Booking App
Software Design Document

Class:

Advanced Software Engineering – Spring 2018

Instructor:

Patrick Mckee

Table of Contents

1. Preface

The purpose of this document is to showcase the design of the restaurant BookingApp. The reason that we decided to make the application is because there are many different services that let you have food delivered to you like GrubHub and Uber eats. However, there is only one service that lets you make a reservation at a restaurant which is how this project came to be. BookingApp would allow reservations for specific tables at a restaurant. The contents below will showcase the booking app in various ways ranging from a general overview of the system to how the system can be evolved over time.

2. Introduction

BookingApp is a web-based application that is composed of frontend and backend components. The frontend components consist of: home, login, restaurant, reservation, and table. Each one of them interact with one or more services that communicate with firebase backend. Home is used to display the home screen where users define their selection. Login is used to register an user or log in. The table module is used to display the available tables at a restaurant based on a given criteria. It also adds the user's reservation to the database.

3. System Overview

When designing the Restaurant BookingApp we wanted it to be available on a web-based system that way it would be easy for the users to access whether they be owners or customers. From a customer experience with the software, the customer would first input their information which would consist of their party size, the time and date they wanted their reservation, and their location. After that, the user would be able to see the restaurants that are in their area with tables available for that time. Next the user would be asked to either create an account or login to the site to be able to confirm the reservation. Finally, after they have either logged in or signed up they would be able to finalize their reservation.

we decided to use the Angular framework, which enforces the use of Firebase as the database. Firebase is a NoSQL realtime database with built-in authentication. Since it is a Google product, they conveniently host it. The reason we decided to use Angular is because there is plenty of documentation online, although less for newer versions. Other benefits are that it allows developers to quickly build components in a manner similar to Ruby on Rails and uses typescript which allows more object oriented like development, vice javascript.

4. System Design

Development Tools

Tools that used in the development of the application are: Git and Github for code repository hosting, Windows Visual Code and Webstorm for editing the code, Google Hangouts for group messaging, TSlint for static analysis checks, NPM for package management, and Google Chrome for testing the website.

Design methods and standards

Our goal is to follow the single responsibility principle. Modules, Components and Services should have a well defined responsibility.

Function documentation

Documentation will be generated using an Compodoc, an npm package. It has been deprecated and only used around the Angular 2 days but it still works for later Angular version projects. It generates a graphical overview of the system, all its modules, classes and services, which can be looked into detail.

Known Bugs

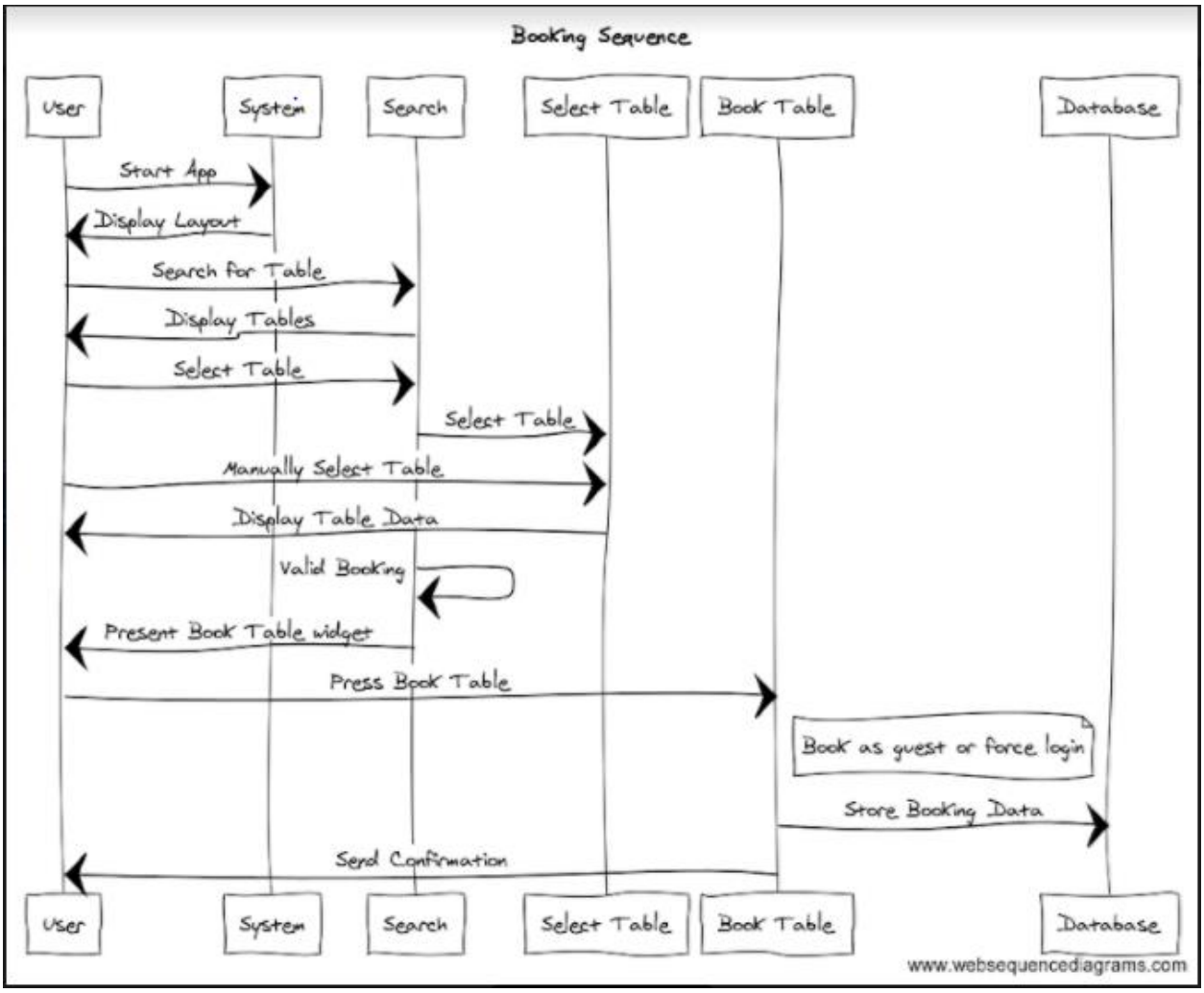
As of right know there are no known bugs that we are encountering we are expecting to find these when we enter and progress through the maintenance portion of the design process.

