|  |  |  |
| --- | --- | --- |
| **Qualification** | | **Module Number and Title** |
| **HD in Computing and Software Engineering** | | **CSE5011 - Mobile Application Development** |
| **Student Name & No.** | | **Assessor** |
| **Mohamed Rashard CL/HDCSE/CMU/112/15** | |  |
| **Hand out date** | | **Submission Date** |
| **2024-06-19** | | **2024-08-07** |
| **Assessment type**  **WRIT1-**  **Coursework** | **Duration/Length of Assessment Type**  **3000 words equivalent** | **Weighting of Assessment**  **100%** |
| **Learner declaration** | | |
| **Mohamed Rashard CL/HDCSE/CMU/112/15 I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Marks Awarded** | | | |
| **First assessor** | |  | |
| **IV marks** | |  | |
| **Agreed grade** | |  | |
| **Signature of the assessor** |  | **Date** |  |

**FEEDBACK FORM**

**INTERNATIONAL COLLEGE OF BUSINESS & TECHNOLOGY**

**Module:**

**Student:**

**Assessor:**

**Assignment:**

**Strong features of your work:**

**Areas for improvement:**

**Marks Awarded:**

**Task A**

**Critical Comparison of Mobile Operating Systems, Development Tools, and Technologies for Dog Food App**

Developing a dog food app requires careful planning and selecting the right tools. Android and iOS are the two dominant mobile operating systems, offering flexibility and a wider range of devices. Native development requires separate codebases for Android and iOS, offering better performance but increased development time and cost. Cross-platform development, using tools like Flutter, React Native, and Ionic, allows for faster development, lower costs, and easier maintenance.

Technology stacks contribute to app functionality, including backend, payments, user interface, and content management systems. The optimal approach depends on budget and time, performance, and team expertise. Cross-platform development is ideal for faster time-to-market and lower upfront costs, while native development may be necessary for advanced functionalities.

Flutter is a great choice for a dog food app due to its focus on educational content, not heavily reliant on device features, and potential for a wider audience. However, native development might be more suitable for features like a dog food scanner or AR integration.

Beyond the initial choice, maintaining the app is crucial. Optimizing images and resources to keep the app size manageable and considering long-term maintenance can help maintain the app's success. By carefully evaluating these factors, you can make an informed decision that ensures both happy dogs and satisfied owners.

(Matellio, How to develop a dog food calculator app? 2024)

**Mobile Operating Systems**

**iOS**

**Advantages:**

* Apple's strict guidelines and controlled hardware ecosystem provide a user-friendly experience for dogs seeking top-notch dog food
* Ensures a seamless journey across all iOS devices, offering a familiar and hassle-free experience for dog owners
* Apple's strict security measures create a safer environment for personal information and financial transactions
* iOS users are more engaged with apps, potentially leading to greater profits for dog food apps
* It's important to consider potential drawbacks before making a final decision
* Overall, iOS presents a promising platform for dog food apps

(Fowler, 19 must-have apps for dog owners in 2024 2024)

(Apple Pay Security and Privacy Overview)

**Disadvantages:**

* Restricted ecosystem
* Costly development requirements
* Need for Apple devices to use Xcode IDE
* High initial investment in development tools and hardware
* Developing for iOS can be challenging due to the above factors

However, iOS offers a premium user experience, robust security, and strong monetization potential

(17 et al., Android vs IOS Development: A comprehensive guide for developers)

(Pros and cons of IOS app development)

**Android**

**Advantages:**

* Android is widely used and open-source, developers can reach a wider audience, which increases its market share in the worldwide mobile operating system industry.
* Developers may create unique app experiences for a variety of devices and user preferences thanks to Android's open-source nature and development freedom.
* Due to its freely available tools and resources, which do not require a specific hardware investment, Android lowers the entrance barrier for developers and makes it more accessible for individuals and small teams.

(Android Statistics (2024) - business of apps)

**Disadvantages:**

* Due to fragmentation between hardware and OS versions, Android's open-source design and wide range of supported devices can lead to compatibility problems and require intensive testing and optimization to ensure correct application functionality across various devices.
* The Android platform's open nature increases vulnerability to security threats, allowing malware and malicious applications to infiltrate, potentially compromising user data and device security, hence developers must be cautious when selecting third-party libraries.

(Moscaritolo, Android fragmentation up with 18,796 different devices 2014)

(Insights that connect you to Success | opensignal)

**Development Tools and Technologies**

**Android Studio**

**Advantages:**

* The official integrated development environment (IDE) for Android apps, Android Studio offers tools for testing, debugging, and performance optimization on all Android devices, including TVs, tablets, phones, and wearables, all in a single, unified environment.
* The large developer community that supports Android Studio provides tools and documentation, facilitating rapid problem-solving and keeping up with the most recent Android development best practices. This community also helps to drive Android Studio's expansion.

(Ghanchi, The major advantages of Android Studio App Development 2022)

**Disadvantages:**

* The large developer community that supports Android Studio provides tools and documentation, facilitating rapid problem-solving and keeping up with the most recent Android development best practices. This community also helps to drive Android Studio's expansion.
* Android Studio is a resource-intensive IDE that demands significant system resources like RAM and processing power, which can be a challenge for developers working on older or lower-spec computers, potentially slowing down the development process.

(Android Studio Reviews 2022 - Capterra)

**Xcode**

**Advantages:**

* Due to its close interaction with Apple's platforms, Xcode is the official integrated development environment (IDE) for iOS, macOS, tvOS, and watchOS applications. It provides excellent support and performance.
* A variety of sophisticated tools for developers are available in Xcode, such as interface design, performance analysis, and debugging. Developers can use it to step through code, check variables, and find problems. App performance is profiled using the Instruments tool, which also helps to find bottlenecks and maximize efficiency. Designing user-friendly interfaces for Apple's platforms is made easier with the help of Interface Builder and SwiftUI.

(Admin, How to use Xcode and its advantages? 2022)

**Disadvantages:**

* Xcode is exclusively available on macOS, limiting accessibility for developers using Windows or Linux operating systems, requiring access to a Mac computer for app development.
* When working on sophisticated code or huge projects, Xcode, a powerful Mac operating system, can be resource-intensive and requires enough RAM, processing power, and storage to run smoothly. This can be a costly investment for developers because it uses macOS and occasionally runs slowly, which could make it less expensive overall.

(Duck et al., How to prevent xcode compiling a package for a target which is not designated? 1969)

**Flutter**

**Advantages:**

* The programming language Dart, which Flutter uses, compiles to native code for optimal performance on the iOS and Android platforms, guaranteeing that apps are developed with platform-specific technologies.
* Flutter provides pre-made, editable widgets that follow the design standards of both iOS and Android, minimizing the requirement for independent UI development and guaranteeing a unified user experience on both platforms.
* Developers may immediately iterate on their code and view the changes in real-time without having to restart the app thanks to Flutter's "Hot Reload" feature. This allows for a more efficient workflow and considerably accelerates the development process.

(AltexSoft, Pros and cons of Flutter App Development 2022)

**Disadvantages:**

* Though it is growing quickly, the Flutter community is still smaller than that of well-known cross-platform development tools like React Native, which means there are less third-party integrations, libraries, and resources accessible. The Flutter community is growing, which means there aren't as many third-party libraries and packages available as there are for other frameworks, as one reviewer on G2 pointed out.
* Flutter apps are typically larger than native iOS and Android apps due to the inclusion of the entire Dart runtime and rendering engine within the app package. This is a major drawback, as noted in a Capterra review, as it results in a more complex and detailed app structure.

(Flutter reviews 2022: Details, pricing, &amp; features | G2)

**React Native**

**Advantages:**

* React Native is a popular tool that uses JavaScript, making it accessible to many developers, reducing the learning curve for those already proficient in the language.
* Apps may operate at speeds similar to native iOS and Android apps thanks to React Native's usage of native components for near-native performance, which guarantees a flawless user experience.
* React Native's robust community support and extensive ecosystem enable developers to effortlessly incorporate third-party services and features into their apps.

(Jones, 15 best mobile apps built with react native in 2023 2023)

**Disadvantages:**

* React Native faces challenges with complex UI and animations due to its JavaScript rendering, potentially causing performance issues and UI complexity limitations compared to native development.
* React Native may experience performance issues due to the bridging process between JavaScript and native code, which can introduce latency and slow down the app's performance, particularly for complex tasks.

(Performance overview · REACT NATIVE 2024)

**Ionic**

**Advantages:**

* Ionic lowers the learning curve for web developers by making use of common web technologies like HTML, CSS, and JavaScript. This makes the transition from web development to mobile app creation easier.
* Ionic offers a wide range of plugins and integrations that let developers take advantage of the physical features of the device, such as the camera, GPS, and sensors.
* Ionic offers a cross-platform development solution that enables developers to write code once and deploy it across multiple platforms, saving time and resources.

(Ionicframework, The cross-platform App Development leader)

**Disadvantages:**

* Ionic apps may not work as well as native apps or apps created with other cross-platform tools like Flutter. This is because Ionic renders its user interface (UI) via web views, which might result in latency and performance overhead.
* Ionic's dependency on web views makes it difficult to provide a native look and feel. Even though Ionic offers UI customization options, it might be challenging to replicate the native user interface in its entirety, particularly for intricate or highly interactive apps.

(Ionic: An OutSystems company reviews &amp; product details - G2)

**Critical Comparison and Justification**

For the dog food app scenario, a cross-platform approach using Flutter or React Native emerges as the most suitable option due to the following reasons:

1. **User Experience and Performance**

* **Flutter**
* Flutter is an open-source mobile application development framework developed by Google, offering a high-performance and consistent user interface across iOS and Android platforms. Its key features include a native-like user experience, cross-platform development, a custom rendering engine, an extensive widget library, reactive programming, and an active community of developers.
* Flutter's reactive programming model, based on the Dart language, allows for efficient animations, transitions, and UI updates, resulting in a more responsive and efficient user experience. It also provides a wide range of libraries, tools, and resources, making it a popular choice for building modern, engaging mobile applications.
* Flutter's design is built on the Dart programming language, allowing for fast and efficient rendering of UI elements, even on low-end devices. Its extensive widget library, ranging from basic elements to complex components, allows developers to quickly build feature-rich applications.
* Flutter's active community of developers contributes to a rich ecosystem of libraries, tools, and resources, providing developers with access to a wide range of solutions to common development challenges. Overall, Flutter's combination of high performance, consistent user interface, and cross-platform capabilities makes it a popular choice for mobile application development.

(Flutter architectural overview)

* **React Native**
* React Native is an open-source framework that focuses on building high-performance, cross-platform mobile applications. It achieves near-native performance by using native components and a robust community ecosystem. Native components, such as `` and ``, directly render underlying platform-specific UI elements on iOS and Android, allowing apps to achieve a native look and feel. The architecture is based on a reactive programming model, allowing for efficient and smooth UI updates without manual DOM manipulation. This approach maintains a high frame rate and a responsive user experience.
* React Native's active community of developers contributes a wide range of libraries, tools, and resources, accelerating the development process and ensuring a consistent user experience across applications. However, developers may still need to apply specific optimization techniques to ensure smooth application running. Strategies like using `FlatList` or `SectionList` to render large lists, leveraging memoization with `useMemo`, and optimizing image loading and bundle size can help maintain a responsive and efficient user interface.
* In summary, React Native's use of native components, reactive programming model, and strong community support enable the development of responsive and interactive mobile applications that closely match the performance of native counterparts. By applying targeted optimization techniques, developers can further enhance the user experience and ensure their React Native apps deliver near-native performance.

(Core components and native components · REACT native 2024)

1. **Development Time and Cost**

* **Cross-Platform Development**
* Cross-platform development frameworks like Flutter and React Native enable code sharing across iOS and Android, reducing development time and costs. These frameworks are essential for quickly bringing applications to market and iterating based on user feedback. Flutter, an open-source UI framework developed by Google in 2018, is known for its high performance and ease of use, making it suitable for complex applications. Its hot reload feature allows developers to make changes to the code and see the changes immediately, making the development process faster and more efficient. React Native, created by Facebook in 2015, is a mobile development framework that transforms JavaScript components into native ones, providing a native experience to users. Both frameworks offer significant advantages for cross-platform development, but the choice depends on the specific needs of the project and the developer's familiarity with the programming languages used by each framework.

