MACoursework Stage 3 App Building Report Group 1

Tran Duy Anh - 23085483
Nguyen Duc Trung - 23085492
Nguyen Nhat Minh - 23085487

January 7, 2026

Contents

1 Introduction

This mini report documents the development of the application for **Stage 3: Building Your App**. The purpose of this report is to describe the final implementation, clearly identify individual coding contributions, document the development environment, and explain any deviations from the original design.

While some aspects of the original design evolved during implementation, the final application remains true to the original concept.

2 App Overview

App Concept

The developed application is a mobile **shopping application** designed to provide users with a convenient and intuitive platform for browsing products, managing purchases, and completing orders on Android devices. The app aims to simulate the core functionality of a modern e-commerce system, allowing users to explore products, manage their shopping preferences, and complete transactions efficiently within a single application.

The target users are general consumers who wish to browse products, save items of interest, and place orders using a streamlined and user-friendly mobile interface. The application focuses on usability, clear navigation, and responsive interaction to enhance the overall shopping experience.

Key Features Implemented

- **User Management**: User account creation, login, and profile management to personalise the shopping experience.
- **Product Browsing**: Display of products with relevant details such as images, descriptions, prices, and ratings.
- **Shopping Cart**: Ability to add, update, and remove products from the cart prior to checkout.
- **Order Management**: Placement of orders, viewing order summaries, and tracking order status.
- **Payment Card Management**: Secure management of payment card information for checkout.
- **Address Management**: Storage and selection of delivery addresses during the checkout process.
- **Wishlist**: Saving favourite products for future reference.

- **Voucher System**: Application of discount vouchers to reduce order total.
- **Notifications**: In-app notifications to inform users about order updates, promotions, and important events.
- **Chatbot Assistance**: An integrated chatbot to assist users with product recommendations and general shopping queries.

Platform and Device Support

- Platform: Android
- Programming Language: Kotlin
- UI Framework: Jetpack Compose
- Orientation Support: Portrait / Landscape
- Target Devices: Android smartphones

3 Design Evolution and Deviations

Original Design	**Final Implementation**	**Reason for Change**
Welcome screen	Welcome screen	None
Login and signup	Login and signup	None
Forgot password screen	None for implementation	
Homepage screen	Homepage screen	None
Discover screen	Discover screen	None
Filter screen	Filter screen	None
Found results screen	Found results screen	None
Product screen	Product screen	None
Cart screen	Adjust the nav bar for cart screen	More flexible used for user
Checkout screen 1	Change the customer shipping details to write down straight away instead of like in the wireframe	To save the address of the user for next time use

Checkout screen 2	Give user 2 options to pay and adjust the UI	Make the app checkout screen more compatible and more efficient in performance
Checkout screen 3	Checkout screen 3	None
My orders screen	Adjust total of the product final display to the left	For better look
Order information screen	Order information screen	None
Rate product screen	Rate product screen	None
Side bar function	Side bar function	None
Notification screen	Notification screen	None
Collection screen	None for implementation	
Profile screen	Profile screen	None
My Wishlist screen	My Wishlist screen	None
Profile setting screen	Profile setting screen	None
Payments screen	Payments screen	None
Add new card screen	Add new card screen	None
Voucher screen	Voucher screen	None
Address screen	Address screen	None
Chat support screen	Chat support screen	None
Device setting screen	Device setting screen	None
Notification setting	Notification setting	None

All changes were made to improve usability, performance, or technical feasibility.

4 Development Environment and Configuration

Tools

- IDE: Android Studio Narwhal 3 Feature Drop — 2025.1.3
- Language: Kotlin
- UI Framework: Jetpack Compose

SDK Configuration

- Compile SDK Version: API level 36 (Android 16)

- Minimum SDK Version: API level 33 (Android 13)
- Target SDK Version: API level 33 (Android 13)

Testing Environment

- Emulator Device(s): Medium Phone API 36.1
- Physical Device Testing: Xiaomi Redmi 12 4G (model number 23053RN02A)

5 Group Collaboration and Work Structure

To support effective collaboration and parallel development, the application was structured into clearly defined and relatively independent modules. Each module represents a logical subsystem of the application, allowing group members to work concurrently with minimal overlap and merge conflicts.

