Oregon State
University

College of Engineering
Electrical Engineering and Computer Science

Software Innovation Lab

Master's Project Report

Kedar Prabhakar Dhere

Flow Magic

Screen Flow Simplified

Defended 1st December, 2023

Commencement December, 2023

Abstract

Flow Magic is a tool, composed of a software development kit (SDK) and a web portal, that allows real-time modifications to an iOS mobile application's screen flow without requiring code redeployment. 'Screen flow' refers to the sequence and navigation between various screens in an application. Traditionally, the screen-flow changes in a mobile application involve a lengthy process where the product owner creates a ticket, developers make code changes, and the rebuilt binary is submitted to the App Store, Apple's application marketplace, for approval. Flow Magic streamlines this process by eliminating developer intervention and App Store verification, thereby allowing product owners to directly implement and push changes to production.

Acknowledgments

I would like to take this opportunity to acknowledge all those who have significantly contributed to my academic journey. First and foremost, I extend my deepest gratitude to my advisor, Dr. Will Braynen. His constant support, encouragement to learn new things, and admirable work ethic have been invaluable. I am particularly thankful to him for teaching me to strive for excellence and give my best in all situations.

I want to thank my committee members, Dr.Mike Bailey and Dr.Arash Termehchy, for making time in their busy schedules. Their support and feedback have been very valuable.

I want to thank Anush Suresh Kumar for his mentorship and consistently guiding me in the right direction throughout every project stage.

I wish to thank all my professors, the EECS staff, parents, sisters, and friends. Their unwavering support has been a cornerstone of my journey. Without them, I would not have been able to complete my academic journey.

Table of Contents

List of Figures

1. Introduction

The introduction of mobile applications to the mobile phone was a big leap in the technology world. Currently, two main operating systems dominate the mobile phone market: Android and iOS. These are managed by two technological giants, namely Google and Apple, for Android and iOS operating systems, respectively [1]. My project focuses on iOS application development.

Apple, in 2008, launched a marketplace for mobile applications named 'App Store' [2]. The App Store allows developers to release their developed applications, and iPhone users can download and use them. The App Store has a feedback system that allows users to rate and review mobile applications. These ratings and reviews play a vital role in the popularity of a mobile application. End users are typically hesitant to download applications that have bad reviews or lower application ratings. To avoid lower ratings or bad reviews developers are keen to make their application user friendly.

One way to address user needs and design user-centric mobile applications is to follow User Interface (UI) / User Experience (UX) principles while designing the application. UI/UX is a field of study used to create the user interface to enrich user experiences. This process involves research, data analysis, and testing to enhance design choices, focussing on aesthetics and user preferences. The outcome of these tests might suggest the product owners to change the application's user interface to make the user experience better.

In addition to applying UI/UX principles, some companies conduct A/B testing. A/B testing is a form of testing in which a few users are presented with option A, and other users are presented with another alternative. A/B testing evaluates an application based on certain parameters such as conversion rate, retention rate, bounce rate, average time on page, and others. Analyzing these parameters assists the product owner in determining which of the presented options is better.

Incorporating feedback from usability testing and A/B testing is essential as users today have numerous alternatives for similar tasks and may easily switch to other available applications

if their needs are not met. Tools that help with speedy adjustments to the application's interface without diving deep into the code are essential to remain competitive and quickly address user issues. Such solutions would expedite time to market, decrease support expenses, and enhance user experience, ensuring that apps remain competitive and user-friendly.

One such example that would be effective to enhance user experience would be to deal with screen flows. Screen flow here refers to a sequence of interconnection between various screens to help the user navigate to select various services offered by a mobile application. All mobile applications typically have multiple screens, and every screen is connected to some screen or other. Various screens and the connection between them can be considered as different steps the user needs to complete to achieve the desired goal. Each user's actions might lead to a different set of screen flows. Figure 1 below shows the typical screen flow pattern to complete the simple sign-in functionality of an imaginary shopping application. Here, Selecting the "SignUp" option will display a different set of screens than selecting the "Login" option. These progressions of screens or the navigations among them are called Screen Flow. If this screen flow is complex and unclear, the user might get confused and unable to perform the desired task.

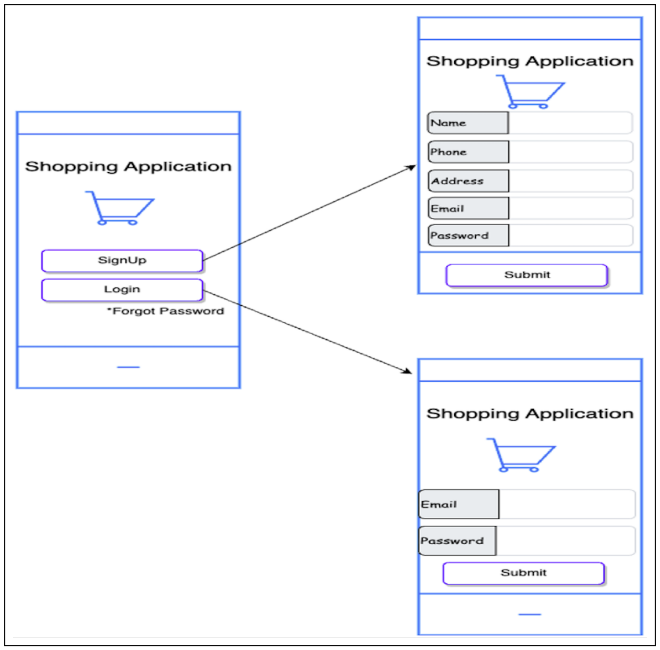


Figure 1.1: Screen Flow

Screen Flow, also referred to as User Flow, plays an important role in the user experience [3]. Screen Flow decides how the user can achieve the desired functionality of the mobile application being used. Hence, it is essential to address any issues related to the screen flow quickly. However, deploying these changes to the production can be time-consuming as changes usually have to go through the typical Software Development Life Cycle (SDLC).

The Software Development Life Cycle (SDLC) typically involves the following steps:

1. Collecting requirements and creating a development ticket.
2. Development phase
3. Testing phase
4. Deployment and App Store verification.

This process often takes considerable time; 'App Store' verification alone might take 24 hours [4].

To summarize, screen flow is crucial in mobile applications and user experience. Currently, fixing the screen flow-related changes might take some time, which might result in a bad user experience, leading to user drop-offs. Flow Magic addresses these issues and provides an easy way to change the screen flow of the iOS application in real-time.

This report offers a detailed overview of Flow Magic. In section 2, we examine existing solutions and discuss their shortcomings. Section 3 highlights our proposed idea. We then break down the chosen technological tools, the user-friendly design, and the overall setup. To better understand the context, images of the application in action are included. We concluded with thoughts on potential improvements for the future.

2. Existing Solutions

The user interface of the mobile applications should be simple and user-centric. Screen Flow is critical in making these UI simple and user-centric. Sometimes, deciding the most appropriate screen flow to satisfy different types of user facets can be challenging. Therefore, tools should be available to help quickly fix user interface-related issues without requiring any code changes. This chapter details some existing methods that deal with UI-related issues without needing much or no code change.

2.1 Server Driven UI based solutions

Server-driven UI, as the name suggests, uses the server to update the application's user interface. Generally, this technique uses the JavaScript Object Notation(JSON) file to represent the user interface. A JSON file is a set of key-value pairs, and human-readable text is used to store and transmit the data [5]. Server-driven UI updates the UI dynamically just by changing the key-value pair. This eliminates the need for code changes, and the changes are available to the user in real-time. The following section will explore a few similar methods in detail to effectively update the application's user interface in a dynamic and efficient manner.

2.1.1 Beagle

Beagle is an open-source framework driven by Server-driven UI principles and offers compatibility across multiple technologies, including Android, iOS, Angular, and React [6]. In line with Server-driven UI, implementing Beagle involves specifying the server URL where the JSON file is hosted. Modifications to the UI are then made by altering the JSON file rather than the code itself. Figure 2.1 shows the required JSON format and the UI rendered with the corresponding JSON [6]. This approach with Beagle eliminates the need for code changes and reflects the changes to UI in real-time. Also, Beagle provides the regular updates and

has a very detailed documentation.

