POLITECNICO
MILANO 1863

DESIGN DOCUMENT

Massimiliano Sica 10558133
Carlo Pezzoli 10500665

February 10, 2022

Contents

1 Introduction

1.1 Purpose

The aim of this project is reporting the logic behind the design of ChefEasy, a cross-platform mobile application developed in React Native that allows registered users to connect with one another in order to offer catering services. The document here reported will cover the main design choices taken in order to implement the above application and provide some UML visualizations.

1.2 Scope

1.2.1 Definitions, Acronyms, Abbreviations

- Chef : person registered in order to offer services to other users
- User: unregistered person willing to get in touch with a chef
- Registered User: person registered in order to be able to leave reviews and upgrade to chef
- Filters: a series of elements used in a search by the user in order to express preferences
- Public page: page containing information about the chef

1.2.2 Description of the given problem

ChefEasy is an application-to-be for phones and tablets that allows chefs and users to get in touch and exchange services. Users will be able to choose a distance in kilometers and a cuisine type to visualize a list of all the chefs in that area which satisfy the given conditions. After choosing, the user will be able to visualize the public profile of the chef where all the basic information, social media links and main dishes pictures are shown. Registered users in addition to all the operations listed above can also leave scores to the chefs and become chefs themselves. A chef can create a public page containing contact information, pictures, a brief description and social media links.

1.2.3 Goals and functional requirements

In order to provide a more complete description of the goals, we decided to split them according to their reference actors namely users/registered users and chefs.In this section we list the functional requirements associated to each goal.
User and Registered User:

G1 : **The users and registered users should be able to find the chef that is most suited for their needs**

R1 : The system must show in the home page a map containing an overview of all the chefs in the area

R2 : The system must provide the user and registered user a list of chefs that satisfy the provided filters (distance from current position, type of cuisine and name)

R3 : The system must show, for the selected chef, the public page

G2 : The user should be able to register and become a registered user

R1 : The system must allow the user to register creating a new account requiring name, last name, email, phone number, city, photo and password.

G3 : The registered user should be able to upgrade to chef anytime

R1 : The system must allow the registered user to create a public chef page (see chef section)

G4 : The registered user should be able to leave an evaluation over a chef's performance from 0 to 5 stars

R1 : The system must allow the user to publish a score on the chefs public page after receiving authorization from the chef

G5 : The registered user should be able to delete their profile anytime

R1 : The system must allow the registered user to delete all of their data

G6 : The registered user should be able to keep up to date their profile

R1 : The system must allow the registered user to update name, last name, city, phone number, photo and password anytime

Chef:

G1 : The chef should have a public profile

R1 : The system must allow the chef to insert a description

R2 : The system must allow the chef to add links to Instagram, YouTube and Twitter

R3 : The system must allow the chef to add up to 4 representative pictures of their dishes

R4 : The system must allow the chef to choose a cuisine macro area (European, Asian, North American, South American, African and Fusion)

R5 : The system must allow the chef to change their profile image

G2 : **The chef should be able to delete their profile anytime**

R1 : The system must allow the chef to delete all of their data

G3 : **The chef should be able to update all their info**

R1 : The system must allow the chef to update password, pictures, social media links, description and phone number.

G6 : **The chef should be able to downgrade to user**

R1 : The system must allow the chef to downgrade to user and delete their public profile

1.3 Product Perspective

This section is devoted to providing an explanation of the external interfaces and of their interaction. This product is not supposed to be a follow-on of another project or device. Our software is thought as an independent, first of a kind implementation developed from scratch. The system is going to relate with the user/chef, which is able to access it through a graphical interface. Concerning hardware and software, the app is going to run on the two main operating systems, namely iOS and Android (latest versions). It furthermore requires an adequate Internet connection (3g,4g) to be able to interact with the system. GPS is required as well, because the app retrieves the position of the users, registered users and chefs in order to allow for more specific searches and visualizations.

1.4 Product Functions

The software is meant to allow people to get in touch with chefs (either professional or not) that will cook and serve them during an organized event.The users can perform searches based on their position and can register in order to leave reviews. The chefs, on the other hand, can advertise themselves and their cooking style in order to find new customers. They can upload pictures and link their YouTube channels and Instagram accounts to show more of their work.

1.5 User Characteristics

This section is devoted to provide a description of the actors in our scenario.

