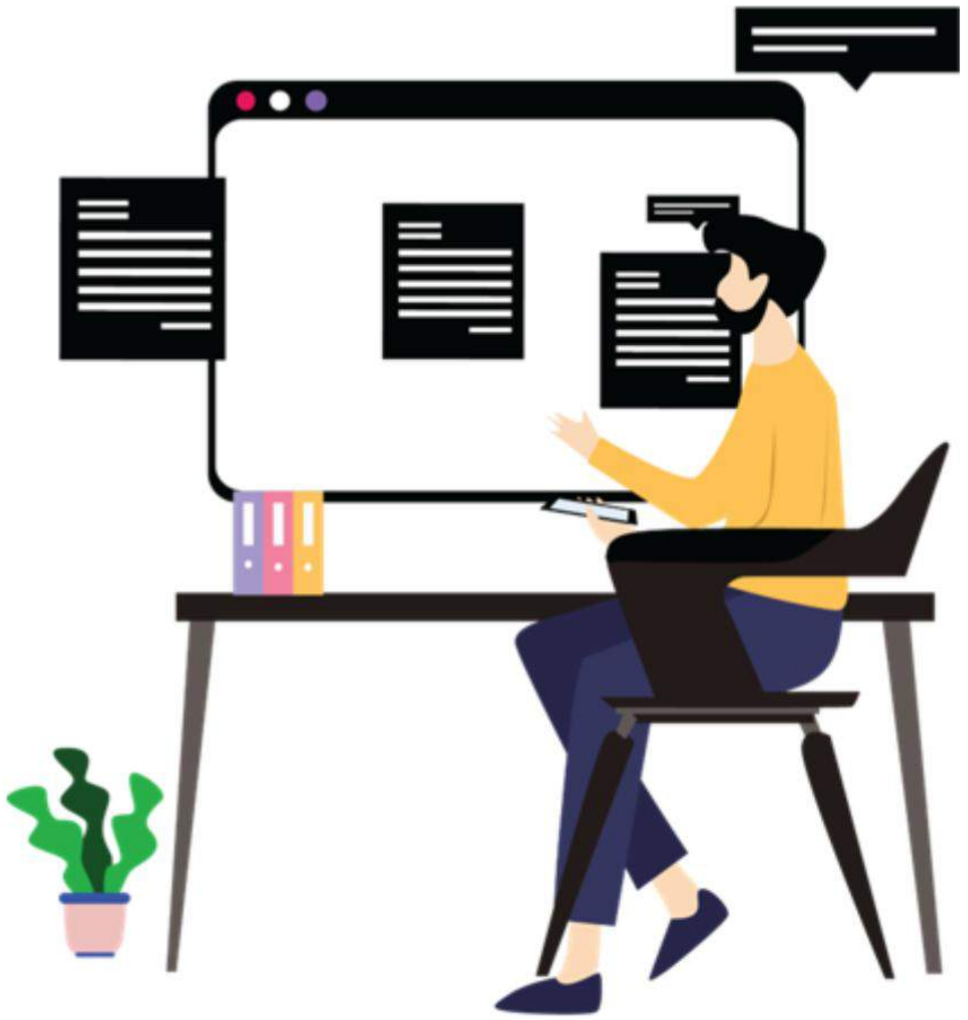
Project Statement
Milk In Minutes



Points to Remember

- Use Angular for developing SPA solution for Milk In Minutes project.
- Follow the design principles for identifying components and services for the project
  - DRY – Do not Repeat Yourself Principle
  - SRP – Single Responsibility Principle
- Use @Input() and @Output() decorators to share data between the components having parent-child relationship
- Create Angular Services for developing reusable application logic.
- Use HttpClient for making server requests.
- Use appropriate life cycle method of component for fetching data from server and reading route data.
- Use Angular Material components, themes and schematics for styling components.
- Use Template-driven forms for designing interactive views.
- Use Angular router for enabling navigation in the application and Route Guards to protect routes with restricted access.