The project follows a layered architecture, separating data handling, navigation, user interface, and utility logic. This structure improved code readability, maintainability, and individual accountability.

Module-Based Work Allocation

- **Data Layer (`data`)**: Responsible for managing application data, including local storage, remote data sources, and chatbot logic. This separation allowed independent development of data handling without affecting UI components.
- **Navigation Layer (`navigation`)**: Handles screen navigation using Jetpack Navigation Compose. Navigation routes and flow control were implemented independently to ensure consistent user transitions across screens.
- **User Interface Layer (`ui`)**: Contains reusable UI components, individual screens, and theme definitions. Each screen was treated as an independent view, enabling different group members to implement features such as product browsing, cart management, and user profile screens in parallel.
- **Utility Layer (`util`)**: Provides shared utility functions such as input validation and login requirement checks, reducing code duplication and improving reusability across the app.

Collaboration Practices

Version control was used throughout development to manage code integration and track individual contributions. Regular communication and code reviews were conducted to

share best practices, resolve conflicts, and ensure consistency in coding style and architectural decisions.

This modular approach ensured that each group member could make a clear and measurable contribution to the codebase while maintaining a cohesive and well-integrated final application.

6 Individual Contributions

Student Name: Tran Duy Anh

Student ID: 23085483

Responsibilities

- Application Architecture Design
- Welcome screen
- Authentication system
- Navigation system
- Check login or guest user for navigation and security purposes
- Search and filter system, filter sidebar components
- Order screens, Order management
- Review system, review screen
- Wishlist system
- Notifications system
- NLP Chatbot, Chat components and screens
- Hooking up Home screen, Search screen, Order screen, Notification screen, Wishlist screen to Room database

Files Implemented

- `data` layer:
  - chatbot: ChatRepository, LocalChatEngine
  - local:

        * dao: NotificationDao, OrderDao, OrderItemDao, ProductDao, ProductReviewDao, UserDao
        * database: AppDatabase, Converters
        * entity: NotificationEntity, ProductReviewEntity, UserEntity, WishlistEntity
        * model: ProductWithAvgRating, ReviewWithUser
        * relation: OrderItemWithProduct
        * repository: NotificationRepository, OrderRepository, ProductRepository, ProductReviewRepository, UserRepository, WishlistRepository
        * session: SessionManager

- `navigation` layer: AppNavHost, SearchNavigation
- `UI` layer:
    - components:
        * chat: ChatBubble, ChatInputBar, ChatTopBar
        * common: BottomNavigationBar, CollapsibleHeader, ShopStatusTabs, ShopTab, TopAppBar
        * order: OrderCard, OrderStatus, OrderStatusTabs
        * product: ProductCard, ProductGrid, StarRating
        * search: FilterProduct, SearchFilterSheet
    - screens:
        * auth: LoginScreen, SignUpScreen
        * cart: CartScreen, CartViewModel
        * chat: ChatMessage, ChatScreen, ChatViewModel
        * checkout: CheckoutPaymentViewModel
        * home: HomeViewModel
        * notification: NotificationScreen, NotificationViewModel
        * onboarding: WelcomeScreen
        * order: OrderScreen, OrderViewModel
        * orderInfo: OrderInfoScreen, OrderInfoViewModel
        * product: ProductScreen, ProductViewModel
        * resultSearch: ResultSearchViewModel
        * review: ReviewScreen, ReviewViewModel
        * search: SearchScreen, SearchViewModel
        * setting: NotificationSettingViewModel
        * wishlist: WishlistViewModel
- util layer: RequireLogin, ValidationUtils

Contribution Description

I was responsible for implementing several core features that directly support the functionality and user experience of the application. This included the authentication system, application navigation, and multiple features such as search and filtering, order management, wishlist, notifications, and the integrated chatbot.

I designed and implemented the navigation structure using Jetpack Navigation Compose to ensure a clear and consistent flow between screens. Authentication logic was integrated with navigation to restrict access to protected screens and improve application security.

