IOS APPLICATION DEVELOPMENT

FINAL PROJECT

Module Title	Final Project	**Module Code**
Title of Assignment		Final Project (100%)
Weighting within the module		100%
Supporting Materials		Push the final code base to github public repository, documentation of project

Hand-in Date	10-04-2023
Hand Out Date	10-05-2023

Scope of Work

Scenario :
Fitness is essential for maintaining overall health and well-being. Regular physical activity and exercise play a crucial role in keeping our bodies and minds in optimal condition.

Create an iOS application to set up a fitness routine and find some important set of exercises. The app can start by allowing users to input their fitness goals, such as weight loss, muscle building. Based on their goals. Users should be able to track their workout performance, log their sets, reps, and weights, and monitor their progress over time. The app can also send reminders to users to complete their workouts based on their chosen schedules.

1. User Details capture [Height/ Weight/ Fitness Goal/ Age/ etc..]
2. Calculate BMI and suggest fitness plans
3. Home Page (Display Excesses List)
4. Details Page (Display Rep count, time, workout video, and effective body part)
5. Create custom schedule
6. View custom schedule

Learning Outcomes to be assessed

1. Analyse the platform components and their uses across multiple devices and formats.
2. Design software suitable for mobile architectures.
3. Develop apps that can interact with external APIs and devices.

App UIs

- Need to design an **unique UI** for the app screens (All screens and Pages)
- It's **free to use** any color, fonts, and sizes in order to make the app more User Friendly.
- Should be compatible for **iPhone 8 or later** devices.
- Should follow apple Usability Guidelines, Designing Interfaces.

Technical Requirements

- Use XCode 12.0 or later as a development IDE.
- Application must compile using Swift 5.0 or a later version of Swift
- Use custom API or BASS for backend operations. [CloudKit, Firebase]
- Use **UIKit** and programmatic UI using Layout Constraint or SnapKit
- Third-party dependencies can integrate using Swift Package Manager

Deliverables

- Application Source code GitHub repository
- Recorded ScreenCast video
- Documentation with User Interfaces
- Presentation and Demo

Assignment Outcome

Students should fulfil the following criteria on completions of assignment

1. The student should complete full application with all the screen.
2. The student should follow the apple design guidelines [HIG] when the create UI and component.
3. The student should test the application for several scenarios, that app can be crashed.
4. The student should come up with a mobile architecture that suitable for application solution to provide good software solution.
5. The student should follow iOS standers and best practices to improve the code. Code should be maintainable, and properly commented. All the method should be in meaningful name.
6. Students should commit daily in order to prove their work.
7. The student should have an idea about performance testing, Application should suitable for app store.

Submission

Students need to create their own GitHub account and create a public repository. Students should commit daily in order to prove their work. Final project repository URL should be emailed before the deadline date with the Project Documentation

Application Profiling will evaluate from the VIVA, Student should able to explain and simulate the memory profiling

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	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	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	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	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-	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.