Final Project Requirements Document

Introduction

The final project for the iOS Development Introduction course is designed to consolidate and demonstrate your understanding of the key concepts covered throughout the course. You will develop a fully functional iOS application that incorporates essential UI components, adheres to best practices in layout and design, and utilizes networking and data storage capabilities.

Project Overview

Create an iOS application that showcases your ability to build user interfaces, manage navigation, handle data, and integrate external resources. The theme of the app is open-ended—you are encouraged to choose a topic that interests you, whether it's a news reader, a simple game, a productivity tool, or any other concept that allows you to meet the requirements outlined below.

Technical Requirements

Your application must include the following features and technologies:

1. Basic UI Components

- **UILabel**: Display text content to the user.
- **UIImageView**: Show images within your app.
- **UIButton**: Allow user interactions through tappable buttons.
- **UITextField** and **UITextView**: Enable text input from the user.
- **UISwitch**, **UISlider**, **UISegmentedControl**: Incorporate additional interactive elements as needed.
- and any other components from **Components Library**

2. Auto Layout with Storyboards

- Design all screens using **Storyboards**.
- Apply **Auto Layout** constraints to ensure the interface adapts to various screen sizes and orientations.
- All constraints must be valid—no warnings or errors in Interface Builder.
- Utilize **Stack Views** where appropriate to simplify your layout.

3. UITableView/UICollectionView with Custom Cells

- Implement a **UITableView** or **UICollectionView** to display a list or grid of items.
- Design **custom cells** to present content uniquely suited to your app.
- Manage data sources and delegates to handle interactions and data display.
- Support dynamic data loading and cell reuse for performance optimization.

4. Multi-Module Application Structure

- Use a **UITabBarController** to manage multiple modules or sections within your app.
- Implement a **UINavigationController** to handle hierarchical navigation within modules.
- Each tab should represent a distinct feature or section of your app.

5. Networking (URLSession or Alamofire)

- Perform network requests to retrieve data from an API.
- Use **URLSession** for networking tasks, or integrate **Alamofire** as an external library.
- Parse **JSON** responses and map them to your data models.
- Handle network errors gracefully with appropriate user feedback.

6. External Libraries (Optional)

- Integrate external libraries such as **Alamofire** for networking or **Kingfisher** for image downloading and caching.
- Manage dependencies using **CocoaPods**, **Carthage**, or **Swift Package Manager**.
- Ensure third-party code is properly attributed and licensed.

7. Local Data Storage

- Use **UserDefaults** to store user preferences or small amounts of data.
- Optionally, implement **Core Data** or **SQLite** for more complex data persistence needs.
- Ensure data integrity and handle storage errors appropriately.

8. Additional Recommended Features

To enhance your app and demonstrate a deeper understanding, consider implementing the following:

- **Error Handling**: Provide meaningful error messages and recovery options.
- **Loading Indicators**: Show activity indicators during network requests.
- **Pull to Refresh**: Allow users to refresh content in table or collection views.
- **Pagination**: Load data in chunks to improve performance with large datasets.
- **Search Functionality**: Enable users to search through content.

9. User Experience Enhancements

- **Animations**: Use basic animations to improve user interaction and feedback.

10. Code Quality and Testing

- Write clean, maintainable code following Swift best practices.
- Comment your code where necessary to explain complex logic.
- Implement **unit tests** using **XCTest** to verify the functionality of critical components.

Deliverables

- **Xcode Project**: The complete project folder, ready to build and run.
- **README File**: Instructions on how to set up and run your app, including any dependencies.
- **Demo video:** Screen recording of your app usage

Evaluation Criteria

Your project will be assessed based on the following:

- **Functionality**: The app meets all specified requirements and works without crashes.
- **User Interface**: The UI is intuitive, visually appealing, and responsive across devices.
- **Code Quality**: Code is well-organized, follows naming conventions, and is free of unnecessary complexity.
- **Technical Implementation**: Effective use of technologies and frameworks taught in the course.
- **Creativity**: The app demonstrates originality and thoughtful design.
- **Documentation**: Clear and comprehensive documentation and comments.
- **Optional Features**: Implementation of recommended additional features will be viewed favorably.

Submission Guidelines

- Upload your project to Git repo.
- Ensure all necessary files are included and that the project builds successfully.
- Make sure you have commits history of your project

Support and Resources

- Refer to the course materials and recommended readings.
- Utilize Apple's Developer Documentation.
- Seek assistance during office hours or via the course forum for any clarifications.