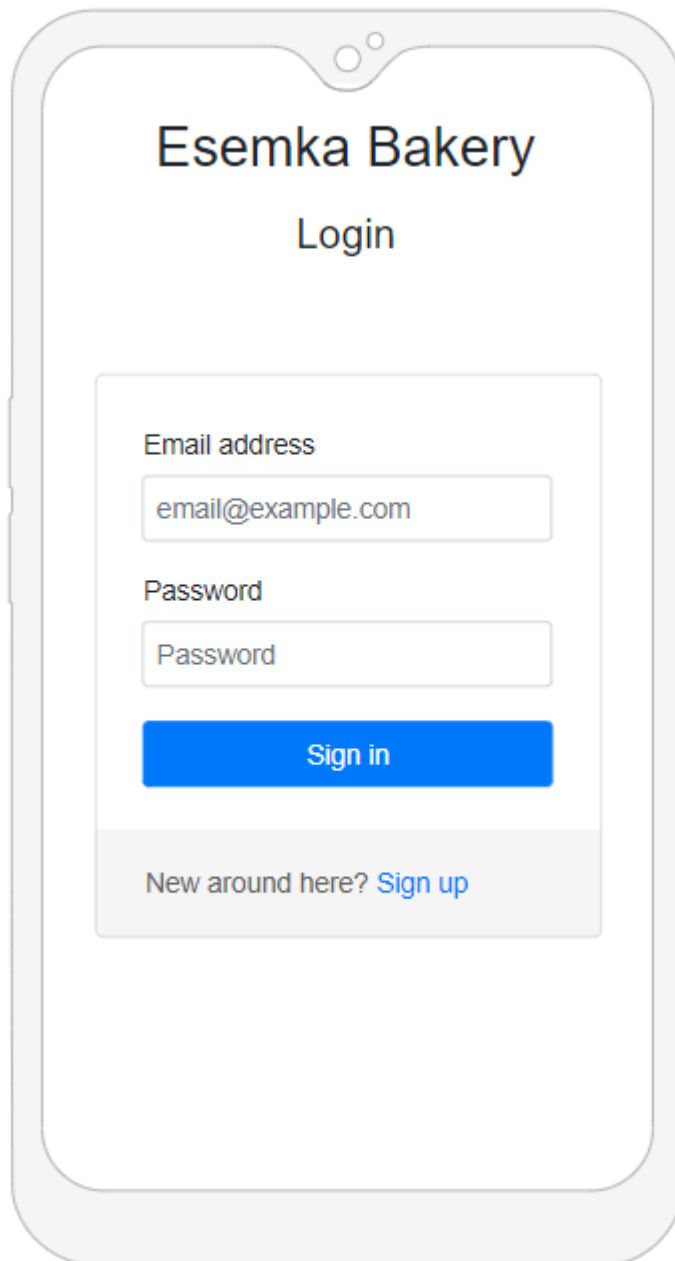


# Esemka Bakery

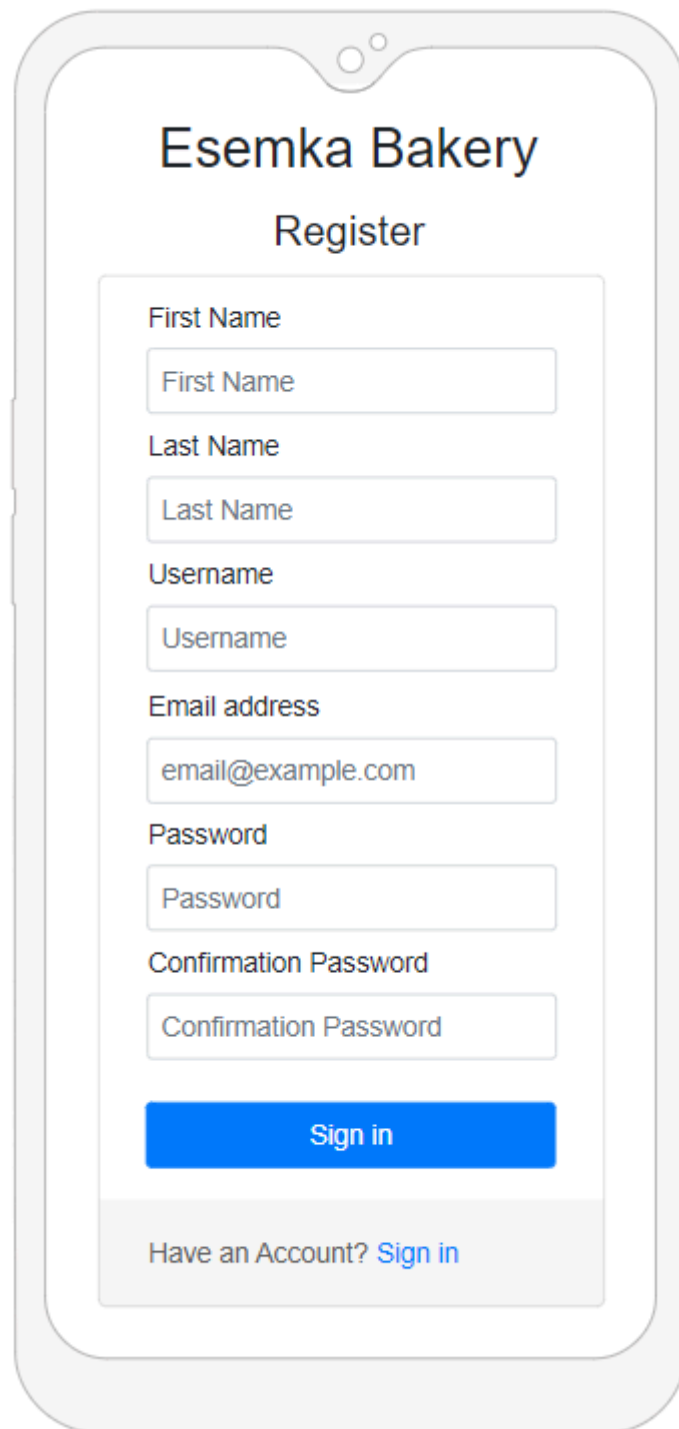
## 1. Login Page



The image shows a mobile application interface for Esemka Bakery. At the top, the app's name "Esemka Bakery" is displayed in a large, bold, black font. Below it, the word "Login" is centered in a smaller, black font. The main content area is a light gray rounded rectangle containing two input fields. The first field is labeled "Email address" and contains the text "email@example.com". The second field is labeled "Password" and contains the text "Password". Below these fields is a prominent blue button with the text "Sign in" in white. At the bottom of the form area, there is a link that says "New around here? Sign up" in a smaller, blue font.

In the login page, the user is required to fill in their email/username and password in the provided columns. When the user clicks the Sign In button, send the data using the **/Auth/Login** endpoint with the **POST** method. Redirect the user to the register page when they click the sign-up button.

