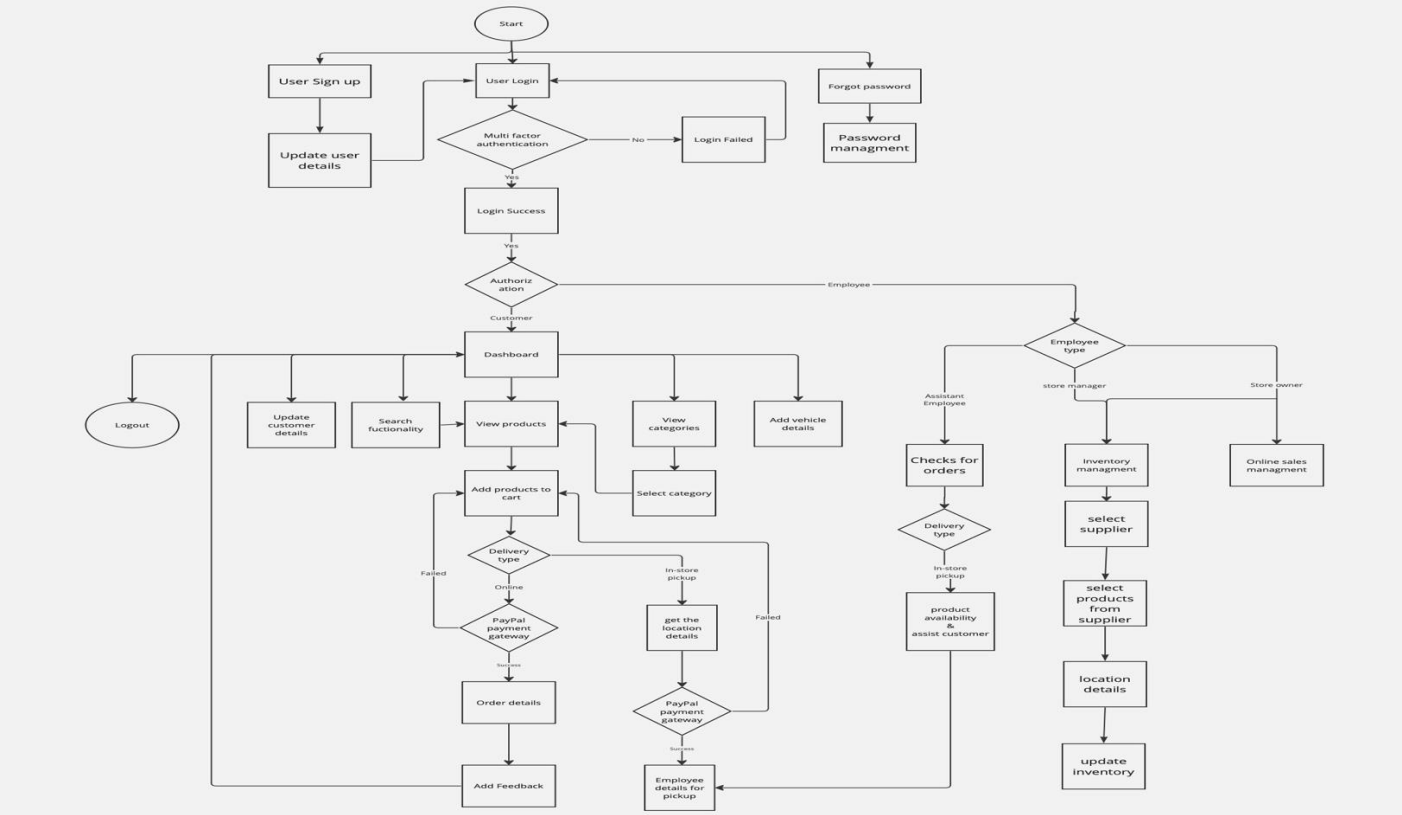
Deliverable 2

System Structure:

Top-Level Description:

All types of users will initially sign up for the application, once user successfully signs up into the application, they can login into the system. We have two types of users.
If the user is of type customer, the order of sequence would be, he/she would land to dashboard page. We can add vehicle details to buy vehicle related parts. The customer could be able to view products and categories. The customer could click on any of the products to view the product details. The customer could also click on any category to view all products based on the category. Once customer selects the products, he can review all the products in checkout page and the products will be added to cart. Customer could complete the payment through secured payment gateway like PayPal and if the payment was success, the customer can view the order details. If the payment was declined, the customer would land again to checkout page to retry for payment. Customers can give feedback and rating once they have landed on order details page.
If the user is of type employee, there would be different permissions and roles for each type of employee. The owner can add, update, view and delete the products from the inventory. Store owner could add products by selecting suppliers and each supplier would have different products. Store owner could also manage online sales performed by the customer. The manager could also add, update, view and delete the products from the inventory. Assistant employee could assist the customers if customer selects for in-store pickup delivery. The assistant employee could only view online sales for which he/she has assisted the customers. The application has capability of having multiple store locations and each location has the inventory management.