For product discovery, I implemented search and filtering functionality to allow users to efficiently locate relevant products. Order management features were developed to enable users to view order details and order history in a clear and structured manner.

Additional features such as the review system and wishlist were implemented to enhance user engagement. I also developed the notification system to inform users of important events such as order updates and available vouchers.

Finally, I implemented an Natural Language Processing chatbot to assist users with app navigations and product recommendations.

I made sure that every component was modular, readable, and maintainable by adhering to the established project architecture and coding conventions throughout the development process.

Student Name: Nguyen Nhat Minh

Student ID: 23085487

Responsibilities

- UI components
  - Common components (TopAppBar, BottomNavigationBar, etc.)
  - Address
  - Cart
  - Checkout
  - Order Info
  - Payment
  - Product
  - Voucher
  - Wishlist
- UI screens

  - Home screen
  - Cart screen
  - Address screen
  - Checkout screen
  - Notification screen
  - Order Info screen
  - Payment screen
  - Search and Result search screen
  - Setting screen
  - Voucher screen
  - Wishlist screen

Files Implemented

- `navigation` layer: AppNavHost
- `UI` layer:
  - components:
    - address: AddressCard
    - cart: CartHeader, CartItemCard, OrderSummaryCard
    - checkout: CastPaymentOption, CheckoutFormField, CheckoutHeader, ShippingMethodRadio
    - common: BottomNavigationBar, EmptyState, SideBar, TopAppBar
    - payment: CardPreview, PaymentCard, PaymentMethodOption
    - product: ImageCarousel, ProductCard, ProductGrid, VerticalProductGrid
    - voucher: VoucherCard
    - wishlist: WishlistProductCard
  - screens:
    - about: AboutUsScreen
    - address: AddressScreen
    - cart: CartScreen, CartViewModel
    - checkout: CheckoutCompletedScreen, CheckoutPaymentScreen, CheckoutScreen, CheckoutViewModel
    - home: HomeScreen
    - payment: AddNewCardScreen, PaymentScreen

    - product: ProductScreen
    - resultSearch: ResultSearchScreen
    - setting: NotificationSettingScreen, PrivacyPolicyScreen, SettingScreen, TermsOfUseScreen
    - voucher: VoucherScreen
    - wishlist: WishlistScreen

Contribution Description

In this project, my primary focus was on developing the user interface components and screens that form the core of the app's interactive experience. I designed and implemented a range of UI components for example the TopAppBar and BottomNavigationBar, to ensure consistent navigation and branding across the application. My focus was on creating components for key features, including AddressCard for managing delivery details, CartItemCard and OrderSummaryCard for the shopping cart, and ProductCard with ImageCarousel for displaying items in grids or details views. These components were built using Kotlin and Jetpack Compose, prioritizing reusability, responsiveness to Material Design principles to enhance usability on various Android devices.

I also in charge for several critical screens, starting with the Home screen, which integrates featured products and collections for an engaging entry point. The Cart screen handles item management and summaries, while the Checkout screen manages multi-step processes for example shipping and payment selection. Other screens like the Search and Result Search screens for product discovery, the Wishlist screen for saving favorites, and utility screens such as Voucher, Payment, Address, Order Info, Notification, and Settings. Each screen was structured with view models (e.g., CartViewModel, CheckoutViewModel) to separate concerns, facilitating data handling and state management.

Throughout implementation, I ensured integration with the app's navigation layer via AppNavHost, enabling smooth transitions between screens. Any deviations from the initial wireframes, such as refined layouts for better touch interactions, were made to improve performance and user feedback. Overall, my work focused on creating an intuitive and visually consistent interface that backs up the app's main features, while still making the solid best practices in mobile dev to meet our project targets.

Student Name: Nguyen Duc Trung

Student ID:

Responsibilities

- Session Management: Jetpack DataStore auth persistence.
- Database Architecture: Room Database, Repository pattern.