Points to Remember: Prompts to ChatGPT

- Give sufficient context to ChatGPT to begin with.
- Give prompts to design the data model first.
  - Refactor the data model if the suggested code is not matching with your requirements.
- Give prompts to create the identified components and integrate with services to fetch data from json-server
- Provide separate prompt for creating each component
- Ensure to use Angular Material components by ChatGPT while designing the UI. Ask the ChatGPT to generate respective CSS for each component
- Give detailed information about each sections like specific icons and its positions, various form fields etc.
- Avoid giving all information at one time, ChatGPT may not be able to keep track and certain things may be missed out resulting in re-work.
- Avoid depending fully on ChatGPT for small bugs which can be fixed easily. This will result in avoiding multiple to-and-fro conversations with ChatGPT.

Milk In Minutes – Problem Statement

- Develop a single page application using Angular – Milk In Minutes, that allows customers to make an online request for milk, cheese, butter or yogurt of their choice.
- The dairy products are displayed with attractive images and crisp details, allowing the users to select the item of their choice and provide the order details.
  - The app should seek confirmation from the users before allowing them to navigate away from the order view without submitting the details.
- The app can search and filter the items by the user's preference for a quick selection.
- The site administrator can view the incoming order requests.
  - The app should redirect user to first validate his identity as site administrator before providing access to the order request view.

Task 1 – Design Landing View

- The landing view of the app must display the images of dairy products like milk, cheese, yogurt and butter.
  - The data must be fetched using `json-server`.
  - This view must be the default view.
- These items should be selectable by the user.
  - Upon selection, the user should be navigated to the order view.
- The landing view must also allow users to search / filter these items by their preference.
  - Search allows user to search by item name.
  - Filtering allows user to filter items by category.

Task 2 – Design Order View

- The user will be navigated to the order view once he selects the item on the landing view.
- The view should display the details of the item selected.
- This view should also allow users to provide the details required for placing order for the selected item.
  - The details should include the item details as well as the customer details.
- The details should be persisted, and the customer should be acknowledged after the order is successfully placed.
- The app will request confirmation from the user to leave the view, if the user attempts to navigate away from this view without submitting the request.

Task 3 – Design Login View

- The user will be navigated to the login view if he attempts navigation to the dairy-products-order view.
- The view should request the user to enter the security code to login as Administrator.
- Upon successful validation, the user should be navigated to the dairy-products-requests view.

Task 4 – Design Dairy-Products-Order View

- This view will display all the dairy products order requests received in tabular format.
- This view has restricted access and is accessible only to the site administrator.