- User : person that is searching for a personal chef for an event, he can find him through the app's DB.
- Registered User : a user that went through the registration process and is able to leave reviews
- Chef : person that is offering his services to the users, they can set up and personalize his profile with photos,descriptions and social media links. They can act as a User anytime.

1.6 Assumptions, Dependencies and Constraints

1.6.1 Assumptions

D1 : Each registered user/chef can be identified unequivocally by the registration procedure.

D2 : All the actors involved in this application are assumed to behave honestly to one another. Hence a chef will always authorize a user that he worked for to leave a review.

1.6.2 Constraints

C1 : The application should not require registration in order to work

C2 : The application must be a mobile application, as it should be possible to use it from everywhere

C3 : The mobile application can only be installed on phones/tablets that have a camera, GPS positioning system and can access the Internet.

2 Specific Requirements

2.1 External Interface Requirements

In this section, we analyze Hardware and Software Interfaces.

2.1.1 Hardware Interfaces

The app is running on smartphones which must have:

- Working internet connection (3G/4G)
- Camera
- GPS
- Space for app package

2.1.2 Software Interfaces

Concerning the software interfaces, an updated version of the operating system is required, on the client device.
The application will interface with the Google Maps API.

3 Architectural Design

3.1 Component View

In this section we are showing the component diagrams for the main components namely : App, HomePage, ChefPage and SignUp We should give for granted that every component talks to HomPage, Search and WelcomePage by means of a navigation bar, for simplicity we are going to omit that from the diagrams.

- HomePage: With this component the user is able to quick search the preferred type of cuisine and to view a map showing their position and the location of the surrounding chefs.
- Search: With this component the user is able to do an in-depth search using as attributes any number of cuisine styles, the name and last name of the chef (if known in advance) and the desired distance.
- WelcomePage: With this component the registered user is able to log in to the application
- SignUp: With this component a generic user can become a registered user
- SignUpChef: With this component the registered user can sign up to become a chef
- ChefPage: With this component the chef can showcase their ability by uploading pictures, social media links and a description. Furthermore, from here the chef can authorize the registered user to leave a review. Plus allows the registered user to leave reviews to the chef if authorized.
- UserPage: With this component the user can customize their profile with personal information and images
- ChefPageUpdate: With this component the chef can update their Chef-Page info

In figure 1, we show the general structure of the application. From a central file called App.js we can access all the other components of the software since it is the main file responsible for the routing and for the display.

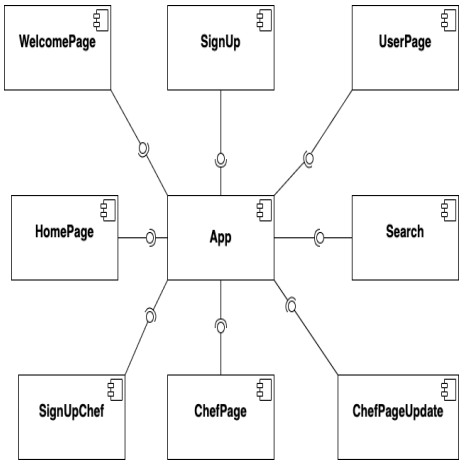

Figure 1: App

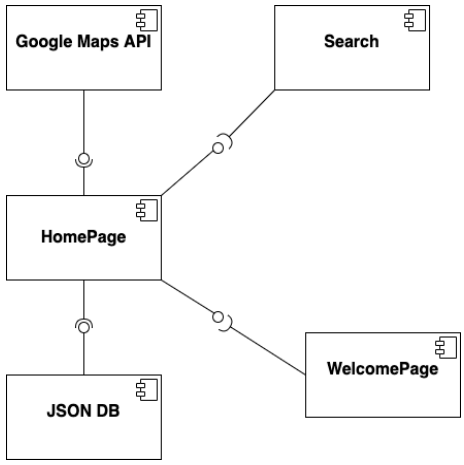

Figure 2: HomePage

In figure 2 instead, we can see the component diagram for the HomePage file. This file is the first that is shown to the user and in order to serve its functionalities it interfaces with the Google Maps API and our JSON database. In figure 3, we observe the component diagram for the search page and in Figure 4 for the Chef Page