- Cart System: Persistent cart storage and logic.
- Address Management: Multi-address and validation.
- Payment Card: Secure storage, VisualTransformation UI.
- Voucher System: Discount database and calculation.
- Checkout Logic: Dynamic pricing and totals.
- Order Tracking: Purchase history, Order-Item database.
- Auth & Profile: Login, SignUp, Profile screens.
- System Integration: End-to-end navigation and data flow.

Files Implemented

- `data` layer:
  - local:
    - dao: AddressDao, CartDao, CartItemDao, OrderDao, OrderItemDao, PaymentCardDao, ProductDao, UserDao
    - database: AppDatabase
    - entity: AddressEntity, CartEntity, CartItemEntity, OrderEntity, OrderItemEntity, PaymentCardEntity, ProductEntity, VoucherEntity
    - repository: CartRepository, OrderRepository, ProductRepository, UserRepository
    - session: SessionManager
  - remote: DivisionApiService, NominatimApiService
- `navigation` layer: AppNavHost
- `UI` layer:
  - components:
    - address: SearchableDropdown
    - auth: AuthComponents
    - cart: CartItemCard
  - screens
    - address: AddressControl, AddressScreen, AddressViewModel, DivisionViewModel
    - auth: LoginScreen, SignUpScreen

* cart: CartViewModel
* checkout: CheckoutPaymentScreen, CheckoutPaymentViewModel, CheckoutScreen, CheckoutViewModel
* payment: AddNewCardScreen, PaymentViewModel
* profile: ProfileScreen, ProfileSetting, ProfileViewModel
* search: SearchScreen
* voucher: VoucherViewModel

Contribution Description

I was primarily in charge of the UWE Shopping App's core architecture and the integration of its essential commerce features. Establishing a scalable foundation while guaranteeing a safe and smooth user experience was my goal.

I structured the data layer using Room Database combined with the Repository pattern. This approach ensured a clean separation of concerns and reliable data handling between the database and UI. For session management, I utilized Jetpack DataStore, which provided a secure method for storing user login states and preferences without the overhead of heavier solutions.

On the functional side, I engineered the Persistent Cart and a dynamic Voucher System. These modules required complex business logic to handle real-time price adjustments and discount calculations accurately. I also built the Checkout Logic, focusing on security by implementing address validation and custom VisualTransformation to mask sensitive payment inputs.

Finally, I integrated these features into a cohesive workflow. I developed an Order Tracking module linked to the order history database and utilized Jetpack Navigation Compose to manage the routing between Authentication, Profile, and Shopping screens. This guaranteed a fluid data flow and a modular codebase that adheres to modern Android development standards.

7 Code Structure and Quality

This project uses usual MVVM (Model–View–ViewModel) structure, which makes project easy to read, lets it grow for an later, and can be maintained well.The source code files are arranged in logical way with different folders where data folder is for things like local DAO, database and entity types; ui.screens folder is for composables found in screens; ui.components folder is used for the UI stuffs that can be reused; an navigation section is kept separately.

Reuse of composable UI pieces is done with separation, like for the navigation bars made by user, product display grids or input box fields so repetition is decreased.

Functional programming approach are sometimes used in ViewModel's data logic, with features like filter(), sortedByDescending() or map() from Kotlin's tools.

Strictly, this code base uses nothing but Jetpack Compose and Kotlin rules for names, like PascalCase is for composable items, camelCase is for functions and variables which increases clarity and easier to manage

8 Flow, Navigation, and User Experience

The final application closely follows the user flow defined in the design stage, with refinements introduced during implementation to improve usability and technical robustness. Navigation is implemented using **Jetpack Navigation Compose**, providing a centralized and scalable navigation architecture.

A single `AppNavHost` acts as the navigation coordinator for the application and manages over twenty routes corresponding to authentication, product browsing, checkout, profile management, and support features. Parameterized navigation is used extensively, allowing screens to receive type-safe arguments such as product identifiers, order references, and pricing values. This approach ensures consistency and reduces navigation-related errors.

User flow begins with an authentication state check performed at application startup using lifecycle-aware side effects. Based on the login state, users are automatically redirected either to the authentication flow or directly to the main application content, ensuring a seamless entry experience without unnecessary user input.

