E-commerce Site

This project aims to build an e-commerce platform using React, Redux, and Firebase. The goal is to complete key features such as product registration, management, search, cart, payment, review submission, admin page, etc. The implementation will be production-ready, using Redux Store for state management and Firebase for data integration.

Purpose of choosing this project:
To gain real-world service-level development experience, enhance frontend/backend integration skills, use and optimize React, Redux, Firebase stack, UI/UX design and performance optimization, and experience project management and maintenance.

Basic implementation and features (basic structure):

- Main / Product Detail / Login
- Product Listing with Filter Functionality
- Search Functionality
- Add to Cart
- Slide Carousel
- Cart & Checkout
- Payment System Integration
- Apply Payment API
- Update Cart Items
- Show Total Price
- My Page & Order History
- Ordered Product List
- Show Order Completion Info
- Shipping Address Management
- Product Review Submission

Admin:

- Order Management (with Chart Display)
- Filtering (Search & Pagination)
- Update Order Status
- Edit and Delete Products

Development Stack:
React, Next.js, Firebase, Redux, JavaScript, TypeScript, CSS, HTML

Testing:
Jest, Playwright, Sentry, Webpack, Node.js, React-query

12.25/2024 - Commit comment: Add new files

- Created global styles
- Created login UI
- Created and styled Loader component
- Created Input component

12.27/2024 - Commit comment: Update file

- Styled Input
- Created Icon component
- Created and styled Auto Sign-In Checkbox
- Created and styled Tooltip component
- Created and styled Divider component
- Created and styled Button component
- Created sign-up UI
- Created and styled Change Password page UI
- Created Heading component

12.28/2024 - Commit comment: Second update

- Connected to Firebase
- Applied authentication, database, storage
- Added toast notifications using React Toastify
- Implemented Firebase login, sign-up, password reset features

12.29/2024 - Commit comment: Create Footer UI

- Created and styled Footer UI
- Added logo, customer service info, SNS

→ From this point, commit comments are written as dates

12.30/2024 ~ 12.31/2024

- Created and styled Header UI
- Displayed displayName based on login state
- Used onAuthStateChanged to detect auth status change
- Removed header on pages except admin page
- Created and styled InnerHeader component
- Set columns and rows using grid

1/2/2025

- Created Redux slice and store
- Saved user data in store on login and removed on logout
- Tracked state changes using action and reducer
- Stored email, userID, userName in Redux store
- Wrapped app in Provider to re-render based on store info

1/3/2025

- Created and styled Main page carousel
- Used setInterval to auto-switch every 5s
- Used clearInterval when component unmounted
- Added passive previous/next buttons
- Used useCallback to regenerate on dependency change

1/4/2025

- Created and styled Product Creation page UI
- Saved registered product to Firestore
- Displayed image upload progress and stored image URL
- Handled inputs using useState
- Used map in category to loop elements
- Saved product info after uploading image URL from Firestore
- Moved to product list or showed error message after save

1/5/2025

- Fetched data from Firestore
- Used custom useFetchCollection Hook
- Created and styled basic product UI
- Stored product data in Redux store
- Used Redux toolkit to manage product code
- Set price range for products
- Created min/max values for all products in Redux store
- Updated code to allow external image URLs

1/6/2025 ~ 1/7/2025 early morning

- Implemented product filtering feature
- Function to display all categories in Redux store
- Apply/reset specific selected category
- Used useEffect and dispatch to update filter
- Created Filter Slice component
- Implemented full filtering logic
- Updated filtered data in Redux store
- Created and styled filter UI
- Added category, brand, price, reset buttons
- Styled active filters using conditional class styles

1/8/2025

- Implemented Product List feature
- Wrote Redux store update code
- Wrote sorting code by price and grouped by Radio button
- Used slice function for products per page
- Created and styled List component UI
- Created UI for sorting by latest, low price, high price
- Generated product blocks from sliced items
- Created UI for productsPerPage
- Wrote page number logic

1/9/2025

- Created and styled Product Item component

- Displayed product image, name, price
- Navigated to product detail on image click
- Allowed image loading from Firebase storage
- Truncated long product names
- Used rating library to display ratings

1/10/2025

- Fixed rocket delivery image next to price
- Fetched review data for each product from Firebase
- Collected documents with same id into array
- Calculated and displayed average rating
- Used isNaN to handle no-review case
- Implemented product search
- Used handleSearch on enter or icon click to filter
- Showed all products if input is empty, else filtered
- Filtered if product name or category includes search term
- Used handleKeyDown for enter key to trigger search
- Displayed filtered products using state.filteredProducts
- Created and styled Pagination component
- Generated page numbers based on product count
- Used pageNumberLimit to limit pages shown at once
- When product count exceeds limit, used arrow button to show next pages
- Used remainder logic to navigate back
- Styled to disable arrow buttons when first/last page

1/15/2025

- Created Product Detail page
- Fetched product data from Firestore
- Used useParams to dynamically get URL
- UI to add to cart (function not yet implemented)
- Created and styled Product Detail UI
- Displayed brand, name, rating, price, description, reward points, quantity, add to cart button
- Disabled minus button when quantity is 1
- Used Divider to separate sections
- Fixed image loading bug from incorrect id in ProductItem