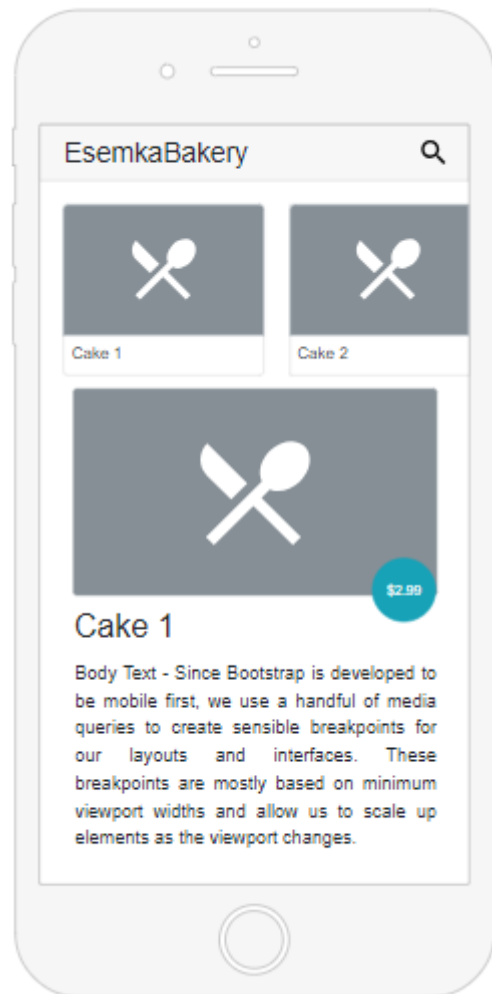
## 2. Register Page



The image shows a mobile application interface for a registration page. At the top, the app title "Esemka Bakery" is displayed in a large, bold, black font. Below the title, the word "Register" is centered in a slightly smaller, bold, black font. The registration form is contained within a light gray rounded rectangle. It features seven input fields, each with a label above it: "First Name", "Last Name", "Username", "Email address", "Password", and "Confirmation Password". The "Email address" field contains the placeholder text "email@example.com". Below the input fields is a prominent blue button with the text "Sign in" in white. At the bottom of the form, there is a light gray footer area containing the text "Have an Account? [Sign in](#)".

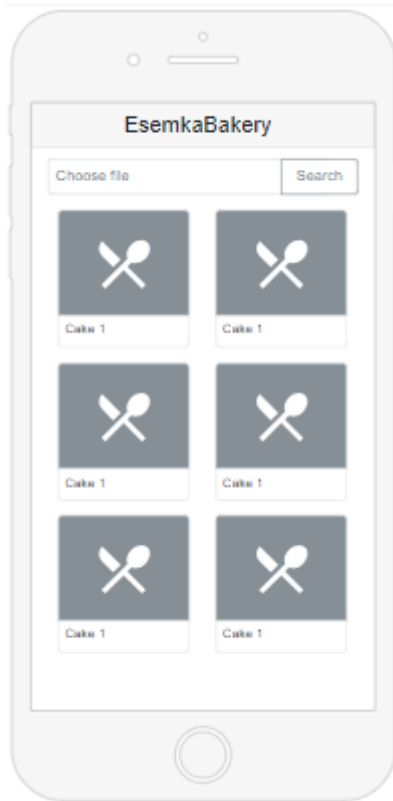
Users must fill in the fields on the registration page. Send the validated data when the Sign Up button is pressed and redirect to the Login page. Use the **/Auth/Register** endpoint with the **POST** method. Direct the user to the Login page when the Sign In button is pressed.

### 3. Home Page



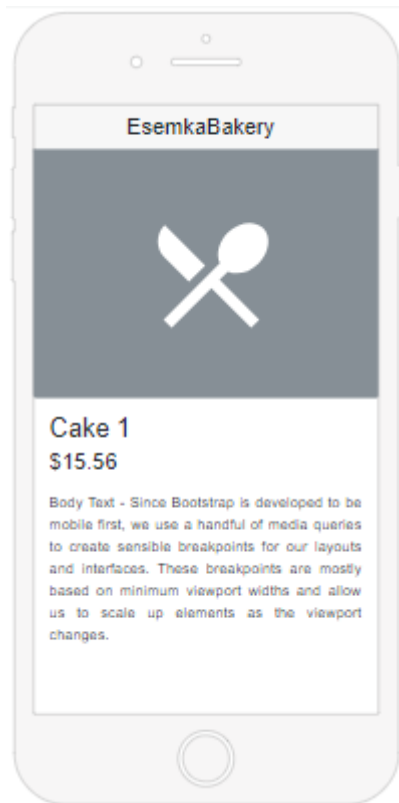
Redirect the user to the home page upon successful authentication. On the home page, display all cakes in a simple design by showing **images and names** at the top of the page. The cakes should be presented in a **horizontally scrollable list**. When the user clicks on one of the cakes, display its details below. Use the **/Cake** endpoint with the **GET** method to retrieve all cake data and the **/Cake/:id** endpoint with the **GET** method to retrieve detailed cake data. There is also a search button that will direct the user to the search page and an order now button to direct the user to the order page.