(Flutter vs. react native: Choosing the Right Cross Platform Mobile App Development Framework)

1. **Maintenance and Updates**

* A single codebase simplifies the maintenance and updating of an app by requiring changes to be made once and deployed across both platforms. This simplifies the maintenance process and ensures consistent updates. It also simplifies bug fixes, feature enhancements, and performance improvements, reducing the need for separate updates for each platform or environment. This also facilitates more efficient testing, as changes can be applied uniformly across all supported platforms.
* A single codebase also simplifies deployment and version control, reducing the risk of inconsistencies or compatibility issues across different platforms or environments. Version control systems like Git can be used more effectively, enabling developers to track changes, collaborate seamlessly, and roll back if necessary with greater agility.
* Utilizing a single codebase can significantly reduce development costs and time, as it allows the group to write code once and deploy it over multiple stages or conditions. This methodology improves the advancement lifecycle and quickens time-to-advertise.

Simplified testing and maintenance are also advantages of a single codebase, as updates are deployed simultaneously across both channels, ensuring consistency and uniformity. This enables faster coding and debugging, as well as consistent coding practices and patterns throughout the project, enhancing maintainability and simplifying future updates or enhancements.

(Benefits of maintaining a single code base for Project Development)

1. **Community and Eco-System**

* **Flutter**
* Flutter, a cross-platform app development framework, is supported by Google and has over a million active developers. Its hot reload feature enhances developer productivity by allowing rapid iteration and testing. Flutter's package ecosystem supports various hardware and services, and its open source nature encourages community contributions. With Google's support, a growing user base, and extensive package ecosystem, Flutter is a reliable choice for cross-platform app development with a promising future.

(Lawson, What's next for flutter after layoffs hit Google Team 2024)

* **React Native**
* The building of mobile applications is facilitated by the extensive third-party library and tools available in the well-established React Native ecosystem. It consists of parts, frameworks, libraries, and other tools that developers may use to speed up app development, boost efficiency, and take advantage of device capabilities. NativeBase, React Navigation, Auth0, React Native Firebase, and Expo are a few well-liked tools. These libraries are continuously maintained and expanded upon by the React Native community, which also hosts conferences, online forums, and local meetups. Developers have access to a wide range of tools to improve app functionality and expedite the development process thanks to this established environment and supportive community.

(The One Technologies, Understanding the react native ecosystem: Your complete guide 2024)

1. **Customization and Flexibility**

* **Flutter**
* The customizable widget system in Flutter enables developers to design distinctive and customized app experiences. With characteristics like flex and fit, the Flexible widget regulates how a child of a base flex widget occupy available space. It is essential for developing apps that are responsive, meaning they adjust to various screen sizes and orientations while maintaining a consistent user experience. Widgets may respond to local size restrictions thanks to Flutter's LayoutBuilder widget, which makes bespoke wrappers possible.

(GeeksforGeeks, Flutter - flexible widget 2022)

* **React Native**
* React Native is a platform that allows developers to integrate native code for customization and optimization of specific app features. Its architecture allows for seamless integration of native code, allowing developers to create custom modules and components. Key benefits include access to platform-specific features, performance optimization, reuse of existing native code, and gradual migration. By providing a bridge between JavaScript and native code, React Native enables developers to create highly customized and optimized mobile apps, leveraging cross-platform development capabilities and the flexibility to integrate native code when necessary.

In conclusion, the choice between Flutter and React Native for developing a dog food app should be based on the development team's expertise and the app's specific requirements. React Native is suitable for highly customized and performance-intensive apps, as it allows for the integration of native code libraries and optimizes app features. Flutter, on the other hand, offers a consistent and visually appealing user interface across multiple platforms due to its flexible widget system and extensive customization capabilities. Both frameworks have strengths and weaknesses, but the decision should be based on the development team's familiarity with them and desired user experience.

**Task B**

**ER Diagram**

**Explanation**

An Entity-Relationship (ER) diagram shows the relationships between entities such as users, goods, orders, and reviews in pawpal nutritions. It is a visual depiction of a database structure. Orders, reviews, items, and users are important components. Users engage with the system, and products stand in for things that can be bought. Reviews are the opinions of users, while orders are the purchases made by customers. Multiple orders, the inclusion of products in orders, and user writing are examples of relationships between entities. Every entity has qualities that are described by its attributes. Entities for monitoring user profiles, payment methods, order histories, and instructional content are also included in the model. An e-commerce or educational platform database may be organized and related to using this ER diagram as a guide.

**Evaluation of the ER Diagram**

**Excellent Design:**

* **Highly detailed diagram:** The diagram includes a comprehensive set of entities and attributes, providing a clear overview of the system's structure.
* **Backed by relevant assumptions:** While not explicitly stated, the diagram seems to be based on reasonable assumptions about the system's functionality. For instance, the inclusion of 'Educational Content' suggests an e-commerce platform with educational resources.
* **Multiplicity, navigability, aggregation & compositions visible in class diagrams:** The diagram effectively utilizes UML notation to represent relationships between entities. For example, the relationship between 'User' and 'Order' is correctly depicted as a one-to-many association.
* **Excellent use of UML notation:** The diagram adheres to standard UML conventions, making it easy to understand and interpret.

**Evaluation:**

The diagram's structure is logical and consistent, but additional context about the system's requirements would strengthen the evaluation. The diagram accurately represents the core components of an e-commerce system with educational content, but without specific use cases or functional requirements, it's difficult to assess its completeness. To improve, consider areas for improvement such as normalization, inheritance, and performance optimization, focusing on data redundancy reduction, design flexibility, and performance implications.

**Overall Assessment:**

The ER diagram is a good foundation for an e-commerce system with educational content, but more information about the system's requirements and functional specifications is needed for a comprehensive evaluation. Improvements include providing clear reasons for design decisions, evaluating database design for optimization, considering inheritance opportunities, analysing data structures and relationships' impact on system performance, and creating complementary diagrams to enhance understanding.

**Use Case Diagram**

**Explanation**

This Use Case diagram illustrates the interactions between Admin and Dog Owner roles in the Pawpal Nutritions. Admin functions include managing users, handling orders, adding new items, and controlling payment options. Dog Owners can browse products, view product details, add to cart, review items, update quantity, checkout, maintain shipping and billing addresses, cancel orders, learn about breeds, read articles, watch videos, view guides, update profiles, sort products, and log in with credentials. The system also offers educational videos, guides, and personal information management.

**Excellent Design**

* **Highly detailed diagram:** The diagram comprehensively covers a wide range of functionalities for both Admin and Dog Owner roles.
* **Backed by relevant assumptions:** The diagram implies a robust e-commerce platform with educational content focused on dog care.
* **Excellent use of UML notation:** The diagram adheres to standard UML conventions, making it clear and easy to understand.

**Evaluation**

* **Good justification of the design:** The diagram's structure is logical and aligns with the expected functionalities of an e-commerce platform for pet products and information.
* **Judge validity of results:** The diagram accurately represents the core features a user would expect from such a platform. However, specific use case descriptions would provide further clarity on system behaviour.
* **Use critical reflection to evaluate the work and justify with valid explanations:**
  + **Potential for improvement:** Consider adding use case descriptions to elaborate on system interactions and pre/post conditions.
  + **Alternative scenarios:** Explore exceptional flows (e.g., out-of-stock items, failed payments) to enhance robustness.
  + **System boundaries:** Clearly define the system's scope and interactions with external systems (e.g., payment gateways, shipping services).

**Fluency (Of design)**

* **Evidence of critical analysis on different perspectives covering how, use case diagram support in designing:** The diagram effectively captures the primary functions of the system from both Admin and Dog Owner perspectives. However, additional diagrams (e.g., activity diagrams, sequence diagrams) could provide more detailed insights into system workflows and interactions.

**Overall Assessment:** The provided use case diagram presents a solid foundation for an e-commerce platform focused on dog products and information. It effectively captures the core functionalities for both Admin and Dog Owner roles. However, further refinement through use case descriptions, alternative flow considerations, and potentially additional diagrams would enhance the design's comprehensiveness.

**Specific Recommendations:**

* **Enrich use cases:** Provide detailed descriptions for each use case, including actors, preconditions, postconditions, and main flow/alternative flows.
* **Identify system boundaries:** Clearly define the system's scope and external interfaces.
* **Consider additional diagrams:** Explore the use of activity diagrams or sequence diagrams to visualize system dynamics and interactions.
* **Evaluate user experience:** Consider user-centric design principles to enhance usability and satisfaction.

By incorporating these recommendations, the use case diagram can be further strengthened to provide a more comprehensive and robust representation of the system.

**Activity Diagram**

**Explanation**

The Activity diagram illustrates a user flow within Pawpal Nutritions, specifically dog products and information. It starts with the sign-in page, then checks if the user is already signed in. If so, they proceed to the main features, browse products, order products, access educational content, manage addresses, and update profiles. The diagram emphasizes the core functionalities of browsing and purchasing products, educational content for dog owners, and managing address and profile information. However, it lacks specific product categories, payment or checkout processes, and information about user roles. Despite these limitations, the diagram provides a high-level overview of the user journey within a pet care platform.

**Evaluation of the Activity Diagram**

**Excellent Design:**

* **Highly detailed diagram:** The diagram presents a clear and comprehensive flow of user actions within the system.
* **Backed by relevant assumptions:** The diagram implies a system that offers product browsing, purchasing, and educational content related to dog care.
* **Excellent use of UML notation:** The diagram adheres to standard UML conventions for activity diagrams, ensuring clarity and consistency.

**Evaluation:**

* **Good justification of the design:** The diagram's flow is logical and aligns with the expected user journey on an e-commerce platform with educational content.
* **Judge validity of results:** The diagram accurately represents the core user interactions within the system. However, specific details about decision points, alternative flows, and error handling would enhance the diagram's completeness.
* **Use critical reflection to evaluate the work and justify with valid explanations:**
  + **Potential for improvement:** Consider adding swimlanes to differentiate between system and user actions.
  + **Alternative flows:** Include paths for actions like returning products, cancelling orders, or encountering payment errors.
  + **Error handling:** Illustrate how the system handles exceptions and provides feedback to the user.

**Fluency (Of design):**

* **Evidence of critical analysis on different perspectives covering how, activity diagrams support in designing:** The diagram effectively captures the primary user flow within the system. However, additional diagrams (e.g., class diagrams, sequence diagrams) could provide complementary perspectives on system interactions and responsibilities.

**Overall Assessment:**

The provided activity diagram presents a solid foundation for understanding the core user flow within the system. It effectively depicts the main user actions and decisions. However, further refinement through the inclusion of swimlanes, alternative flows, error handling, and potentially additional diagrams would enhance the diagram's comprehensiveness.

**Specific Recommendations:**

* **Introduce swimlanes:** Differentiate between system and user actions to clarify responsibilities.
* **Consider alternative flows:** Include paths for exceptional scenarios or user choices.
* **Address error handling:** Show how the system responds to errors or unexpected situations.
* **Complement with other diagrams:** Use Class or sequence diagrams to provide additional insights into system behaviour.

By incorporating these recommendations, the activity diagram can be further strengthened to provide a more robust and informative representation of the system's workflow.

**Relational Schema**

**Navigation Map**

**Task C**

**Implementation of User Interfaces**

**Low-Fidelity Wireframes**

**Signup Page**

**Key Features**

The user account features a username field for creating a unique identifier, an email field for communication, password recovery, and targeted promotions, a secure password field for protecting personal data, a confirmation password field for correct password entry, a sign-up button for account creation, and a "Already Registered? Login" link for existing users to log in directly instead of creating a new account. These fields are crucial for efficient account management and security, ensuring a secure and user-friendly experience.

**Justification**

The Pawpal Nutritions sign-up form features a username field, email field, password field, confirmation password field, sign-up button, and "Already Registered? Login" link. Each field has a clear label, making it easy for users to understand the required information. Placeholder text can be added to provide additional guidance, such as "Enter your username." A password strength indicator can encourage users to create more secure passwords. Error handling is included to inform users of any input issues, such as invalid email formats or weak passwords.  
  
Data privacy is clearly communicated in accordance with privacy regulations. By incorporating these features and considerations, the sign-up form for Pawpal Nutritions can enhance user experience, security, and data management.  
  
The username field allows users to create a unique identifier for their account, making it easy to log in and access their information. The email field collects the user's email address, which is essential for communication, password recovery, and potentially targeted recommendations or promotions. The password field allows users to create a secure password to protect their account information and personal data. The sign-up button initiates the account creation process, allowing users to proceed after entering their credentials. The "Already Registered? Login" link provides a convenient option for existing users to log in directly instead of creating a new account.

**Login Page**

**Key Features**

A website or application's login form is a specific place where users may enter their login information, including their username and password, to access their account or the system. A text input box called the username field is used by users to enter their email address or username. Users can enter their password in the text input box labelled "password field." Clicking on the sign-up button takes users to the sign-up page, where they may register for a new account. The wording "Not yet Registered? Sign Up" directs users who have not yet registered to the sign-up page so they can make a new account.

