Lazy Pizza Milestone #2 Requirements

This document describes the requirements for the first development milestone (Milestone #2) of the Lazy Pizza app.

The goal of this milestone is to implement the **Cart screen**, the **Order History screen** in an unauthorized state, and introduce the **Bottom Navigation Bar**.

The mockups define the app's appearance and behavior. Feel free to decide how you implement specific things (e.g., how you display a specific loading progress or how you specify error messages).

You can find the Lazy Pizza **mockups here**:

> https://www.figma.com/design/7EgItcq6KfY1Ct9H6uOitQ/Lazy-Pizza?node-id=4-12542

Milestone #2 Goal

- Implement a bottom navigation bar with three main screens: Menu, Cart, History.
- Build the Cart screen with a product list, order summary, and a "Recommended Add-ons" block.
- Create the Order History screen with a simple UI. All order data must be stored remotely and fetched by the app.
- Adapt the UI for both mobile and wide screens.

📡 Remote Data

All product catalog and order data must be **stored remotely** and **fetched dynamically** by the app.

> 📌 We recommend using **Firebase** as the primary backend solution. Other alternatives are possible, but be mindful of time constraints, as implementing custom solutions may require significantly more effort.

Comparison of Approaches: Firebase vs Custom Backend

In modern mobile app development, there are two common approaches to organizing the backend: using ready-made solutions such as **Firebase** or building a **custom backend**. Both options have their pros and cons, which should be considered when choosing an architecture.

Firebase

✅ Great to quickly get started and prototype

✅ Easy to scale

❗ Can become incredibly expensive when scaled (you pay for users, regardless of how profitable your app is - and small bugs causing plenty of requests can create cost spikes)

❗ Limited control and querying potential

Custom Backend

✅ Full control over everything from DB to storage to programming language

✅ Can be maintained for relatively little money in comparison to cloud providers (however, it's typical to stick to a hybrid model by having both a custom backend and some cloud providers, e.g. for image storage to get the best of both worlds)

❗ Steeper learning curve

❗ Scalability depends on how you structure it

Recommended Solution: Firebase

- We strongly recommend using **Firebase** for this project:
  - Firebase Storage for images
  - Firebase Firestore or Realtime Database for product data
- Firebase provides excellent Android integration, real-time capabilities, and is free for development purposes.

Icons

You may use Material Design icons where appropriate. If a suitable Material icon is not available, use the custom icons provided in the Figma mockups.

In Figma, any icon or image can be **exported** by selecting the element and clicking "Export" in the **right-hand panel**. In this panel, you can also choose the desired format (PNG, SVG, etc.).

Adaptive Layouts

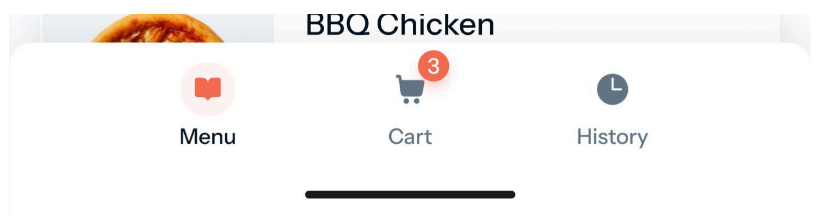
The app must support two breakpoints:

- **Up to 840 dp** → mobile layout.
- **From 840 dp and above** → wide-screen layout.

Both versions must be implemented as shown in the mockups.