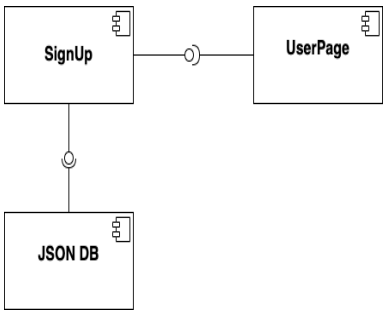

Figure 3: SignUp

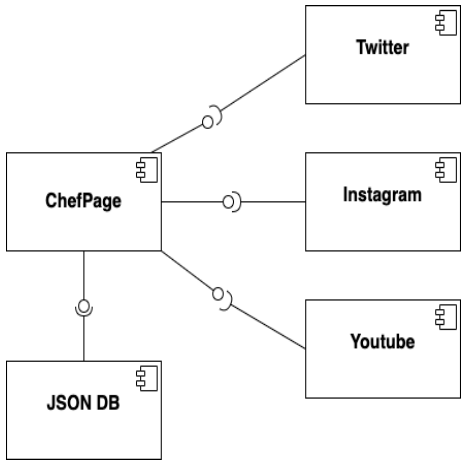

Figure 4: ChefPage

3.2 Deployment View

We can now move to the deployment section and understand how the different components are deployed on our devices. It is worth pointing out that for the purpose of this exam a mock back-end has been used (JSON database stored

in local). However in a real scenario we could consider deploying our back-end using a cloud service like AWS. In Figure 5 is shown a possible deployment where all the components listed in the previous section are mounted within the mobile phone of the user while the database for storing the user info is deployed on the cloud taking into account possible scalability and availability issues. We provide the app two endpoints: one for writing (much less common operation) and one for reading. The reading endpoint will point to a number of read replicas of the master database (accessed through the writing endpoint) that will scale horizontally depending on the number or read requests.

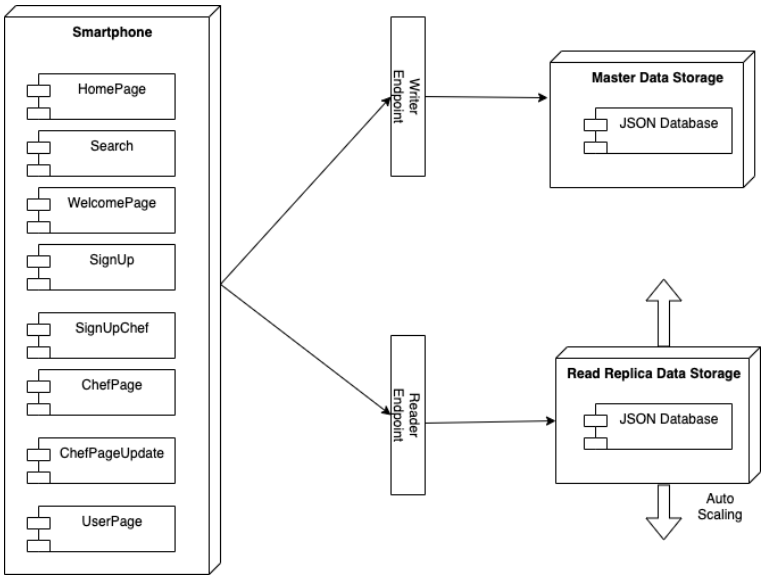

Figure 5: Deployment

3.3 Runtime View

In this section we explain how the different components of our system interact in the following scenarios:

- Chef search: the diagram shows a search for a chef cooking Asian food within 20 km from your position (Figure 6)
- Chef sign up: the diagram shows the procedure a user needs to take to become a chef (Figure 7)