**Justification**

Users may type their login information in a concentrated area on the login form, which improves usability and facilitates self-authentication. Users may easily access their accounts by providing a unique identifier for themselves in the username box. Enabling secure authentication through the password field safeguards user accounts and information. A clear call to action for new users to register an account is provided via the sign-up button. By making it simpler for users to establish a new account, the "Not yet Registered? Sign Up" phrase offers an alternate route for those who do not currently have one, enhancing accessibility and user experience.

**User Homepage**

**Key Features**

The feature of the platform and a search bar are prominently displayed in the header of the "Get a Way" platform at the top of the homepage. material cards offer particular titles and descriptions of the material, arranged into parts. The information is divided into "Educational Content" and "Most Viewed," the two primary areas. For convenience, a "View all" link is included in each part. Indicating choices such as accessing the main menu, browsing user profiles, or moving to other significant portions of the platform, the navigation symbols located at the upper corners of the page function as visual signals.

**Justification**

The header of the platform explains who it is and what it does, giving users a clear idea of what it offers. Additionally, a search option is included for convenient access to certain material. Cards that are visually appealing, simple to read, and have a uniform layout are used to present content for effective consumption. The material on the site is arranged logically across divisions for ease of navigation. Users are able to narrow down on areas of interest, and "View all" options are available for more investigation. The platform's navigation symbols are thoughtfully incorporated to offer rapid and simple access to other areas, facilitating seamless section transitions and information retrieval. All things considered, the platform improves accessibility and user experience.

**User Profile**

**Key Features**

The app has a profile symbol, a section for payment methods, a settings section, a username display, and a toggle for alerts. The avatar or chosen picture for the user is represented by the profile icon. The selected username of the user is displayed in the username display, frequently with a bigger letter size or highlighted region. Users can add, amend, or remove bank accounts or credit card details, as well as adjust their payment choices. Users may change the app's notifications, account privacy, and language settings in the settings area. The notifications toggle can be enabled by users to get alerts or updates. Users must accept the conditions stated in the app by checking the privacy policy checkbox, which guarantees that their personal data is managed and safeguarded.

**Justification**

The app has a profile symbol, a section for payment methods, a settings section, a username display, and a toggle for alerts. The avatar or chosen picture for the user is represented by the profile icon. The selected username of the user is displayed in the username display, frequently with a bigger letter size or highlighted region. Users can add, amend, or remove bank accounts or credit card details, as well as adjust their payment choices. Users may change the app's notifications, account privacy, and language settings in the settings area. The notifications toggle can be enabled by users to get alerts or updates. Users must accept the conditions stated in the app by checking the privacy policy checkbox, which guarantees that their personal data is managed and safeguarded.

**Products Page**

**Key Features**

The platform's header section prominently displays the platform's feature and a search bar for easy product search. Below the header are multiple product cards arranged in a grid layout, each with an eye-catching image, clear title, brief description, and price. Each card also has an "Add to Cart" button, allowing users to conveniently add the product to their online shopping cart with a single click. This grid layout ensures a user-friendly experience and helps customers find the right products.

**Justification**

The header establishes the identity and purpose of the platform and serves as the initial point of engagement. The platform's name and logo should be shown prominently, and a major search tool should be included for simple product searching. Users will locate what they need more quickly and have a better overall experience as a result. Product cards have a standardized structure for easy browsing and are intended to visually and conveniently highlight particular objects. It is imperative that they exhibit crucial product information, such crisp and clear photos, an evocative title, a succinct description, and a pricing, to let customers make well-informed judgments. A noticeable "Add to Cart" button would also be a good addition to product cards since it would streamline user navigation and make shopping easier.

**Cart**

**Key Features:** The item list includes a title, description, price, quantity selector, subtotal and total calculation, and checkout button. The title displays the item's name, description provides an overview, and price shows the cost of each item. The quantity selector allows users to adjust the number of items before checkout. The page likely includes real-time calculations for subtotal and total cost. The checkout button directs users to the purchase flow efficiently.

**Justification:** The cart page offers user convenience by providing title, description, and price information. It also offers flexibility and control through quantity selectors and action buttons. The price for each item, along with real-time calculations, provides transparency for informed purchases. The checkout button streamlines the purchase process, and the clear organization of information enhances usability. Overall, the cart page is designed for easy navigation and management of selections.

**Educational Content**

**Key Features:** The educational content is organized into multiple topic sections, each with a subheading and a description, and is listed sequentially down the page, with the title at the top showcasing the main topic.

**Justification:** The title of the page is clear and concise, providing users with immediate understanding of the content. It aids navigation and helps users locate the main subject of the educational material quickly. The content is organized into manageable sections, with topic subheadings for easy comprehension. The description provides detailed information about each topic, ensuring comprehensive coverage. The structure enhances usability and learning experience by ensuring a logical flow, easy navigation, information accessibility, and clear headings. The organized content encourages user interaction and engagement, making it more engaging and less overwhelming. The structure aligns the content with specific learning objectives, making it easier to achieve educational goals.

**Reviews Page**

**Key Features:** The review list displays the reviewer's name, rating, and text, presented sequentially. A dedicated section for users to submit reviews is provided, including fields for the reviewer's name, rating, and text, and a submit button for posting the review.

**Justification:** The review list includes a reviewer's name, rating, review text, and a submit button. The name field encourages transparency and personalization, while the rating field allows users to quantify their experience. The review text field allows users to elaborate on their rating and provide constructive feedback. The submit button allows users to post their review, ensuring their feedback is captured and displayed. This enhances user experience by providing comprehensive information, fostering user engagement, and building trust. The review submission section invites user interaction, making the page dynamic and continually updated with new content. The platform's clean and structured layout makes it easy for users to find the information they need.