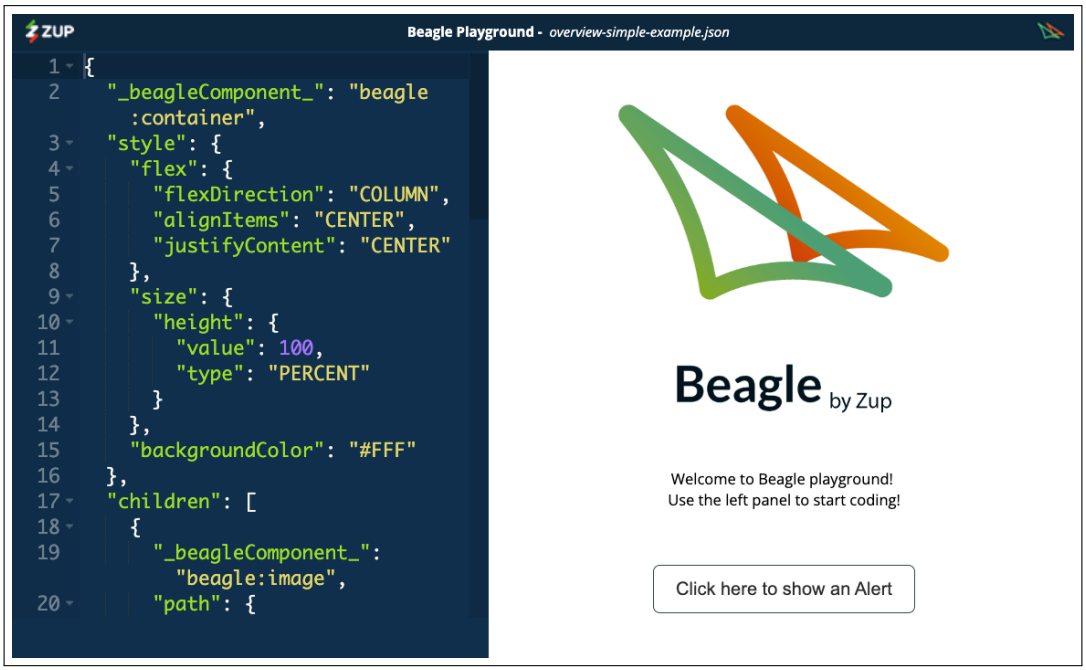


Figure 2.1: Beagle UI: The left displays the JSON content, while the right part illustrates the user interface shaped by that JSON. https://docs.usebeagle.io/v2.1/overview/

Beagle, while innovative, has certain limitations to consider:

1. Individuals without technical expertise, like product owners, might struggle to comprehend and modify the JSON, maintaining some dependency on developers.
2. Beagle supports changing the navigation flow but requires the use of its custom class instead of the standard "UIViewController" class provided by Apple [7]
3. Beagle is compatible with UIKit but does not yet support SwiftUI [6].

2.2 Reshape

Reshape [8] is a tool developed in-house at the Software Innovation Lab to dynamically change the screen flow for Android applications without requiring any code changes. Reshape

provides a web portal, as shown in Fig. 2.2 [8], that allows users to change the screen flow of their mobile application's UI. This portal displays the images of the Android applications screens with the screen number. To change the screen flow, two dropdowns are provided labeled "Screen Number 1" and "Screen Number 2". The user has to select the screen numbers that need to be swapped from the dropdown and click the "Replace" button.

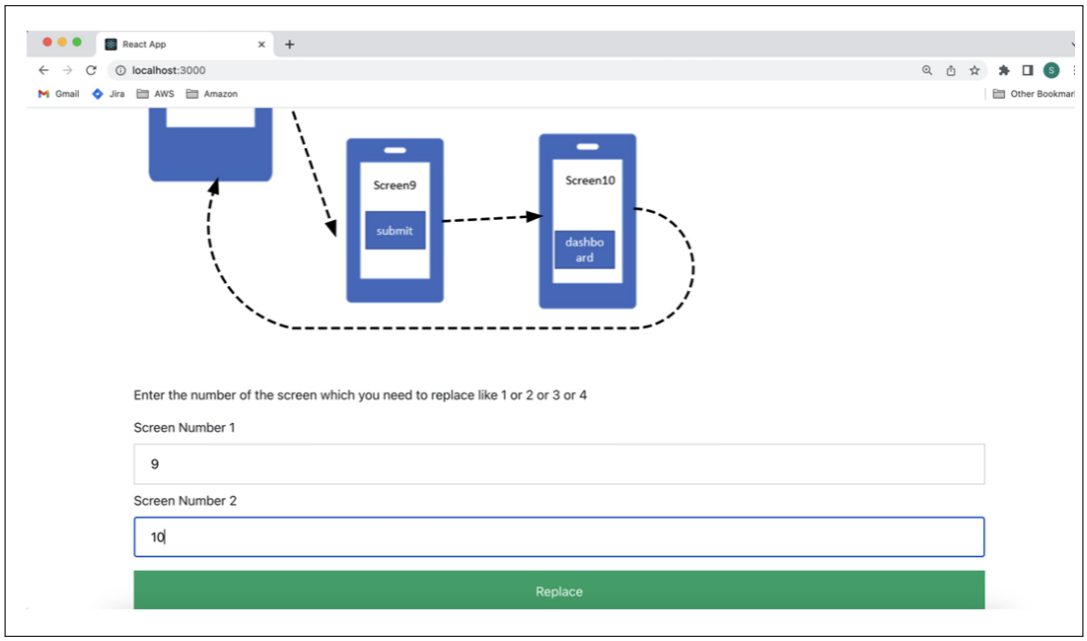


Figure 2.2: Reshape UI

Reshape has the following limitations:

1. Screens should be independent and not depend on each other. It means, that if one screen's info comes from another screen's action (like an API call), Reshape can't modify it.
2. Reshape allows changes to the screen flow only for screens with up to two buttons.
3. The UI uses a dropdown to update screen flow, but there is room for improvement to enhance user-friendliness.
4. As the number of screens increases, using screen numbers to change the screen flow can become challenging.

2.3 A/B Testing Tools

A/B testing platforms, such as Optimizely and Leanplum, enable the modification of screen navigation through feature flags, as explored in the Reshape project report [8]. These feature flags act as a switch; whichever flag is set on the A/B testing portal, the production will display that flow to the end user. For example, if the product owner has two navigation options, A and B, each option will be associated with its own feature flag. The screen flow linked to the active feature flag will be displayed to the end user. However, introducing more options in the future necessitates adding new switches and their corresponding flags and screen flows. Any changes to the existing flow may require code modifications.

In this chapter, various solutions were discussed that facilitate real-time UI modifications without needing code changes. Tools like Beagle and Reshape allow changing the screen flow in real-time. Reshape provides a web portal to change the screen flow but has some limitations. Beagle, alternatively, requires making changes to JSON. JSON requires following rules for proper functioning and might be difficult to maintain if the JSON is longer. Individuals with non-technical backgrounds might not be aware of those rules and might find it difficult to change the UI using JSON. A/B testing tools allow real-time modification to the screen flow. However, they do not allow changes in the screen flow itself. This creates some dependency on the developers and might cause a delay in the implementation. The next chapter will introduce a proposed solution to overcome the limitations highlighted in this chapter.

3. Proposed Solution

The proposed solution, "Flow Magic," is a tool that will enable product owners to modify and deploy screen flow in real-time without needing any code changes. This chapter discusses "Flow Magic," exploring its advantages, detailing how it can be implemented, and the fundamental principles underlying its design with a focus on action ports in the Flow Magic design.

Flow Magic will help the product owners, who are typically not aware of the codebase, independently implement their ideas about the mobile application's screen flow and incorporate the results from user surveys without the help of developers. Also, this tool will help to reduce the time to market, enhance customer satisfaction, and decrease customer drop-offs by addressing customer queries quickly. It will also allow for experimentation.

This tool provides a web portal with a user-friendly, drag-and-drop interface, enabling product owners to modify the flow of existing screens independently. Only authorized users can access the web portal, making sure only authorized users can change the screen flow. Moreover, the portal is designed to be singular for an organization, meaning that if a company manages multiple applications, it can utilize one portal for all its applications.

3.1 Phases of Flow Magic implementation

Flow Magic consists of two phases of implementation:

I) Development Phase (Developer as a user)

In this phase, the developer first imports the Flow Magic SDK, a Swift Package, into their application using Swift Package Manager (SPM). SPM requires a GitHub URL to import the package and its related dependencies. Once the package is imported, the developer must register the screens with unique names and define their navigation paths using Flow

Magic SDK's methods. These methods auto-generate the necessary information and provide dynamic navigation.

II) Post-production Phase (Product Owner as a user)

Once the application is deployed to production, the product owner can use the web portal to change the screen flow. Access to this portal is secured through Google authentication using Gmail. Once authenticated, all registered applications become accessible. Product owners can then select the specific application they wish to modify. Any changes to the portal's screen flow are immediately reflected in the server's data file. Subsequently, the SDK within the iOS application retrieves this updated data, ensuring the app's screen flow is current with the latest modifications.

3.2 Action Ports

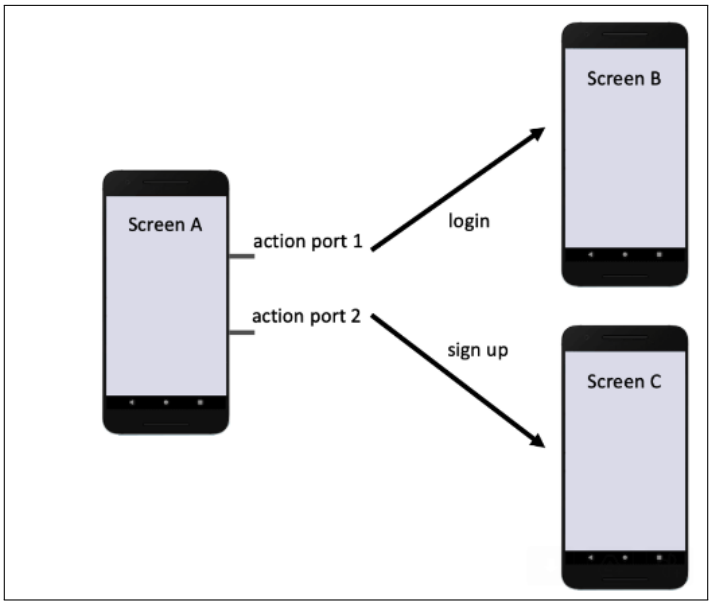


Figure 3.1: Action Ports

In standard application development, screen-to-screen navigations are pre-set by the developer, thus establishing a fixed screen flow. Flow Magic, however, adopts a more dynamic

approach:

- **Screen Identification and Action Ports:** Each screen within the application is assigned a unique name. Each screen also has associated 'action ports,' which serve as potential navigation endpoints from the given screen. For instance, a screen named "Screen A" may have action ports like "Login" and "SignUp," representing the possible navigational paths from "Screen A."
- **Web Portal for Screen Flow Modification:** Flow Magic features a specialized web portal where product owners can alter the app's screen flow. Modifications made in this portal change the configuration of the action ports on the backend. This capability facilitates real-time, dynamic modifications to the screen flow within the app.