Technical Requirements

🧭 Bottom Navigation Bar

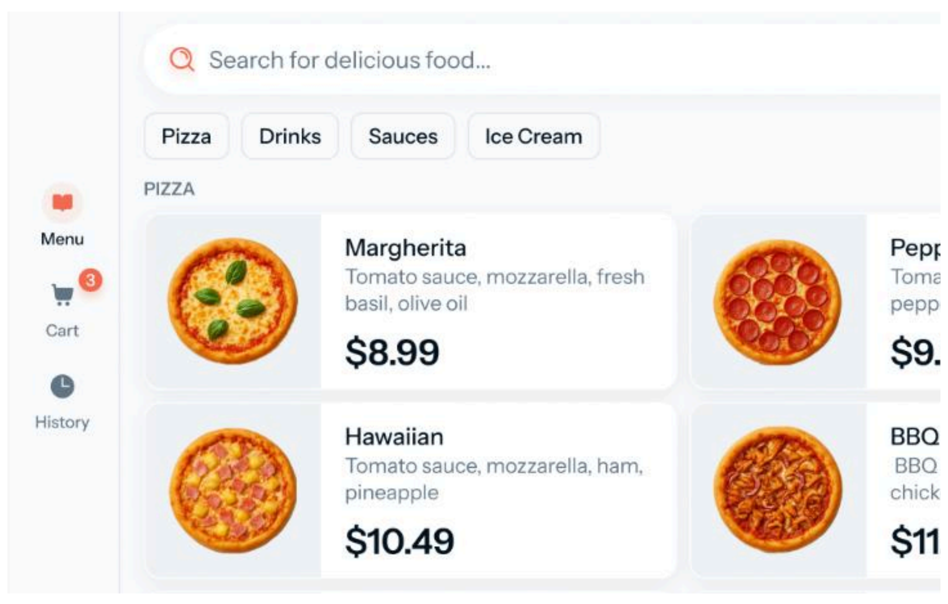


- Placed at the bottom of the screen **(up to 840 dp)**.
- Contains 3 tabs:
  - **Menu** — main product catalog.
  - **Cart** — cart screen.
  - **History** — order history screen.
- The active tab is highlighted.
- **Cart tab** behavior:
  - If items are added to the cart → a **badge with the number of items** appears in the top-right corner of the icon.
- A shadow is applied above the bar to visually separate it from the content.

BackStack behavior

- Switching between **Menu / Cart / History** clears the history stack.
- Each tab opens as a root-level screen (no history preserved between tabs).
- Pressing the **system Back button** on any root-level bottom navigation destination (Menu, Cart, or History) → exits the app.
- Inner screens (e.g., Product Details) are pushed onto the BackStack and return with Back as usual.

Wide-screen behavior (from 840 dp)


Search for delicious food...
Pizza
Drinks
Sauces
Ice Cream
PIZZA
Menu
3
Cart
History
Margherita
Tomato sauce, mozzarella, fresh basil, olive oil
$8.99
Hawaiian
Tomato sauce, mozzarella, ham, pineapple
$10.49

- Instead of the bottom bar, use a NavigationRail (Material component).
- Placed on the **left edge** of the screen.
- Contains the same 3 items: Menu, Cart, History.
- Icons are **centered vertically** within the rail.
- Active tab is highlighted in the same way as on mobile.

🍕 Product Details Screen (Updated)

- When the user taps the **Add to Cart** button, the app must **add the selected pizza** to the cart and then **return the user to the main Menu screen** so they can continue browsing other products.
- The added pizza must be stored in the cart with all selected toppings and quantities, ready to be displayed the next time the Cart screen is opened.
- The **Cart icon** in the Bottom Navigation Bar must update to show the **current number of items** in the cart as a badge in the top-right corner of the icon.

- This badge should dynamically reflect any changes — for example, when items are added or removed from the cart.

