**Podcast Listing application**

### Document Properties

| Item | Details |
| --- | --- |
| Document | Architecture Design Document |
| Application | Podcast Listing Application |
| Author | Arpita Jain |
| Version | V1.0 |
| Creation Date | 09/7/2020 |
| Last Updated | 09/7/2020 |
| Status | Completed |

Contents

[Document Properties 1](#_Toc45989188)

[**1.** **Introduction** 3](#_Toc45989189)

[**2.** **Architecture Diagram** 3](#_Toc45989190)

[**3.** **Use-Case View** 4](#_Toc45989191)

[**4.** **Flow Chart** 5](#_Toc45989192)

[**5.** **Components** 6](#_Toc45989193)

[**6.** **Services** 6](#_Toc45989194)

[**7.** **Validations** 6](#_Toc45989195)

[**8.** **Tools and Libraries** 7](#_Toc45989196)

[**9.** **Unit Testing** 7](#_Toc45989197)

[**10.** **Responsive** 8](#_Toc45989198)

[**11.** **Accessibility** 8](#_Toc45989199)

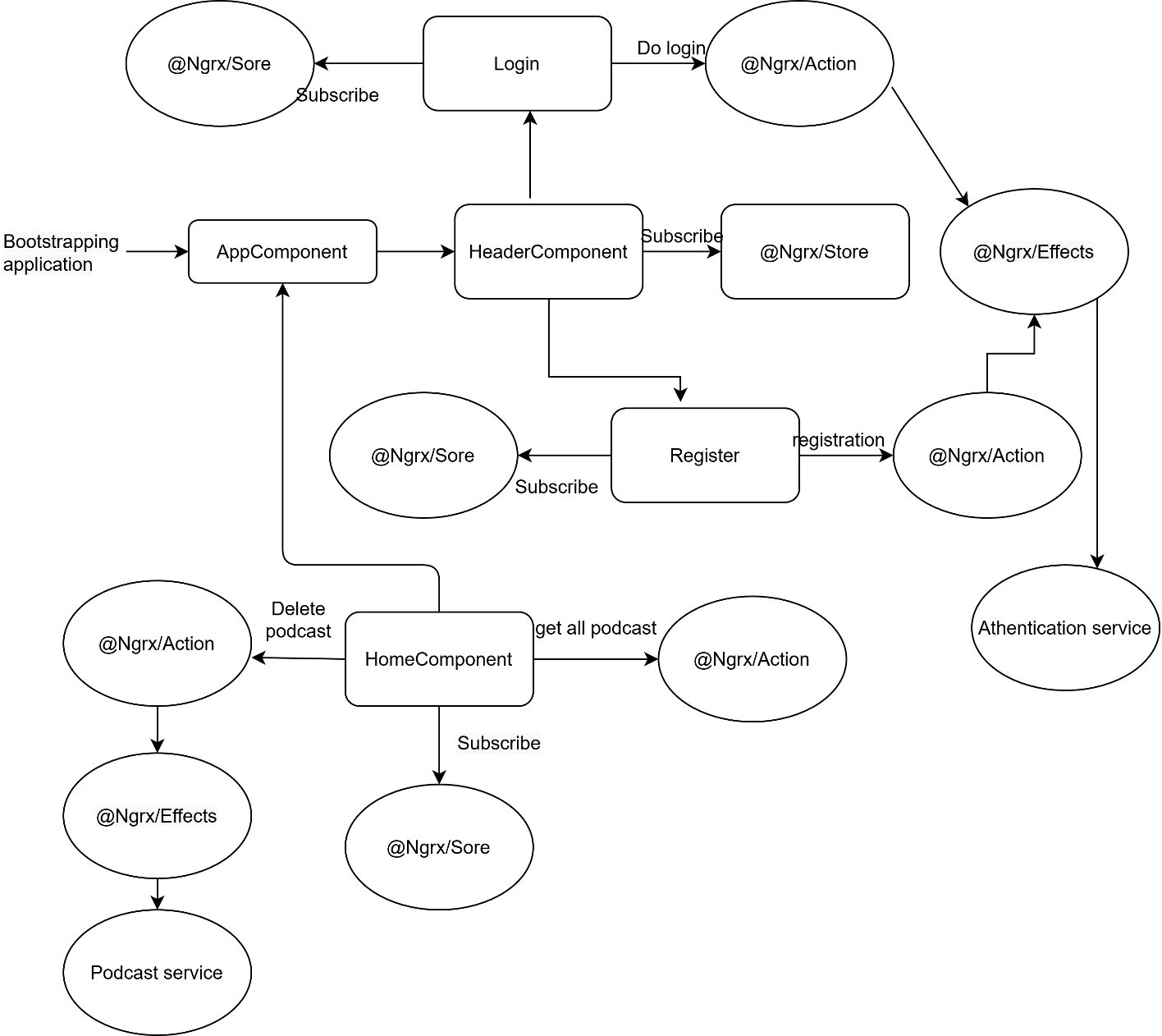
[**12.** **List other enhancement (at least top 3)** 9](#_Toc45989200)