Primary navigation between core sections (such as Home, Search, Cart, Orders, and Profile) is supported through a persistent bottom navigation bar. This enables quick transitions between major features while preserving user context and minimizing disruption to ongoing tasks.

The browsing and purchasing flow was implemented to match real-world shopping behaviour. Users can navigate from product listings to detailed product screens, add items to the cart, and proceed through a structured checkout process consisting of order review, address selection, payment method confirmation, and final order submission. Visual feedback mechanisms, including loading indicators and state-based UI updates, are displayed during asynchronous operations to maintain responsiveness and clarity.

Additional flows, including order history review, wishlist management, notifications, and chatbot interaction, were integrated into the overall navigation structure without disrupting the primary user journey. Consistent back navigation and predictable screen transitions were applied across all views to ensure a smooth and intuitive user experience.

Overall, the implemented navigation and flow reflect the original design intent while benefiting from practical adjustments made during development to improve clarity, responsiveness, and scalability.

9 Functionality Coverage

Most planned features defined during the design phase have been fully implemented. The application supports essential e-commerce functionality, including user authentication, product browsing, cart and checkout workflows, address management, wishlist handling, search with filtering and sorting, customer support chat interaction, and order tracking.

Layouts are designed to be responsive across multiple screen sizes using Jetpack Compose responsive design principles, such as flexible modifiers, constraint-aware layouts, and adaptive grid components. Any missing or partially implemented features, particularly real-world payment integration, are justified by academic and technical constraints, with simulated logic used to represent payment flow behavior.

10 Complexity and Skill Development

The project demonstrates complexity beyond a basic Android application by incorporating modern development techniques such as reactive state management, structured navigation, and custom UI construction. State is managed using a combination of local composable state and observable data streams from the persistence layer.

Asynchronous data processing is handled using Kotlin Coroutines and lifecycle-aware scopes, ensuring database operations do not block the UI thread and preventing application unresponsiveness. The implementation of custom UI elements, including navigation sidebars, gradient-based banners, and multi-condition filtering logic, further reflects advanced skill development beyond introductory tutorials.

11 Use of External Resources and Tools

Based on the `build.gradle.kts` configuration file, the project integrates the following professional tools and libraries:

- **Networking and API**
  - `com.squareup.retrofit2:retrofit:2.9.0`: Serves as the primary client for connecting to and retrieving data related to the 63 provinces from an external API.
  - `com.squareup.retrofit2:converter-gson:2.9.0`: Handles automatic serialization and deserialization of JSON data into Kotlin data classes.
- **Local Persistence**
  - `androidx.room:room-runtime` and `room-ktx`: Implement the local database system for persisting product and order data.

  - `androidx.datastore:datastore-preferences`: Manages lightweight data storage for user authentication state (sessions) and user preferences.

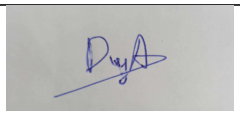
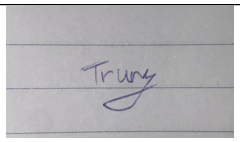
- **UI Framework**
  - `androidx.compose.material3:1.4.0`: Utilizes the Material Design 3 system for building a modern and responsive user interface.
  - `androidx.compose.material:material-icons-extended:1.7.4`: Provides a comprehensive set of standardized icons for the application.

- **Navigation and Logic**
  - `androidx.navigation:navigation-compose:2.8.0`: Manages the navigation graph and screen transitions within the application.
  - `androidx.lifecycle:lifecycle-runtime-ktx`: Handles application lifecycle events and integrates Kotlin Coroutines for efficient asynchronous task management.

- **Development Tools**
  - **KSP (Kotlin Symbol Processing):** Utilized to optimize build performance and annotation processing for the Room Database.

12 Declaration

We confirm that the contributions listed in this report are accurate and have been agreed upon by all group members.

Name	**Student ID**	**Signature**	**Date**
Tran Duy Anh	23085483		07/01/2026
Nguyen Nhat Minh	23085487		07/01/2026
Nguyen Duc Trung	23085492		07/01/2026