🛍️ Order Flow Behavior

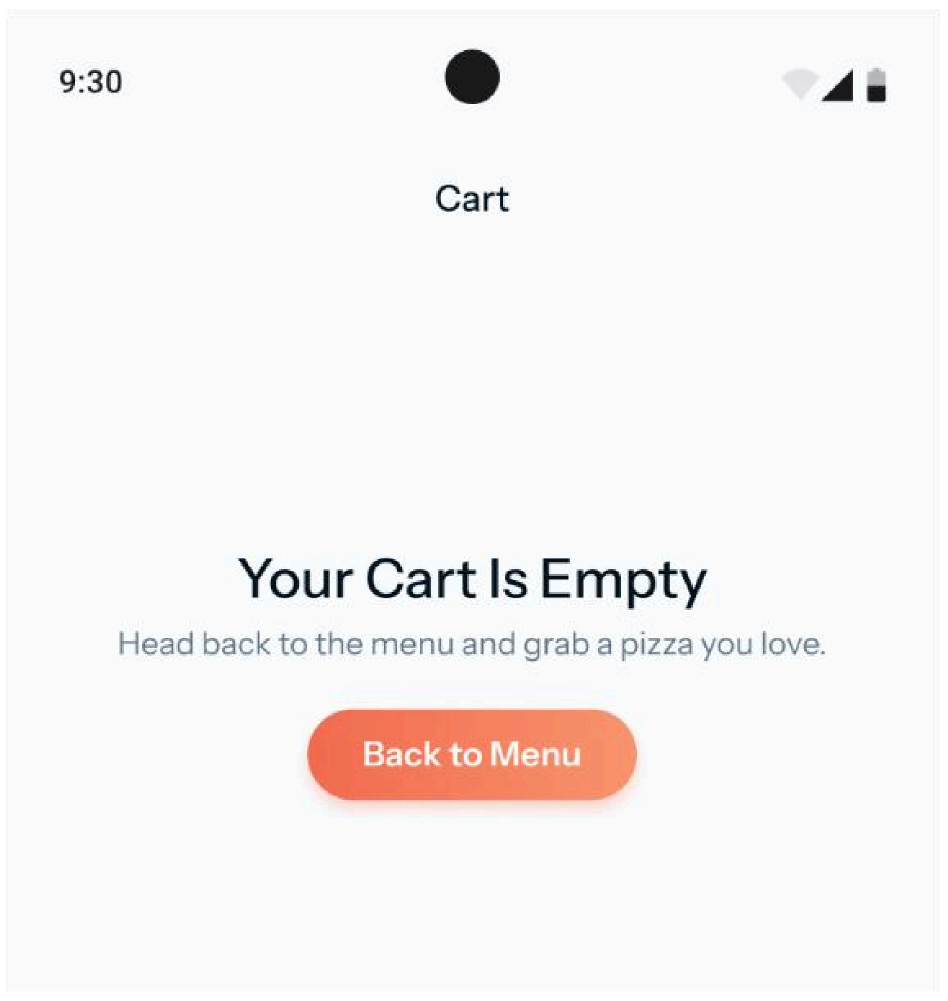
- When the user taps the **Add to Cart** button on any product card from the main screen (e.g., drinks, sauces, or ice cream), the app **should not** automatically open the Cart screen.
- The user must receive **feedback confirming** that the item was successfully added to the cart — the implementation method is up to you (for example, a short vibration or a snackbar message).
- The **Cart icon** in the bottom navigation bar must **instantly update its badge** to show the current number of items in the cart.
- If the user changes the product quantity on any product card (using the − 1 + selector), both the **badge count** and the **cart data** must update dynamically.
- For pizzas, after tapping the **Add to Cart** button on the **Product Details screen**, the app should **return the user to the main Menu screen** so they can continue selecting other products.
- In this case, the **badge count** must also update accordingly.

Why this flow:

This flow allows users to add multiple items quickly without unnecessary navigation steps, maintaining a smooth and efficient ordering experience. It aligns with common UX practices in modern food ordering apps.

🛒 Cart Screen

The **top bar** displays only the centered title: *"Cart"*.



Empty State

If the cart has no items:

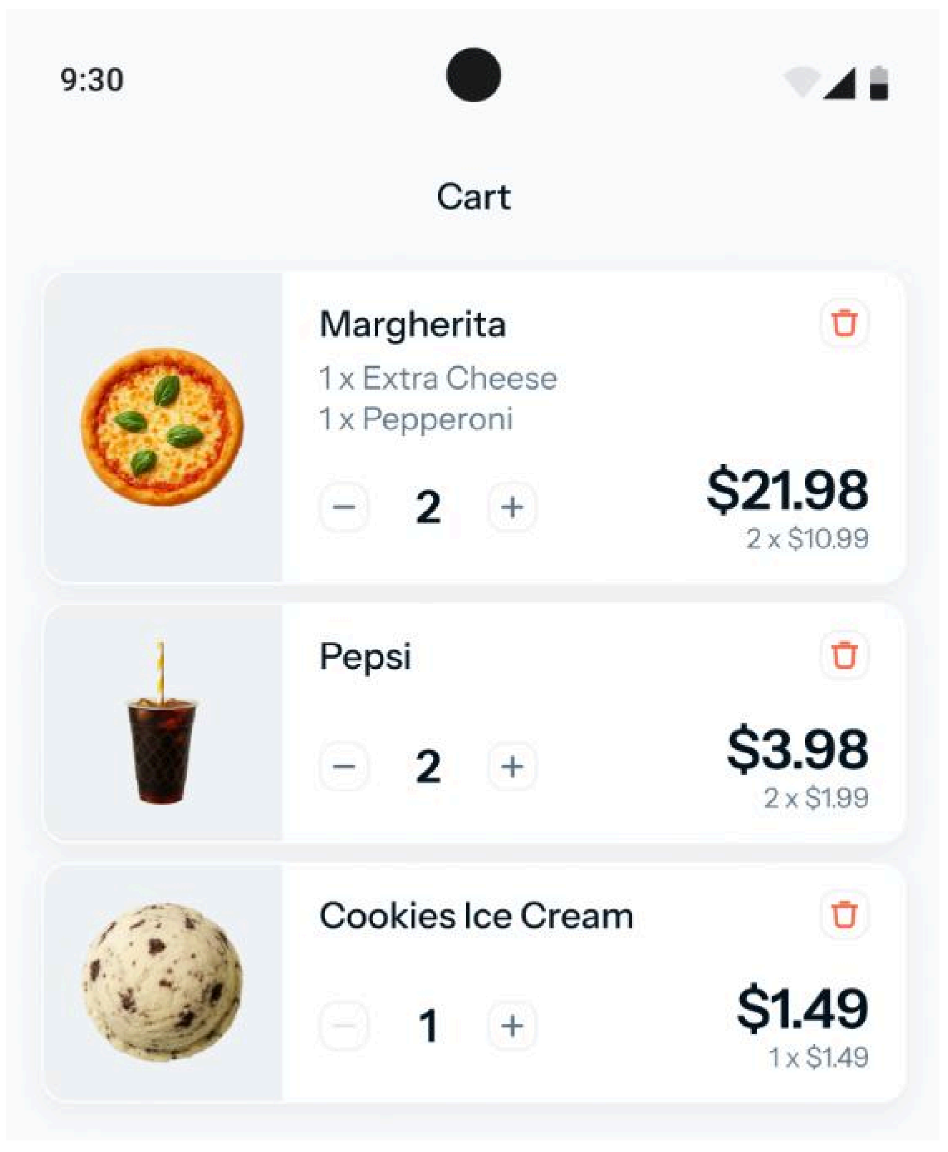
- Show centered title (with **120 dp** padding from the top bar): *"Your Cart Is Empty"*.
- Subtitle/description below: *"Head back to the menu and grab a pizza you love."*
- Centered button: **Back to Menu** (navigates user to the Menu tab).

Product List

Vertical list of **product cards**:

- Product image.
- Product name.

- If the pizza includes **extra toppings**, they are displayed directly below the name as a list of selected add-ons.
    - Each topping is listed on a separate line.
    - If multiple units of the same topping are selected, use the format: *1×Extra Cheese*, *2×Olives*, etc.
- Unit price.
- Quantity selector (– 1 +).
- Total product price (e.g., $21.98).
- If the pizza includes extra toppings or add-ons, the total product price must include the base pizza price plus the cost of all selected add-ons.
- Below it — a mini-label in format 2 × $10.99.
- Remove icon (trash/X) in the top-right corner.

