|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **National Institute of Business Management**  **School of Computing and Engineering**  **Course work | Assesement Announcement Sheet** | | | |  | |
| **Course Name** | BSc Hons Computing | | | | |
| **Module Name** | iOS Application Development | | | | |
| **Batch** | 22.2P | | | | |
| **Learning Outcomes Covered (Mention according to the Module Descriptor)** | Analyze the platform components and their uses across multiple devices and formats | | | | |
| Design software suitable for mobile architectures | | | | |
| Develop apps that can interact with external APIs and devices. | | | | |
| Develop apps that can communicate with sensors built into the phone hardware. | | | | |
| **Assesement | CW No** | CW 1 | | | | |
| **Assesement Mode** | **Individual | ~~Group~~** | | **Group (if it is group mode only)** | | |
| **Group Size** | | **Grouping Criteria** |
| **-** | | - |
| **Assesement Type** | **~~Practical Test~~ | Report | Software | ~~Presentation | VIVA | MCQ~~** | | | | |
| **If other specify** |  | | | |
| **Hand in Date | Time** | 6/3/2024 | | | | |
| **Hand out Date | Time** | 3/31/2024 | | | | |
| **Submission Details (Format and Location)** | LMS | Git Hub Link | | | | |
| **Plagiarism Criteria** | Similarities will be set zero. | | | | |
| **Assesement | CW Description** | | | | | |
| In this coursework, you will be tasked with developing an iOS application for an online clothing brand. The application should provide users with a seamless shopping experience, allowing them to browse, search, and purchase clothing items directly from their iOS devices. You will need to implement various features such as product listings, user authentication, shopping cart management, and checkout functionality. Additionally, the app should adhere to design guidelines to ensure a visually appealing and user-friendly interface.  **Objectives :**   * Develop a fully functional iOS application for an online clothing brand. * Implement features for browsing, searching, and purchasing clothing items. * Utilize user authentication to provide personalized experiences. * Design and implement a responsive and intuitive user interface. * Integrate backend services for data and content management. * Test and debug the application to ensure functionality and reliability.   **Tasks:**  User Interface Design:  Design the user interface for the app, including screens for product listings, product details, shopping cart, and checkout.  Implement responsive layout designs suitable for various iOS devices.  Product Listings:  Fetch and display clothing products from a backend API.  Implement features for sorting and filtering products based on categories, price, etc.    Product Detail View:  Create a detailed view for each clothing product, displaying images, descriptions, sizes, prices and color variations.  Allow users to add items to their shopping cart from the product detail view.    User Authentication:  Implement user authentication from Backend API.  Enable users to sign up, log in, and manage their accounts within the app.    Shopping Cart Management:  Develop functionality for adding, removing, and updating items in the shopping cart.  Display the current contents of the shopping cart and calculate the total price.    Backend Integration:  Integrate backend services for managing product data, user accounts, and orders.  Ensure secure communication between the app and the backend server.    Testing and Debugging:  Test the application on various iOS devices and simulators.  Debug any issues or errors encountered during testing.  **Submission Guidelines:**  Submit the completed iOS project files along with any additional resources or documentation.  Include screenshots or a video demonstration of the app in action.  Provide a brief overview of the development process and any challenges faced.  Project Report and GitHub Commit history.  **Assessment Criteria:**  Functionality: Does the app fulfill the requirements outlined in the assignment description?  User Experience: Is the user interface intuitive and visually appealing?  Code Quality: Is the code well-structured, readable, and maintainable?  Backend Integration: Are backend services effectively utilized for data management and processing?  Presentation: How do you present your application to the audience, and what capabilities do you bring in? | | | | | |
| Marking Scheme | | | | | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Marking Criteria** | **Analyze the platform components and their uses across multiple devices and formats** | **Design software suitable for mobile architectures.** | **Develop apps that can interact with external APIs and devices** | **Develop apps that can communicate with sensors built into the phone hardware** | | **Marks** | **(30/100)** | **(20/100)** | **(20/100)** | **(30/100)** | | **Description** | The gathered requirements are irrelevant or mainly incomplete.  Wrong analyze work with interface builder and UI Component. Poor usage of the platform components. Poor understanding of the Scenario, completely wrong and does not reflect in the documentation. | Incomplete flow of the application. Did not use proper design patterns. Lack of importance for creating flow for the application. Not creating correct Storyboard flow of the application design. Not follow the any of the mobile architectures | Incomplete api client service. Unable to complete external api using given platform (firebase). | Unable to use the given sensor. Not following the guidelines. Application not interacting with hardware features in the application | | Scenario is understood to an extent which is acceptable.  Relevant platform component usage.  Analyze work with interface builder and UI component. Reflecting of no crash in multiple devices and formats. | App Flow are created but not in details. The visibility of the process is unclear in the flow, but overall flow of the app design is acceptable. Not follow the best practices | Cover all the api.  Correct and relevant api creation in meaningful way, non-clear Api response and data binding with UI component. Unnecessary api calls. | Work with at least one sensor in the device. Sensors that used is suitable for the requirement. | | Correct requirements analyze, usage of platform component, interface builder and UI component.  Nothing impressive | App flow are up to the standard and designed. Follow best practices and Any of the design Patten given bellow   * MVC * MVP * MVVM * VIPER | Cover all the api and clear idea about Https methods. Correct error and success states handling. Model creating for each data object form Api response. | At least one of sensor usage. Follow correct guidelines when using the hardware features of the application. smooth interaction with sensors. | | Proper understanding of the UI component and interface builder, Consider platform components and its uses across devices and formats. Follow IOS design guidelines for interfaces. Clear scaffolding for UI component. Common usage for reuse component. Represent the user interaction with data field and user interaction validations | Clear meaning of the methods. Usage of design Patten, and clear idea about the design Patten. | Represent selected design Patten When handling the api. Common client for connect with api. Correct manipulation for data and data model. Reduce unnecessary api calls. | Usage of both sensors align with the application requirement. Follow correct integration with hardware sensors. | | Component reuse, create common methods for render UI component that used in multiple areas. Follow IOS design guidelines for interfaces when working with multiple device and formats. Manageable code structure when design Interfaces. Contain App theme in separate file. Organize scaffolding for UI component. Clear usage of interface builder (easy to understand and maintain) meaning full validation for failed user actions | Used design Patten should reflect of Used architecture should balance distribution of responsibilities among entities with strict roles.  Code should represent of Ease of use and a low maintenance cost.  Each method should represent the Testability (ex: unit) | The api client and connecting with api should be clean, effective and impressive at all prototyping levels with clearly scoped design problems and prototypes that make sense to users.  Clear error and success response handling. Right test case for all the api, and represent created mock object in previous github code, Before the api integration. Unit test for check api Response and Payloads. Clear lading indicator handling while api request in in-progress. | Clear and non-stuck flow with sensors. Should represent the clear idea about application performance when dealing with the hardware sensors. Sensors implementation and deferent type of device should not block with the current flow of the application. | | | | | | |