats, redundancy, or contradictions)

7. Submit the final document and the 6-7 working documents created in the earlier steps. (One per team member, plus one per subgroup, in addition to the final document)

Functional Requirements: needs to complete task

1. The app needs to allow the user to be able to select a file.
2. The app needs to have a way to input numbers.
3. The app needs to be able to handle errors.
4. The app needs to provide a way for the user to run the program.
5. The app needs to keep track of the accumulator value.
6. The app needs to keep track of the location in memory of each value.
7. The app needs to have a way to quit the program.
8. The app needs to have a functioning UI.
9. The app needs to have a UI that the user can understand in order to use the program.
10. The app needs to correctly read the user’s file.
11. The app needs to properly dissect BasicML code.
12. The app needs to be able to grab the memory location in the code.
13. The app needs to be able to grab the positive or negative signs in the code.
14. The app needs to accept numbers without a positive or negative sign.
15. The app needs to correctly apply the BasicML code.

Non-Functional Requirements: user friendliness, not necessary

1. For user friendliness, the app could display the accumulator value.
2. Perhaps for user friendliness, it might make sense to separate all the information into different clickable sections.
3. One other thing for user friendliness is to provide test files for the user in case they don’t want to write their own.