[**13.** **Setup** 9](#_Toc45989201)

# **Introduction**

This document provides a comprehensive architectural overview of the application, using number of different architectural views to depict different aspects of the application.

# **Architecture Diagram**



# **Use-Case View**

The application use cases are -

* **Add Podcast** – It provide functionality to add Podcast, also having required validation with it.
* **Edit Podcast –** Any podcast can be editable while clicking on edit link.
* **Delete Podcast** – Each podcast will have a delete link on home page for respective user, the list podcast be deleted by clicking on it, it also confirms about the deletion of podcast, when user confirm by ok the podcast will be deleted.
* **Login** – If user is registered and the they can login with email and password.
* **Register** – User can register with their details including email and password,

These use cases are initiated by the user.

# **Components**

* AppComponent – This component is used for Application Bootstrapping
* HomeComponent – Home Page of application that show all the podcast and their respective details and if user is logged-In then it displays edit and delete feature and links.
* HeaderComponent – Header of application that show links for login, registration and home and after login it shows link to add podcast.
* AddEditPodcastComponent – This component contain form for add list and edit Podcast.
* LoginComponent – This component adds a functionality for login.
* RegisterComponent – This component adds a functionality for Register.

# **Services**

There are three services –

1. PodcastService – This service generally having logic about podcast in the application and this contain code for getting only podcast, edit podcast details, add new podcast and delete podcast logic.
2. AuthenticationService – This service generally having logic about login and registering users and to fetch their respective detail.
3. TokenService – This service generally adds and fetch token and respective data for logged in user in the application from the storage.
4. ThemeService – This service is related to theme; it stores CSS variables according to theme.

# **NGRX Store**

We have use Ngrx with creation actions, effects, reducers and selectors for state management.  
for auth and podcast we create separate action, reducer and effects file and create one single state for selector.

# **Route Guard**

We have created route guard CanActivate for accessing link, if state contain authentication to true, they the routs will be activated for logged in user.

# **Interceptor**

AuthInterceptor - We have created Interceptor for each request, after login each request contain token in its request header to verify user at backend.

ErrorInterceptor - We have created Interceptor for handling error, so for each request if any error come then it will throw that error.

# **Validations**

1) Application covered the required validation for podcast and login, registration form.

**Validation strategy:**

- Reactive form for validation

- There is corresponding html template for form elements

- Required attribute true for individual form element, this can be extended to have in a separate service file for re-usability.

# **Tools and Libraries**

- Angular version: 10.0

- TypesScript: 3.9

- Rxjs: 6.5

- Jest: 26

This is a unit testing framework; it consists assertion and mocking support for writing test cases.

# **Unit Testing**

Added Unit Test cases for services, guard, interceptors and components.