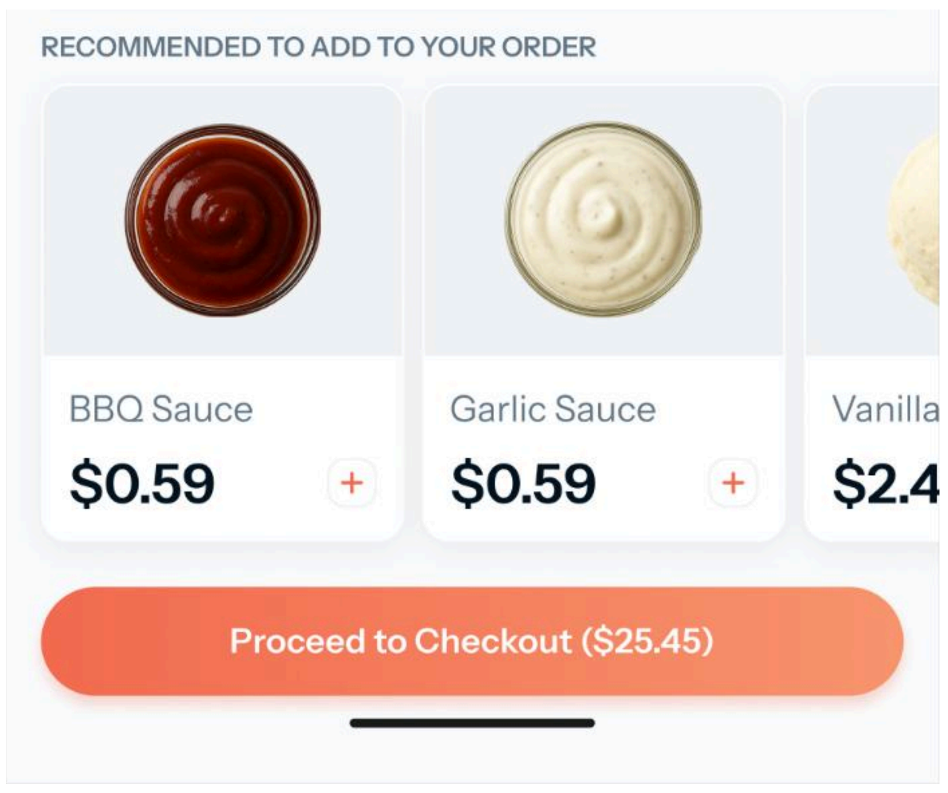


Logic:

- Changing quantity dynamically updates:
    - The product's total sum.
    - The multiplication label (N × $X.XX).
    - The cart's total sum in the sticky button.
- Tapping the **remove (trash) icon** deletes the item from the cart list.
- If the current **quantity = 1** → the minus (–) button is disabled (inactive).
- Pressing the **plus (+) button** increases the quantity and updates the item's total sum accordingly.
- If the user adds the same pizza with **different sets of toppings** (e.g., one Margherita without extras and another with Olives + Bacon), they must appear as **two separate items** in the cart.
    - Each unique combination of a base product and selected add-ons is treated as an **independent cart entry**.

Recommended Add-ons

- Title: "*RECOMMENDED TO ADD TO YOUR ORDER*".
- Horizontal scrollable row of cards.
- Each card includes:
    - Image.
    - Name.
    - Price.
    - + button to quickly add to cart.


RECOMMENDED TO ADD TO YOUR ORDER
BBQ Sauce
$0.59
Garlic Sauce
$0.59
Vanilla
$2.4
Proceed to Checkout ($25.45)

Logic:

- Content is generated randomly from the list of **sauces** and **drinks**.
- Items are displayed in a mixed, randomized order each time.

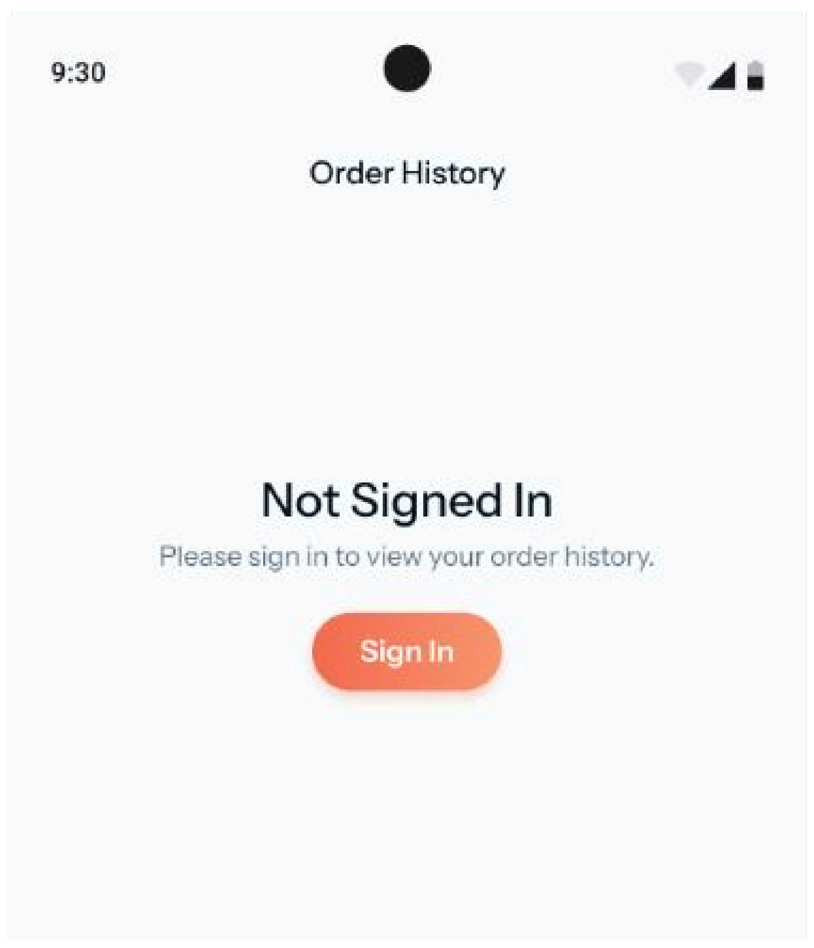
- When tapping the **+ button**, the selected item is:
    - Added to the cart list above as a new product card with the same structure as other cart items.
    - Removed from the Recommended Add-ons list so it no longer appears there.
- If the item is later removed from the cart, it should **reappear** in the Recommended Add-ons list.