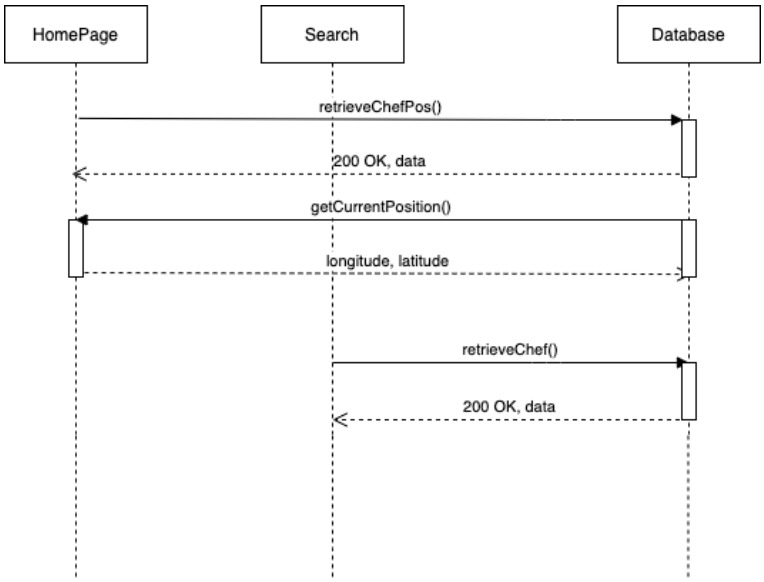

Figure 6: Chef Search sequence diagram

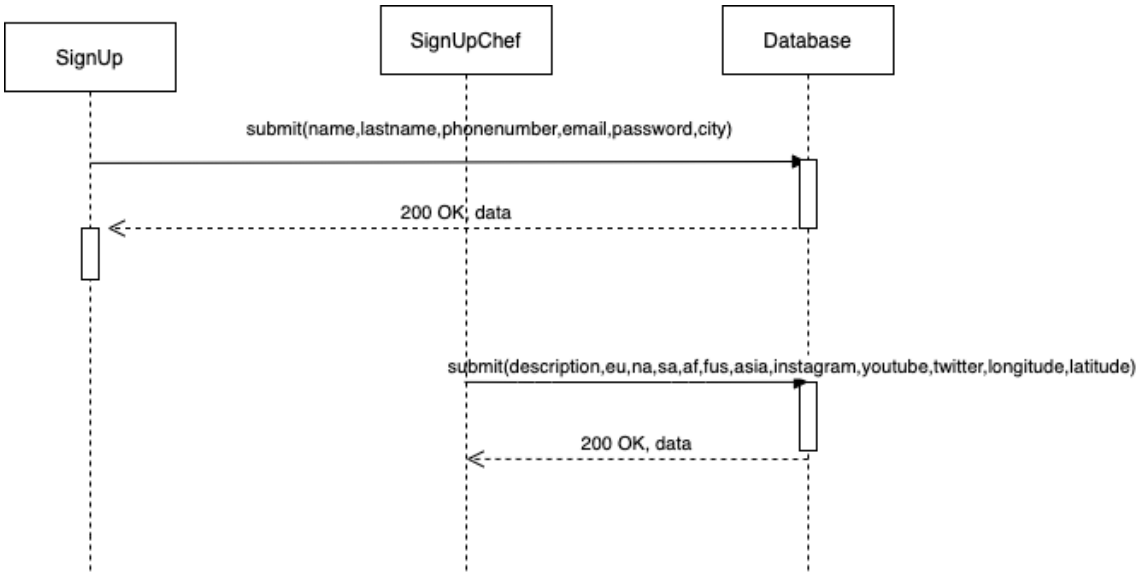

Figure 7: Chef Sign up sequence diagram

3.4 Selected Architectural Styles and Patterns

The applications is based on three layers :

- Data Layer: The application has a data layer, where the data is stored in a JSON server. For the purposes of the demo the server was stored in local, but in an actual implementation it would be stored in the cloud
- Logic Layer: The application has a logic layer which is responsible for querying the JSON database and the Google Maps API. All the queries are carried out using http by means of the JavaScript fetch API
- User Interface Layer: This layer is the one the user directly interacts with, it provides similar UI to users, registered users and chefs. The difference is that chefs are allowed to modify things in their ChefPage while the others are only allowed to look at it and, if authorized, leave a review.

3.5 User Interfaces

In this section all the user interfaces are shown on both ios and android for cellphone and tablet. In particular in Figure 31 we also show the authorization button that only the chef owning the profile can see in order to allow reviews.