**High-Fidelity Wireframe**

**Signup Page**

**User Profile**

**Products Page**

**Home Page**

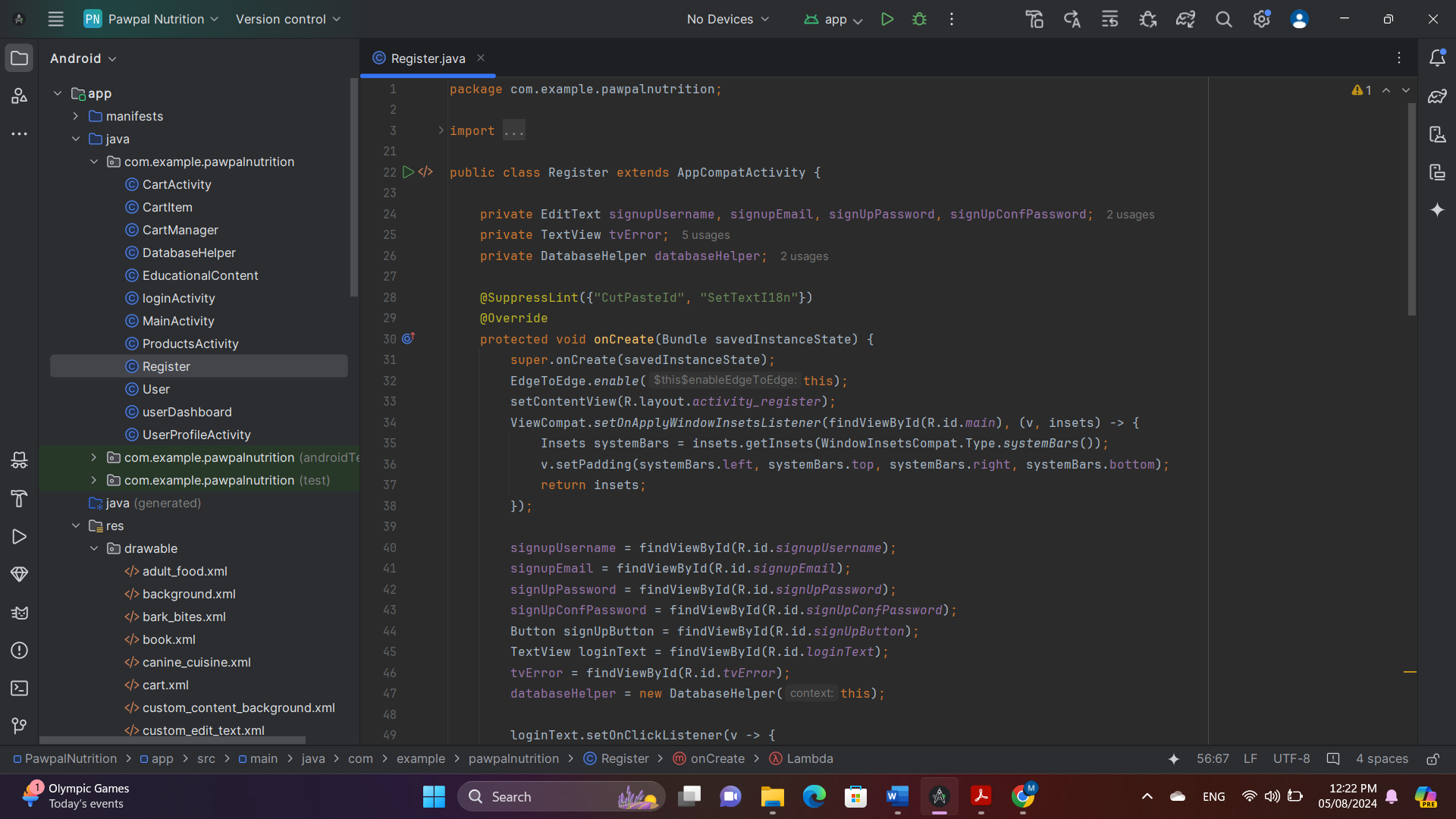
**Reviews Page**

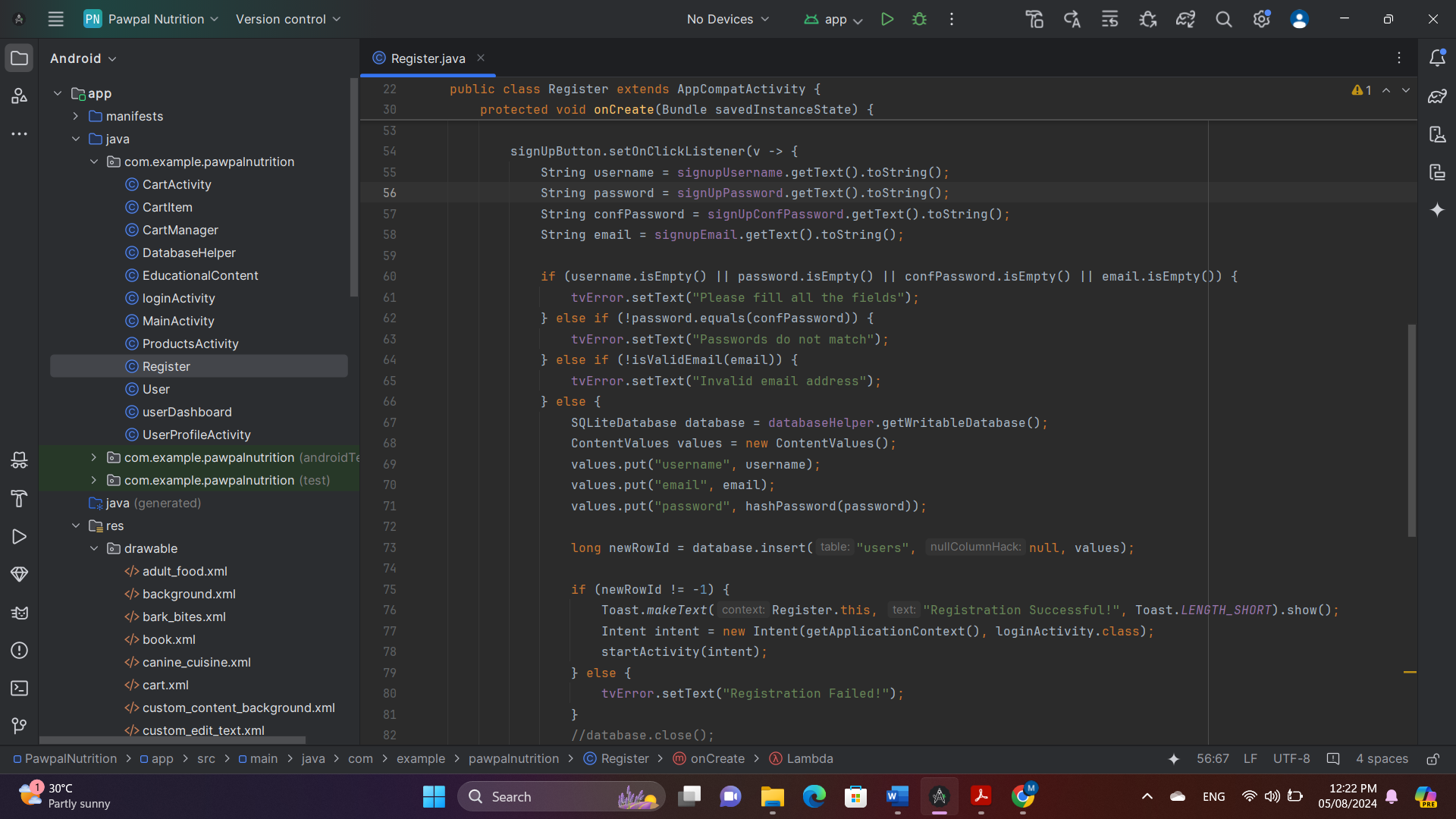
**Educational Content Page**

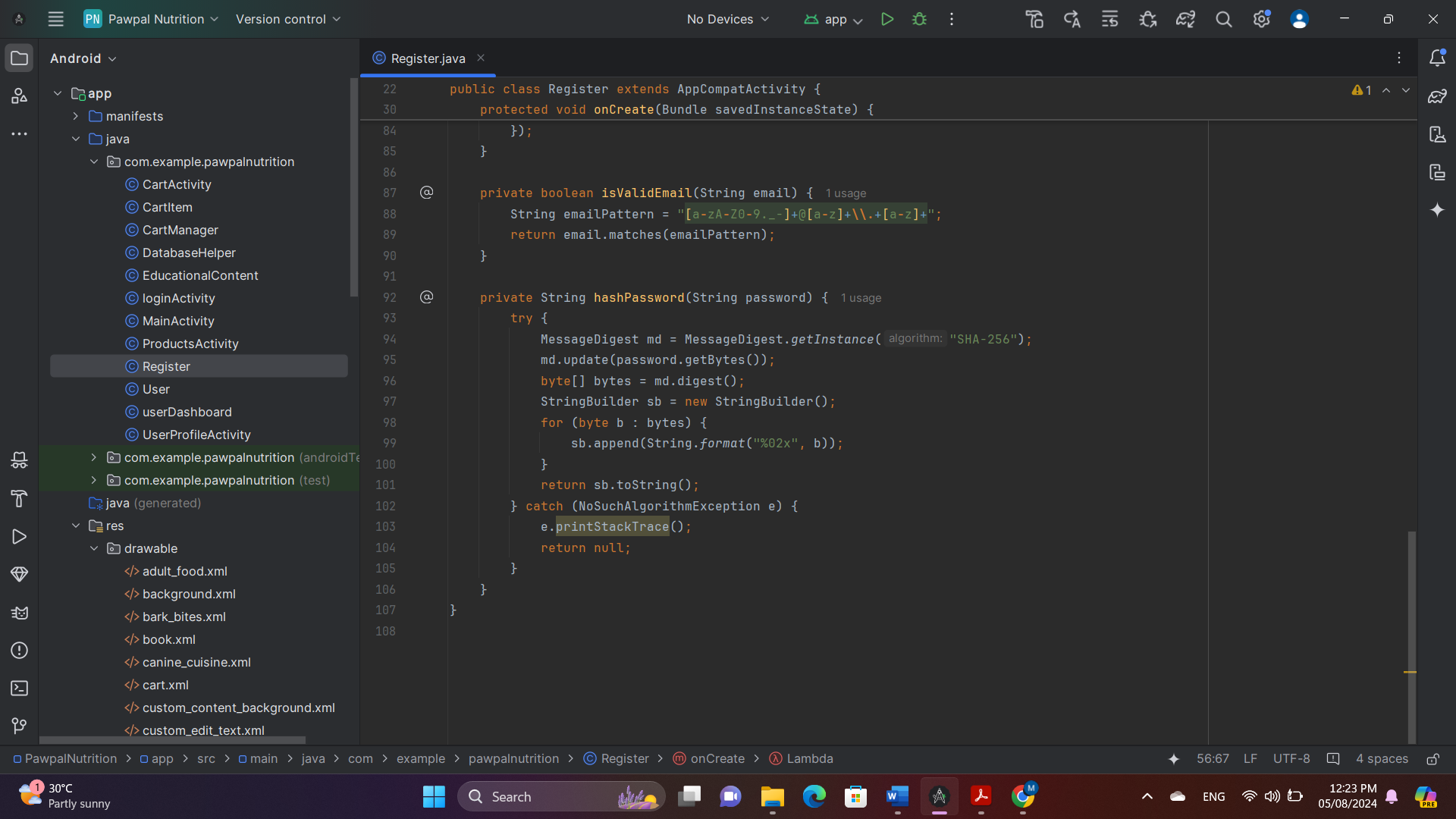
**Cart Page**

**Task D**

**Signup page**

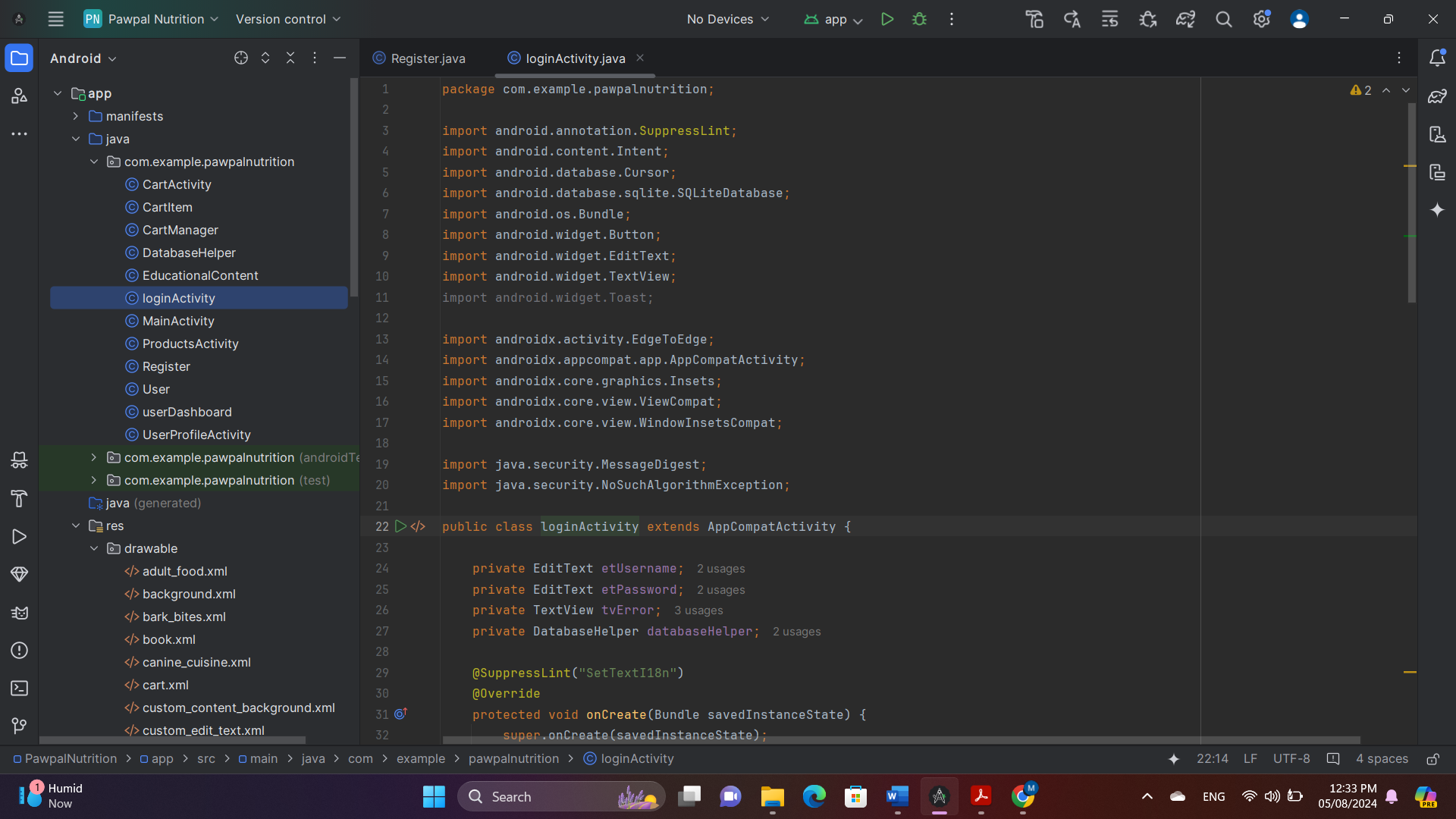


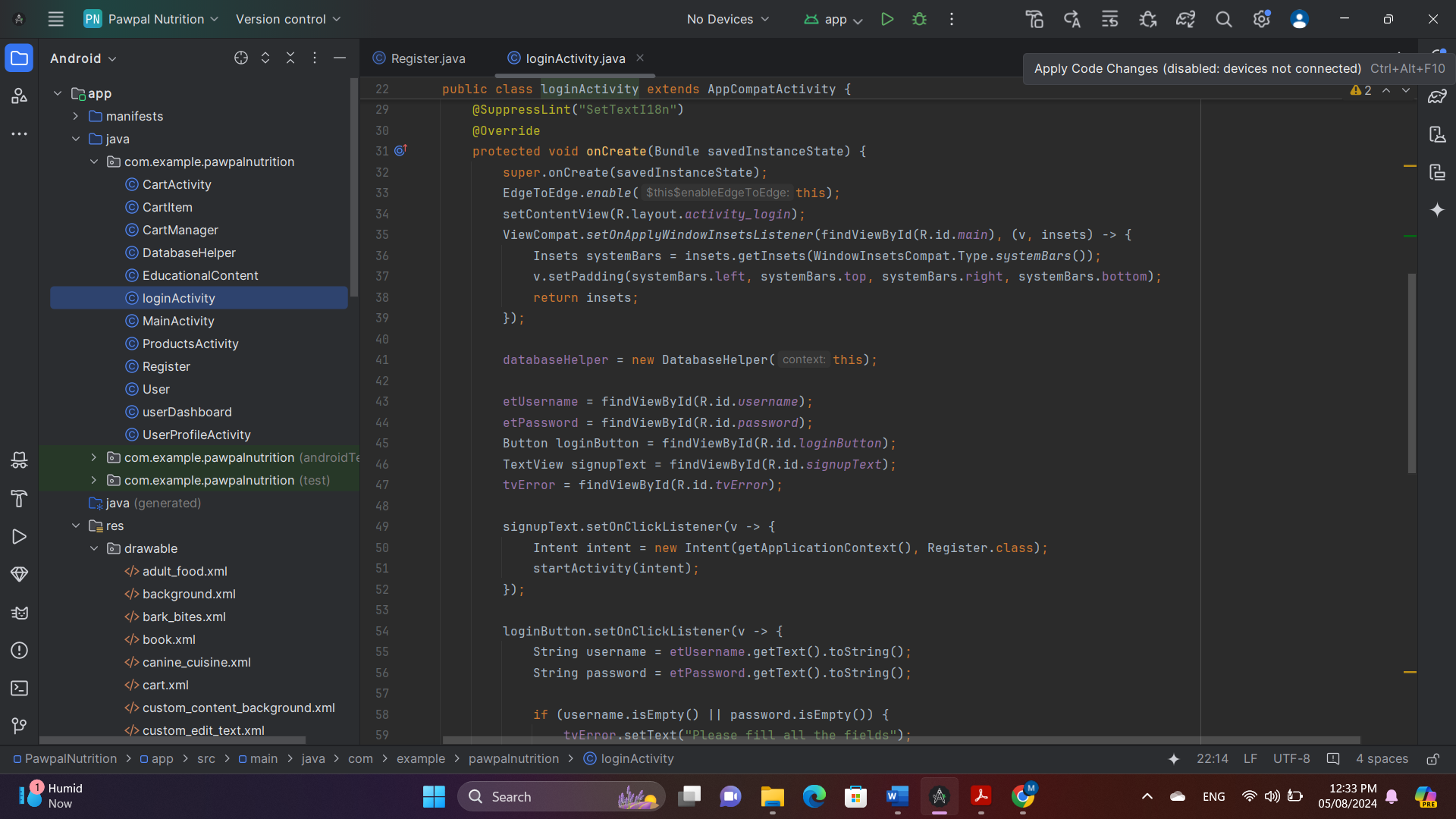


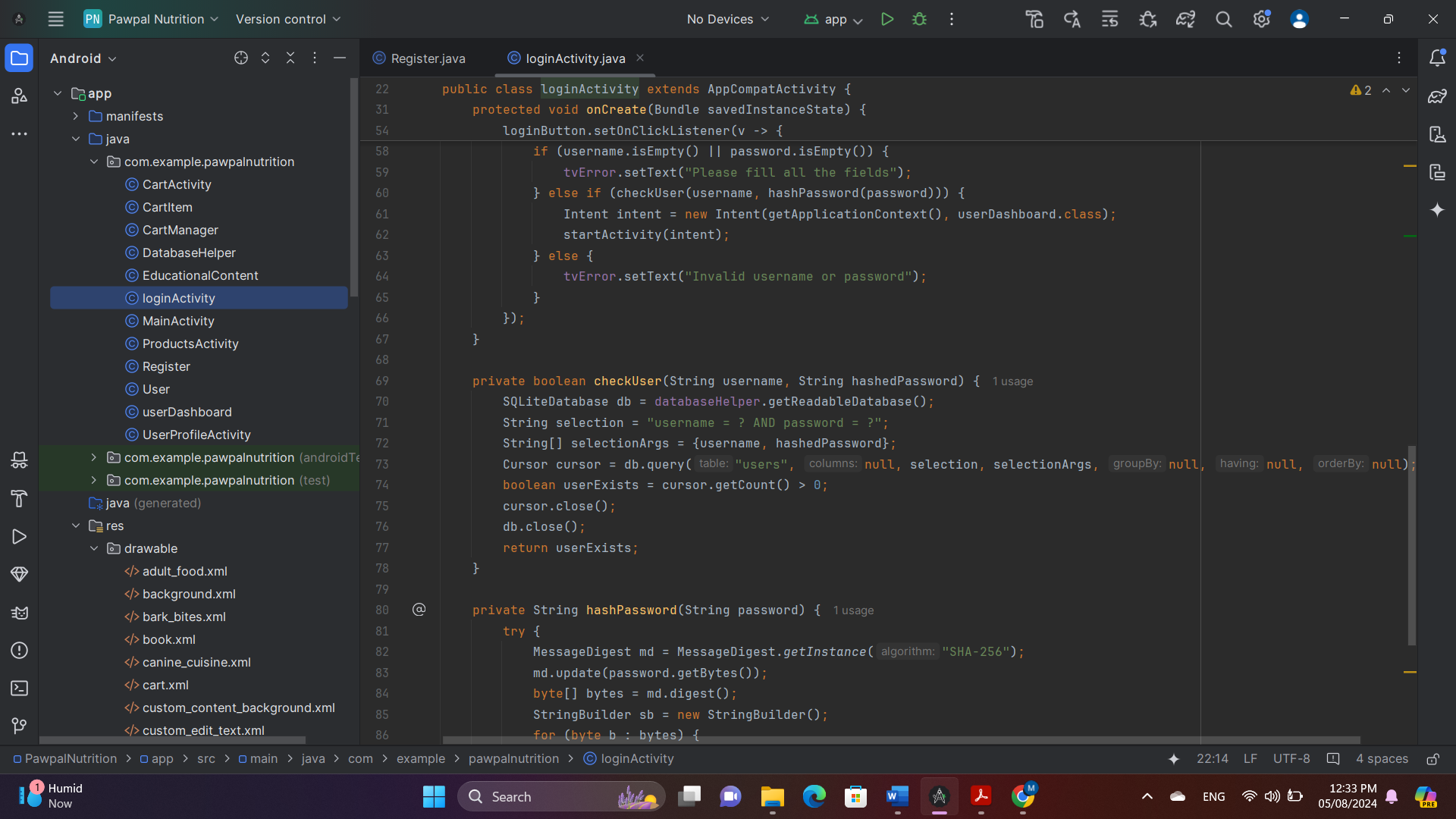


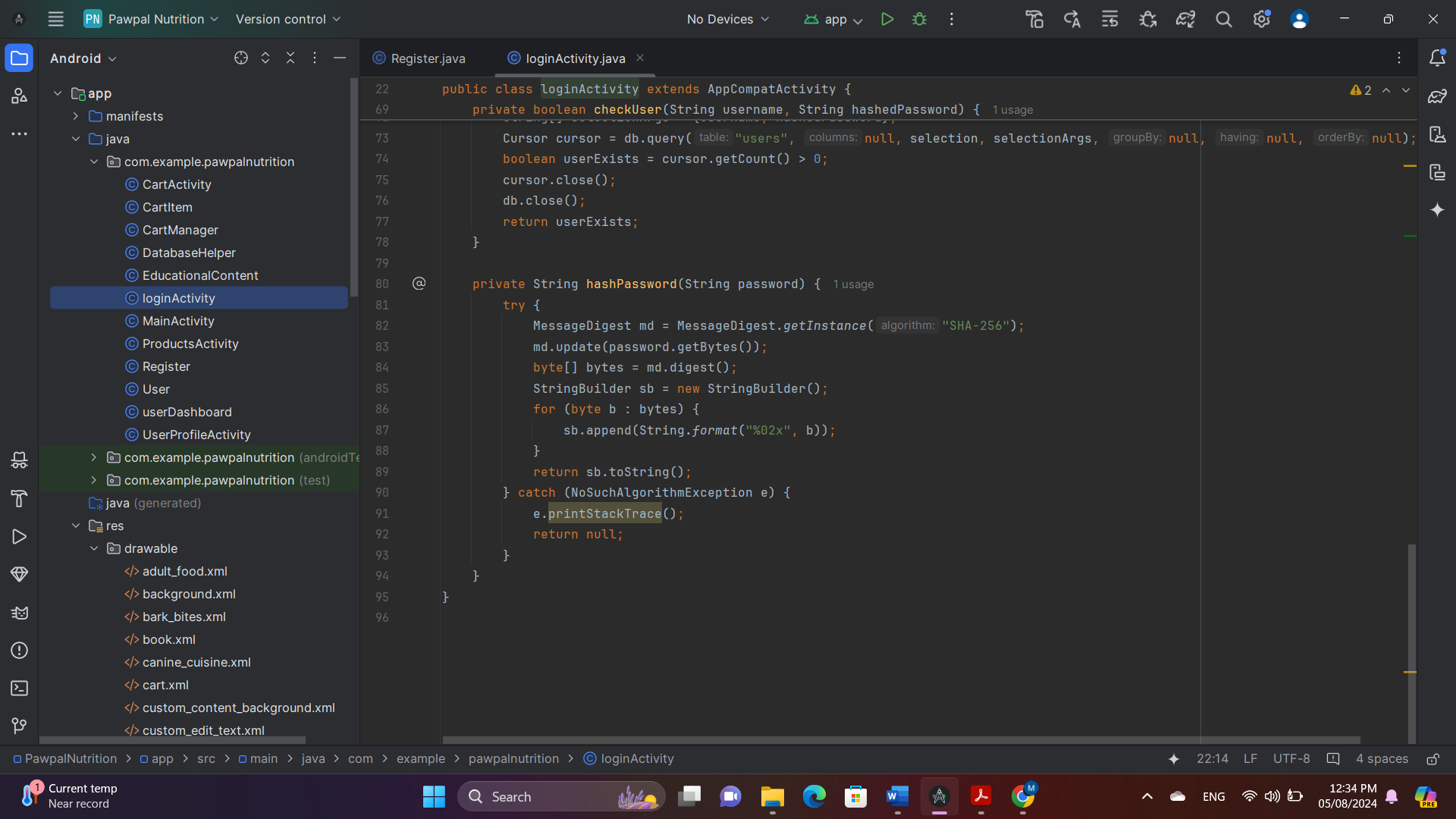
**Explanation**  
The code imports essential libraries for various functionalities within the application, including libraries for creating user interface components, handling database operations, and implementing cryptographic features for data security. The `Register` class is declared, extending `AppCompatActivity`, and initializes several variables to act as references to UI elements. The `onCreate` method sets up the user interface for the activity, defining the layout resource file, enabling full screen usage, and adjusting padding to accommodate system UI elements. References are established between declared variables and corresponding UI components, allowing programmatic interaction with UI elements. A click listener is set up for a text view prompting the user to log in, and a sign-up button click listener defines the behaviour of the button when clicked. The code checks input validation, database insertion, and hashing functions to ensure user registration. Helper methods, such as isValidEmail and hashPassword, are included to facilitate tasks related to user registration.