#### 4. Search Page



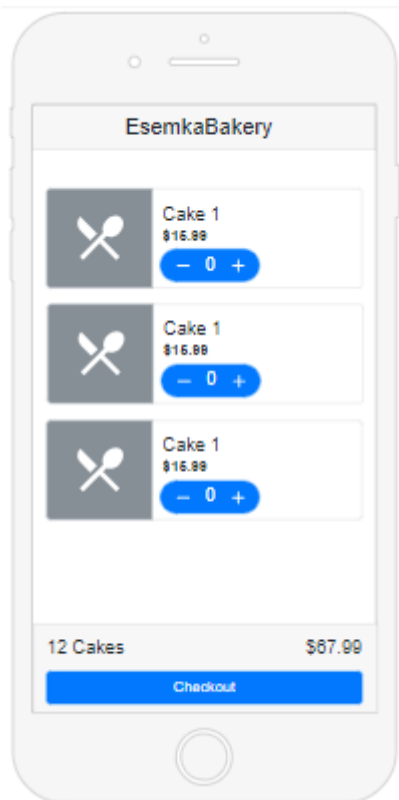
On the search page, users can search for cakes based on the **cake name**. Use the **/Cake?search=:search** endpoint with the **GET** method. Display cake data in a vertically scrollable list and in a **2-column layout**. When a cake data is pressed, redirect the user to the cake detail page.

## 5. Detail Page



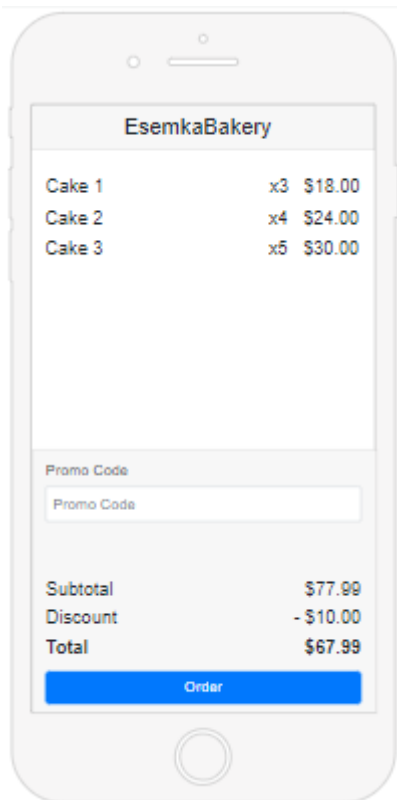
On the cake detail page, display the **image, name, price, and complete description** of the cake. Use the **/Cake/:id** endpoint with the **GET** method to retrieve cake data based on the cake's id.

## 6. Order Page



Users can order cakes by going to the order page and selecting the cakes they want to order. Users can choose **more than one cake with a quantity greater than 1 for each selected cake**. When a user selects a cake, **update the subtotal column according to the price and quantity** of the selected cake. Redirect to the checkout page when the user clicks the checkout button. Users can only proceed to checkout if there are selected cakes with **a quantity greater than 1**.

## 7. Checkout Page



Display all selected cakes on the checkout page. The information to be shown for each cake includes the **Cake Name, Quantity, and Subtotal Price**. At the bottom, there is a promo code column that can be filled in by the user. When the user enters a promo code, ensure its validity by fetching data from the API using the **/Order/PromoCode/:code** endpoint with the **GET** method. If the promo code is valid, apply a discount to the discount column and update the value in the total column. Also, display the subtotal of all selected cakes. When the user clicks the order button, send the order data to the API using the **/Order** endpoint with the **POST** method.

~ A general does not use the same troops over and over again. ~