GENASYS

JOB LEVEL: Frontend Developer Assessment

GENASYS SOFTWARE SOLUTIONS

Document Issue: V1.1
April 2025

genasystech.com

Setup Instructions:

Requirements:

Angular CLI: 14

Setup Steps:

1. Create a new angular project:
   - 1.1. Use ng new frontend-assessment to create a new angular project.
   - 1.2. Ensure routing is enabled and choose SCSS for styling.
2. Install dependencies:
   - 2.1. Run npm install to install dependencies.
3. Code submission:
   - 3.1. Push your project to your GitHub repository or zip project.
   - 3.2. Share the repository/zip link upon completion.

Assessment Objectives:

1. Implement Routing

   1.1. Set up Angular routing to navigate between different views.

   1.2. Ensure that the routing module is properly configured with lazy-loaded modules where applicable.

2. Customer Management Page

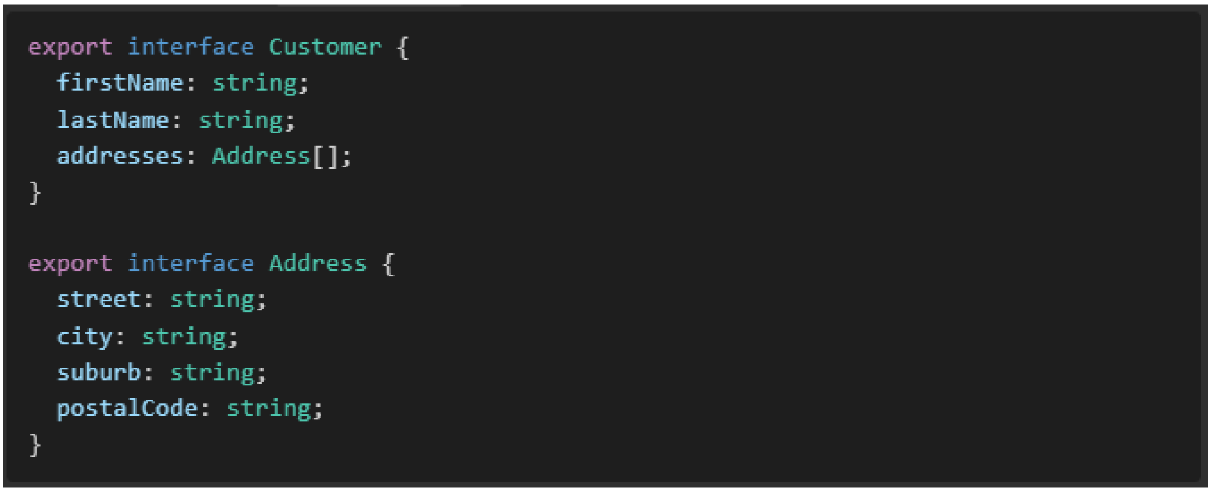
   2.1. Create a new Customer Management page that lists customer data in a table.

   2.2. Use Angular Material Table to display the data.

   2.3. Implement RxJS observables to fetch and subscribe to data from the NgRx Store.

   2.4. Load initial data from a json assets file based on the following interface:

      2.4.1. The file should contain similar data based on the below mock customer data interface example.



```
export interface Customer {
  firstName: string;
  lastName: string;
  addresses: Address[];
}

export interface Address {
  street: string;
  city: string;
  suburb: string;
  postalCode: string;
}
```

      2.4.2. Allow for the filtering and sorting of customer data inside of the table.

3. Quote Management Page

   3.1. Create a separate Quote Management page where users can view, create, and update insurance quotes associated with customers.

3.2. Define a Quote interface that includes fields such as quote ID, customer full name, quote amount, quote status, and created date.

3.3. Display the list of quotes in an Angular Material Table, allowing filtering by customer or status.

3.4. Implement navigation from the Customer Management Page to the Quote Management Page, showing only quotes related to a selected customer.

3.5. Ensure updates to quotes reflect correctly in the NgRx Store and UI.

4. Implement Add, Edit, and Delete Functionality

4.1. Implement functionality to add, edit, and delete customers using NgRx Store.

4.2. Use actions, reducers, and effects to manage state.

4.3. Ensure that operations update the store and reflect in the UI.

4.4. Allow modifying addresses by adding, editing, and deleting them within each customer entry.

4.5. Implement confirmation dialogs before deleting a record.

5. Handle Record Selection and Detail View

5.1. Clicking on a customer row should open a detail view with additional customer information.

5.2. Display the customer's list of addresses in the detail view.

5.3. Allow inline editing of customer and address details, persisting changes through NgRx Store.

Development Guidelines:

Mock Data Setup:

- Create services using Angular's Http Client to fetch mock data.
- Define appropriate data models representing the JSON structure.

NgRx Store Integration:

- Implement actions, reducers, selectors, and effects.
- Define actions for fetching, updating state, and handling errors.
- Configure effects for asynchronous operations (e.g., fetching mock API data).
- Reference: NgRx Documentation

RxJS Usage:

- Use RxJS observables for data changes.
- Dispatch actions to fetch data when the component initializes.
- Subscribe to data streams and update the component state accordingly.
- Unsubscribe from observables to prevent memory leaks.
- Reference: RxJS Documentation

Angular Material Integration:

- Use Angular Material components for UI consistency.
- Reference: Angular Material Components

Acceptance Criteria:

- Seamless integration with NgRx Store, RxJS, and Angular Material.
- Clean, readable, and well-formatted code (naming conventions, proper imports).
- Minimal console errors or warnings while navigating.
- Maintainability and best practices followed.
- Enjoy the challenge and have fun!