Sticky Button

- Text: *“Proceed to Checkout ($XX.XX)”*. Always displays the current cart total.
- A semi-transparent overlay (gradient) is displayed above the button, visually separating it from the content above (toppings list).
- In this milestone, the button is non-functional. Checkout logic will be implemented in the next milestone.

📜 Order History Screen (Unauthorized State)

The **top bar** displays only the centered title: *“Order History”*.



Content

- Centered **title** (with **120 dp** padding from the top bar): *“Not Signed In”*
- Centered **subtitle** below: *“Please sign in to view your order history.”*
- Button: **Sign In** (centered).

Note:

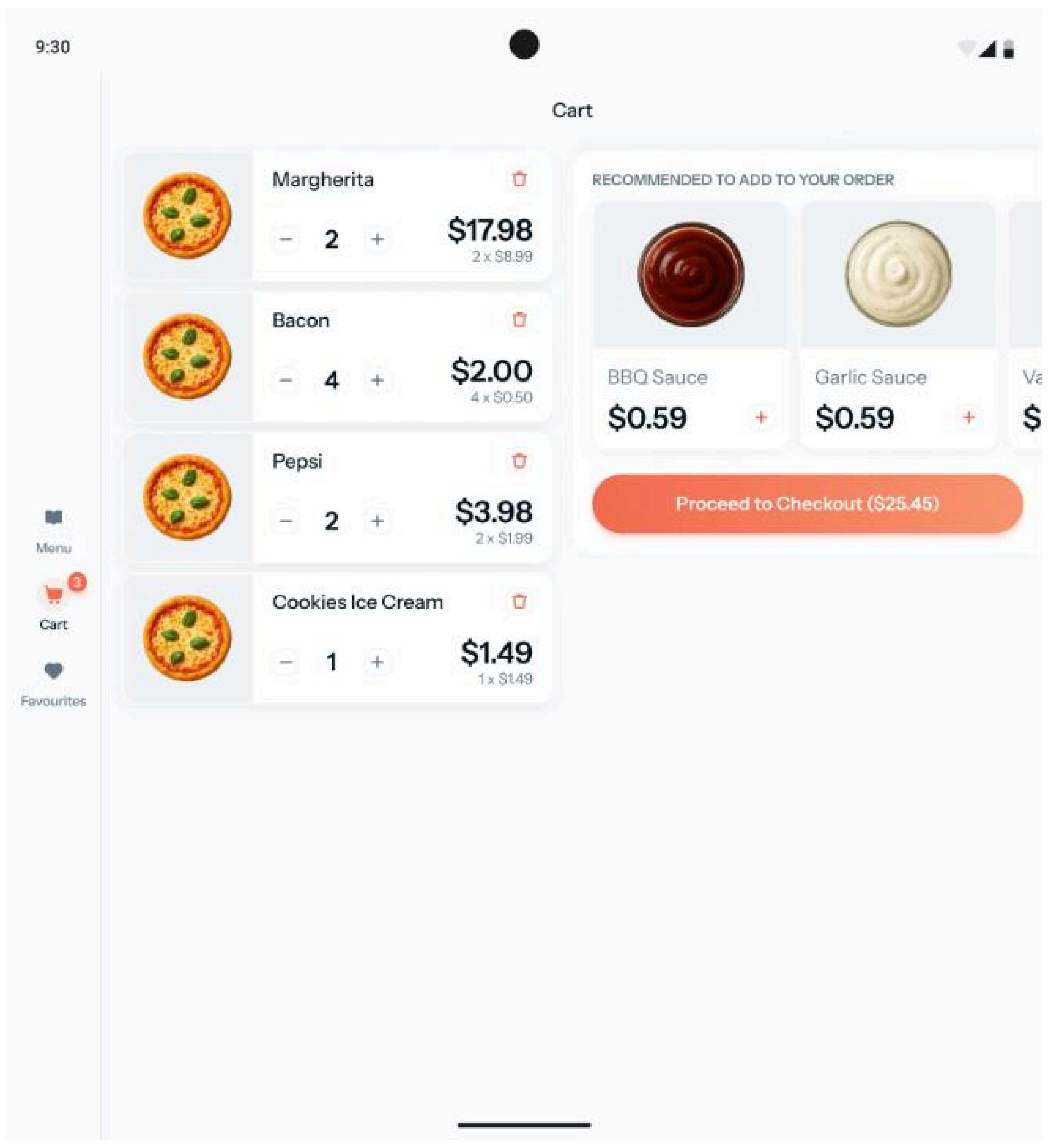
- The **Order History feature** and sign-in logic will be implemented in the **next milestone**.
- All order data must be **stored remotely** and **fetched dynamically** by the app. We recommend using **Firebase** for this, but any other suitable backend solution may also be used.

