# React Native CLI Assignment: "GlowCart" – Beauty E-commerce App

## Objective

Build a minimal cosmetic e-commerce app using **React Native CLI**, replicating the provided UI. This task assesses your skills in building high-performance mobile apps with clean code architecture, accurate UI replication, and smooth UX.

---

## API Source

Use:
**https://dummyjson.com/products**

Filter/mimic cosmetic products based on title, image, and description. You can hardcode values like “Essence Mascara” using similar-looking products.

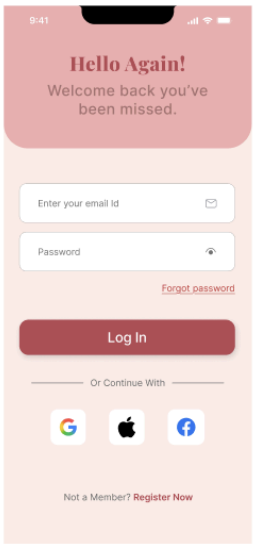
---

## Screens to Build (Match Provided Figma or Image):

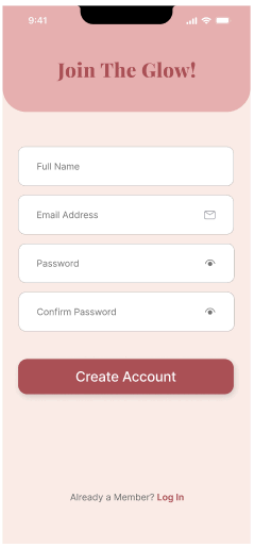
Here is the link of Figma Design: Figma

onboarding

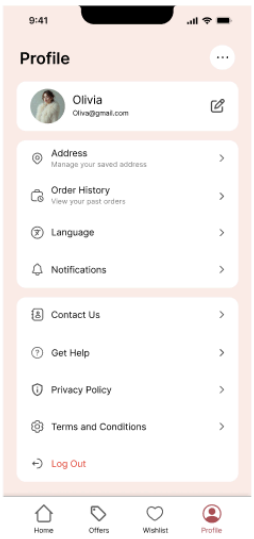

login



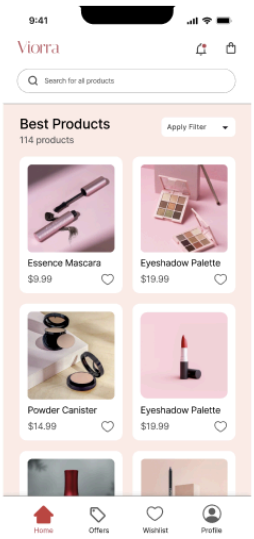

Register



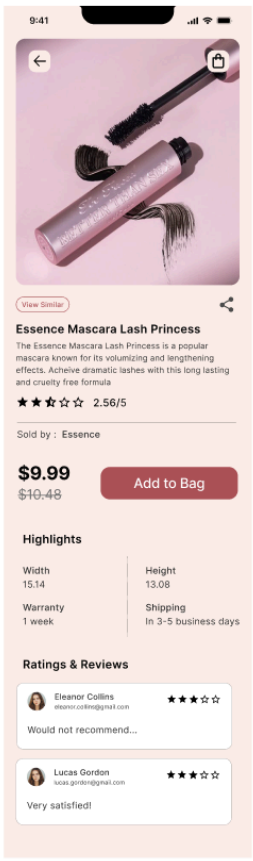

profile




Product list




product detils




1. **Onboarding Screen**

- Display logo/image, tagline (“Your Beauty, Delivered”), and a **“Get Started”** button.
- Navigates to the login screen.

2. **Login Screen**
    - Email and password input fields.
    - Login button.
    - Optional: Google/Apple/Facebook login buttons (UI only).
    - Link to “Register Now”.

3. **Register Screen**

    - Full name, email, password, confirm password fields.
    - "Create Account" button.
    - Link to "Login" screen.

4. **Home/Product List Screen**

    - Fetch and display products from the API.
    - Include search bar and filter icon (functionality optional).
    - Each card shows image, name, and price.

5. **Product Details Screen**

    - Large image of product.
    - Title, description, ratings, price.
    - "Add to Bag" button.
    - Highlights (dimensions, warranty, shipping).
    - Ratings and reviews section (mock reviews).

6. **Profile Screen**

    - Mock user info (name, email).
    - Sections:
        - Address
        - Order History
        - Language
        - Notifications
        - Contact Us, Get Help, Privacy Policy, T&C
        - Logout

---

## Technical Requirements

- **React Native CLI** project setup (`npx react-native init GlowCart`)
- Use **React Navigation** for screen transitions.
- **Axios** or `fetch` for API calls.
- **State Management**: Context API / Redux / Zustand (your choice).
- Use `FlatList` for product rendering.
- Custom reusable components (e.g., `ProductCard`, `Header`, `Button`).
- Style with `StyleSheet` API or `tailwind-rn`.
- Avoid hardcoding; use dynamic data where applicable.

---

## Bonus (Not Required but Appreciated)

- TypeScript
- Animations (e.g., fade-in product cards, button taps)
- Pagination or Lazy Loading
- Theme support (light/dark)

## Evaluation Criteria

| Criteria | Details |
|---|---|
| **Speed** | How quickly the app is completed. |
| **Accuracy** | UI closely matches the provided design. |
| **Code Quality** | Modular, clean, DRY code. |
| **Best Practices** | Folder structure, reusable components, logic separation. |
| **API Integration** | Efficient fetching, error handling. |
| **Performance** | Smooth navigation, responsive UI. |

| | |
|---|---|
| **Responsiveness** | App looks good across devices. |

---

## Deliverables

- Public GitHub repository link.
- Please provide a video demonstration of the completed project by submitting a link. You may upload the video to any cloud platform or utilize a video sharing service such as Jam or Loom.
- Include a `README.md` with:
    - Setup instructions
    - Time taken
    - Screenshots/GIFs
    - Any assumptions or known issues