4. Technology Stack

Flow Magic is developed using the latest and most popular technology in the current market. The following diagram gives an overview of the technology stack used. This chapter provides an overview and discusses why they are used.

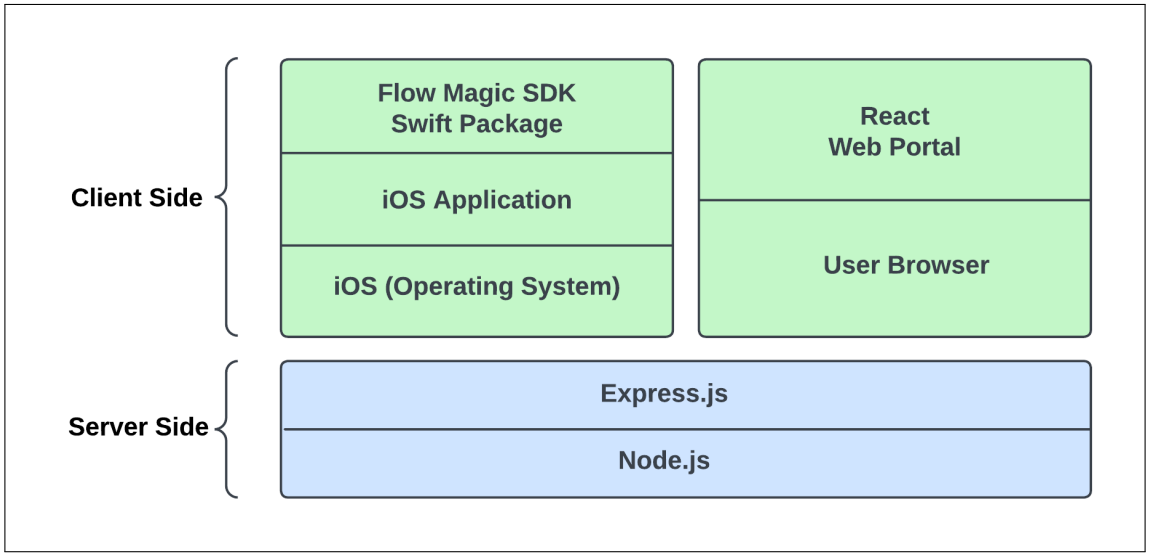


Figure 4.1: Technolgy Stack

4.1 Frontend

4.1.1 Swift Package

According to Apple Developer documentation, Swift packages are components of Swift, Objective-C, Objective-C++, C, or C++ code, which can be reused in various projects. They encapsulate source files, binaries, and resources for convenient use in application projects.[9] The Flow Magic SDK is developed as a Swift Package. This format enables seamless integration into projects using Swift Package Manager, a dependency manager. The decision to

develop the SDK as a Swift package facilitates code reusability and sharing of functionalities with other developers. GitHub URL: https://github.com/KedarDhere/FlowMagic-SDK.

4.1.2 React

Flow Magic's web portal is built using React. React is a JavaScript library developed by Meta and used to build the frontend [10]. React is widely used in the current market, has a strong community, and has detailed documentation. Meta regularly releases updates and makes it robust. React was selected to build the Flow Magic web portal, considering these factors.

4.1.3 React Flow

React Flow is a library designed for creating node-based applications [11]. In the context of Flow Magic, each mobile application screen is treated as a node (as shown in Figure 1.1). React Flow's capability to create custom nodes aligns well with the main concept of action ports in Flow Magic. It enables easy configuration of mobile screen flow visualization, allowing for drag-and-drop functionality and zoom-in and out features.

4.2 Backend

4.2.1 Node

"Node.js® is an open-source, cross-platform JavaScript runtime environment. [12]" It enables the execution of JavaScript code outside a web browser. Flow Magic uses Node.js to set up its backend server by leveraging its efficiency and scalability in handling backend processes.

4.2.2 Express

Express is a backend node-based web application framework for building REST APIs [13]. It is widely used as a standard server framework for Node.js applications [13]. The framework's ease of use in setting up servers and implementing REST API routing makes it a preferred choice for developers. Flow Magic uses Express for REST API routing.

4.2.3 Google OAuth

O-Auth2 is a protocol in which the resource owner(an individual) does not need to provide the credentials to the client(a website or an application); instead, the resource owner grants permission to the third-party service (like Google or Facebook) to provide the ability to the client to access the protected resource. In this case, the end user is the product owner who wants to access the Flow Magic Web portal(client), so the end user authenticates itself with Google. Google, in turn, returns an access token to the Flow Magic server. Flow Magic server uses this access token to verify the user and grants access if the user is authorized. Flow Magic server uses the npm module Passport for authorizing the user.

5. System Architecture

This topic gives a detailed overview of the Flow Magic system architecture. It starts by discussing the high-level architecture, then discusses in detail the MVVM architecture of the Flow Magic SDK, and then explains the interaction between the components through the sequence diagrams.

The following diagram provides a high-level system architecture of the Flow Magic. It mainly consists of four parts.

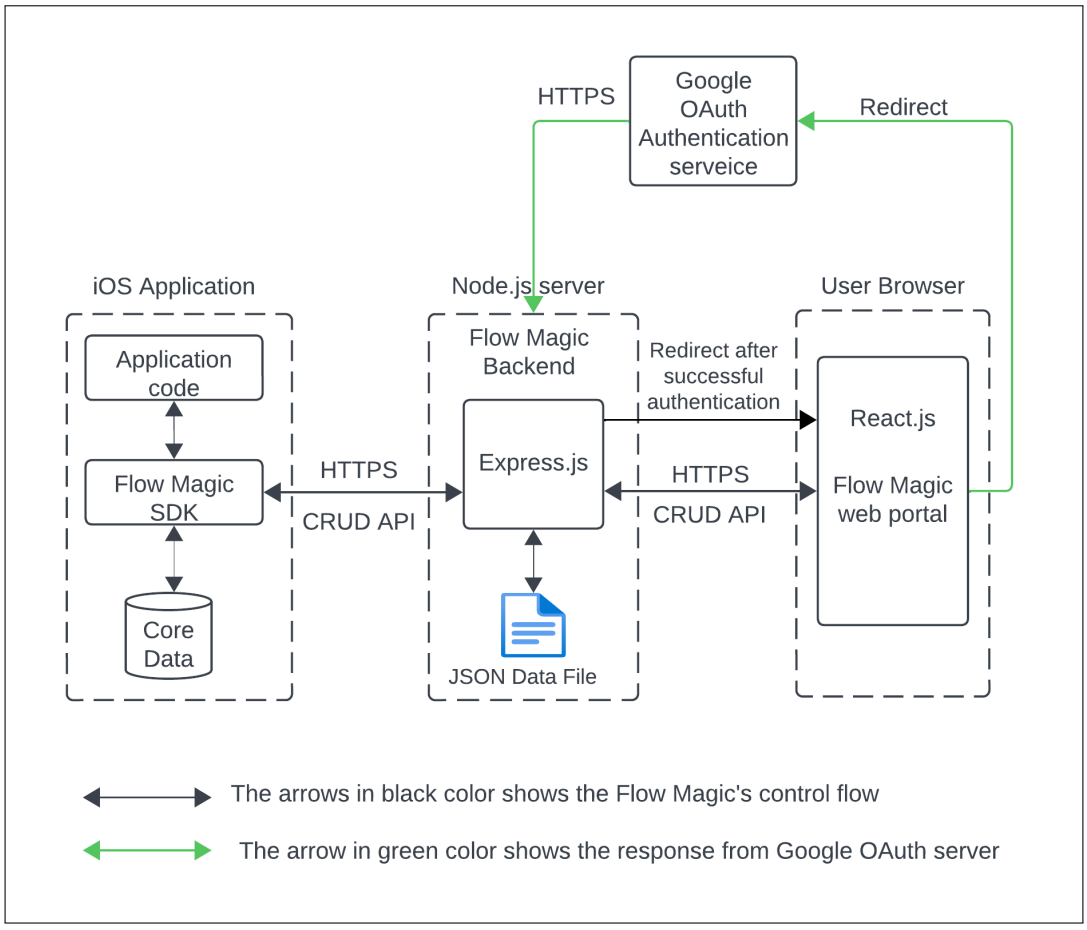


Figure 5.1: System Architecture

1. **iOS Application with Flow Magic SDK:** Developers need to integrate the Flow Magic SDK into their iOS application. This will enable developers to dynamically change the navigation between the existing views/screens. Integration of the Flow Magic SDK into the iOS application development will retain compatibility with the core Swift UI functionalities, allowing developers unhindered access to these standard features.

2. **Node. js Server:** The next block is a backend server built in Node.js using the Express framework. It interacts with both the iOS application and the web portal through REST APIs. The server then handles the routing of the APIs, processing the request and sending the response back to the particular clients.

3. **Flow Magic Web Portal (React-based):** The web portal is built using React, and the React Flow library is used to display the screen flow using customized nodes and to provide the change of the flow using a drag-and-drop mechanism. Each organization requires only a single portal for all its applications. For example, Oregon State University (OSU) manages various mobile applications, such as OSU Mobile, OSU Bever bus, and Safe Ride, which are all accessible through a single portal, thus removing the need for separate accounts for each application.