SYSTEM DIAGRAM:



Description of System Components:

Multi Factor Authentication: Customers/Employees would login into the application by entering username and password. If he/she was able to authenticate successfully, OTP will be generated and once OTP is verified, the customer/ employee would land into their dashboard page.

PayPal Payment Gateway: A subsystem which handles all payment transactions done by the customer. The payment channels would interact with banking systems to process the customer payments.

User: An initial sub system which contains personal information of all the users as they register for the application, potentially it would be customer or the employee.

Customer: An entity who would use this application to purchase vehicle products. The customer could also add vehicle details and provide feedback etc.

Vehicle Details: A data entity which could store customer's vehicle information and display the products based on vehicle related details.

Inventory: The subsystem which manages item quantity, item status in the stock. It contains products and there would be inventory in each store location.

Product Category: A system of classification that divides products into various categories according to similar characteristics.

Products: This represents the items which would be sold to customers. Products would be part of inventory and it would be linked with suppliers as there would multiple products under a supplier and each supplier would also supply multiple products.

Supplier: An entity that represent an individual who supplies products to all the part pro locations. They have the linkage with all the locations and with the products.

Employee: The individuals who work for part pro company, which includes store owner, store manager and assistant employees. They have main line of business work on inventory management, online sales management.

Location: An important entity that represents the physical location of all the stores for part pro company.

Online Sales: A subsystem that contains sales of all the products, who has purchased the products, and which employee got engaged for customer assistance.

Cart: A temporary entity which holds the products what customer wishes to purchase. It plays a major role for payment processing and order management.

Order Details: A data component which has each transaction and tracking details like items purchased, total price and status of the order.

Feedback: A subsystem which contains customer review and feedback of the products purchased for getting valuable suggestions and insights from the customers.

Owner, Manager and Assistant Employee: The individual components which represents roles of each type of employee and each role has their own set of permissions and access levels for part pro website. All types of employees need to login to get their permission set groups.

Requirements Specifications:

Functional Requirements:

1. **Login Module:**

   Sign Up Functionality: User will be asked to enter all the details like username, password, email Id, mobile number, zip code, user type role etc.. If the username is not found and if password is unique user will be created.

   Authentication Functionality: If the user is able to open account successfully, He/she should provide username and password in login screen. If the login details are valid i.e., it will query in the database with that username provided, if the user record was found. It will retrieve password from the same user record. The password field will be encrypted and stored in database. If the decrypted password matches with password entered by the user, the user would successfully login into the application and land on MFA page. Both customers and employees would be able to reuse the same login screen for authentication.

   Forgot Password Functionality: If the user has forgot password, he/she can update the password by entering username. If the username was found, user can change password by entering new password. The new password would be updated in database for the same user record.

   Multi Factor Authentication Functionality: Once the user can login using username and password, user needs to complete the MFA using OTP validation. The OTP would be generated and it would be stored in database with fields like OTP, userID and expiry time. The OTP would be shared to user's emailID. If the user enters the OTP which was sent to the user, OTP verification would be done. If the OTP was expired or OTP was incorrect, he/she should retry for OTP. If the OTP entered was correct, user would land on dashboard page.

   User Authorization: As there are two types of user roles, if the user role is of customer they would land on customer dashboard page. If the user role is of employee they would land on employee dashboard page. When user creates account, based on the user role, if there are any common attributes related to the user, customer, and employee. The common fields will be automatically populated into the customer and employee tables during user creation.

2. **Customer module:**

   Add Vehicle Information: Once customer can land on dashboard, customer adds vehicle details like vehicle Type, mileage, color, vehicle model etc. The customer related vehicle details would be stored in vehicle entity. One customer could also add more than one vehicle details. Based on the vehicle details, the products would be automatically filtered and displayed in dashboard page.