**Login Page**

****



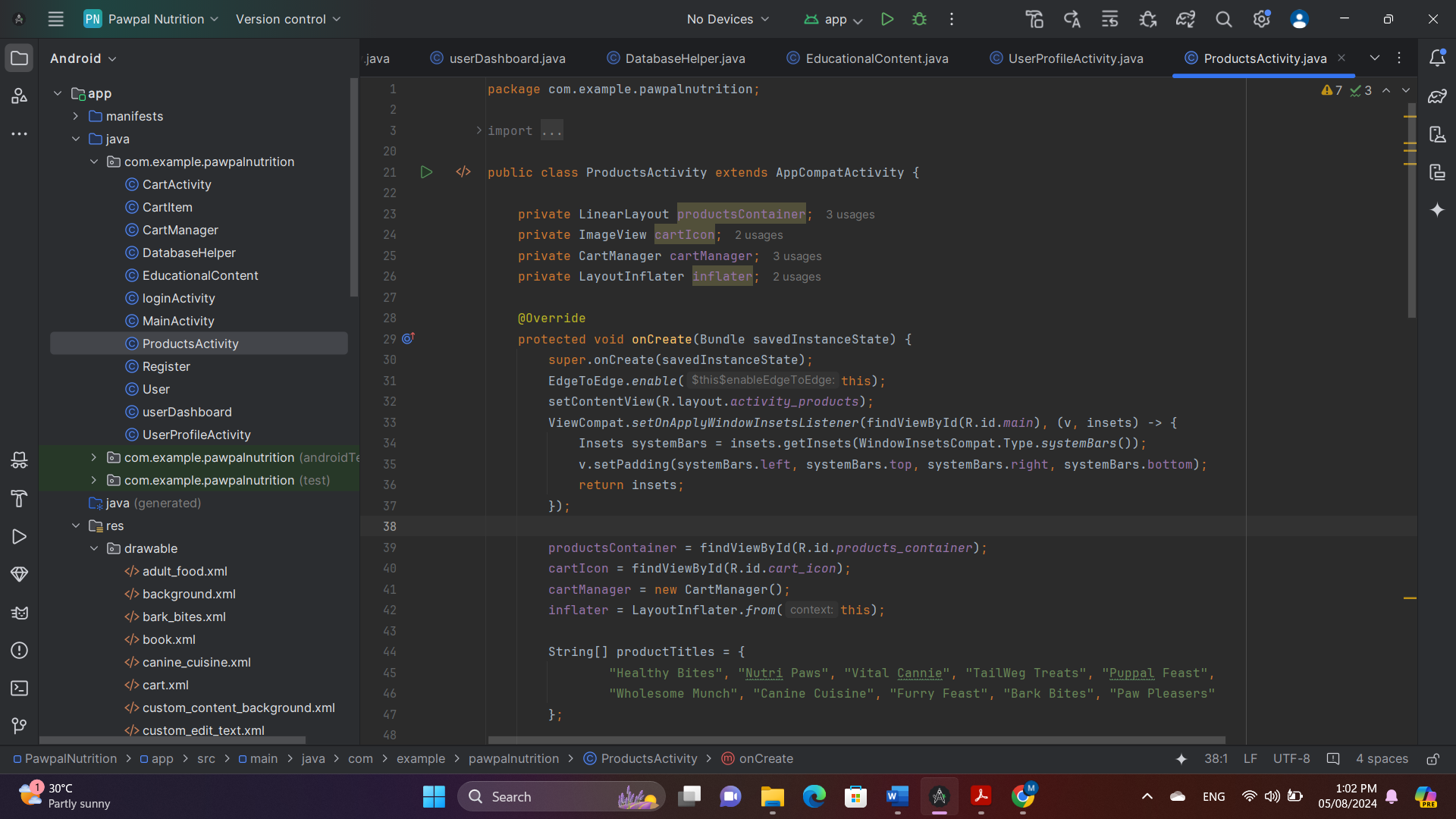


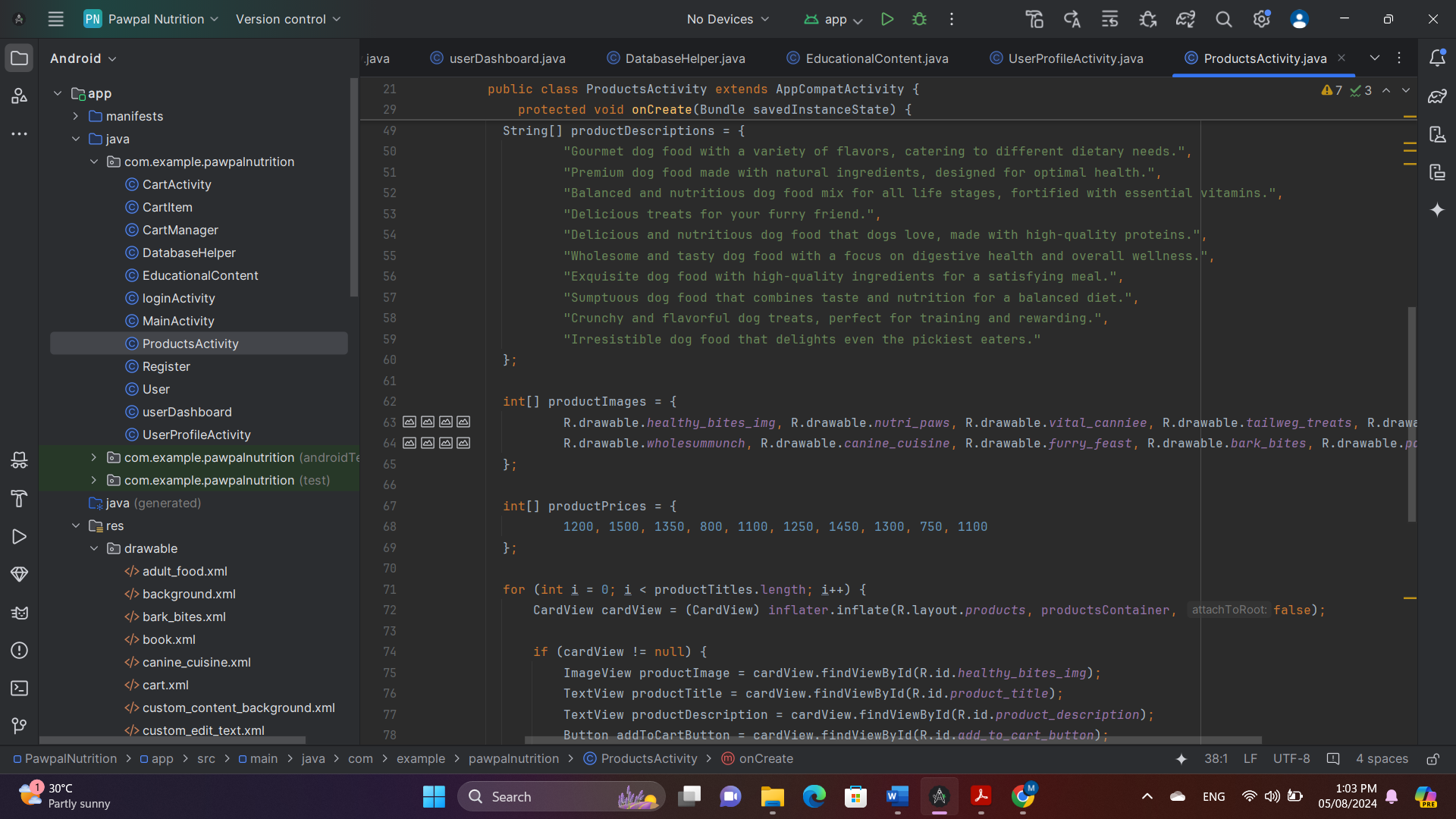


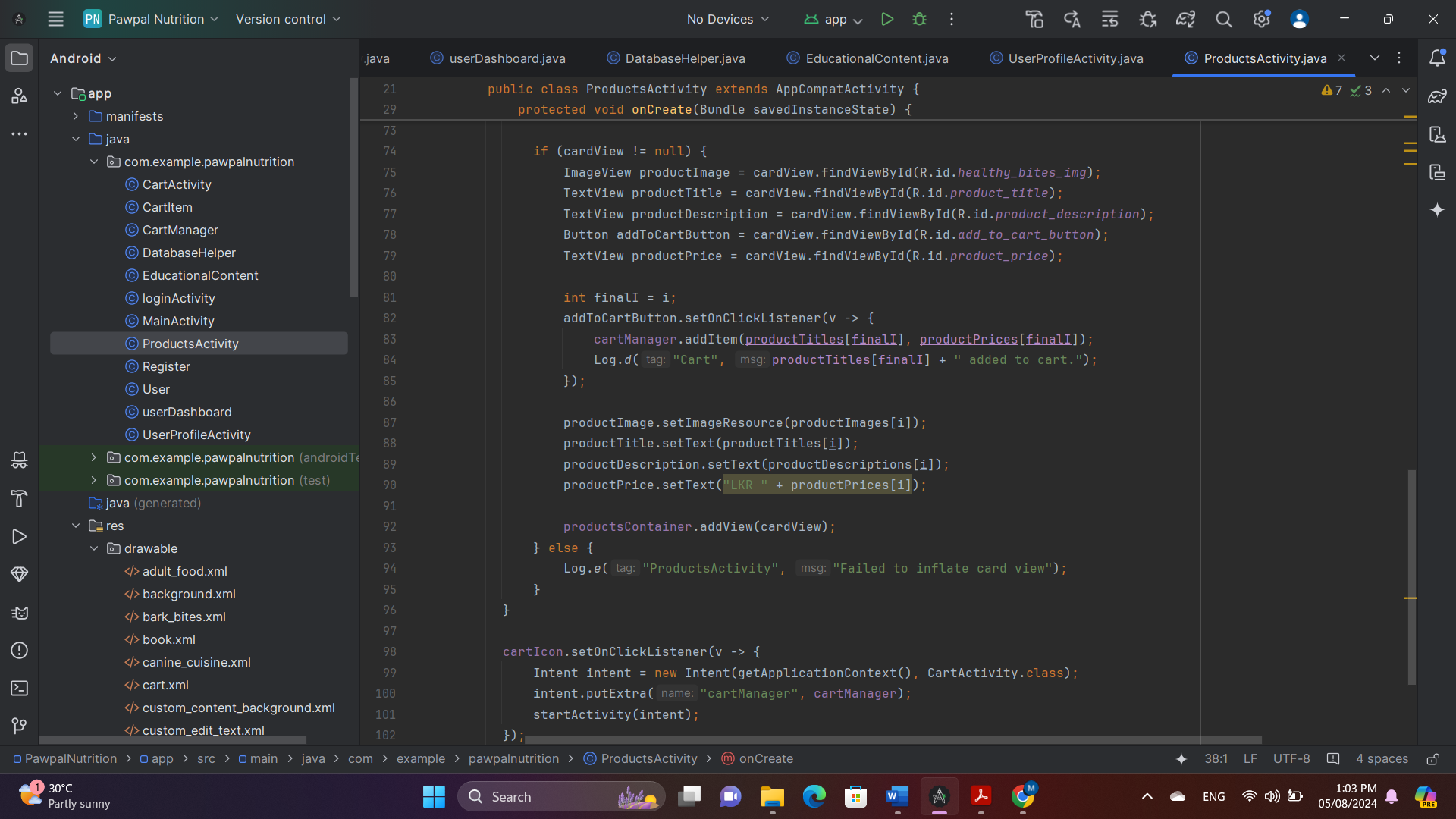
**Explanation**

This code is responsible for handling the login functionality for the PawPalNutrition app, verifying user credentials against a local SQLite database, and allowing access to the user dashboard upon successful login. It imports necessary libraries for UI components, database handling, and cryptography, and creates an activity called 'loginActivity' that extends 'AppCompatActivity'. Variable initialization includes setting the layout, enabling the activity, and adjusting padding. The code also references UI elements, such as text fields and buttons, and a helper for database operations.The onCreate method sets the layout, enables the activity, and sets the input fields, buttons, and text views to their corresponding UI elements. The 'Sign-Up Text Click Listener' starts the 'Register' screen when the "sign up" text is clicked. The 'Login Button Click Listener' checks if any fields are empty and calls the 'checkUser' method to verify if the username and hashed password exist in the database. The 'hashPassword Method' hashes the password using SHA-256 for security.In summary, the code provides a complete login process, including user inputs, validation, hashing, database checks, messages based on login success or failure, and redirecting to the user dashboard upon successful login.