1/16/2025 ~ 1/17/2025 early morning

- Created Product Review UI and showed reviews
- Reviewed allowed after purchase
- Verified product id in Firebase matches website id
- Used map to render review data
- Modified useFetchDocuments to get doc id
- Passed data to review component
- Displayed username, rating, review content, and date
- Used dayjs.js for localized date display
- Implemented Cart feature (data management)
- On cart button click, called addToCart
- Used localStorage to persist cart
- If product exists, increase quantity; else add full product info

- Tracked total quantity and price
- Stored cart data in Redux store
- Fixed slider scss warning by setting width

1/17/2025 ~ 1/18/2025 early morning

- Created Cart page
- Retrieved cart and user data from Redux
- Calculated total cart price
- Adjusted/removable item quantity
- Removed item if quantity becomes 0
- Implemented delete all cart items
- Synced cart changes with localStorage
- On checkout, redirected to login if not logged in, then back to cart after login
- Styled Cart page
- Added Link to homepage if cart is empty
- Used table to display items with map and key by id
- Used thead to match item info layout

1/19/2025 ~ 1/20/2025 early morning

- Created address input page for checkout
- On Buy Now, redirect to address input
- Created address input state
- Created logic for state management
- Submit button redirects to order page
- Styled address input page UI

- Created order page
- Summarized products in cart
- Retrieved cart data using useSelector
- Displayed items, total quantity, total price, payment button
- Redirect to homepage if cart is empty

1/23/2025

- Generated payment service key
- Created env file and stored key
- Implemented payment feature
- Payment popup, success, fail, transaction history
- Used fetch to send HTTP request
- On success, stored data in Firestore
- Cleared cart after payment
- Saved data to Firestore

1/26/2025

- Created and styled Payment Success page
- Retrieved product, order ID, card number, price, date from payment system API
- Created order history page
- Used dispatch to update order list and useSelector to get data
- Used useFetchCollection to get data from Firebase
- Wrote orderSlice to update store
- Disabled timestamp error with serializableCheck: false
- Created history UI

- Used filter to get desired order data
- Created detail navigation function on click
- Used table to show quantity, date, order ID, price, order status
- Fixed error by correcting dayjs installation (installed day instead of dayjs)

1/31/2025

- Created and styled Order Detail page
- Got product id from Redux and data from Firebase
- Displayed name, quantity, price, status, image
- Used click and router to move to review page
- Created Product Review submission page
- Retrieved product id and data
- Rendered rating and review
- Displayed name, image, rating, review, submit button
- Stored review on submit
- Redirected to Order Detail page

2/11/2025 ~ 2/12/2025 early morning

- Created Admin Navbar
- Displayed navbar only on Admin pages
- Created nav for dashboard, product list, add product, orders
- Applied active style based on pathname
- Created and styled Admin product page
- Fetched all products and stored in Redux
- Created search function

- Displayed product data in table
- Created search component with icon and scss
- Added edit page link using icon
- Implemented delete product using Notiflix and Firebase
- Created and styled Admin Edit page
- Fetched and edited data
- Removed old image when new image uploaded
- Added edit date to original date
- Reused Add Product code for edit
- Exported categories for reuse

2/13/2025

- Created Admin Order History page
- Fetched order data
- Stored in Redux store
- Fixed typo in navbar causing 404
- Created Admin Order Detail page
- Fetched product id and data
- Added order status update feature
- Styled UI
- Fixed null handling for card number and price due to test payment

2/15/2025

- Created Order Status Update component
- Changed orderStatus and updated in Firestore

- Redirected after update
- Styled UI
- Created Admin Dashboard page
- Created necessary icons
- Fetched product, order, total price data
- Stored in Redux
- Created order total price function in orderSlice
- Styled UI
- Created order status chart
- Used vertical-bar-chart from react-chartjs-2
- Fetched all orders from Redux
- Counted status using array length
- Styled chart

2/18/2025 ~ 2/19/2025

- Studied theory and setup for JS → TS conversion for maintenance and reusability
- Converted Input component to TypeScript
- Created interface for props
- Assigned appropriate types, used [x: string]: any for unknown props
- Used ChangeEvent for e object in onChange
- Converted AutoSignInCheckbox, Checkbox, Tooltip, Button components to TS

2/21/2025

- Converted Login, Signup, Loader, Reset, Heading pages to TS

- Converted Redux slice to TS
- Created `types` folder for interfaces used in Redux
- Used store.getState and typeof to create types
- Used ReturnType to use returned type
- Automatically inferred types in action payload

2/22/2025

- Converted remaining component files to TS
- Used React.ReactNode for children props
- Used '!' to ensure email is not null
- Created necessary interfaces
- In utils, if error is an object, used instanceof Error to show message; else stringified it
- Used DocumentData from Firebase for types
- Used WhereFilterOp from Firebase for arg[1] type definition

3/2/2025

- Updated header and footer logos
- Deployed using Vercel
- Prevented pageNumbers from recalculating on every render
- Escaped apostrophe in ProductReviewItem with '
- Added type Promise<{ orderId: string }> for PageProps searchParams
- Deleted interface and inserted IProduct directly in ProductList
- Deployment succeeded on Vercel
- Fixed Google login issue by adding Vercel domain to Firebase Authentication
- Added TOSS test key to Vercel environment variables (env not uploaded to GitHub)

- Fixed issue with TOSS test API data not being fetched
- Confirmed null values for card number and total price due to test system
- Added handling for null in approvedAt in formatTime function

Deployment complete – payment system uses Toss test API