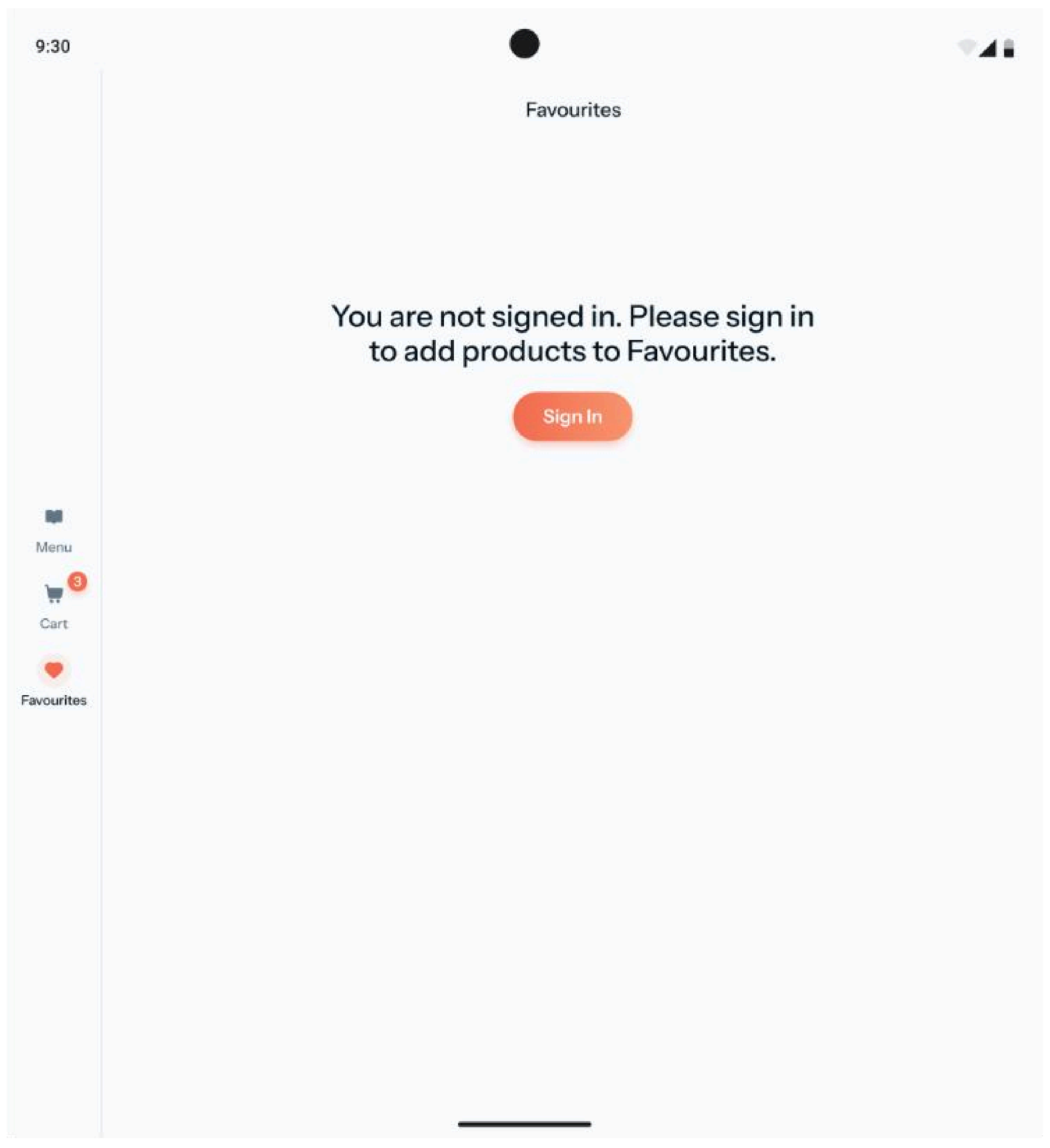
💡 Be mindful of time constraints when choosing alternatives to Firebase, as implementing custom solutions may require significantly more effort.