**Products Page**

****

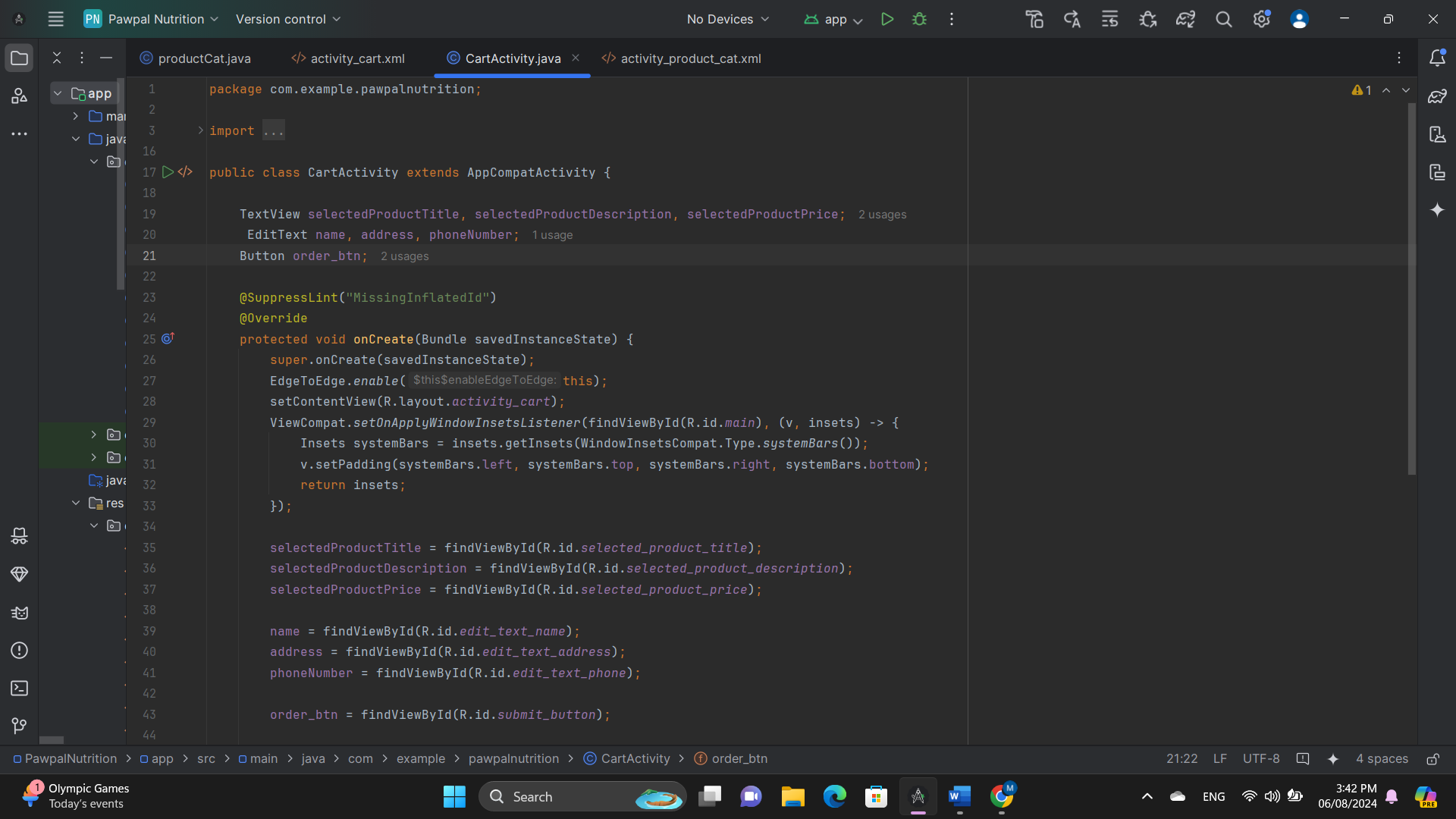


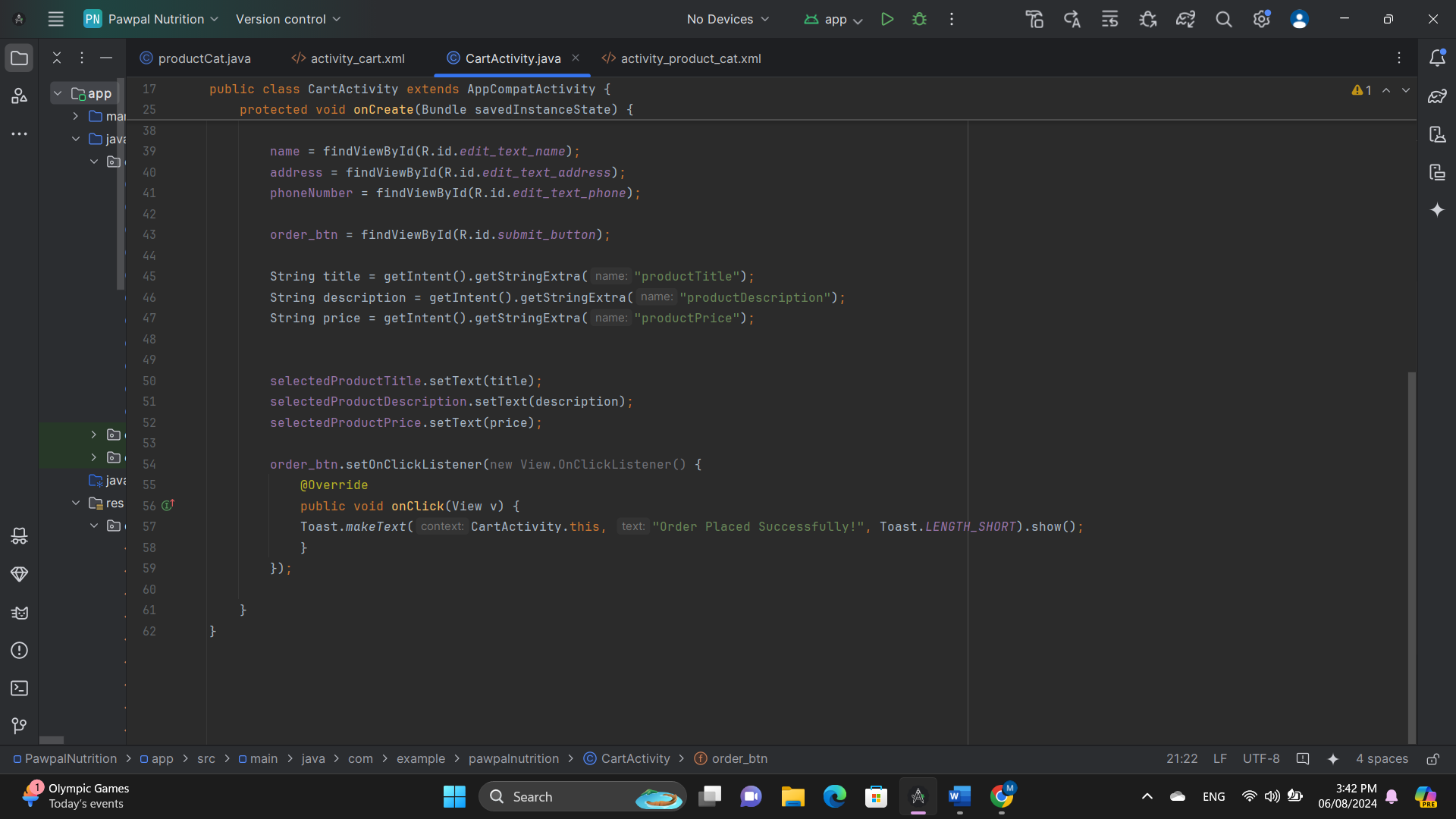


**Explanation**

The `ProductsActivity` class is an Android activity that displays a list of products for a dog food app, allowing users to view product details and add items to their shopping cart. It includes a linear layout, an image view representing the shopping cart icon, a cart manager, and an inflater used to inflate the product card layout from XML. The `onCreate` method initializes the activity, sets up the user interface, and populates it with product data. The activity initializes views by finding their IDs in the layout, defines product data using arrays for `productTitles`, `productDescriptions`, `productImages`, and `productPrices`, creates product cards by iterating through the product data, sets up the "Add to Cart" button, and adds the card view to `productsContainer`. The activity also handles cart icon clicks by setting an `OnClickListener` for `cartIcon` and passing `cartManager` to `CartActivity` via `Intent`. The activity logs messages when an item is added to the cart and an error if card view inflation fails.

**Cart**





**Explanation**

This `CartActivity` class for the PawPalNutrition app manages the display and ordering of a selected product. When the activity is created, it sets up edge-to-edge display, applies system bar insets, and initializes UI components including `TextView` elements for displaying the product title, description, and price, and `EditText` fields for user input (name, address, phone number). The product details are retrieved from the intent that started this activity and set to the corresponding `TextView` fields. A `Button` is set up to handle order submission, displaying a toast message confirming the order placement when clicked.

**Explore PawPal Nutritions and discover top-quality dog food for your furry friend. Here's the link to get started:** [**Dog Food App**](https://drive.google.com/file/d/1cRhNK3ltPrMEB5xH5aR3fwHoTaJZFh9e/view?usp=drivesdk)

**Task E**

**Introduction**

This test plan aims to verify the reliability and effectiveness of a dog food app that helps dog owners find, purchase, and manage high-quality dog food and nutritional products. The app includes user authentication, product catalog browsing, shopping cart management, and educational content about dog nutrition. The test plan will address multiple components of the app, ensuring it meets requirements, functions correctly, and delivers a smooth user experience. Testing will include functional, usability, performance, and security testing to identify and resolve potential issues before the app is launched to the public. The goal is to ensure the app meets all specified requirements and delivers a smooth user experience.

**Scope**

The main goals of this test plan are to make sure that the system works and is dependable in crucial areas such user profiles and authentication, the product catalogue, the shopping cart, and instructional material. Testing user accounts, including those for registration, login, and profile maintenance, is part of it. The product catalogue's sorting and filtering capabilities will accurately show the goods. There will be smooth operations for adding, organizing, and checking out goods in the shopping cart. Users will have access to accurate instructional information. The test strategy is to thoroughly assess these essential components for the best possible functioning and user experience.

**Objectives**

The system should validate user authentication, allow users to create new accounts with valid credentials, and ensure they can log in and manage their profiles smoothly. The product catalogue should display accurate information, including product names, descriptions, prices, images, and availability status. Test filtering options and sorting options to ensure users can easily narrow down their search results. The shopping cart functionality should be tested, ensuring users can add, remove, and adjust products without issues. The total prices should be accurate, taking into account discounts, taxes, or shipping fees. The educational content related to dog nutrition and breeds should be displayed properly, with appropriate formatting and readability. The information should be accurate, up-to-date, and based on credible sources, providing reliable guidance on dog care. The content should be engaging and accessible, with clear explanations and relevant images or illustrations.

**Test Cases**

**User Registration**

**Test Case 1: Verify Registration with Valid Email and Password**

* **Description:** Confirm that a user can register with a valid email and password.
* **Steps:**
  1. Open the registration page.
  2. Enter a valid email address.
  3. Enter a valid password.
  4. Submit the registration form.
* **Expected Result:** Registration is successful, and a confirmation message is displayed.
* **Actual Result:** Same as Expected
* **Screenshots:**

**Test Case 2: Verify Registration with an Already Registered Email**

* **Description:** Ensure the app prevents registration with an email that is already registered.
* **Steps:**
  1. Open the registration page.
  2. Enter an already registered email address.
  3. Enter a password.
  4. Submit the registration form.
* **Expected Result:** An error message is displayed indicating the email is already registered.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Test Case 3: Verify Registration with Invalid Email Format**

* **Description:** Confirm that the app does not allow registration with an invalid email format.
* **Steps:**
  1. Open the registration page.
  2. Enter an invalid email address (e.g., invalid-email).
  3. Enter a valid password.
  4. Submit the registration form.
* **Expected Result:** An error message is displayed indicating the email format is incorrect.
* **Actual Result:** Same as Expected
* **Screenshots:**

**Test Case 4: Verify Registration with Password that Does Not Meet Complexity Requirements**

* **Description:** Ensure that the app enforces password complexity requirements.
* **Steps:**
  1. Open the registration page.
  2. Enter a valid email address.
  3. Enter a password that does not meet complexity requirements (e.g., password).
  4. Submit the registration form.
* **Expected Result:** An error message is displayed indicating the password does not meet complexity requirements.
* **Actual Result:** Same as Expected
* **Screenshots:**

**Test Case 6: Verify Registration with Unmatching Password and Confirm Password**

**Description**: Confirm that users receive an error message when the password and confirm password fields do not match during registration.

**Steps**:

1. Open the registration page.
2. Enter a valid email address.
3. Enter a password in the password field.
4. Enter a different password in the confirm password field.
5. Submit the registration form.

**Expected Result**: An error message is displayed indicating that the passwords do not match.

**Actual Result**: Same as Expected.

**Screenshots**:

**Test Case 7: Verify Login with Incorrect Password**

* **Description:** Ensure that login fails with an incorrect password.
* **Steps:**
  1. Open the login page.
  2. Enter a registered email address.
  3. Enter an incorrect password.
  4. Submit the login form.
* **Expected Result:** An error message is displayed indicating the password is incorrect.
* **Actual Result:** Same as Expected
* **Screenshots:**

**Test Case 8: Verify Login with Unregistered Username**

* **Description:** Confirm that login fails with an unregistered username.
* **Steps:**
  1. Open the login page.
  2. Enter an unregistered username.
  3. Enter any password.
  4. Submit the login form.
* **Expected Result:** An error message is displayed indicating the username is not registered.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Test Case 9: Verify Login with Empty Fields**

* **Description:** Ensure that login fails when fields are left empty.
* **Steps:**
  1. Open the login page.
  2. Leave both email and password fields empty.
  3. Submit the login form.
* **Expected Result:** An error message is displayed indicating that fields cannot be empty.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Test Case 10: Verify Register with Empty Fields**

* **Description:** Ensure that register fails when fields are left empty.
* **Steps:**
  1. Open the login page.
  2. Leave both email and password fields empty.
  3. Submit the login form.
* **Expected Result:** An error message is displayed indicating that fields cannot be empty.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Test Case 11: Verify that Product Ratings are Displayed and Accurate**

* **Description:** Ensure that product ratings are displayed.
* **Steps:**
  1. Navigate to the most viewed items.
  2. Select a product with ratings.
  3. Check that products are displayed with ratings.
* **Expected Result:** Products are displayed correctly with accurate ratings.
* **Actual Result:** Same as Expected
* **Screenshots:**

**Educational Content**

**Test Case 12: Verify that Articles about Dog Nutrition are Accessible and Correctly Formatted**

* **Description:** Confirm that educational articles on dog nutrition are accessible and well-formatted.
* **Steps:**
  1. Navigate to the educational content section.
  2. Access articles about dog nutrition.
  3. Check formatting and readability.
* **Expected Result:** Articles are accessible, well-formatted, and easy to read.
* **Actual Result:** Same as expected.
* **Screenshots:**

**Educational Content about Dog Breed**

**Test Case 13: Verify that Articles about Dog Breeds are Accessible and Correctly Formatted**

* **Description:** Confirm that educational articles on dog nutrition are accessible and well-formatted.
* **Steps:**
  1. Navigate to the educational content section.
  2. Access articles about dog breeds.
  3. Check formatting and readability.
* **Expected Result:** Articles are accessible, well-formatted, and easy to read.
* **Actual Result:** Same as expected.
* **Screenshots:**

**Product Catalogue**

**Test Case 14: Verify that Products are Displayed with Correct Details (Name, Description, Price)**

* **Description:** Confirm that products are displayed with accurate details.
* **Steps:**
  1. Navigate to the product catalogue.
  2. Check that each product has a name, description, and price.
* **Expected Result:** Product details are displayed correctly.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Reviews**

**Test Case 15: Verify that Reviews are Displaying.**

* **Description:** Confirm that reviews are displayed with accurate details.
* **Steps:**
  1. Navigate to the reviews page.
  2. Check that each review has a name, description, and ratings.
* **Expected Result:** Reviews are displayed correctly.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Add to Cart**

**Test Case 16: Verify that Products are viewing in the cart.**

* **Description:** Confirm that products are displayed with accurate details.
* **Steps:**
  1. Navigate to the Products page.
  2. Click add to cart button.
  3. After redirecting to cart, ensure that product name , description and title are displayed
* **Expected Result:** Details are displayed correctly.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Placing an Order**

**Test Case 16: Verify that Products are viewing in the cart.**

* **Description:** Ensuring that the order is placed.
* **Steps:**
  1. Navigate to the Products page.
  2. Click add to cart button.
  3. After redirecting to cart, ensure that product name , description and title are displayed.
  4. After filling the empty fields, click submit button.
* **Expected Result:** A success message should be displayed.
* **Actual Result:** Same as Expected.
* **Screenshots:**

**Task F**

**User Documentation for the Dog Food App**

**Introduction**

Welcome to the Dog Food App! This user-friendly application is specifically tailored for dog owners seeking high-quality dog food and nutrition-related products. This guide aims to familiarize you with the app’s features and functionalities so you can make the most of your experience.

