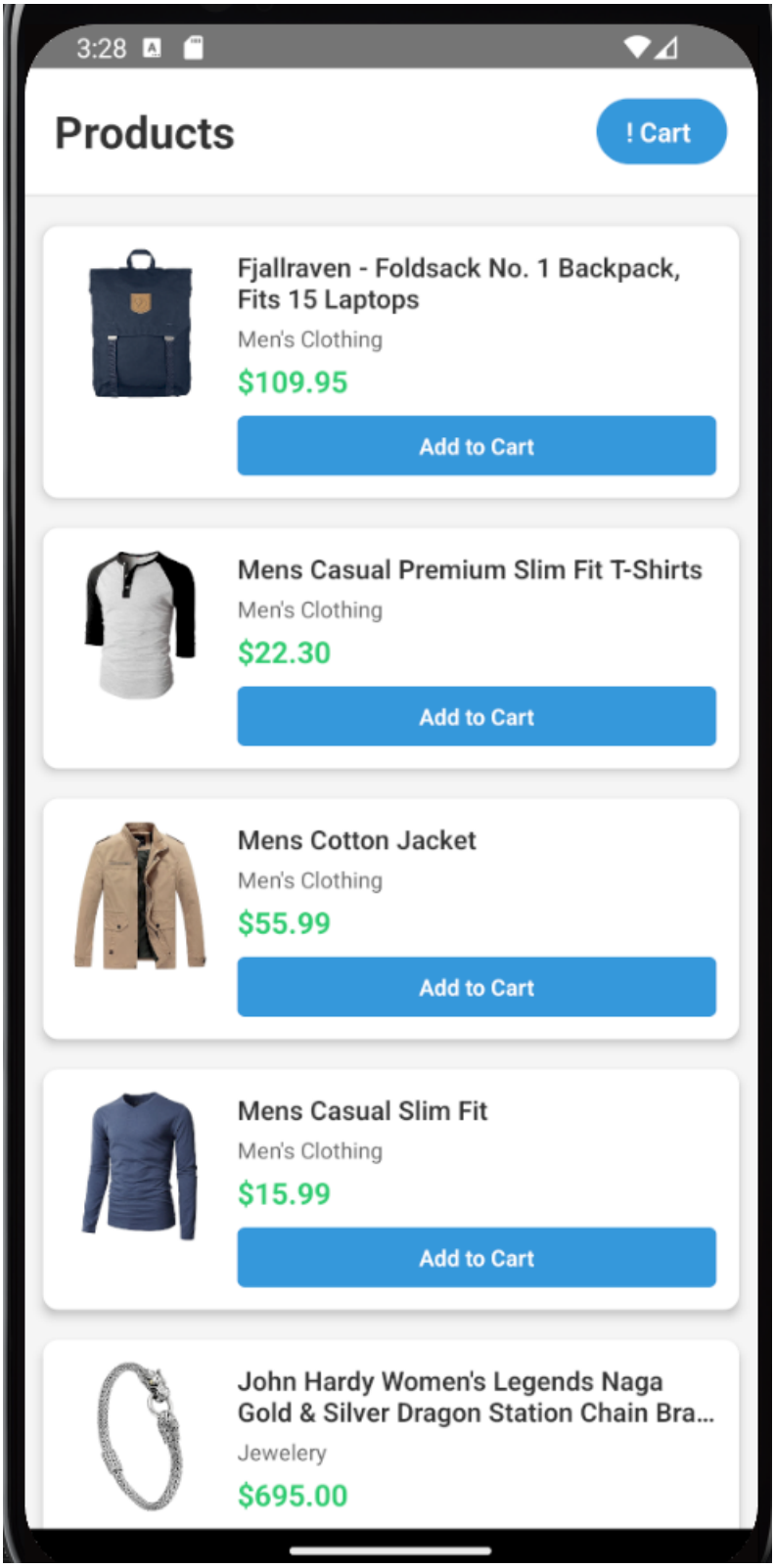
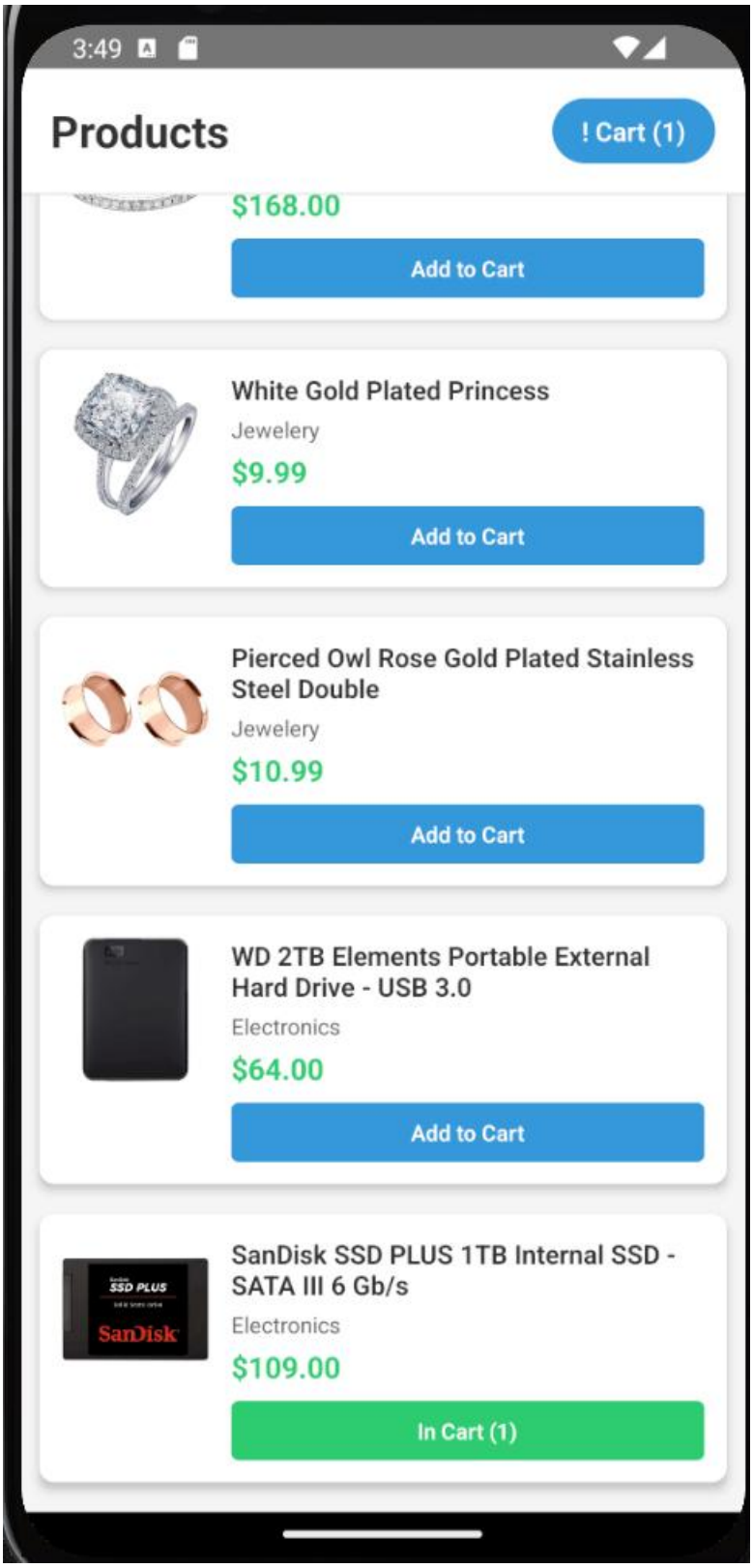
LAB 7