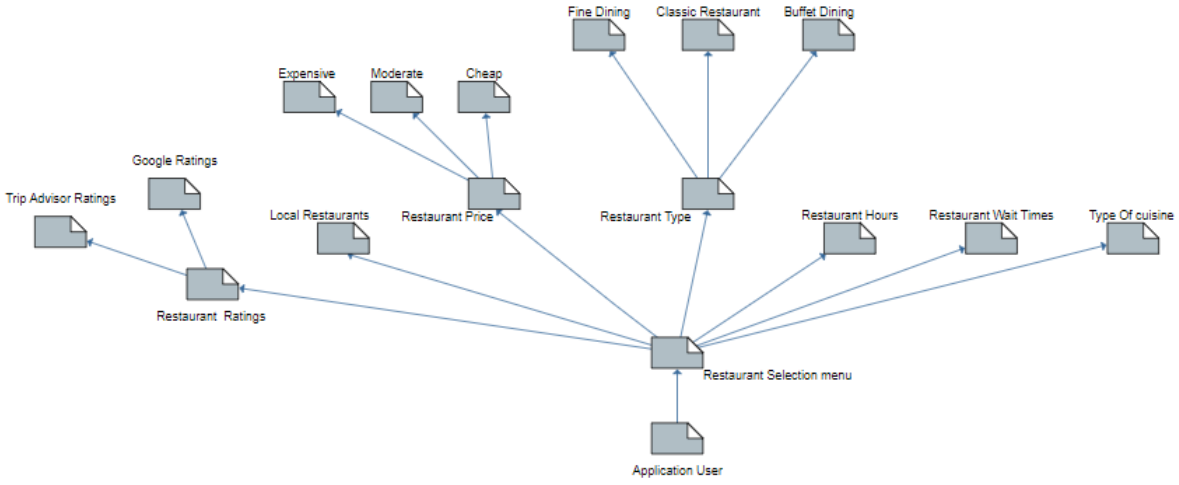
Programming standards

Developers are the following out of Microsoft's coding standards for contributors to Typescript:

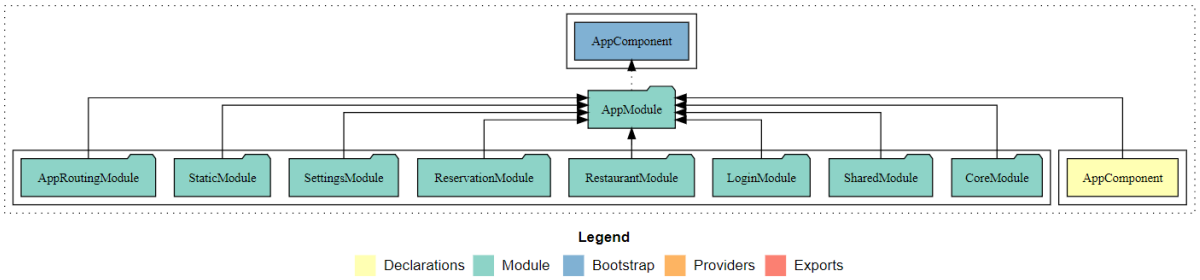
- Names
    - Use PascalCase for type names.
    - Do not use "I" as a prefix for interface names.
    - Use PascalCase for enum values.
    - Use camelCase for function names.
    - Use camelCase for property names and local variables.
    - Use whole words in names when possible.
    -
- Components
    - 1 file per logical component (e.g. parser, scanner, emitter, checker).
    - Do not add new files. :)
    - files with ".generated.*" suffix are auto-generated, do not hand-edit them.
- Types
    - Do not export types/functions unless you need to share it across multiple components.
    - Do not introduce new types/values to the global namespace.
    - Shared types should be defined in 'types.ts'.
    - Within a file, type definitions should come first.