**Key Features**

**Here are the main features you can explore within the app:**

* **User Authentication and Profiles:** Create and manage your personal account.
* **Product Catalogue:** Access a wide range of dog food and nutrition products.
* **Shopping Cart:** Conveniently manage your selected products before purchase.

**Getting Started**

1. **User Registration and Login**

**To access the app, you will need to create an account or log in to an existing one. Here are the steps:**

**Registration:**

1. Tap the "Sign Up" button on the Login Page.

2. Fill in the required fields, including your username, email, desired password, and password confirmation.

3. After filling in all information, tap the "Sign Up" button to create your new account.

**Login:**

1. Following registration, the user may either click the Login button to access the login page, or it will automatically send them there in a few seconds.

2. Enter your registered username and password.

3. Tap "Login" to access your personalized account.

1. **User Profile Management**

**Once logged in, you can manage your profile as follows:**

**Access Profile:**

Tap on the profile icon located at the top right corner of the app.

**Update Personal Information:**

1. Here, you can edit details such as your name, shipping address, and preferred payment methods.

2. After making edits, ensure you tap the "Save" button to update your profile information.

1. **Browsing Products**

**Product Catalogue**

**Explore the extensive selection of dog food and nutrition-related products:**

**Browse Products:** Scroll through the catalogue to view available items.

**Product Details**

**To learn more about a specific product:**

1. You can learn about product just by viewing there will be a detailed description about the product
2. **Managing the Shopping Cart**

**Add to Cart**

**To add products to your cart:**

1. Once on the product detail page, tap the "Add to Cart" button.

**View Cart**

**To review your selected items:**

1. Tap the cart icon located in the top right corner of the screen. This will show the contents of your cart, including product details, quantities, and the total price.

**Adjust Quantities**

**If you need to change the number of items in your cart:**

1. Use the “+” and “-” buttons next to each item to adjust the quantity.

**Remove Items**

**To delete items from your cart:**

1. Tap the "Remove" button next to the item you want to eliminate.

**Checkout**

**When you are ready to make a purchase:**

1. Tap the "Checkout" button.

2. Enter the necessary payment details and review your order.

3. Confirm your order to complete the purchase.

Thank you for using the Dog Food App! We hope this documentation helps you navigate through the app and enjoy a seamless shopping experience for you and your furry friend.

**Technical Documentation for PawPal Nutrition App**

**Overview**

The PawPal Nutrition App aims to help dog owners find and purchase high-quality dog food and nutrition-related products. Users can browse products, read detailed descriptions, make purchases, and manage their orders. The app also provides educational content about dog nutrition.

**Key Features**

1. **User Authentication and Profiles**
   * User registration and login with an email account.
   * User profile management with the ability to update personal information, address, and payment methods.
2. **Product Catalogue**
   * Display a list of dog food and nutrition-related products.
   * Filter and sort products based on criteria such as brand, type, age, etc.
   * Each product has a detailed description, price, and customer reviews.
3. **Shopping Cart**
   * Enable users to add products to their cart.
   * Show the cart's contents, including product details, quantities, and total price.
   * Allow users to adjust quantities, remove items, and view the subtotal.
4. **Educational Content**
   * Include a section with articles, videos, and guides about dog nutrition.
   * Provide information on different dog breeds, life stages, dietary requirements, and health tips.

**Database Schema**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| **ID** | **Integer** | **Primary Key, Auto Increment** |
| **Username** | **Text** | **Unique username** |
| **Password** | **Text** | **User Password** |

**User Interface Design**

**Layout and Components**

1. **Login Screen**
   * **Layout**: Simple and clean design with a focus on user input fields.
   * **Components**:
     + **Username Field**: Text input for the username.
     + **Password Field**: Text input for the password, with masking.
     + **Login Button**: Button to submit login credentials.
     + **Sign Up Link**: Link to navigate to the registration screen.
2. **Product Catalog Screen**
   * **Layout**: List view of products with filtering and sorting options at the top.
   * **Components**:
     + **Search Bar**: Text input to search for products.
     + **Filters**: Dropdowns or checkboxes to filter products by brand, type, age, etc.
     + **Sort Options**: Dropdown to sort products by price, popularity, etc.
     + **Product List**: Grid or list of product items, each showing an image, name, price, and a brief description.
3. **Shopping Cart Screen**
   * **Layout**: List view of items in the cart with a summary section at the bottom.
   * **Components**:
     + **Cart Items**: List of products in the cart, each showing an image, name, quantity, price, and remove button.
     + **Subtotal**: Display of the total price of the items in the cart.
     + **Checkout Button**: Button to proceed to checkout.
     + **Continue Shopping Link**: Link to return to the product catalog.
4. **User Profile Screen**
   * **Layout**: Sections for personal information, address, and payment methods.
   * **Components**:
     + **Personal Information**: Text fields for name, email, and phone number.
     + **Address Information**: Text fields for address details.
     + **Payment Methods**: List of saved payment methods with options to add or remove.
     + **Save Button**: Button to save updates to the profile.

**Workflow and Use Cases**

**User Registration Workflow**

1. **Access Registration Page**:
   * User navigates to the registration page from the login screen.
2. **Fill Registration Form**:
   * User enters required information such as username, password, email, and optional details like name.
3. **Submit Form**:
   * User submits the registration form.
4. **Validate Input**:
   * The app validates the input data.
5. **Create User**:
   * If validation passes, a new user record is created in the database.
6. **Confirmation**:
   * User receives a confirmation message or email.

**Product Browsing Workflow**

1. **Login**:
   * User logs in to access the product catalogue.
2. **Access Product Catalogue**:
   * User navigates to the product catalogue screen.
3. **Apply Filters and Sort**:
   * User applies filters and sorts products based on preferences.
4. **View Product Details**:
   * User selects a product to view detailed information.
5. **Add to Cart**:
   * User adds the product to the shopping cart.
6. **Continue Browsing or Checkout**:
   * User can continue browsing or proceed to checkout.

**Adding Items to Cart Workflow**

1. **Select Product**:
   * User selects a product from the catalog.
2. **View Product Details**:
   * User views the product details page.
3. **Add to Cart**:
   * User clicks the "Add to Cart" button.
4. **Confirmation**:
   * User receives a confirmation message that the item has been added to the cart.

These diagrams and descriptions provide a clear and detailed understanding of the user interface and workflows for the PawPal Nutrition App.

**Testing and Quality Assurance**

**Testing Strategies**

1. **Unit Tests**
   * **Scope**: Focus on individual functions and methods to ensure they work as expected.
   * **Tools**: Jest (JavaScript), JUnit (Java), or equivalent testing frameworks.
   * **Example**: Testing the function that calculates the total price in the shopping cart.
2. **Integration Tests**
   * **Scope**: Test the interaction between different modules or services.
   * **Tools**: Mocha with Chai (JavaScript), PyTest (Python), or equivalent frameworks.
   * **Example**: Testing the integration between the user registration endpoint and the database.
3. **End-to-End Tests**
   * **Scope**: Simulate user interactions to ensure the entire application workflow functions correctly.
   * **Tools**: Selenium, Cypress, or Playwright.
   * **Example**: Testing the complete user journey from logging in to adding items to the cart and checking out.

**Automated Testing Frameworks**

1. **Jest**
   * **Usage**: For writing unit and integration tests in JavaScript.
   * **Integration**: Integrated into the CI/CD pipeline to run tests on every commit.
2. **Cypress**
   * **Usage**: For end-to-end testing with a focus on frontend workflows.
   * **Integration**: Tests are run automatically during the CI/CD pipeline to ensure new changes do not break existing functionality.

**Integration into Development Pipeline**

1. **Continuous Integration (CI)**
   * **Setup**: Use CI tools like Jenkins, GitHub Actions, or Travis CI.
   * **Process**: On each commit, the CI pipeline runs all tests (unit, integration, and end-to-end) to validate the changes.
2. **Continuous Deployment (CD)**
   * **Setup**: Use CD tools integrated with CI tools to automate deployments.
   * **Process**: After successful tests, the application is automatically deployed to staging or production environments.
3. **Test Coverage Reports**
   * **Tools**: Istanbul (JavaScript), Coverage.py (Python).
   * **Process**: Generate test coverage reports to ensure that a significant portion of the codebase is covered by tests.
4. **Code Reviews**
   * **Process**: Peer reviews of code changes are conducted to ensure code quality and catch potential issues early.

(*Informit*)

(Watkins & Mills, *References - testing it*)

**References**

* (No date a) Insights that connect you to Success | opensignal. Available at: https://www.opensignal.com/ (Accessed: 16 July 2024).
* (No date a) Ionic: An OutSystems company reviews & product details - G2. Available at: https://www.g2.com/products/ionic-an-outsystems-company/reviews (Accessed: 16 July 2024).
* (No date) Android Statistics (2024) - business of apps. Available at: https://www.businessofapps.com/data/android-statistics/ (Accessed: 16 July 2024).
* (No date) Android Studio Reviews 2022 - Capterra. Available at: https://www.capterra.com/p/246781/Android-Studio/reviews/ (Accessed: 16 July 2024).
* 17, K.R.A. et al. (no date) Android vs IOS Development: A comprehensive guide for developers, Digital Acceleration Company. Available at: https://www.netguru.com/blog/android-vs-ios-development (Accessed: 15 July 2024).
* Apple Pay Security and Privacy Overview (no date) Apple Support. Available at: https://support.apple.com/en-us/101554 (Accessed: 15 July 2024).
* Benefits of maintaining a single code base for Project Development (no date) Benefits of Maintaining a Single Code Base for Project Development. Available at: https://braininventory.in/posts/one-code-base-project-development/1762 (Accessed: 17 July 2024).
* Duck, DuckDuck                      35.7k4848 gold badges259259 silver badges481481 bronze badges and Rob NapierRob Napier                      296k3535 gold badges476476 silver badges637637 bronze badges (1969) How to prevent xcode compiling a package for a target which is not designated?, Stack Overflow. Available at: https://stackoverflow.com/questions/78287509/how-to-prevent-xcode-compiling-a-package-for-a-target-which-is-not-designated (Accessed: 16 July 2024).
* Flutter architectural overview (no date) Flutter. Available at: https://docs.flutter.dev/resources/architectural-overview (Accessed: 17 July 2024).
* Flutter vs. react native: Choosing the Right Cross Platform Mobile App Development Framework (no date) Sendbird. Available at: https://sendbird.com/developer/tutorials/flutter-vs-react-native-choosing-the-right-cross-platform-mobile-app-development-framework (Accessed: 17 July 2024).
* Fowler, C. (2024) 19 must-have apps for dog owners in 2024, Hepper. Available at: https://www.hepper.com/must-have-apps-for-dog-owners/ (Accessed: 15 July 2024).
* GeeksforGeeks (2022) Flutter - flexible widget, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/flutter-flexible-widget/ (Accessed: 19 July 2024).
* Ghanchi, J. (2022) The major advantages of Android Studio App Development, IndianAppDevelopers. Available at: https://www.indianappdevelopers.com/blog/advantages-of-android-studio-app-development/ (Accessed: 16 July 2024).
* *Informit* (no date) *InformIT*. Available at: https://www.informit.com/articles/article.aspx?p=355875&seqNum=5 (Accessed: 07 August 2024).
* Lawson, L. (2024) What’s next for flutter after layoffs hit Google Team, The New Stack. Available at: https://thenewstack.io/whats-next-for-flutter-after-layoffs-hit-google-team/ (Accessed: 19 July 2024).
* Matellio (2024) How to develop a dog food calculator app?, Matellio Inc. Available at: https://www.matellio.com/blog/dog-food-calculator-app-development/ (Accessed: 15 July 2024).
* Moscaritolo, A. (2014) Android fragmentation up with 18,796 different devices, PCMAG. Available at: https://www.pcmag.com/news/android-fragmentation-up-with-18796-different-devices (Accessed: 16 July 2024).
* Pros and cons of IOS app development (no date) DDI Development. Available at: https://ddi-dev.com/blog/programming/pros-and-cons-of-ios-app-development/ (Accessed: 15 July 2024).
* The One Technologies (2024) Understanding the react native ecosystem: Your complete guide, The One Technologies - Software Company. Available at: https://theonetechnologies.com/blog/post/understanding-the-react-native-ecosystem-your-complete-guide (Accessed: 19 July 2024).
* Watkins, J. and Mills, S. (no date) *References - testing it*, *Cambridge Core*. Available at: https://www.cambridge.org/core/books/abs/testing-it/references/C5B53D82BC8BA44B95660D40BD1B863E (Accessed: 07 August 2024).