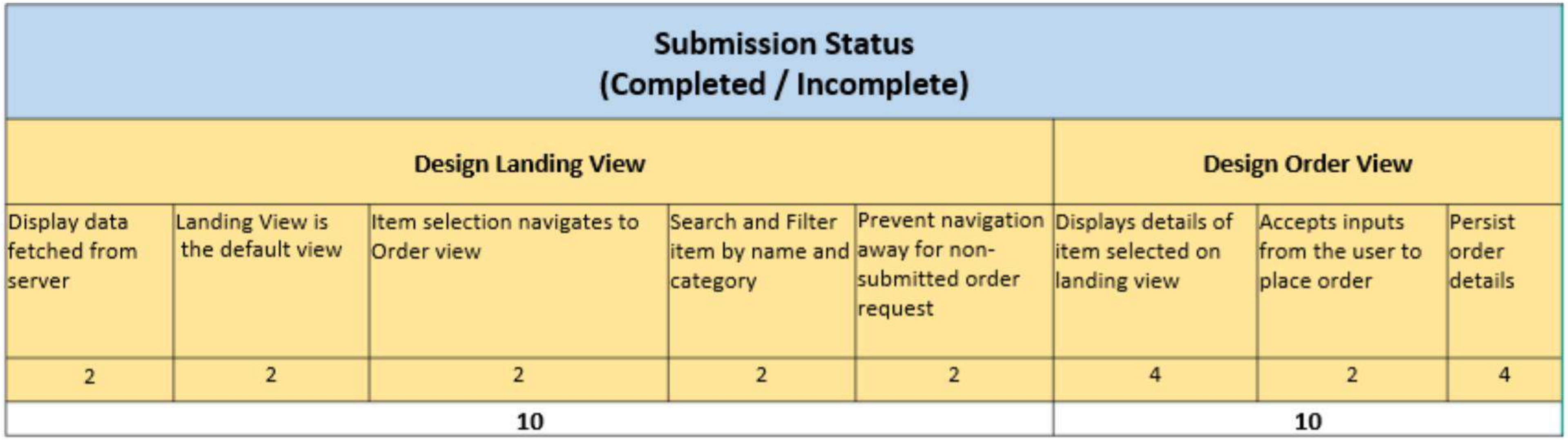
Instructions for Project

- Click here for the boilerplate.
- Read the README.md file in the boilerplate for further instructions about the challenge.
- Fork the boilerplate into your own workspace.
- Clone the forked boilerplate into your local system.
- The boilerplate contains images of various dairy products.
- Copy the images in the solution code and use them in the project.
- Create Angular application and develop the solution for the requirements specified.
- Test the outcome and ensure it fulfills the stated requirements.

Evaluation Rubrics

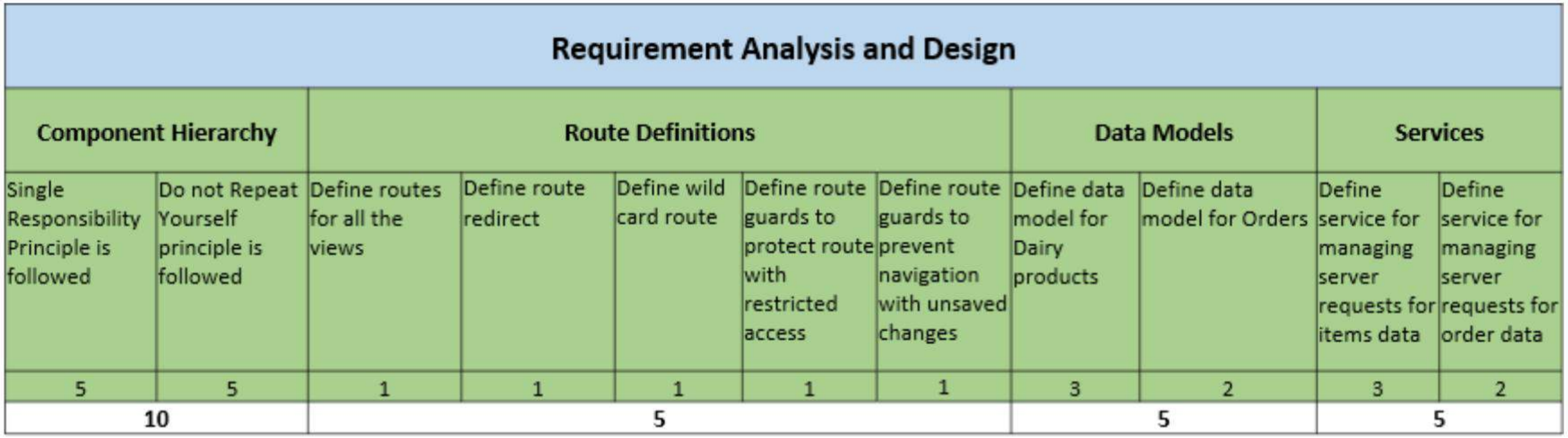
		Submission Status (Completed/ Incomplete)		Requirement Analysis and Design				Implementation						Code Quality			Remarks
Participant Name	Email	Design landing view	Design order view	Component Hierarchy	Route Definitions	Data Models	Services	Naming Conventions	Using Angular Material	Responsive Design	Form Validations	Error Handling	Functional Completeness	Well-Indented code	Adequately Commented code	Existence of unused variables	
		10	10	10	5	5	5	5	5	5	5	5	15	5	5	5	