4. **Security and Authentication via Google OAuth 2.0:** The Flow Magic portal needs the user to be authenticated before gaining access to the portal. This step ensures that only those with authorization can access the platform, enhancing the system's security.

5.1 FlowMagic SDK architecture

Flow Magic SDK follows the MVVM architecture.

MVVM Architecture:

The following diagram explains the workings of MVVM architecture.

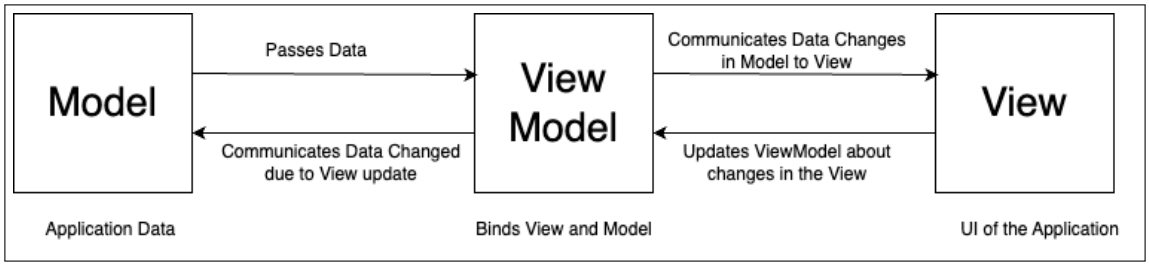


Figure 5.2: MVVM

View: The view is the UI of the application. It is responsible for displaying the data to the user and responding to user input. Sometimes, updating the Model as per the user's action on the UI is required. View calls the functions defined in the ViewModel, and then ViewModel asks the Model to update the data.

ViewModel: The ViewModel is the intermediary between the View and the Model. ViewModel provides data to View for display. ViewModel keeps track of changes in the Model, and as soon as it detects any changes in the data, ViewModel publishes those changes to all the Views. Views that subscribe to these announcements will listen to them and update themselves.

Model: The Model is the application's data layer. It stores the data that is used by the View. This component operates independently of the user interface and the ViewModel, focusing solely on the data's logic and handling.

5.1.1 Flow Magic SDK's MVVM implementation

The following diagram shows the system architecture of SDK.

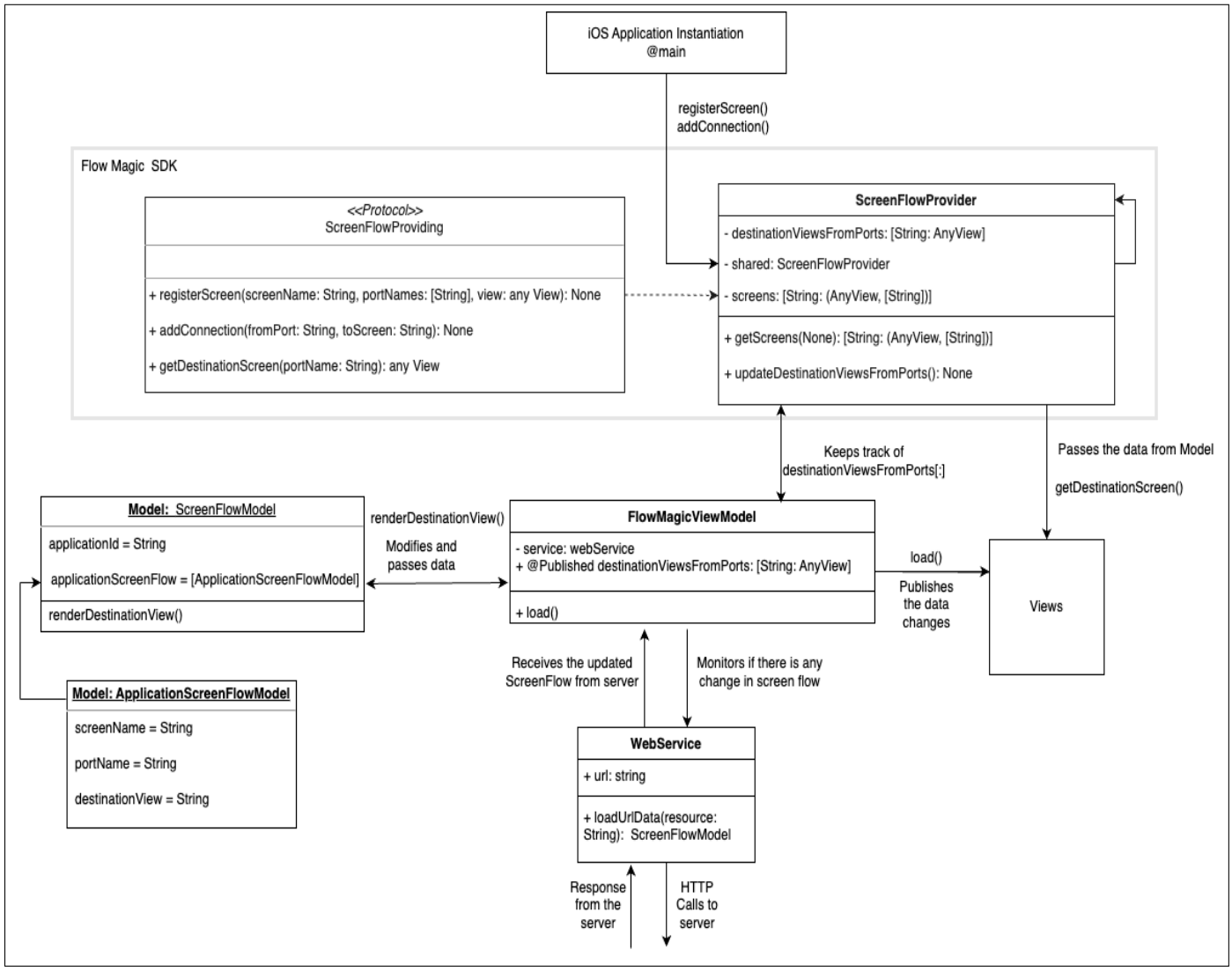


Figure 5.3: SDK System architecture

The Flow Magic SDK provides a structured way for developers to design and manage how users navigate through various screens in an application. Here's a simplified breakdown of its operation:

Initialization: At the start, developers establish the primary screens and their connections. This step determines the layout and flow of screens that users will navigate through.

ScreenFlowProvider's Role: This component acts as a centralized reference, holding essential details about the screens and their connections. It ensures consistency and accuracy in navigation.

User Interface (Views): Views refer to the actual screens or pages that users interact with. They can adapt and change based on the predetermined flow and any updates from the backend.

Data Structure: The `ScreenFlowModel` is a data structure that holds critical identifiers and outlines how different screens in the application are connected.

Server Communication: The `ViewModel` facilitates interaction with external servers. It fetches any updates or changes and ensures the application's screens reflect these modifications in real time.

Dynamic Updates: The `ViewModel` possesses a monitoring mechanism. If there's an alteration in the screen flow or any other updates, it automatically triggers an adjustment in the displayed screen to reflect the most current information.

User Experience: Ensuring a smooth and updated view for users is paramount. To achieve this, the screens (or views) are consistently refreshed based on real-time data and flow adjustments.

5.1.2 Sequence Diagram

All of these flows show the happy path.

1. GetScreenFlow

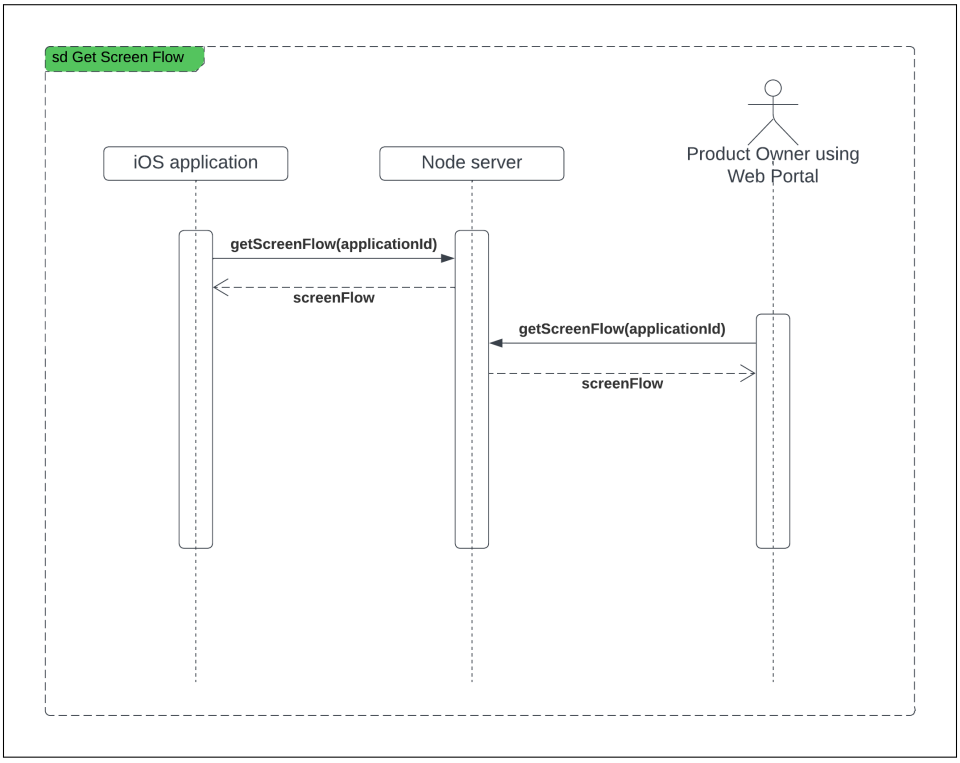


Figure 5.4: Get Screen Flow

A. Communication between iOS Application and server

1. Once the application deploys to Production, it starts communicating with the server to get the updated screen flow.
2. The server sends the screen flow back to the iOS application.
3. The iOS application rebuilds views (if required) as per the response received from the server.