5. Component Design

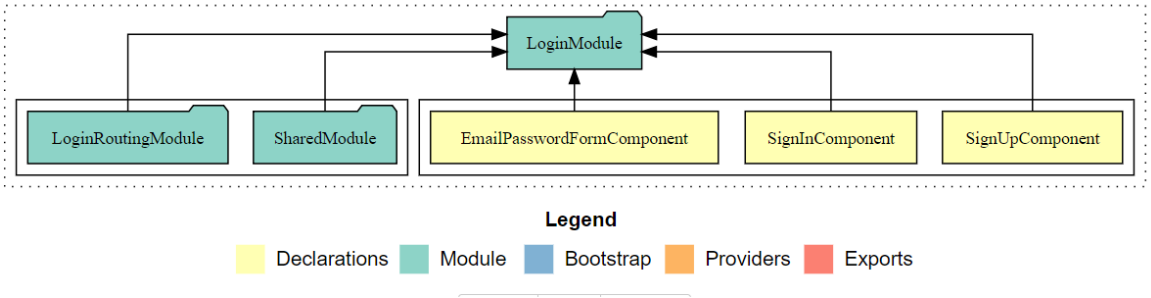

Booking Sequence
User
System
Search
Select Table
Book Table
Database
Start App
Display Layout
Search for Table
Display Tables
Select Table
Select Table
Manually Select Table
Display Table Data
Valid Booking
Present Book Table widget
Press Book Table
Book as guest or force login
Store Booking Data
Send Confirmation
User
System
Search
Select Table
Book Table
Database
www.websequencediagrams.com


Fine Dining
Classic Restaurant
Buffet Dining
Expensive
Moderate
Cheap
Google Ratings
Trip Advisor Ratings
Local Restaurants
Restaurant Price
Restaurant Type
Restaurant Hours
Restaurant Wait Times
Type Of cuisine
Restaurant Ratings
Restaurant Selection menu
Application User

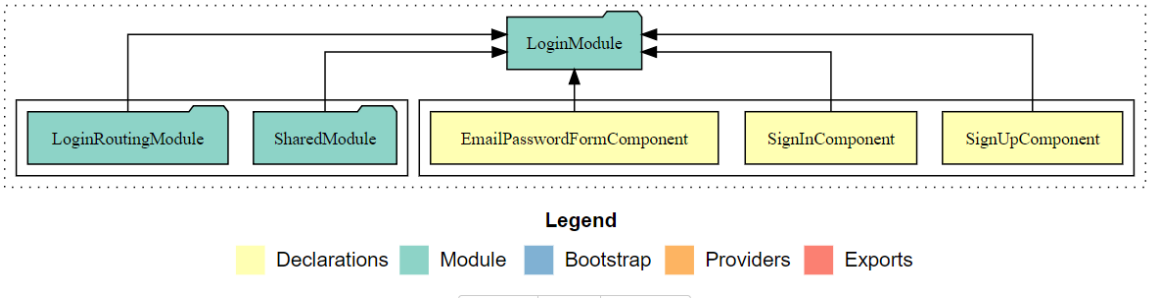
5.1 App Module



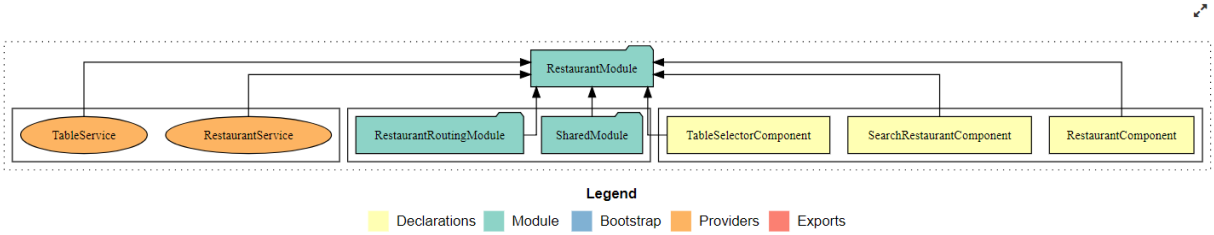
5.2 Login Module



5.2 Reservation Module



5.2 Restaurant Module



Full class documentation is available in html format.

6. System Evolution

Since we couldn't fully implement the restaurant selection menu in the way we wanted to, a way the system can evolve is that tables can chosen in an interactive manner where the user is shown the available tables and then clicks to enter the reservation. Currently, they can only click from a list of tables. With the BookingApp there are several other things that can be done to help the system grow and evolve. The first this would be to allow the addition of restaurants by business owners. Another could be to let the owner/manager customize the layout of the restaurant, for events like the Super Bowl or holidays. Another thing that can be done to grow the system is to have the app make suggestions on new restaurants based on previous restaurant selections.

7. Appendices - ToDo

8. Index - ToDo