   Retrieve All Products: On the customer dashboard, customer can view all the products available in part pro stores. All products will be stored in products entity and all product names will be retrieved and it would be displayed as cards in the form of carousel.

   Retrieve All Categories: On the customer dashboard, customer can view all the categories available in part pro stores. All categories will be stored in categories entity and all category names will be retrieved and it would be displayed as cards.

Update Personal Details: Customer could have an option to update personal details like mobile number, emailId, zip code etc., Once customer updates the details, it would query with the customer Id and updates the customer details. If the user id is invalid, exception will be thrown.

Nearest Store Locator: Once customer lands on dashboard, customer would be able to view the nearest part pro store. Based on customer's current location, nearest store zip codes would be retrieved from location entity and based on distance calculations in back end one of the zip codes and location address would be persisted as nearest location and it would be populated on customer's dashboard.

Products Search and Filtering: Once customer searches either the category name or by product name, all the products will get filtered from the product entity and products would be retrieved based on search inputs. The dashboard gets refreshed with the filtered products.

3. **Product Details Module**:

Get All Product Details: Once customer clicks on any of the products, based on the product ID, it would query in the products entity. If the product ID is found, product related information like product description, price, product image and product rating would be retrieved and it would be displayed in UI.

Get All Products by Category: Once customer clicks on any of the category , based on the category ID, it would query in the category entity. If the category ID is found, category related products information like product name would be retrieved and it would be displayed in UI.

Delivery Mode Selection: Customers have a flexibility of instore pickup and online delivery modes. If the customer selects instore pickup, on successful payment, assistant employee details and store location details would be displayed for the customer for picking up the items. If the customer selects for online, on successful payment, order details like order ID, order status and products which they have purchased should be displayed to customer

4. **Cart Module**:

Add Items to Cart: When customer clicks on Add to Cart Button which would be displayed under each product, it would add that item into cart. We can also increase or decrease the quantity of each item in cart. Item details would be reviewed in the cart page. Total price i.e.,(the summation of the all the product price*quantity)would also be displayed in the cart page. The cart related information would be stored in cart entity with product Id, customer Id and product quantity.

Remove Items from Cart: On the cart page, customer could also click on Remove items from cart button to discard the items. Respective combination of product Id and customer ID would be deleted from cart entity. Remove button would be shown beside each product which got added in the cart.

5. **Payment Module**:

Complete Payment: Once customer clicks on complete payment button, it would redirect to new payment gateway called PayPal, where the customer completes the payment. If the payment was successful, customer would be redirected to order details screen. If the payment was declined due to any technical issues, customer would be redirected to cart details page. Once payment got

initiated payment information entity would be updated with payer ID, payer Name, status, currency, and amount.

Cancel Payment: Once customer clicks on complete payment button, redirect to new payment gateway called PayPal, where the customer completes the payment. If the customer wishes to cancel the payment for any reason, the customer would be redirected back to cart details page. The payment status would be changed to cancelled and we would be updating the payment information by querying with the payment Id from Payment Information entity.

6. **Order Details Module:**

View Order Details: If the customer wishes for online mode of delivery, order details would be displayed which contains the entire information of products purchased, order status and order tracking ID. The order details would be stored in Order entity with order ID, product Id, order status and product quantity. Product related information would be displayed in order details page by querying product ID from products entity.

Order Tracking: Customer could track the order status like order shipped, order delivered and order not shipped. Order tracking Id would be displayed in order details page. On clicking of order tracking ID, order status would be displayed. The order status would be updated in orders entity. Triggers would automatically update the order status in order entity when any of the event got initiated with orders entity.

7. **Employee Module:**

Responsibilities of Store Owner: If the employee is of type store owner, he would able to view employee's dashboard. Role would be categorized based on employee type field in employee entity. Based on his/her permissions the dashboard contains Online Sales Management, Inventory Management and Employee Hire options. Owner would hire for employees and managers and onboard them into part pro stores. Owner would have full access to all the stores of part pro. Store owner would be able to view and manage all the online sales of all the part pro locations. Store owner would also mange inventory by adding products based on product status. If it is 'Out of Stock' items needs to be added to part pro store.

Responsibilities of Store Manager: If the employee is of type store manager, they would have different roles and permissions. Based on his/her permissions the dashboard contains Inventory management where they can see entire products and their quantity in their location. They can have easy access of inventory levels.