B. Communication between Portal and server

1. The product owner can access the application's screen flow, after the application's deployment.The Product Owner logs into the web portal and selects the desired application, triggering a getScreenFlow() request to the server.
2. The server responds by sending the current screen flow configuration back to the web portal.
3. Upon receiving this data, the web portal displays the screen flow for the Product Owner's review.

Both of these communications are running simultaneously.

2. UpdateScreenFlow

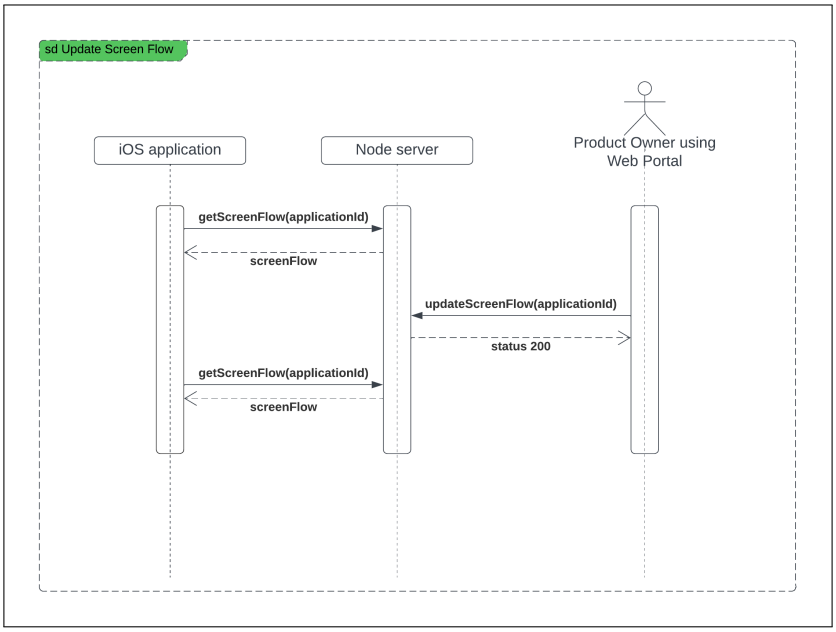


Figure 5.5: Update Screen Flow

1. The product owner changes the screen flow and sends the updated screen flow to the portal.
2. The portal processes the changes and sends the update with a status 200 response and the revised screen flow data.

3. The iOS application requests the server to get the updated screen flow. Once the updated screen flow is received, it displays the new screen flow in real time.

6. API Documentation

REST APIs

1. Get all the registered applications

Request: `GET /applications/{companyName}`

Response body:

```
[
    "applicationsData": [
        {
            "id": "66ceb688-a2b3-11ed-a8fc-0242ac120002",
            "applicationName": "SocialBook"
        },
        {
            "id": "66ceb688-a2b3-11ed-a8fc-0242ac120003",
            "applicationName": "Instagram"
        },
        {
            "id": "66ceb688-a2b3-11ed-a8fc-0242ac120004",
            "applicationName": "Music"
        }
    ]
]
```

2. Get the application's screen flow

Request: `GET /applications/  applicationId / screenFlow`

Response body:

```
{
  "id": "1234667812345678",
  "screenFlow": [
    {
      "screenName": "Home",
```

```
      "portName": "Home.Login",
      "destinationView": "Login"
    }
  ]
}
```

3. Modify application's screen flow

Request: `PUT /applications/  applicationId / screenFlow`

Response body:

```
{
  "id": "1234667812345678",
  "screenFlow": [
    {
      "screenName": "Home",
      "portName": "Home.Login",
      "destinationView": "Login"
    }
  ]
}
```

6.1 Swagger Document

Swagger is a tool that allows for the documentation of REST APIs according to the OpenAPI specification. The OpenAPI specification, formerly known as the Swagger specification, is a format that enables the description of APIs. This includes documenting endpoints, security protocols used, and optional and required parameters, among others11. Apart from the REST API documentation, Swagger also allows API testing and facilitates sharing this documentation with others. Its interactive UI simplifies the understanding of APIs for all application stakeholders11. The following figure illustrates the major Flow Magic API endpoints, their required parameters, and their respective HTTP verbs. The detailed YAML file is available on GitHub https://github.com/KedarDhere/FlowMagic/tree/main/FlowMagic-MockServer/APIDocumentation.

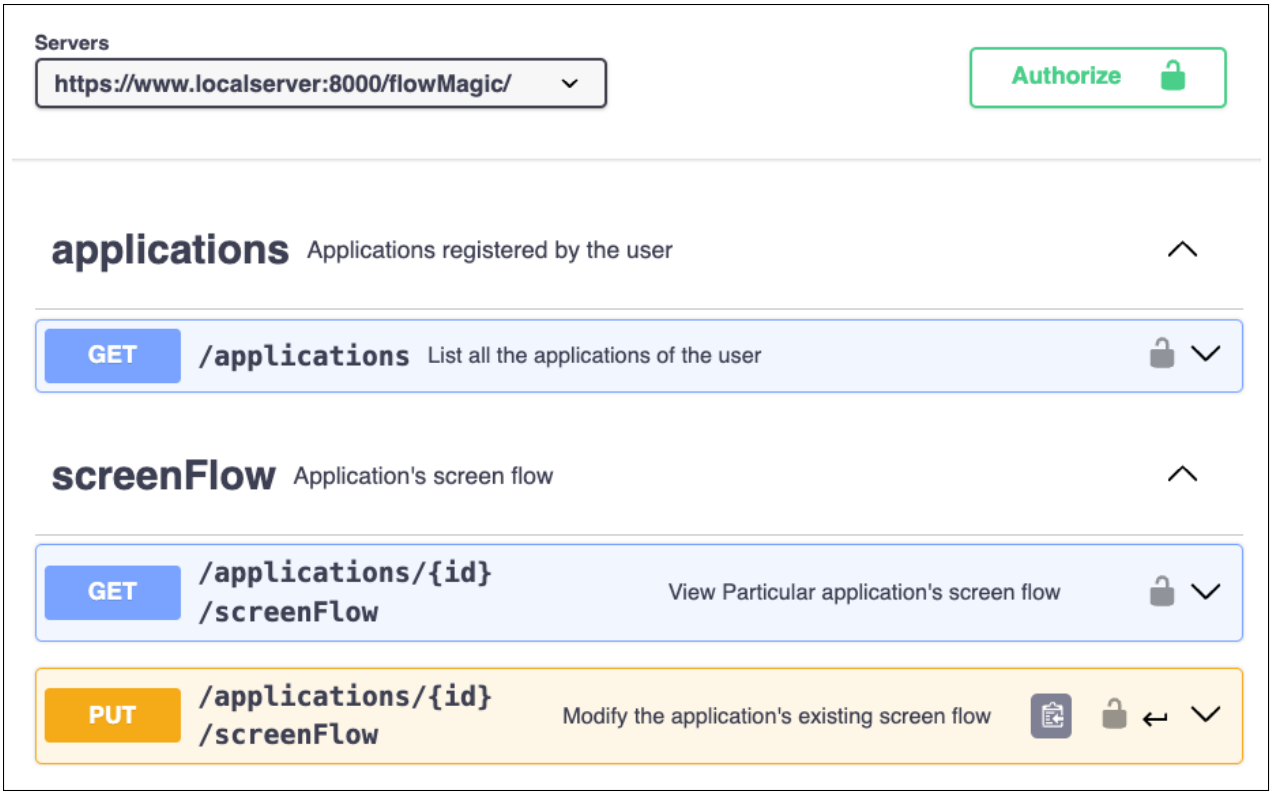


Figure 6.1: Swagger

7. Unit Testing

Unit testing is a technique where the software's "unit" or "individual component" is tested in isolation. Unit tests are generally automated. Automated unit tests help to detect bugs early in the software development process. There are many standard unit testing frameworks available. This chapter explains the various testing frameworks used in the Flow Magic test suite.

7.1 Unit Testing in Flow Magic SDK

7.1.1 XCTest for Swift Package

Flow Magic SDK, a Swift Package, uses Swift's standard testing framework, 'XCTest.' XCTest allows the creation and execution of test cases and also provides performance metrics. XCode integrates with the XCTest, gives the code coverage report, and indicates the code not covered by the unit test in red color. This helps to know which part of the code is not covered by the test suite.

7.1.2 Mocking dependencies

Unit testing often requires mocking dependencies to ensure the isolation of individual components for testing. In the Flow Magic SDK, this is achieved by injecting mock dependency objects of dependent classes instead of actual production instances as parameters into the classes that are the focus of testing.