It consists of two services -

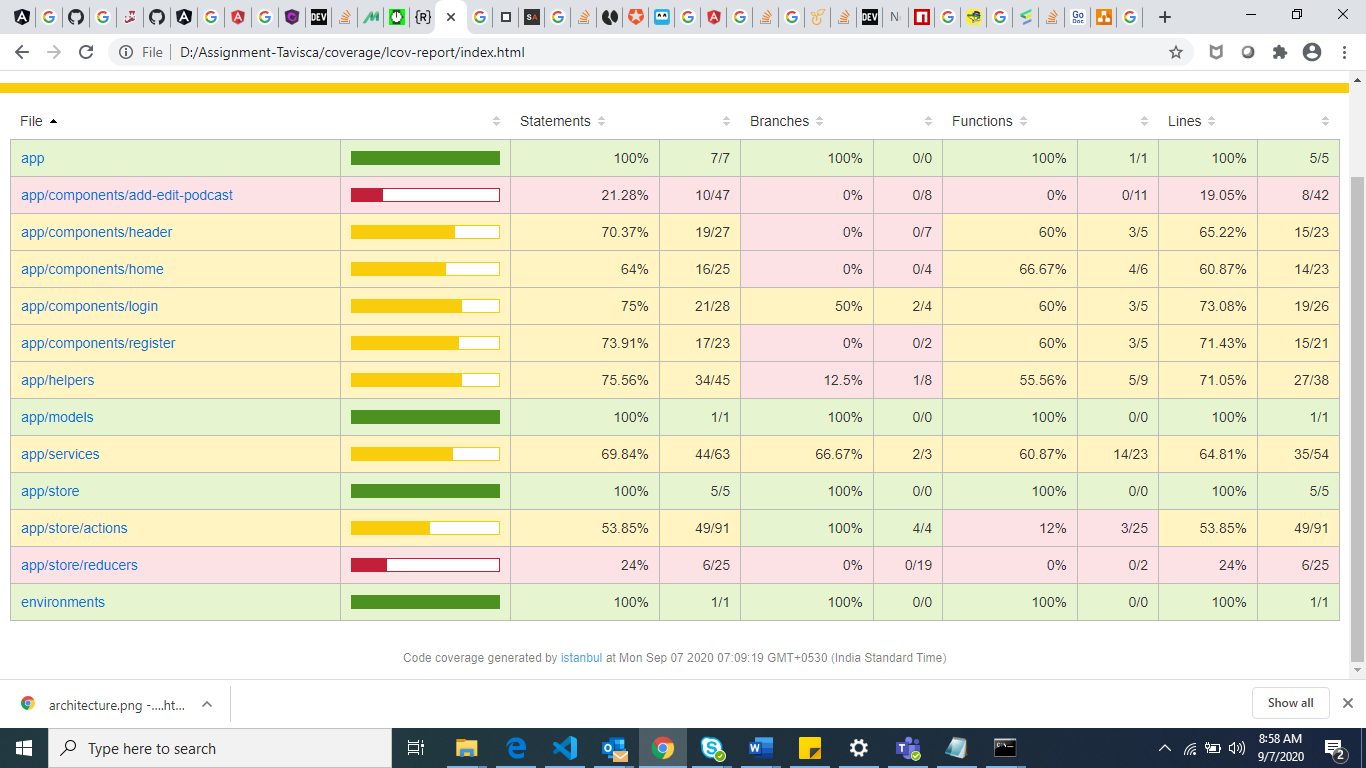
* + Podcast.service.ts – It contain test cases for the all the logic like add podcast, update podcast, delete podcast and get podcast.
  + Authentication.service.ts - It contain test cases for the all the logic like login and register and logout.
  + Auth.guard.ts – It contain test case for CanActivate route.
  + AuthInterceptor – It contain test case for interceptor.

Component related test cases are also included.

**Total Test Suites: 13**

**Total Tests: 26**

**Screenshot for unit test cases coverage -**



# **Responsive**

* The application is responsive with all the device and desktop size.
* No Horizontal scrollbar in application.

# **Accessibility**

For Accessibility, application having appropriate HTML tags that are preferred for accessibility

User can run application via keyboard and tab.  
We have included tabIndex and aria-label for all required html elements.

# **List other Open Items-**

1. Need to implement dynamic loading and webcomponents.
2. Need to add storybooks for application.

# **Setup**

1. Download and install node.js (version: 12.16)
2. Install npm and install @angular (version: 10.0)
3. Git clone ------------------------------ origin master
4. Navigate to cloned folder & run npm install
5. Run ng serve

For testing use -

npm run test

For Coverage use -

npm run test-watch