Alvin

* HistoryLogs (App) – 25/10
  + Finish The Implementation
* Use Case Diagram – 26/10
  + Update the use case diagram based on our current implementation, remove unimplemented functionality
* Boundary Class Diagram – 27/10
  + Update the boundary class diagram based on our current app design
* ViewHistoryLogs (Sequential Diagram) – 27/10
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)

Ming Kai

* Dialog Map – 26/10
  + Update the current dialog map to fit our dialog map, remove unimplemented functions
* RunnerMainMenu (App) – 26/10
  + Asynchronous the markers and the app
* UI Mockup – 27/10
  + Update the current UI to look like our UI, remove unimplemented functions
* Login (Sequential Diagram) – 27/10
* ViewCarParkAvailability (Sequential Diagram) – 27/10
* ViewRoute (Sequential Diagram) -27/10
* CompleteJob (Sequential Diagram) -27/10

Kan Yui

* Use Case Description – 27/10
  + Update use case description based on our current implementation and use case diagram, remove unimplemented functionality
* Entity Class Diagram -27/10
  + Update entity class diagram based on our current tables we are using
* ViewJobs (App) – 27/10
  + Finish up View Job by displaying the job description
* ViewJobs (Sequential Diagram) – 27/10
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)
* ViewJobDetails (Sequential Diagram) – 27/10
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)
* Record Video of the app - 29/10
  + After the main app is functional, record a video showcasing the different tabs in our app as well as the completed job

Andria

* Non-Functional Requirements
  + Update our functionalities, remove functions that we are not implementing
* ManageRunner (Sequential Diagram)
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)
* TrackActiveRunner (Sequential Diagram)
  + TrackActiveRunner is the function in OperatorMainMenu that shows all the active runner and their newlocation and lastlocation
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)
* ManageJobs (Sequential Diagram)
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)
* ViewRunner (Sequential Diagram)
  + Update the BCE accordingly based on usage (useEffect and server.js is controller, boundary is the “html” aka return <div> </div>)

Anxian

* Familiarize with our MERN app – 27/10
  + Familiarise with Github and how to pull the app
  + Launching the app itself
    - npm install on server folder and client folder
    - npm start on both
    - Able to understand the different navigation
* Functional Requirements – 27/10
  + Update our functionalities, remove functions that we are not implementing
* System Architecture Diagram – 27/10
  + After understanding the functionalities attempt to draw the system architecture diagram
  + Can take the “links” from dialog map and the “functions” from boundary class diagram
  + From my understanding there are 4 layers (in order)
    - User
    - Boundary Classes (Presentation Layer)
    - Control Classes (Business Layer)
    - Entity Classes (Persistent Data)