7.2 Testing in Flow Magic Backend Server

7.2.1 Jest, Supertest

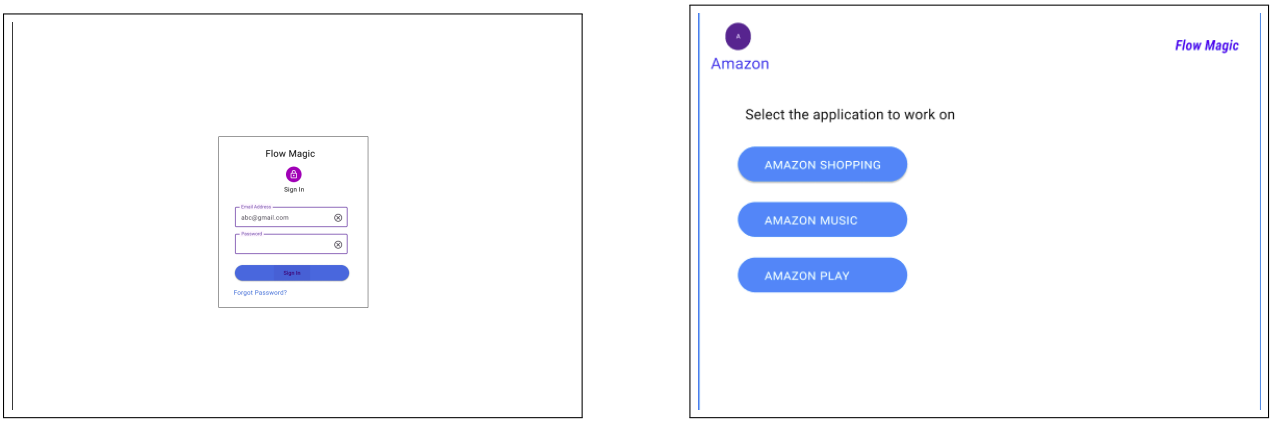
Flow Magic backend server test suite uses three testing frameworks: Jest, Supertest, and Nock. Jest is used to create test cases and provides an environment to execute the test cases. Suptetest is used to test the HTTP requests, and nock is used to mock the HTTP responses. The test coverage report can be obtained through the Jest command jest –coverage.

7.3 Continuous Integration Workflow

Flow Magic implements the Continuous Integration (CI) workflow for the Node backend and Swift Package using GitHub Actions. This CI pipeline automatically executes both test suites before merging any changes to the main branch. This ensures error-free deployment to the main branch.

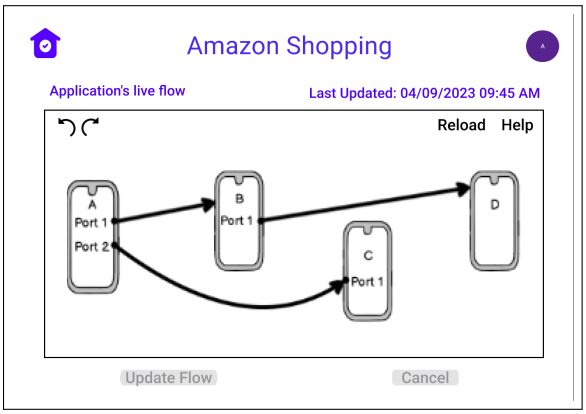
8. Figma Prototypes

Figma operates as a cloud-based digital design and prototyping tool, particularly for creating user interfaces (UI) and user experiences (UX). It enables designers to create responsive, dynamic prototypes for websites, applications, or digital products, all without the need to write code. This functionality allows designers to simulate and review how the final product will look and function in real time. In the design process, Figma plays a critical role. It aids in the early detection and correction of design issues, much like a testing phase in software development. During project implementation, Figma proves invaluable in the visualization and planning stages, allowing for iterative improvements. Adjustments to the final product are often based on insights gained from these initial Figma prototypes, ensuring a functional and aesthetically pleasing.

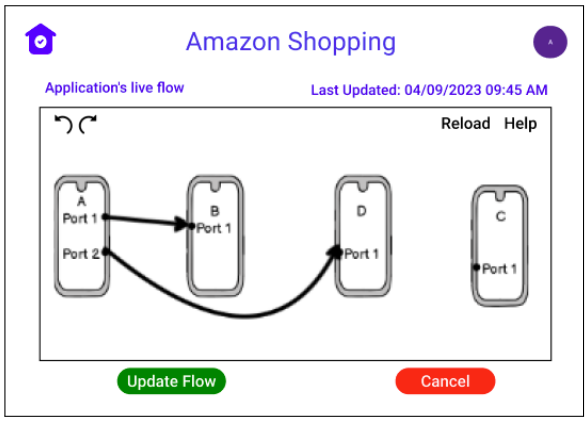


(a) Login Page (b) Application Selection Page

Figure 8.1: Login and Application Selection Pages

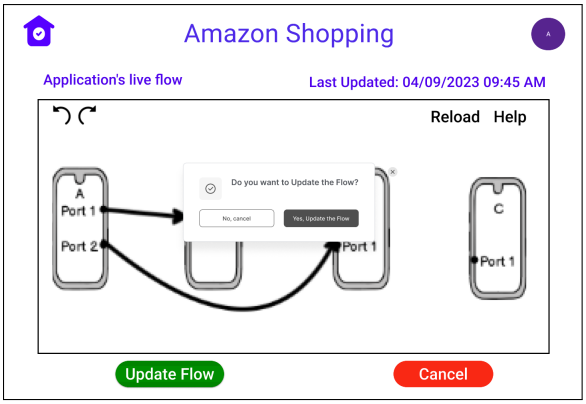


(a) Screen Flow Display Page

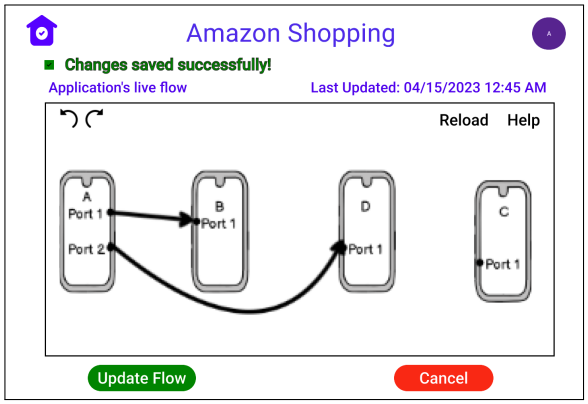


(b) After changing the screen flow

Figure 8.2: Screen Flow Display and After Changing the Screen Flow



(a) After clicking on the update flow button



(b) After successfully updating the Flow

Figure 8.3: After Clicking on the Update Flow Button and After Successfully Updating the Flow

9. Application Screenshots

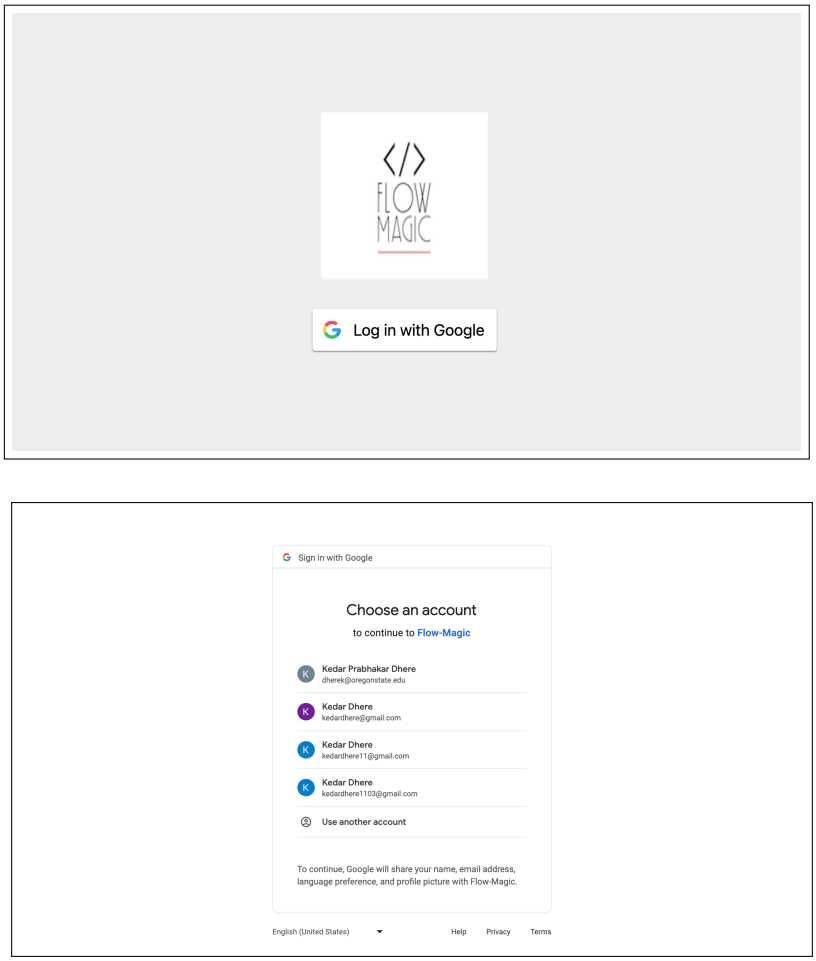


Figure 9.1: Login Page - This figure showcases the initial login page where users have the convenience to authenticate their identity using Google's login service.

