iOS Competencies

A competency is an array of essential knowledge, skills, and abilities required to successfully perform a specific task. The following is a list of iOS competencies by experience levels that Kodeco's curriculum is designed to teach.

- ## Swift Language

  The Swift programming language is crucial for iOS development because it is the official and highly efficient language that powers applications for Apple's ecosystem.

  - **Beginner**
    - Describe the basic syntax of Swift, including variables, constants, functions, loops, and conditional statements.
    - Explain the fundamental syntax and data types in Swift.
    - Apply the fundamental concepts of Swift to solve problems and build simple iOS apps.
    - Debug and troubleshoot Swift code by identifying common errors.
    - Optimize code for better performance and readability.
  - **Intermediate**
    - Use the Swift standard library and APIs to implement advanced features.
    - Apply Swift's concurrency model for efficient asynchronous programming.
  - **Advanced**
    - Demonstrate expertise in advanced language features.
- ## Xcode

  Xcode is the integrated development environment (IDE) for iOS development, providing essential tools for building, testing, and debugging apps.

  - **Beginner**
    - Navigate and use Xcode's interface for creating and running iOS projects.
    - Debug and troubleshoot basic issues in Xcode.
  - **Intermediate**
    - Utilize advanced Xcode features like Instruments for performance analysis, and debugging tools.
    - Optimize workflow with Xcode, including version control integration and workspace management.
  - **Advanced**
    - Provide mentorship on advanced Xcode usage and best practices to junior developers.
    - Create and share custom Xcode templates and extensions.
- ## Version Control

  Version control is a system for tracking changes to files over time. It is essential for iOS development because it allows developers to collaborate on projects and revert to previous versions of code if necessary.

  - **Beginner**

        - Use a version control system, such as Git, to track changes to Swift code.
        - Create and merge branches in a version control system.
    - **Intermediate**
        - Use a version control system to collaborate with other developers on iOS projects.
        - Resolve merge conflicts in a version control system.
    - **Advanced**
        - Lead version control strategies for large-scale projects, including branching and release management.
        - Use Git to manage complex branching and merging scenarios.

- SwiftUI

  SwiftUI is Apple's modern UI framework, simplifying UI development and providing a consistent interface across Apple platforms.

    - **Beginner**
        - Create basic user interfaces using SwiftUI's declarative syntax.
        - Implement navigation and user interaction with SwiftUI.
    - **Intermediate**
        - Develop complex and custom UI components in SwiftUI.
        - Use SwiftUI animations to create dynamic user interfaces.
        - Integrate SwiftUI with existing UIKit-based projects.
    - **Advanced**
        - Architect and implement highly reusable SwiftUI components and libraries.

- Architecture*

  Understanding iOS app architecture is crucial for building scalable, maintainable, and organized apps.

    - **Beginner**
        - Explain the basic principles of software architecture.
        - Apply simple architectural patterns to iOS apps such as MVC (Model-View-Controller)
    - **Intermediate**
        - Implement and advocate for a specific architecture like MVVM, VIPER, MVI, TCA, or Clean Architecture.
        - Refactor existing projects to improve architecture and maintainability.
    - **Advanced**
        - Design and lead the implementation of scalable and maintainable iOS apps.

- Networking*

  Networking is important for iOS development because it allows apps to communicate with other devices on the network. This includes communicating with servers to download data, upload data, and perform other tasks.

    - **Beginner**
        - Perform basic network requests using URLSession.
        - Handle simple JSON parsing and error handling for network responses.
    - **Intermediate**
        - Build robust networking layers with error handling, authentication, and caching.
        - Work with RESTful APIs, GraphQL, or other advanced network protocols.

        - Use networking libraries to simplify networking tasks.
    - **Advanced**
        - Architect and lead the development of highly scalable network infrastructure.

- Data Persistence*

Data persistence is important because it allows apps to store data on the device so that it can be used later. This includes storing user preferences, game data, and other types of data.

- **Beginner**
    - Save and retrieve data in a basic file or using UserDefaults.
    - Explain the concept of data modeling for local storage.
- **Intermediate**
    - Implement more complex data persistence solutions, such as Core Data, SwiftData, Realm, or using a cloud database.
    - Choose the right data persistence solution for an iOS app.
    - Optimize data persistence for efficiency and synchronization with remote data sources.
- **Advanced**
    - Design and implement distributed data persistence solutions for multi-platform and offline-capable apps.
    - Investigate and resolve complex data synchronization and conflict resolution challenges.

- Concurrency*

Concurrency is the ability of an app to perform multiple tasks simultaneously. This is important because it allows apps to be more responsive and efficient.

- **Beginner**
    - Explain the basics of concurrency, such as threads and processes.
    - Implement basic concurrency tasks, such as downloading data from the network.
    - Use OperationQueue to manage multiple concurrent tasks.
- **Intermediate**
    - Use concurrency to improve the performance and responsiveness of apps.
    - Implement complex concurrent tasks, such as streaming audio or video.
    - Simplify the development of concurrent code.
- **Advanced**
    - Design and implement concurrent algorithms and data structures.
    - Optimize concurrent code for performance and scalability.
    - Troubleshoot and debug concurrent code.

- Memory Management & Performance

Memory management and performance are important for iOS development because iOS devices have limited resources. Both are crucial for delivering smooth and responsive iOS apps to avoid performance bottlenecks.

- **Beginner**
    - Comprehend the basics of memory management in Swift.
    - Identify common performance bottlenecks in iOS apps.
- **Intermediate**
    - Implement advanced memory management techniques, including ARC optimization.