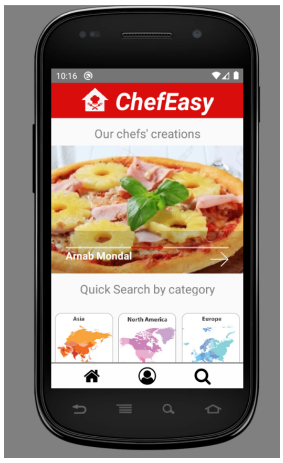

Figure 8: Android: Home Page

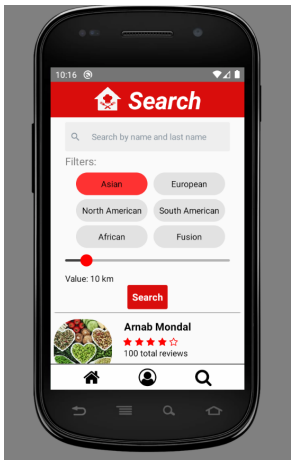

Figure 9: Android: Search Page

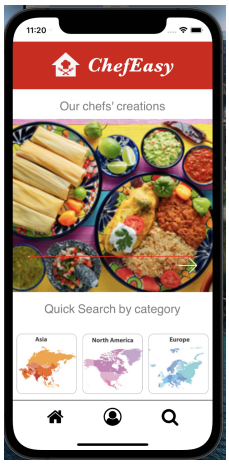

Figure 10: Ios: Home Page

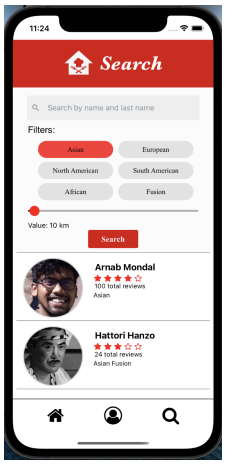

Figure 11: Ios: Search Page

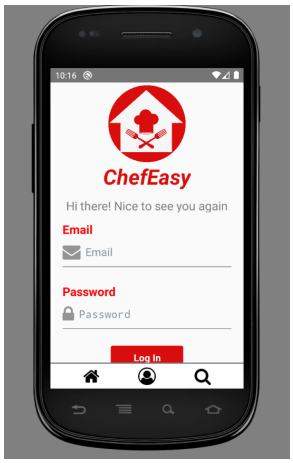

Figure 12: Android: Login

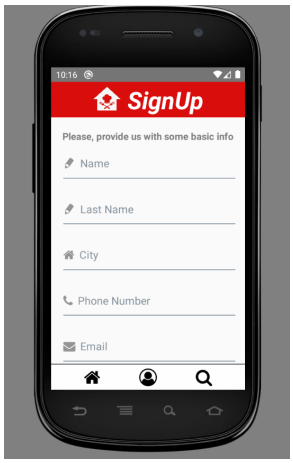

Figure 13: Android: Sign Up

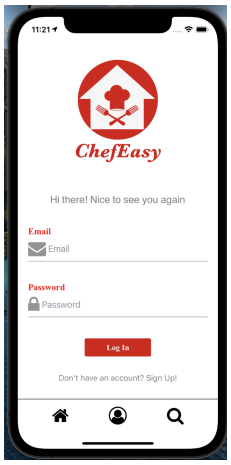

Figure 14: Ios: Login

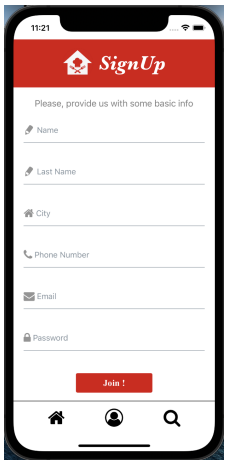

Figure 15: Ios: Sign Up

Figure 16: Android: User Profile

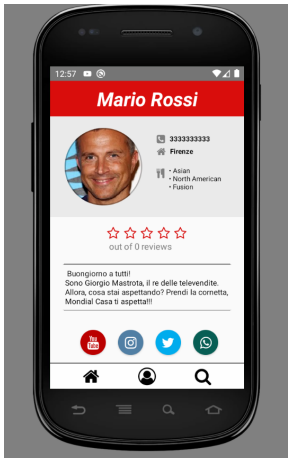

Figure 17: Android: Chef Home Page

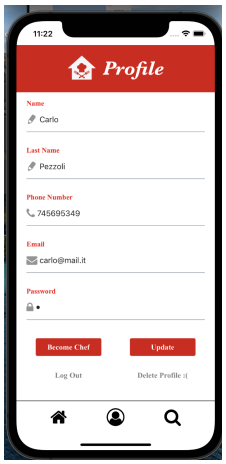

Figure 18: Ios: User Profile

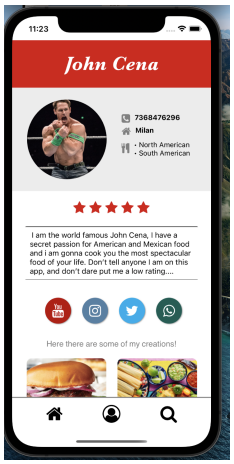

Figure 19: Ios: Chef Home Page

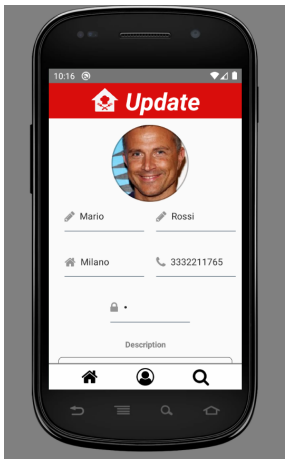