Responsibilities of Assistant Employee: If the employee is of type assistant employee, they would have different roles and permissions. They would not have access for inventory management and online sales management for the stores. Assistant employees would assist customers if the customers have opted for in-store pickup delivery. One of the employees in that specified location would get engaged with customer delivery items and ship it to the customers. Employee assignment would be randomly done based on the availability.

8. **Inventory Management Module**:

Add Products in Inventory: Store owners and managers can add products in the inventory. Employee would select any of the suppliers available. Suppliers would get populated by fetching

all the available suppliers from supplier entity. Once employee clicks on one of the suppliers, they can view all the products that supplier can deliver. Based on the current inventory levels of the store location, we can select the products and update the quantity and status in the inventory entity. Products and suppliers entity would have relationship between them. So, all the products would be retrieved from products entity.

Updating and Deletion of Products: Employees can update products related information like product images, product description, product name etc. The product details would be updated based on product ID in products entity. Unwanted products can also be deleted by employee from the available products.

Add Products in Inventory Based on Inventory Levels: Employee can update products into the inventory based on inventory levels. If the product status was 'Out of status' in the inventory entity employee would add items into the store. If the product status was 'Available' in the inventory entity employee would not take any action like addition/updating of products into the store.

9. **Online Sales Management Module**:

View Online Sales: Store owner would be able to view online sales of all the locations. Once customer completes the payment, sales table would be updated with sales ID, total product price, product ID, customer ID and employee ID. Store owner can view and manage online sales of all the store locations and can sort online sales by ascending or descending order. Sales would be displayed in pagination as there would be multiple records. Store owner could also find entire insights of each employee sales and can also find insights of each manager sales of respective store.

10. **Feedback Module:**

Add and Update Feedback: When customer completes payment and lands on order details page. Customer would be asked to add feedback. Once user gives the rating, Feedback table would be updated with product ID and entered rating value. The average of all entered ratings for each product ID would be calculated and product rating would be getting updated in products entity table.

Non-Functional Requirements:

1. **Code Reviews:** When the developer pushes code to the GitHub repository, code reviews should be done. If the code pushed has any issues or bugs, code needs to be reverted or raised with new revision with updated changes. Code comments need to be added by peer developer for any code concerns or clarifications. Developer needs to resolve all the comments by replying to it. If the code comments require new code revisions, developer need to raise as new revision and get it approved.

2. **Code Deployment**: Once the code got pushed into main branch, deployment pipeline needs to be started in Azure cloud VM. Both back-end and front-end code was pushed to Azure VM and it would get stored in Azure storage. Docker containers would be up and running once the VM starts/ restarts. MySQL docker image would run on one of the docker containers and listens to port 3306. Front end code would be built successfully using docker image and docker container would be created for the same and listens to port 3000. Back-end code would be built successfully using docker image and docker container would be created for the same and listens to port 8000. So, developer needs to deploy their code and investigate the issue if there are any errors in failure of docker container deployments by verifying docker logs.

3. **Optimization**: Code needs to be optimized once the developer completes all the important core and advanced features. Optimization should be done at database calls, API calls and decrease the response time for the users. Unnecessary logging statements and unnecessary code needs to be cleaned up. Reuse of code and modularity of code by writing code based on its functionality would improve performance and decrease the application size.

4. **Performance**: The system should be well designed to handle high amount of web traffic. As inventory, payments and onlines sales are very significant components which requires huge number of interactions by customer, we need to have less reaction time. Multi-threading or concurrency needs to be implemented for these components for better performance. During interactions with the application, users can anticipate quick processing times and prompt feedback, resulting in a flawless operational experience.

5. **Integration Capabilities**: The application has multiple integrations like database integrations, API integrations in UI, third party API integrations, PayPal payment integrations, MFA integration. All capabilities listed should be compatible enough by having efficient system architecture for each and individual systems and subsystems.

6. **Scalability**: Load balancers needs to be implemented in the cloud to route the web traffic to different servers based on the server's availability using round robin or any other techniques. Resources need to be monitored continuously, if resource health is poor or requires additional RAM, the servers which were deployed in VM, should horizontally scaled up by increasing RAM and should also vertically scale up by increasing a greater number of servers.

7. **Usability**: Customers and employees may easily and intuitively interact with the system to its user-centric interface. It complies with accessibility guidelines, guaranteeing that all users regardless of ability can navigate and use the system efficiently, improving user happiness in the process. Navigation within the application should be precise and simple and it should have less reaction time. Error handling need to be done for every module, which gives better user experience if there is any system or unplanned outages.