        - Identify and fix memory leaks in iOS apps.
        - Profile and optimize app performance using tools like Instruments.
    - **Advanced**
        - Lead performance optimization efforts for complex, resource-intensive applications.
        - Perform in-depth memory analysis and advanced profiling, including time profiling and energy efficiency.

- Accessibility*

Accessibility is important because it ensures that apps are accessible to all users, including those with disabilities.

    - **Beginner**
        - Explain the basic principles of accessibility in user interface design.
        - Implement simple accessibility features in iOS apps.
    - **Intermediate**
        - Implement advanced accessibility features, including custom accessibility actions and traits.
        - Conduct thorough testing to ensure compliance with accessibility guidelines.
        - Apply accessibility best practices in both design and audio.
    - **Advanced**
        - Advocate for accessibility at an organizational level and ensure all projects meet or exceed accessibility standards.
        - Develop innovative accessibility solutions and tools to benefit a wide range of users.

- Debugging*

Debugging allows developers to find and fix bugs in their code, ensuring stability and reliability in their apps.

    - **Beginner**
        - Use debugging tools in Xcode to locate and fix basic issues.
        - Employ print statements and breakpoints for basic debugging.
    - **Intermediate**
        - Perform advanced debugging techniques, such as stepping through code and inspecting variables.
        - Analyze crash reports and diagnose complex issues in production apps.
    - **Advanced**
        - Act as a troubleshooting expert, providing guidance and mentorship to resolve the most challenging issues.
        - Develop and maintain custom debugging and diagnostics tools for the team.

- Dependency Management

Dependency management allows developers to easily manage the third-party libraries and frameworks that they use in their apps.

    - **Beginner**
        - Use a dependency management tool, such as CocoaPods, Carthage, or the Swift Package Manager.
        - Install and uninstall third-party libraries and frameworks.
        - Describe the concept of dependency resolution and versioning.
    - **Intermediate**

        - Troubleshoot and resolve dependency conflicts.
    - **Advanced**
        - Architect and maintain the organization's dependency management strategy, including internal package development.

- Security

Ensuring the security of users data and app functionality is a fundamental responsibility for app developers.

    - **Beginner**
        - Describe basic security concepts like data encryption and secure coding practices.
        - Implement user authentication and authorization.
    - **Intermediate**
        - Implement more complex security features in iOS apps.
        - Test iOS apps for security vulnerabilities.
    - **Advanced**
        - Develop and maintain a comprehensive security program, including threat modeling, risk assessment, and compliance management.
        - Lead incident response and coordinate security efforts across the organization.

- Testing*

Testing ensures that apps are bug-free and meet the requirements of the users.

    - **Beginner**
        - Write unit tests for Swift code.
        - Run unit tests in Xcode.
    - **Intermediate**
        - Write integration tests and UI tests for iOS apps.
        - Use continuous integration and continuous delivery (CI/CD) to automate testing.
    - **Advanced**
        - Establish and advocate for testing best practices, including test-driven development (TDD).
        - Develop and maintain a robust testing infrastructure, including continuous testing and test data management.

- UIKit

UIKit remains a fundamental framework for building iOS apps, and understanding it is critical for working with existing projects and supporting older iOS versions.

    - **Beginner**
        - Create user interfaces using UIKit components.
        - Explain UIKit's view hierarchy and view controllers.
    - **Intermediate**
        - Customize and extend UIKit components for complex UI requirements.
        - Integrate UIKit with newer SwiftUI-based features.
    - **Advanced**
        - Innovate and develop custom UIKit components and libraries.
        - Provide expert guidance on transitioning legacy UIKit codebases to modern practices and frameworks,

- ## Continuous Integration

  Continuous integration streamlines the development process, automates testing, and ensures code quality and stability.
  - **Beginner**
    - Set up basic CI/CD pipelines using tools like Jenkins, Fastlane, or Xcode Cloud.
    - Automate build and testing processes with version control integration.
  - **Intermediate**
    - Configure advanced workflows, including automated deployment to the app stores.
    - Implement code quality checks and integration tests in CI pipelines.
  - **Advanced**
    - Architect and lead the organization's CI/CD strategy, incorporating advanced automation and deployment practices.
    - Establish performance monitoring and optimization as part of the CI/CD pipeline.
- ## Data Structures & Algorithms*

  A solid understanding of data structures and algorithms is vital for optimizing app performance, solving complex problems and for passing technical interviews.
  - **Beginner**
    - Learn the fundamental data structures like arrays, dictionaries, and linked lists.
    - Implement basic sorting and searching algorithms.
  - **Intermediate**
    - Apply more complex data structures like trees, graphs, and hash tables.
    - Optimize code using efficient algorithms for specific tasks within apps.
  - **Advanced**
    - Lead algorithmic optimizations and data structure design for performance-critical components.
- ## Interviewing

  Interviewing skills help developers secure job opportunities and advance their careers in the iOS development field.
  - **Beginner**
    - Prepare for technical interviews by practicing coding challenges in Swift.
    - Develop effective communication and problem-solving skills for interviews.
  - **Intermediate**
    - Provide detailed, well-structured explanations for technical questions and coding challenges during interviews.
    - Master system design interview and demonstrate expertise in architectural decisions.
- ## App Distribution

  Understanding the app distribution process ensures that developed apps can reach users effectively and securely.
  - **Beginner**
    - Learn the basics of app signing and provisioning profiles.
    - Deploy apps to physical devices for testing.
    - Submit an iOS app to the App Store.
  - **Intermediate**
    - Manage App Store Connect and navigate the App Store submission process.

        - Implement beta testing and gather user feedback for app improvements.
    - **Advanced**
        - Lead app distribution strategies, including release management, A/B testing, and staged rollouts.

*Indicates some overlap with Android in terms of concepts that can be taught universally.