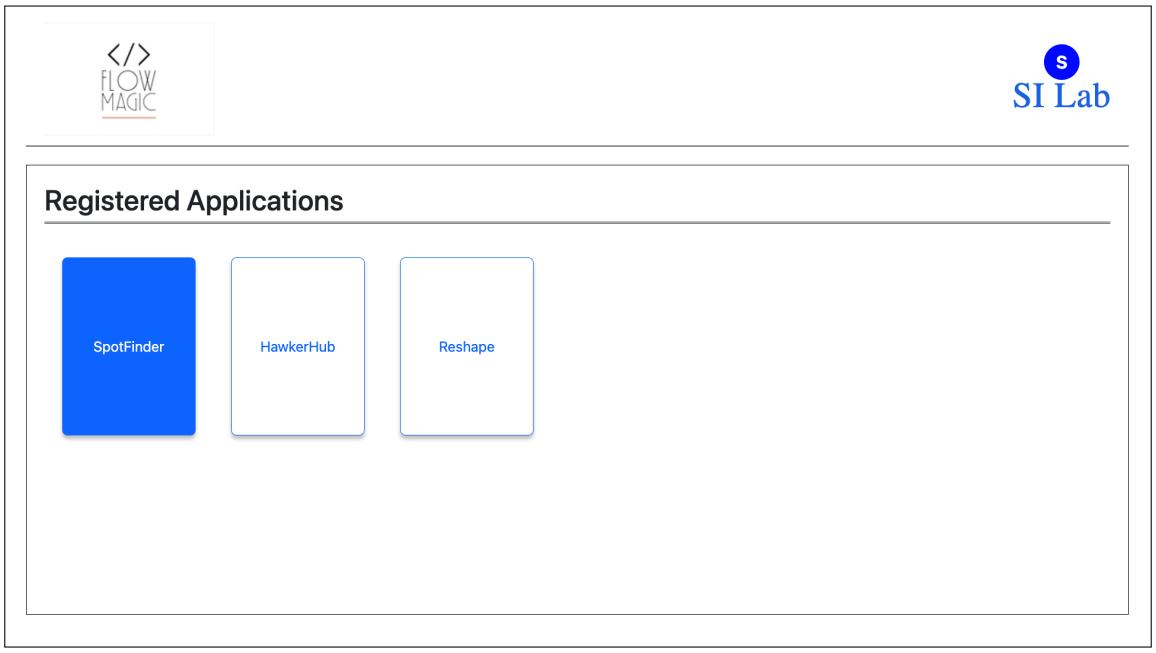


Figure 9.2: Dashboard Post-Login - Once authenticated, this figure illustrates the dashboard where users can view a list of all the applications they have registered on the platform.

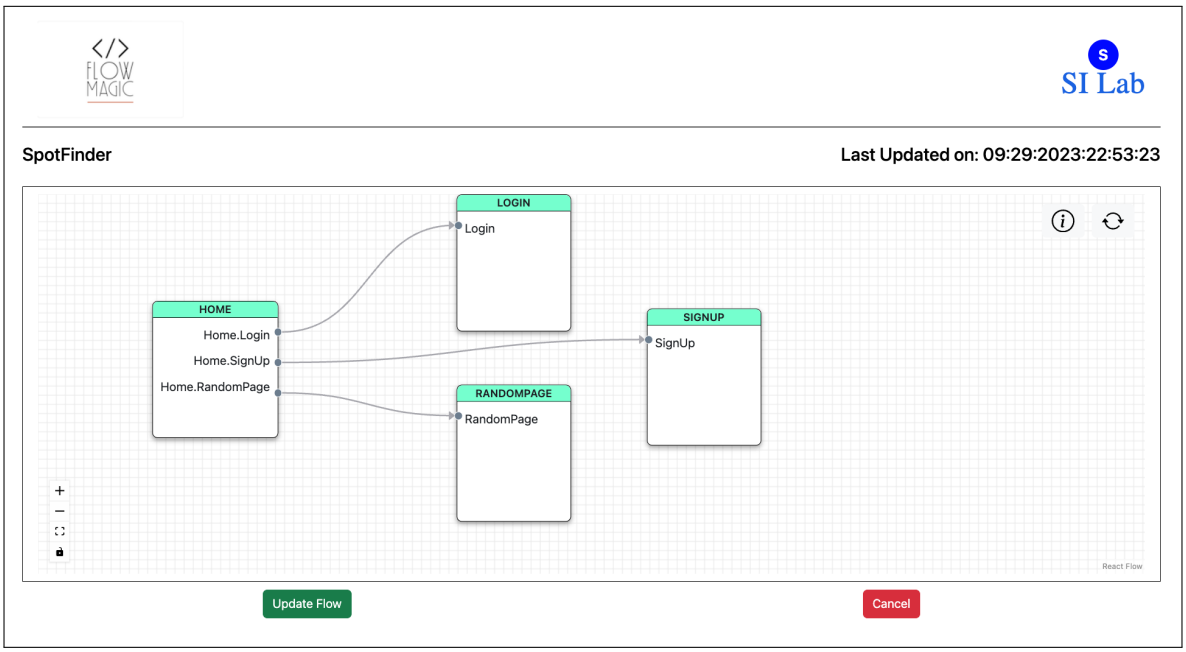


Figure 9.3: Application Screen Flow - This figure presents a detailed flowchart or map of the selected application after a user has chosen an application from their registered list.

Figure 9.4: Screen Flow Update and Pop-up Notification - Depicting a modification to the application's screen flow, this figure also highlights a pop-up notification as part of the change

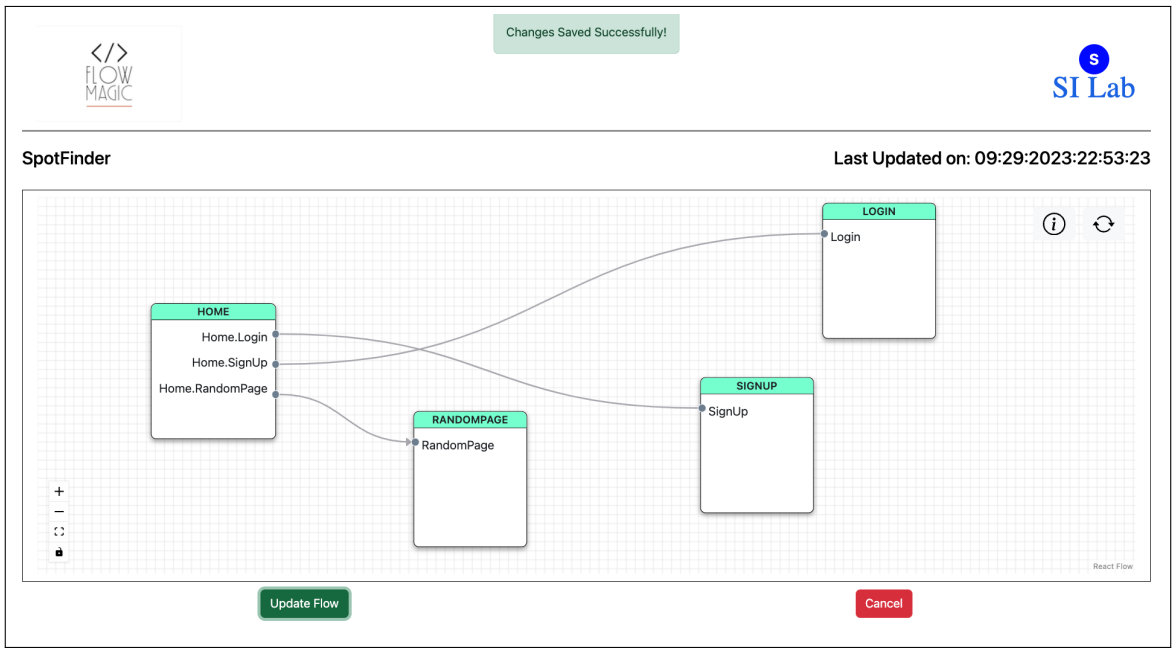


Figure 9.5: Update Confirmation and Timestamp - This figure captures a successful alert message, confirming that the screen flow has been updated. Alongside, it also shows the 'Last Updated On' timestamp reflecting the most recent change.

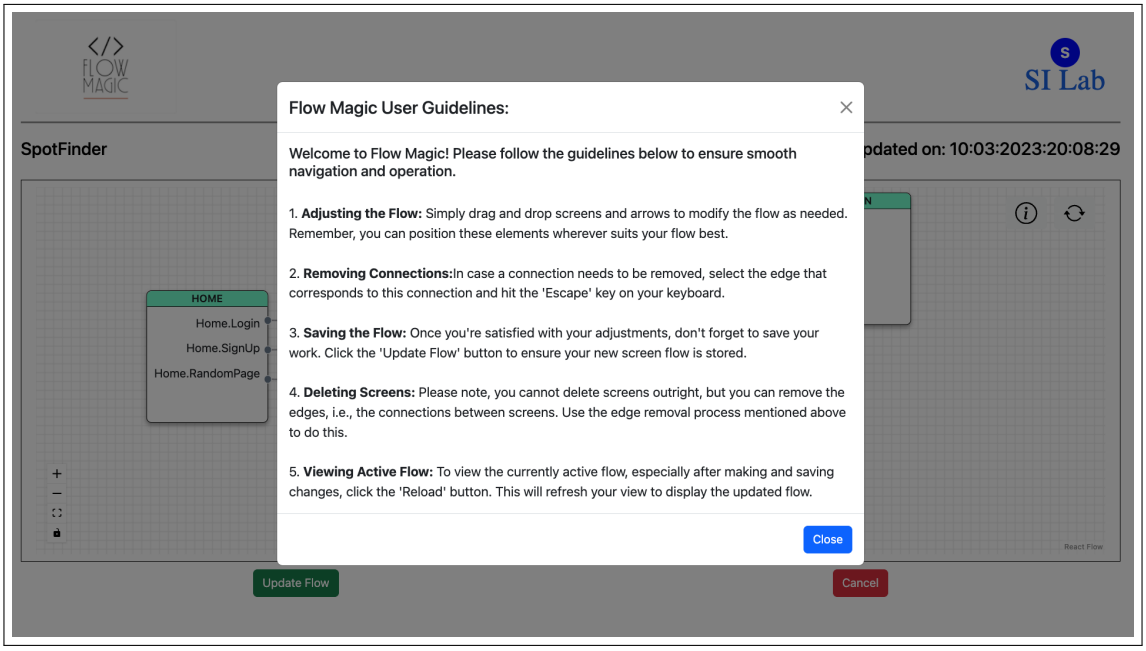


Figure 9.6: Usage Guidelines Information - This figure represents what users see upon clicking the information button: a display of the application guidelines.