8. **Maintainability**: Because of the system's simple design and thoroughly defined procedures, maintenance is expedited and it should be documented. It guarantees that the system develops with the business it supports by enabling simple upgrades and improvements without causing major disruptions to operations. Code modularity should be done and split into sub modules based on the functionality so that each submodule can be easily developed, tested, and deployed.

9. **Reliability**: The architecture of the system prioritizes robustness and low downtime with an emphasis on continuous operation. In the case of a system failure, a thorough backup and recovery plan is in place to guarantee that operations can be promptly resumed, protecting data integrity and business continuity. We need to calculate mean time between failures and mean time to recover failures and need to have less values and we need to design such resilient and stable systems.

10. **Security**: Security procedures are essential to the system and protect private data, including login credentials and credit card information of the customers. Password details and any other sensitive information needs to be encrypted before storing it in database. Application Load balancers needs to in place for enhanced security. The code needs to be obfuscated and the production code should not be in development mode. MFA needs to be implemented for secure authentication. Customer need to be logged in before completing the payments using payment gateways.

11. **Coding Standards**: Coding conventions and naming conventions should be followed as per coding guidelines. The developed code should be well documented and need to be formatted for more readability and it would be easy to understand for other developers.

Interfaces:

1. **Database Integration**: The structured data get stored in database and there would be multiple database calls to retrieve, update, delete, add data on the entities. We have been using My SQL as relational database management system here and related configurations were updated in configuration files to connect to the database.

2. **Back-end API Integration**: We need to design REST API which gets exposed in controller layer in Spring Boot application. All the API's need to be tested using Postman before getting exposed to front end user interface. All the API's designed would have an interaction with database.

3. **PayPal Payment Gateway Integration**: The back end would be integrated with third party integration systems like PayPal SDK and all the payment gateway integration related configurations need to be updated in back-end property files and all the dependencies also need to be updated to have its software development and run environment accessibility into our system.

4. **Front-end UI Integration**: The front end would be exposed to end users and all components should be well structured. The user screens should have good responsive designs. The API which was exposed from back end need to be integrated and data needs to be populated on UI screens dynamically. React library was used to implement front end UI.

5. **Azure Cloud Integration:** The code would be pushed from GitHub to Azure cloud VM and need to deploy the code into the VM. The VM is of Linux OS and of type ubuntu. The public IP address of the VM would be elastic and it would first hit the front-end services and then it would hit back-end API controller layer and then data access layer where entities interact with database.

6. **MFA Integration**: Multi-Factor authentication need to be implemented and spring mail framework related configurations need to be added in property files. As the OTP is sent to the user's email ID we need to have all the dependencies that need to be updated into our system.

Development Phase Plan:

Development Phase 1:

Core Functionality and Foundation:

We are planning to develop all important and core features and which have dependency and linkage with other modules development. This phase is very critical as we would develop all critical features and critical aspects for initial setup. In phase 1 we were going to develop front end user screens for some of the modules and back-end APIs for some of the modules. Some other critical and important aspects also need to be incorporated as part of this phase.

The following requirements which need to be implemented in phase 1 needs to be prioritized as per their criticality. This phase of development is core foundation for other two phases and this phase also requires all the class diagram designs, UML designs and use case diagrams designs. We need to have a roadmap for implementing each requirement as part of this phase to understand the deadlines and their priority.

Initial Non-Functional Requirements:

1. Initial Setup of Back-end repository structure.
2. Initial Setup of Front-end repository structure.
3. Initial Database setup(MySQL Workbench).
4. Initial environment and Azure cloud setup.
5. Initial PayPal Payment Gateway developer account setup
6. Initial Multi Factor Authentication related configurations setup.
7. All software's installation related to back end and front end

Back-end REST API Development Functional Requirements: This covers both back-end code development and unit test coverage for the developed code.

1. Forgot Password Functionality
2. Multi Factor Authentication Functionality
3. User Authorization
4. Add Vehicle Information
5. Retrieval of All Products
6. Retrieval of All Categories
7. Updating Personal Details
8. Nearest Store Locator
9. Get All Product Details
10. Get All Products by Category
11. Delivery Mode Selection
12. Adding Items to Cart
13. Removal of Items from Cart
14. Completing Payment
15. Cancelling Payment

Functional Requirements for Designing Front end UI screens and Integration with Back-end API: This covers both front-end code development and unit test coverage for the developed code.

