Narendra

Project handover plan

Release history

| **Date** | **Version** | **Project Name** | **Author****(Developer)** | **Approved By****(TL)** |
| --- | --- | --- | --- | --- |
|  |  | Mastek | Narendra | Akhand Pratap Singh |

Index

1. Project background 4

1.1 Client 4

1.2 Purpose 4

1.3 Scope 4

1.4 Work role 4

2. Project description 4

2.1 Core work flow 4

2.2 Architecture 4

2.3 Work item 4

2.4 Project Team 4

3. Handover plan 4

3.1 Handover timeline 4

3.2 Handover method 4

4. Notice 4

4.1 Cooperation model 4

4.2 Frequently Asked Questions 5

5. Account information 6

5.1 Site 6

5.2 Source code 6

5.3 Resources 6

5.4 App iTunes/Play Store URL 6

6. Work Status 6

6.1 Current Status 6

6.2 Status project on last day of contract 6

Mastek

Project handover plan

# Project background

## 1.1 Client

*Mastek*

## 1.2 Purpose

*NA*

## 1.3 Scope

*NA*

## 1.4 Work role

*The role of an iOS developer involves designing, developing, testing, and maintaining applications for Apple's iOS platform, which includes iPhone, iPad, and other Apple devices. Here's a breakdown of the typical responsibilities and tasks of an iOS developer:*

*App Development: The primary responsibility of an iOS developer is to create mobile applications for iOS devices. This involves writing code in languages such as Swift or Objective-C to implement the app's functionality, user interface, and features.*

*UI/UX Design: iOS developers work closely with designers to implement the user interface (UI) and user experience (UX) of the app. They translate design mockups and wireframes into interactive and visually appealing interfaces that adhere to Apple's design guidelines.*

*Programming: iOS developers write clean, efficient, and maintainable code to implement various app features and functionalities. They use programming languages such as Swift or Objective-C, as well as frameworks like UIKit, SwiftUI, and Core Data, to build iOS applications.*

*Testing and Debugging: iOS developers are responsible for testing and debugging their code to ensure the app functions correctly and is free of bugs and errors. They use tools like Xcode's built-in debugger and testing frameworks to identify and fix issues.*

*Version Control: iOS developers use version control systems like Git to manage and track changes to their codebase. They collaborate with other team members, such as designers and backend developers, by branching, merging, and resolving conflicts in the codebase.*

*Performance Optimization: iOS developers optimize app performance by identifying and addressing performance bottlenecks, memory leaks, and other inefficiencies. They use profiling tools and techniques to analyze and optimize the app's performance.*

*API Integration: iOS developers integrate APIs (Application Programming Interfaces) to enable communication between the app and external services or databases. This includes integrating RESTful APIs for fetching data, authentication, and performing other operations.*

*Documentation: iOS developers document their code, including APIs, classes, functions, and libraries, to facilitate collaboration, maintainability, and knowledge sharing within the team.*

*Continuous Learning: iOS developers stay up-to-date with the latest trends, technologies, and best practices in iOS development. They attend conferences, participate in online communities, and engage in continuous learning to improve their skills and stay current with industry advancements.*

*App Store Submission: iOS developers prepare and submit apps to the Apple App Store, ensuring they meet Apple's guidelines and requirements for quality, performance, security, and content.*

*Overall, iOS developers play a crucial role in the development lifecycle of iOS applications, from initial concept and design to implementation, testing, and deployment. They work collaboratively with designers, backend developers, and other team members to deliver high-quality, user-friendly iOS apps that meet business objectives and user needs.*

*.*

# Project description

## 2.1 Core work flow

## 2.2 Architecture

*The application was made in VIPER.*

*Separation of Concerns: VIPER enforces a clear separation of concerns by dividing the application logic into distinct components: View, Interactor, Presenter, Entity, and Router. Each component has a specific responsibility, making it easier to understand, maintain, and modify different parts of the codebase. This separation also facilitates unit testing since each component can be tested independently.*

*Scalability and Maintainability: VIPER promotes modularity and loose coupling between components, which enhances the scalability and maintainability of the codebase. By breaking down the application into smaller, independent modules, developers can easily add new features, modify existing functionality, or replace components without affecting other parts of the system. This makes it ideal for large and complex projects where flexibility and maintainability are essential.*

*Testability: VIPER architecture inherently supports unit testing, which is crucial for ensuring the reliability and quality of the codebase. Since each component in VIPER has a well-defined responsibility and clear boundaries, it's easier to write unit tests for individual components in isolation. This enables developers to identify and fix bugs early in the development process, leading to more robust and stable applications.*

*Overall, the VIPER architecture offers several benefits, including separation of concerns, scalability, maintainability, and testability, making it a popular choice for iOS app development, e*

*The interactor contains the business logic of your application.*

*It communicates with data sources (such as APIs, databases, or repositories) to fetch and manipulate data.*

*In Swift, the interactor is a plain Swift class or a protocol with a corresponding implementation.*

*Presenter:*

*The presenter acts as an intermediary between the view and the interactor.*

*It receives input from the view, processes it using the interactor, and formats the data to be displayed by the view.*

*In Swift, the presenter is typically a class or a protocol with a corresponding implementation.*

*Entity:*

*The entity represents the data model used by your application.*

*It encapsulates the data retrieved from external sources and passed between the interactor, presenter, and view.*

*In Swift, the entity can be a struct, class, or protocol depending on your data modeling needs.*

*Router:*

*The router handles navigation and routing logic within your application.*

*It is responsible for presenting and dismissing view controllers, as well as passing data between them.*

*In Swift, the router is often implemented using a coordinator pattern or as part of the presenter.*

## Work item

*NA.*

## Project Team

*project Manager: Nitin / Anirudh*

# Handover plan

## 3.1 Handover timeline

*NA*

## 3.2 Handover method

*NA*

# Notice

## 4.1 Cooperation model

*[State the cooperation model of this project, and client’s particular requirements or habits.]*

## 4.2 Frequently Asked Questions

*[List the FAQ and the mistakes that new member will possibly make]*

#

#  Account information

## 5.1 Site

*NA*

## 5.2 Source code

*https://github.com/mastek*

## Resources

*Design link : https://www.figma.com/files*

## App iTunes/Play Store URL

*NA*

# Work Status

## 6.1 Current Status

*Completed*

## 6.2 Status project on last day of contract

*Completed*