Figure 20: Android: Update Page Chef

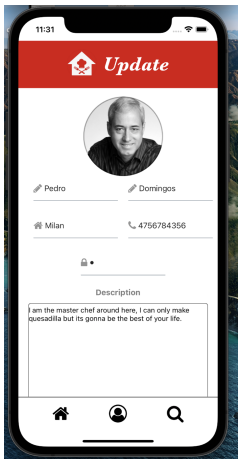

Figure 21: Ios: Update Page Chef

Figure 22: Android: HomePage

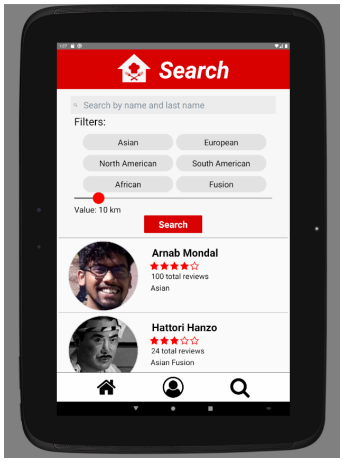

Figure 23: Android: Search Page

Figure 24: Ios: HomePage

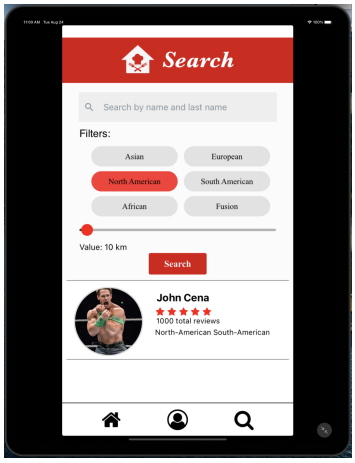

Figure 25: Ios: Search Page

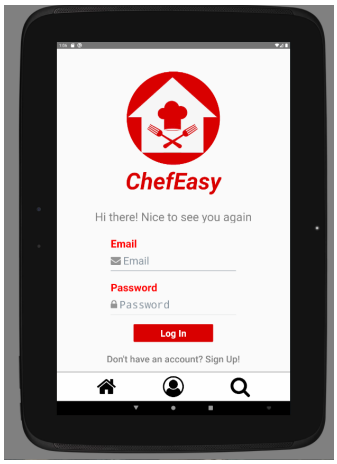

Figure 26: Android: Login

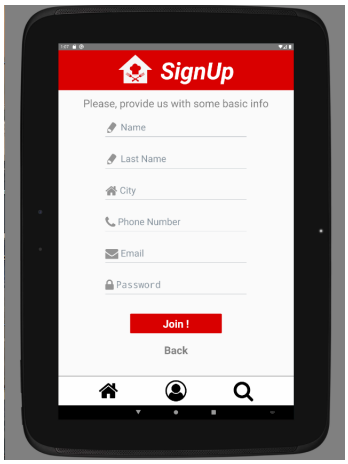

Figure 27: Android: Sign Up

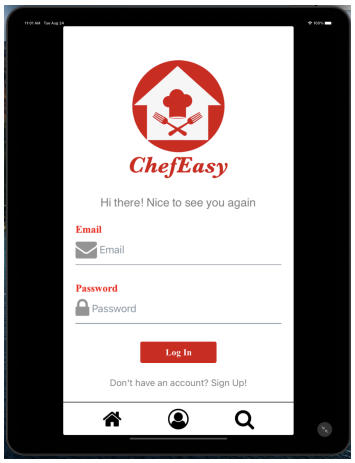

Figure 28: Ios: Login

Figure 29: Ios: Sign Up

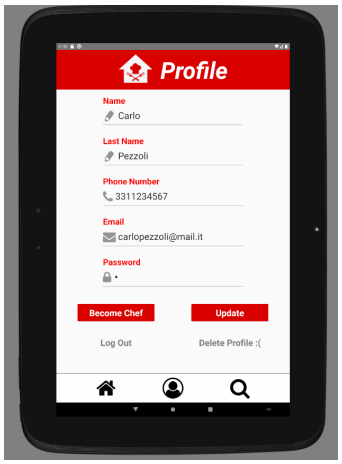

Figure 30: Android: User Profile

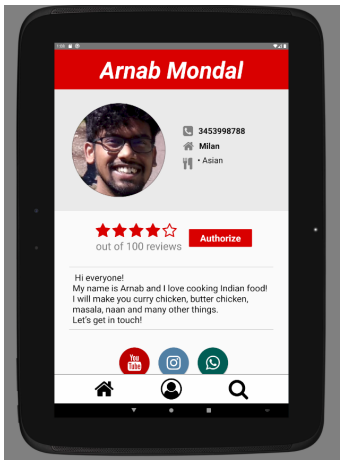

Figure 31: Android: Chef Home Page

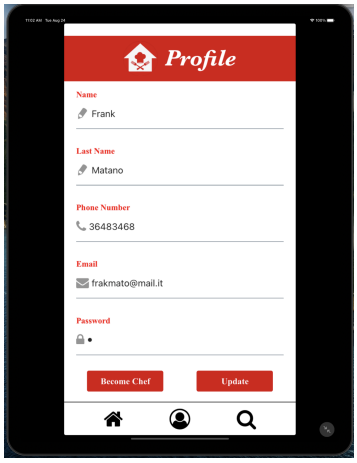

Figure 32: Ios: User Profile

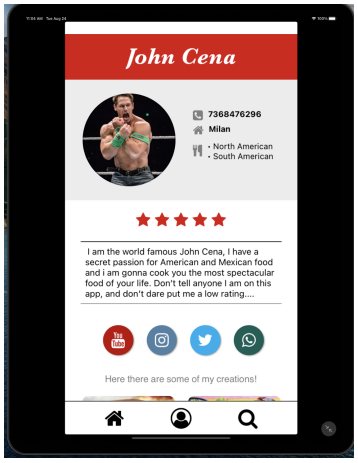

Figure 33: Ios: Chef Home Page

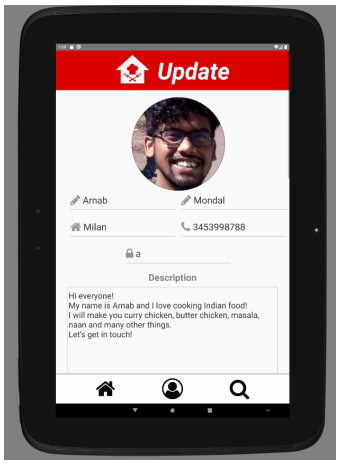