1. Forgot Password Functionality
2. Multi Factor Authentication Functionality
3. Add Vehicle Information
4. Retrieval of All Products
5. Retrieval of All Categories
6. Updating Personal Details
7. Nearest Store Locator
8. Get All Product Details
9. Get All Products by Category
10. Delivery Mode Selection
11. Adding Items to Cart
12. Removal of Items from Cart
13. Completing Payment
14. Cancelling Payment

Other Non-Functional Requirements:

1. Code reviews and merge requests for phase 1 developed code
2. Code deployment for phase 1 developed code

Development Phase 2:
Enhancements and Expansions:

We are planning to develop all other enhanced features .This phase is very critical as we would develop all critical features and critical aspects. In phase 2 we were going to develop front end user screens for other

modules and back-end APIs for other modules. Some non-functional requirements also need to be incorporated as part of this phase.

The following requirements which need to be implemented in phase 2 needs to be prioritized as per their criticality. We need to have a roadmap for implementing each requirement as part of this phase to understand the deadlines and their priority.

Back-end REST API Development Functional Requirements: This covers both back-end code development and unit test coverage for the developed code.

1. View Order Details
2. Order Tracking
3. Responsibilities of Store Owner
4. Responsibilities of Store Manager
5. Responsibilities of Assistant Employee
6. Add Products in Inventory
7. Updating and Deletion of Products

Functional Requirements for Designing Front end UI screens and Integration with Back-end API: This covers both front-end code development and unit test coverage for the developed code.

1. View Order Details
2. Order Tracking
3. Responsibilities of Store Owner
4. Responsibilities of Store Manager
5. Responsibilities of Assistant Employee
6. Add Products in Inventory
7. Updating and Deletion of Products

Other Non-Functional Requirements:

1. Code reviews and merge requests for phase 2 developed code
2. Code deployment for phase 2 developed code
3. Compatibility
4. Maintainability

Development Phase 3:

Advanced Features Development and Optimization:

We are planning to develop all other advanced features .This phase is very critical as we would develop all critical features and critical aspects. In phase 3 we were going to develop the front-end user screens for left-over modules and back-end APIs for left-over modules. Code would be optimized and code would be well documented. Some non-functional requirements also need to be incorporated as part of this phase.

The following requirements which need to be implemented in phase 3 needs to be prioritized as per their criticality. We need to have a roadmap for implementing each requirement as part of this phase to understand the deadlines and their priority.

Back-end REST API Development Functional Requirements: This covers both back-end code development and unit test coverage for the developed code.

1. Products Search and Filtering
2. Adding Products in Inventory Based on Inventory Levels
3. View Online Sales
4. Add and Update Feedback

Functional Requirements for Designing Front end UI screens and Integration with Back-end API:
This covers both front-end code development and unit test coverage for the developed code.

1. Products Search and Filtering
2. Adding Products in Inventory Based on Inventory Levels
3. View Online Sales
4. Add and Update Feedback

Other Non-Functional Requirements:

1. Code reviews and merge requests for phase 3 developed code
2. Code deployment for phase 3 developed code
3. Optimization
4. Security
5. Performance
6. Scalability
7. Usability

Member Contribution Table:

Member name	Contribution description	Overall Contribution (%)	Note (if applicable)
Sai Rahul Padma	I have worked on system diagram and initial description of the document. I have assigned all the tasks to the team.	12.5	
Nikhita Muvva	I have worked on description of all the components and subsystems and worked with Samyuktha for functional requirements.	12.5	
Sai Samyuktha Paspuleti	I have worked on finding functional requirements and worked with Rishika for non-functional requirements.	12.5	
Rishika Yalamanchili	I have worked on finding all non-functional requirements	12.5	
Pavani Venigalla	I have worked on finding interfaces, updated meeting minutes, Note-deliverable-2.txt file in GitHub.	12.5	
Himabindu Chunduri	I have worked on finding requirements which were required for development phase 1 and	12.5	

	prioritized based on their criticality.		
Sneha Reddy Gangannagari	I have worked on finding requirements which were required for development phase 2 and prioritized based on their criticality.	12.5	
Jhasha Sri Ede	I have worked on finding requirements which were required for development phase 3 and prioritized based on their criticality.	12.5	