Acceptance Test for User story:

To manually test U.S 1, cd into the iCare folder, run `npm install` to download all the dependencies you'll need. Then to run the react server you will need to `npm run start` .

Open a new tab in your terminal and cd to iCare/server and run `nodemon app.js` (or node) to run the backend.

Now You can open up localhost:3000 to see the login page, to test whether the login page works, you can enter the credentials, 'email@email.com' with the password 'pass' and press the submit button to get redirected to an upload page.

To test the upload file functionality, while on the /upload page, there should be a clear file drop zone outlined with a dashed border. To test the functionality of this file upload, simply upload any file from your computer by dragging & dropping a file, or clicking and selecting a file from the file explorer.

To test the pre set query functionality, there should be a button right underneath the file drop zone. Simply click this button and a table of results should be displayed right underneath it.

Code Review Strategy:

Our strategy will focus on the readability, maintainability and reliability. As a team, we will try to identify issues regarding styling, functionality and design. Each group member will review approximately 200 lines of code in the session (should last about 30 mins). Specifically, Ishan will review the express server (app.js); Joe and Sai will review various React components; Ricky will review the insertDatabase.js functions. Reviewers should jot down notes or issued on a piece of paper or on their computer for further inspection and improvements. Team members should focus on answers of the following questions:

1. Is there sufficient documentation?
2. Is the code style clean and consistent?
3. Does the code use appropriate naming conventions?
4. Does the code have any magic numbers?
5. Does the code follow SOLID design?
6. Can we implement design patterns?
7. Can one easily understand what the code does?
8. Is there unnecessary debug statements?
9. Is there error handling?

Code Review Summary:

Ishan: Overall, the code in app.js is strong and structured. In app.js, the code is clean and organised. The code is written in such a way that there is no need for documentation, since all the functions are fairly simple and well written and thusly self explanatory. There is no need for design patterns or SOLID design as there is only one server file handling fairly simple post and get requests. However, one issue with this code is that there are some unnecessary debug statements that we can get rid of. Another issue is that the code is still prone to error and exceptions, as some of them have not been handled.

Sai: The code in app.js is well structured with many parts componentized so that it is very easy to read and understand. Since single components are easy to understand, we are able to build larger pages by breaking them down into parts that we can reuse. I noticed that some parts of the pages can still further be broken down into more components so that is something that needs to be fixed. Overall however, it is very clean and follows good design.

Ricky: The code in insert database needs a lot of work to be done stylistically. The code spacing and formatting is inconsistent and hard to read. Specifically, functions with nested callback functions (such as con.query) need to look cleaner with the spacing. There are comments but those are also inconsistent as they are not everywhere. Also, there are lots of things that can be written cleaner. For example, when parsing excel files, there are lots of nested for loops and function calls that could be written cleaner. In addition there are commented out debug statements that can be removed.

Joe: The code in inputBox.js is simple and clean. Since inputBox.js is just a single component(an input box), much of the code is self explanatory and all of the variable names are clear and concise. This also follows good coding practises because this class has only one task, and it can be reused several times by other classes. However, even though the class is very simple, it does not have any documentation whatsoever and I feel like it could use just a little bit. The code in fileUpload.js is also simple and clean. Like inputBox.js, fileUpload.js is a single component with one task and is reusable which follows good coding practices. The code is not as self explanatory as inputBox.js, which means it should have some documentation. However, there is only one line of documentation in the code which is not enough.

Code review debriefing video:
https://drive.google.com/file/d/1GpdsEE9vvWu0CrEjLvDupAQ_0QNnzlP0/view?usp=sharing