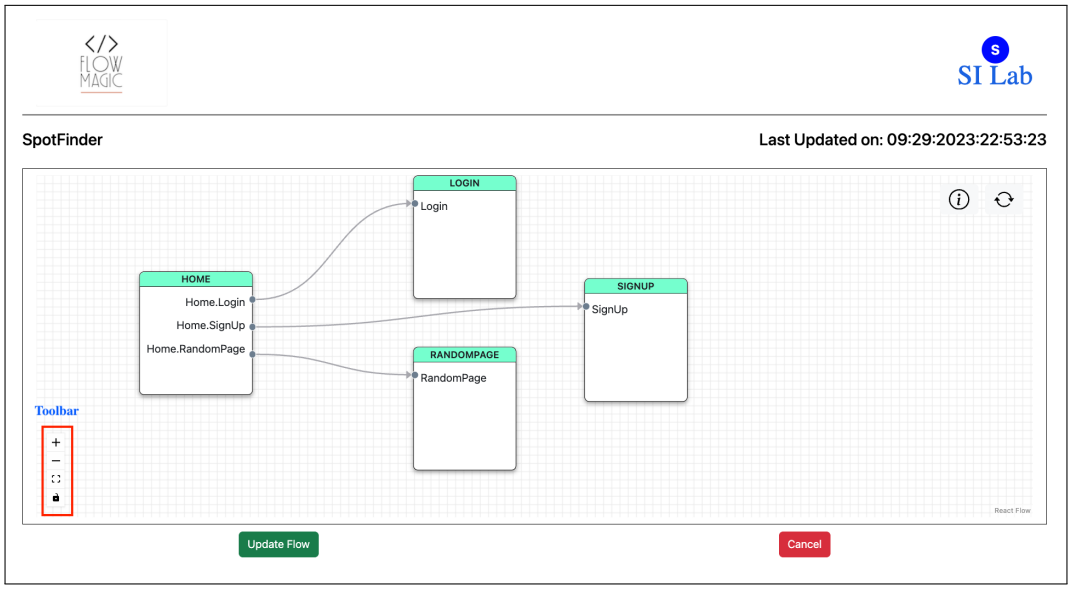


Figure 9.7: Toolbar Functionality - Located at the lower-left corner of the screen, this figure demonstrates the toolbar, providing options for users to zoom in, zoom out, highlight their current location on the minimap, and use a tool for rectangular dragging within the application.

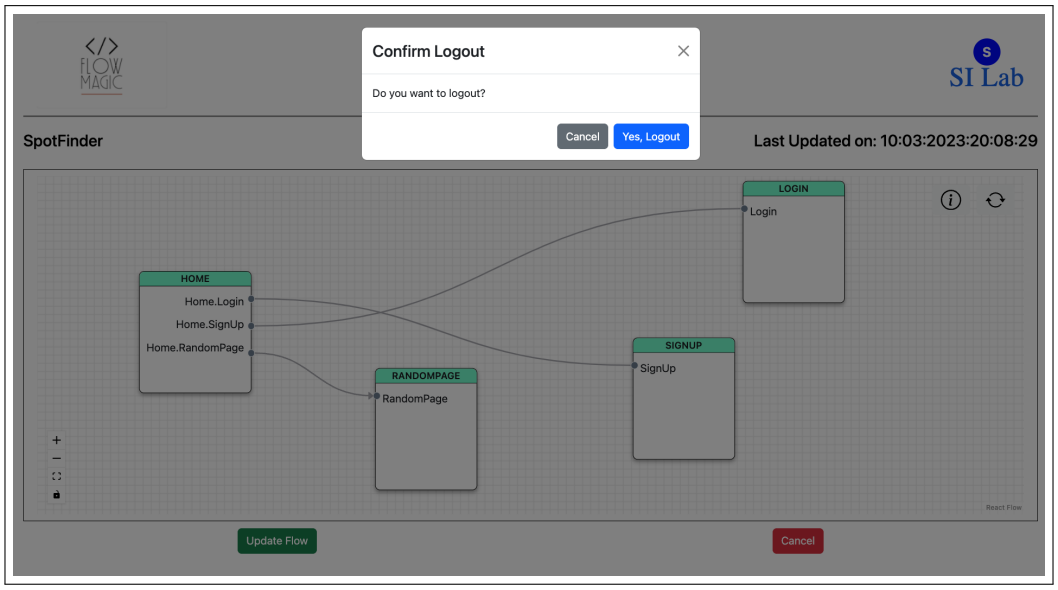


Figure 9.8: Logout Option - This figure shows the dropdown menu that appears when users click on their initials at the top corner of the screen. It includes an option to log out of the application.

10. Future Scope

This chapter will explore the future possibilities for 'Flow Magic.' It discusses potential enhancements and additional functionalities that could be integrated into the tool to further improve its efficiency and user experience.

- Condition-based Screen Flow customization: If the product owner wishes to configure and change the flow of screens based on certain conditions, this capability can be implemented in the future.
- Expanding Screen Functionality: At present, Flow Magic permits changing the flow of an existing screen. If a new screen is required, it must first be introduced using the SDK before the screen flow can be modified via the portal. In future iterations, we could expand the functionality to incorporate adding new screens and establishing connections between them.
- Expanding Platform Compatibility: The current service could be enhanced by extending its compatibility to other development platforms such as Flutter and React Native. This expansion would allow for greater versatility in its application and reach a wider user base in the development community.
- Incorporating A/B Testing Capabilities: The A/B testing tool provides the mechanism to provide different options to the different user sets. In terms of the screen flow, as explained in Chapter 2: Existing Solutions, these tools allow to change the screen flow based on flag. This A/B testing functionality can be integrated into the web portal. This would enable the Product Owner to implement the A/B testing along with dynamically changing the screen flow.

11. Conclusion

Flow Magic facilitates real-time modifications to the screen flow of iOS mobile applications, offering the capability to respond quickly to evolving user needs and feedback. Flow Magic can be beneficial for organizations to improve the usability and retention of their users by reducing the need for extensive reprogramming and bridging the gap between technical development and user demands.

One of the major advantages of Flow Magic is its cost-effectiveness and efficiency compared to traditional application design and development methods. Unlike conventional approaches that often require laborious and time-consuming reprogramming to accommodate changes, Flow Magic's streamlined process allows for rapid responses to necessary alterations, saving valuable time and resources.

In the future, there is potential for extending the platform's capabilities to cater to other development platforms, incorporating A/B Testing capabilities, and utilizing Machine Learning-based recommendations to create even more user-centric screen flows.

Overall, as Flow Magic continues to evolve and incorporate new technologies, it has the potential to become an even more valuable tool for organizations which helps to rapid mobile development and more user satisfaction.

References

[1] "Mobile operating system market share statistics." https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/. [Online; accessed 16-Nov-2023].

[2] "App store (apple)." https://en.wikipedia.org/wiki/App_Store_(Apple)#:~:text=Apple%20announced%20Mac%20App%20Store,6%20%22Snow%20Leopard%22%20update. [Online; accessed 16-Nov-2023].

[3] "What are user flows in ux." https://www.simplilearn.com/tutorials/ui-ux-career-resources/user-flows-in-ux#:~:text=User%20flows%20are%20an%20important,%2C%20flowcharts%2C%20and%20other%20tools. [Online; accessed 16-Nov-2023].

[4] "App review." https://developer.apple.com/app-store/review/. [Online; accessed 16-Nov-2023].

[5] "Json." https://en.wikipedia.org/wiki/JSON. [Online; accessed 16-Nov-2023].

[6] "Documentation: Beagle overview." https://docs.usebeagle.io/v2.1/overview/. [Online; accessed 16-Nov-2023].

[7] "Documentation: Customization for ios applications." https://docs.usebeagle.io/v2.1/ios/customization/beagle-screen-view-controller/. [Online; accessed 16-Nov-2023].

[8] "Project report: Reshape ui." https://research.engr.oregonstate.edu/si-lab/archive/2022_siri. [Online; accessed 16-Nov-2023].

[9] "Swift packages." https://developer.apple.com/documentation/xcode/swift-packages. [Online; accessed 16-Nov-2023].

[10] "React." https://en.wikipedia.org/wiki/React_(software). [Online; accessed 16-Nov-2023].

[11] "React flow." https://reactflow.dev/learn/concepts/introduction. [Online; accessed 16-Nov-2023].

[12] "Node.js." https://nodejs.org/en. [Online; accessed 16-Nov-2023].

[13] "Express." https://en.wikipedia.org/wiki/Express.js. [Online; accessed 16-Nov-2023].