Understanding Evaluation Rubrics - Submission Status



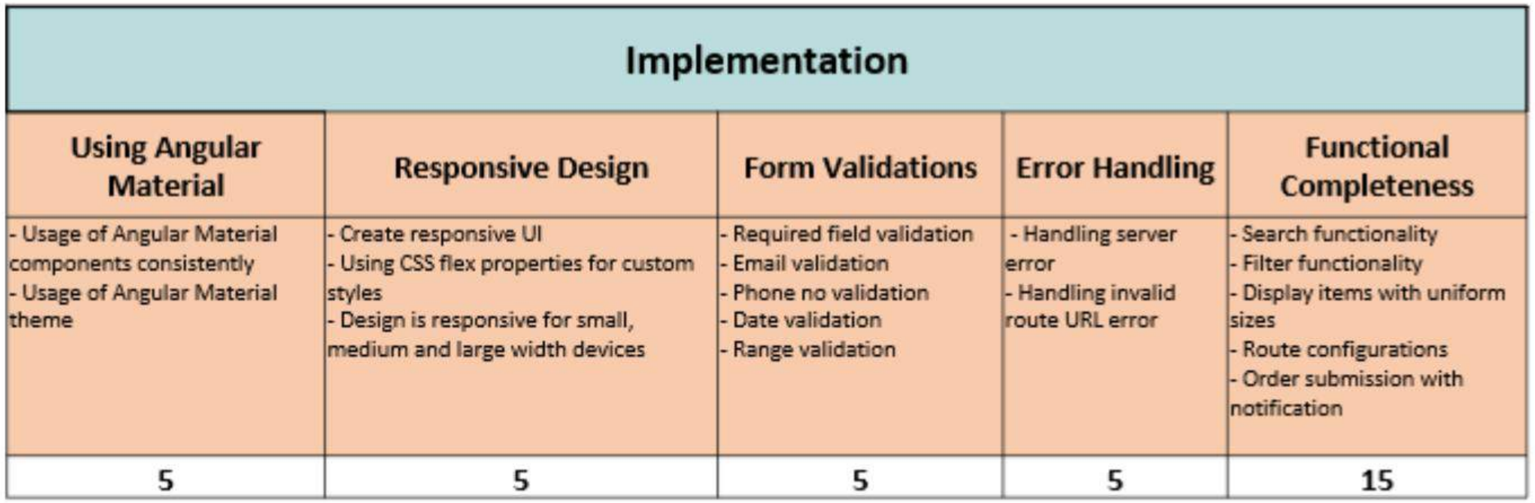
Submission Status (Completed / Incomplete)							
Design Landing View					Design Order View		
Display data fetched from server	Landing View is the default view	Item selection navigates to Order view	Search and Filter item by name and category	Prevent navigation away for non-submitted order request	Displays details of item selected on landing view	Accepts inputs from the user to place order	Persist order details
2	2	2	2	2	4	2	4
10					10		