📏 Screen Adaptation

- **Up to 840 dp:**
    - Cart → product list in 1 column.
    - Recommended Add-ons below the list, horizontally scrollable.
    - Order History → centered message.
    - Bottom Navigation always visible at the bottom.

- **From 840 dp and above:**
  - Cart:
    - Left side → product list (vertical, scrollable if long).
    - Right side → “Recommended Add-ons” (horizontal scroll) with a button below.
    - Right side → “Recommended Add-ons” (horizontal scroll) with a button below.
    - Empty state → display a centered message and button.
  - Order History → Centered unauthorized message.
  - Navigation:
    - Use **NavigationRai**l (standard Material component).
    - Placed **on the left**.
    - Menu icons **vertically centered**.





You are not signed in. Please sign in to add products to Favourites.

🔗 Useful Links for This Challenge

- Add Firebase to your Android project
- Connect your App to Firebase
- Create a Splash Screen
- UX With Material3
- Full Guide to Material3 Theming
- The Full Jetpack Compose Responsive UI Crash Course
- How to Create a Lazy Column With Categories
- How to Save & Restore the Scroll Position of a LazyColumn Persistently
- Stateful vs. Stateless Composables
- State Hoisting in Compose
- Managing State in Jetpack Compose (Codelab)
- Add shadows in Compose

📦 Application Items List

Pizza Menu

- **Margherita** — $8.99

  Ingredients: Tomato sauce, mozzarella, fresh basil, olive oil
- **Pepperoni** — $9.99

  Ingredients: Tomato sauce, mozzarella, pepperoni
- **Hawaiian** — $10.49

  Ingredients: Tomato sauce, mozzarella, ham, pineapple
- **BBQ Chicken** — $11.49

  Ingredients: BBQ sauce, mozzarella, grilled chicken, onion, corn
- **Four Cheese** — $11.99

  Ingredients: Mozzarella, gorgonzola, parmesan, ricotta
- **Veggie Delight** — $9.79

  Ingredients: Tomato sauce, mozzarella, mushrooms, olives, bell pepper, onion, corn
- **Meat Lovers** — $12.49

  Ingredients: Tomato sauce, mozzarella, pepperoni, ham, bacon, sausage
- **Spicy Inferno** — $11.29

  Ingredients: Tomato sauce, mozzarella, spicy salami, jalapeños, red chili pepper, garlic
- **Seafood Special** — $13.99

  Ingredients: Tomato sauce, mozzarella, shrimp, mussels, squid, parsley
- **Truffle Mushroom** — $12.99

  Ingredients: Cream sauce, mozzarella, mushrooms, truffle oil, parmesan

Drinks

1. Mineral Water — $1.49
2. 7-Up — $1.89
3. Pepsi— $1.99
4. Orange Juice — $2.49
5. Apple Juice — $2.29
6. Iced Tea (Lemon) — $2.19

Sauces

1. Garlic Sauce — $0.59
2. BBQ Sauce — $0.59
3. Cheese Sauce — $0.89
4. Spicy Chili Sauce — $0.59

Ice Cream

1. Vanilla Ice Cream — $2.49
2. Chocolate Ice Cream — $2.49
3. Strawberry Ice Cream — $2.49
4. Cookies Ice Cream — $2.79
5. Pistachio Ice Cream — $2.99
6. Mango Sorbet — $2.69

Extra Toppings

- Bacon — $1.00
- Extra Cheese — $1.00
- Corn — $0.50
- Tomato — $0.50
- Olives — $0.50
- Pepperoni — $1.00
- Mushrooms — $0.50
- Basil — $0.50
- Pineapple — $1.00
- Onion — $0.50
- Chili Peppers — $0.50
- Spinach — $0.50