Figure 34: Android: Update Page Chef

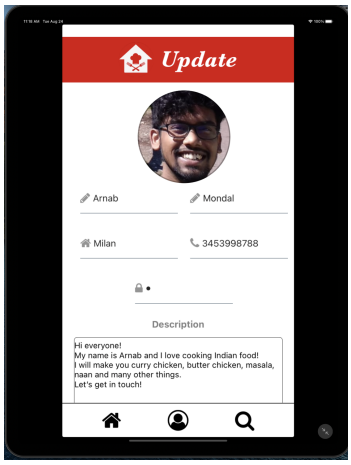

Figure 35: Ios: Update Page Chef

4 Testing

In this section we explain how we carried out testing for our application, we decided to use Jest since it is built in with react native. The very first thing we decided to test was the UI, since we wanted to make sure that it did not change unexpectedly. We then decided to proceed with snapshot testing for every component, namely:

- HomePage
- WelcomePage
- SignUp
- SignUpChef
- UserPage
- ChefPageUpdate
- ChefPage
- Search

The tests look like this:

```
import React from 'react';
import renderer from 'react-test-renderer';
import HomePage from '../views/HomePage';

test('renders correctly', () => {
  const tree = renderer.create(<HomePage />).toJSON();
  expect(tree).toMatchSnapshot();
});
```

Figure 36: Home Page Test