Understanding Evaluation Rubrics - Requirement Analysis and Design



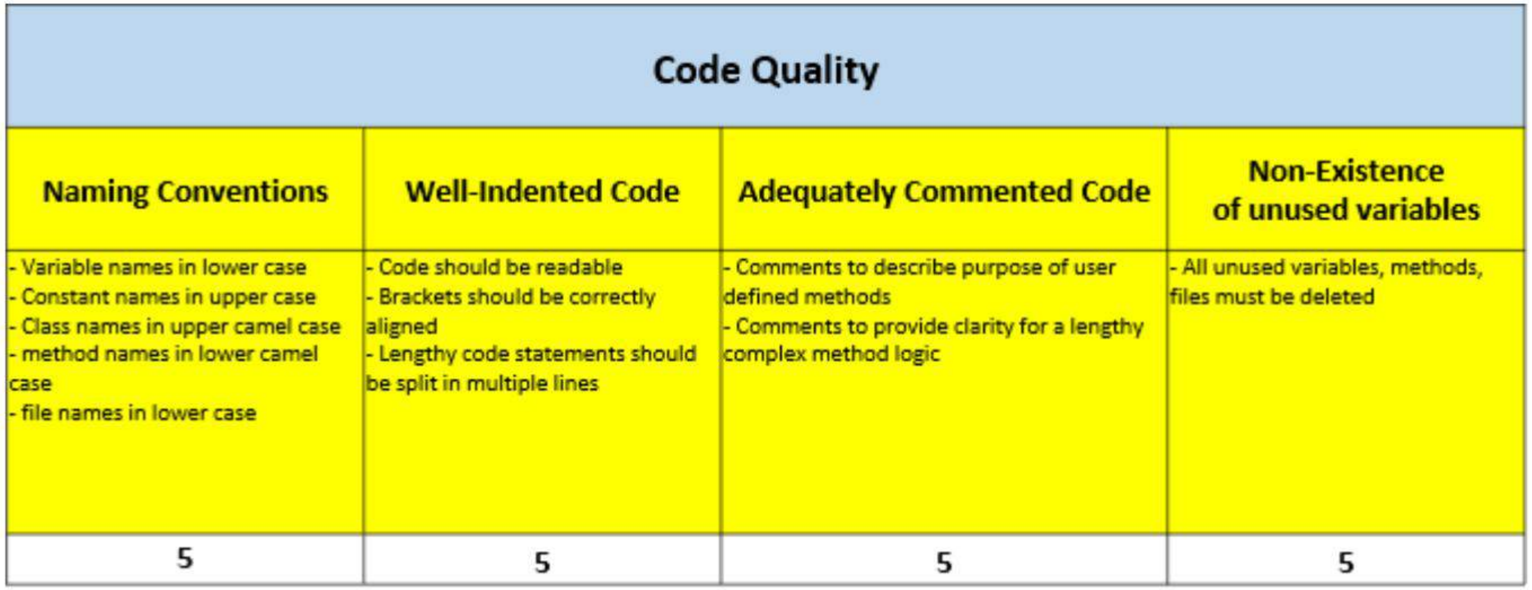
Requirement Analysis and Design										
Component Hierarchy		Route Definitions					Data Models		Services	
Single Responsibility Principle is followed	Do not Repeat Yourself principle is followed	Define routes for all the views	Define route redirect	Define wild card route	Define route guards to protect route with restricted access	Define route guards to prevent navigation with unsaved changes	Define data model for Dairy products	Define data model for Orders	Define service for managing server requests for items data	Define service for managing server requests for order data
5	5	1	1	1	1	1	3	2	3	2
10		5					5		5	

Understanding Evaluation Rubrics - Implementation



Implementation				
Using Angular Material	**Responsive Design**	**Form Validations**	**Error Handling**	**Functional Completeness**
- Usage of Angular Material components consistently - Usage of Angular Material theme	- Create responsive UI - Using CSS flex properties for custom styles - Design is responsive for small, medium and large width devices	- Required field validation - Email validation - Phone no validation - Date validation - Range validation	- Handling server error - Handling invalid route URL error	- Search functionality - Filter functionality - Display items with uniform sizes - Route configurations - Order submission with notification
5	5	5	5	15

Understanding Evaluation Rubrics – Code Quality



Code Quality			
Naming Conventions	**Well-Indented Code**	**Adequately Commented Code**	**Non-Existence of unused variables**
- Variable names in lower case - Constant names in upper case - Class names in upper camel case - method names in lower camel case - file names in lower case	- Code should be readable - Brackets should be correctly aligned - Lengthy code statements should be split in multiple lines	- Comments to describe purpose of user defined methods - Comments to provide clarity for a lengthy complex method logic	- All unused variables, methods, files must be deleted
5	5	5	5