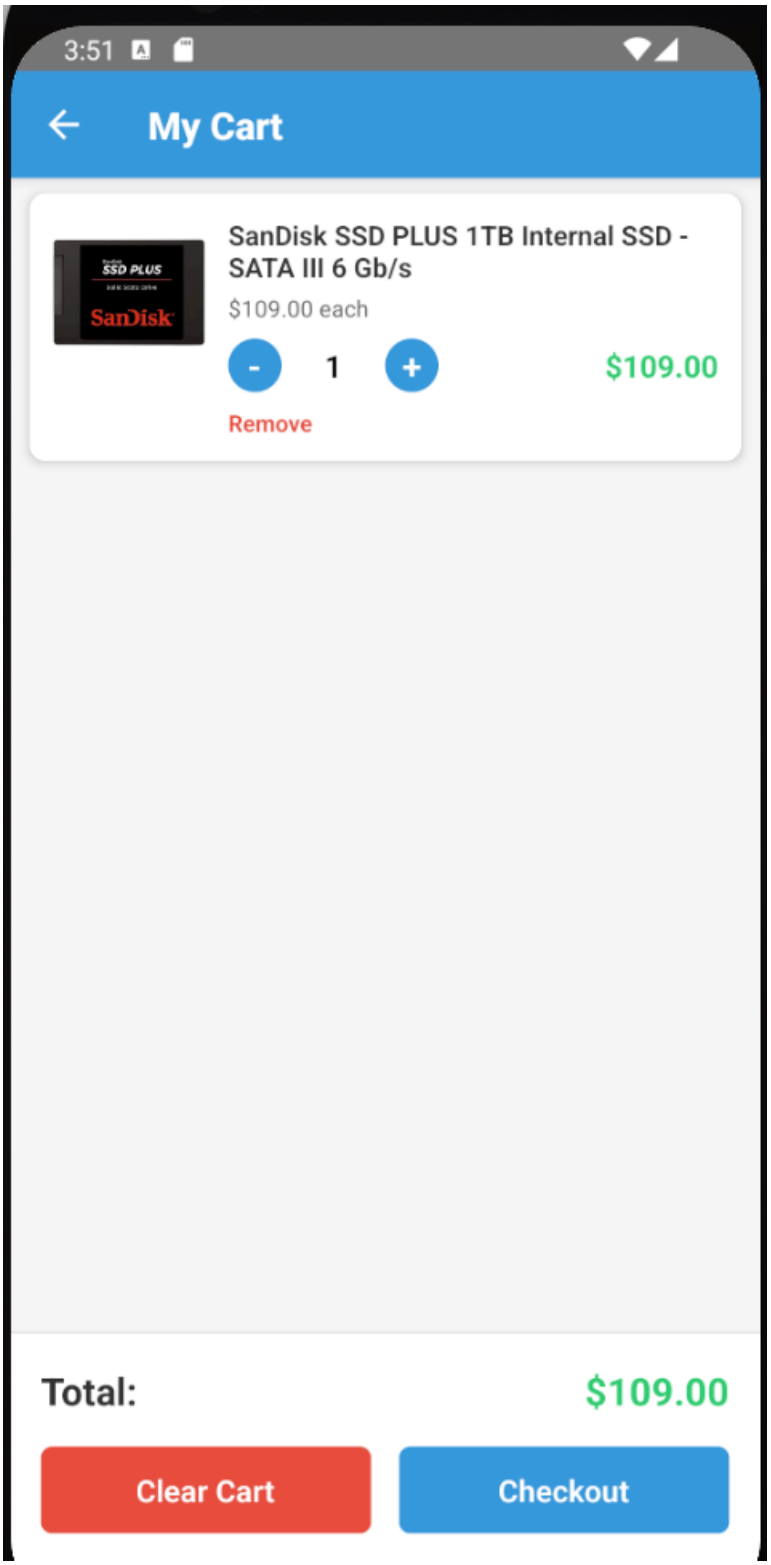
Products list with loaded products:



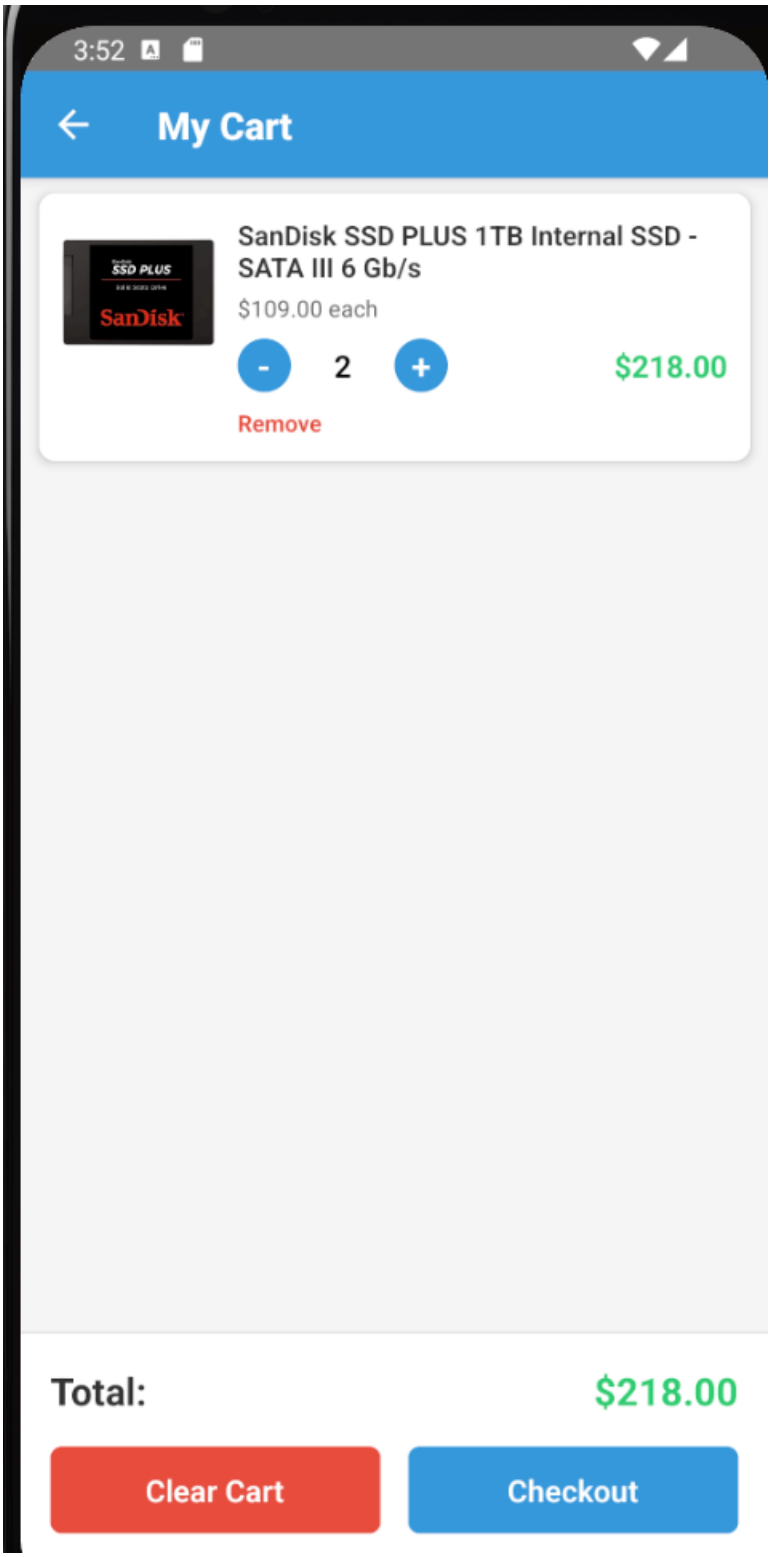
2. Product added into cart:



3. Cart Screen with items:



4.Cart with updated queantities:



Lab report:

Challenges faced during the project:

Honestly this lab was a bit rollercoaster for me . When I first opened up the instructions I thought , “Okay cool, just shopping cart I can do this” But the moment I started actually connecting Redux with the React Native , things got complicated really fast. The slices, selectors, components, and the screens, and it felt like everything dependend in everything else.

I also struggled with keeping the track of what the store looked like at the different points. Sometimes my cart would say it had 0 items even tough I literally just added something . Other times, the product list wouldn’t refresh and I’d sit there and starting the screen thinking, “Why is nothing happening?”

The async part stressed me out too. The FakeStore API failed twice in a row and I thought I broke the entire project. I wasn’t sure if my code or the API being moody. It took me a while to even realize that the mock data fallback existed for a reason.

And then there is the UI ….. getting the CartScreen to look right was way harder than I expected. Images not lining up, text overflowing, buttons were not looking right…. Every tiny thing felt like a new mini project to be honest.

Overall, the challenge wasn’t one big thing , it was the constant little bumps that added up.

HOW I SOLVED THEM :

What ended up helping me the most was slowing down and tackling one issue at a time. Instead of trying to “get Redux,” I focused on just making the products slice work. I used Redux DevTools a LOT — it honestly felt like a cheat code. Seeing my actions appear live helped me understand whether the problem was in the slice, the selector, or the component.

Another thing that helped was literally talking myself through the flow out loud. Like: “Okay, I click this button. That dispatches addToCart. That should update state.cart.items. Then CartScreen should read that using a selector.” Saying it step-by-step made it feel less confusing.

For the UI issues, I took everything apart and tested small parts individually. I rendered just one CartItem by itself before even touching FlatList, and only when it looked right did I put it back. That made things way easier to debug.

The API problem was solved simply by trying again later and relying on mock data when it failed. Once I realized the API wasn't my fault, my stress level dropped instantly.

And honestly, re-reading the lab PDF helped more than I wanted to admit. Sometimes I would get stuck for 20 minutes only to see that I missed one tiny import or named a selector wrong.

In the end, I fixed everything through patience, trial and error, and going back to basics.

WHAT I LEARNED ABOUT REDUX:

Before this lab, Redux felt like this complicated thing that only huge companies used. After actually building a full shopping cart with it, I finally understand why developers like it so much. It's not about making things harder — it's about keeping everything organized when your app starts growing.

The biggest thing I learned is how Redux makes your app predictable. Every action, every state change, every update has a clear path. Instead of guessing where my data is coming from, I know it's coming from the store. Instead of wondering why something didn't update, I can check the action in DevTools and see exactly what happened.

Redux Toolkit especially made things easier. createSlice feels so clean — everything is in one place. I loved not having to manually write action types or worry about immutability because Immer handled that behind the scenes.

Async thunks were another big eye-opener. I never understood how apps manage loading states before. Now it makes sense: pending → fulfilled → rejected. It's a nice flow that matches how real apps behave.

And selectors… I finally get why people use them. They make the code cleaner and keep components simpler.

Overall, this lab made Redux feel way less scary. I walked away from it feeling like I actually understand how bigger apps keep everything consistent and manageable.