**BOILERPLATE**

Product Requirements Document

Team C Language learning AI gane| Boilerplate | 07/07/2024

## **Introduction**

This Product Requirement Document (PRD) outlines the requirements for a boilerplate, a foundational framework designed to provide a flexible and scalable starting point for developers when building various applications and use cases. This PRD will detail the purpose, features, key objectives, assumptions, user personas and key features of the boilerplate. It will also outline the core UX flow, success metrics and roadmap of the product.

## **Table of Contents**

1. [Product Vision](#_zgl8fc9gzz09)
2. [Background](#_cuyzxg7xj3qf)
3. [Problem statement](#_z7h2kbp5xkwu)
4. [Goals/Objectives](#_mmsh9a3dltuc)
5. [Assumptions/Hypotheses](#_643h9gm8jdrj)
6. [User Personas](#_53fu3gsrisju)
7. [Key Features/Functionalities](#_1xoo8maaxsg2)
8. [Core UX flow](#_367c0926acl0)
9. [Success Metrics](#_msfgumudlm0q)
10. [Roadmap](#_lszx1h22n1ac)

## **Product Vision**

The Boilerplate aims to **provide developers and development teams**, who **need an efficient starting point** for their **projects**, with **a blueprint and solid foundation for their work.** It is a comprehensive project framework that accelerates development time by offering a well-structured, feature-rich, and scalable foundation. Our boilerplate targets junior to senior developers and tech startups, primarily located in urban cities across Africa. Unlike other boilerplates that may lack essential features or clear documentation, our product **provides a modular architecture,** **detailed documentation**, and **built-in support** for common development needs, ensuring a smoother and **more productive development experience**.

## **Background**

Many developers seek to create dynamic and robust web solutions quickly, increasing the demand for **efficient**, **maintainable**, and **scalable** applications. The Boilerplate will **provide a solid foundation for products, offering common functionalities,** which significantly reduces the time required to start new projects. Having a reliable and well made boilerplate allows the development teams to **focus on building unique features** rather than starting from scratch.

Boilerplate code covers basic setup arrangement like **setting up tools**, **organising folders**, and **handling basic errors**. It gives developers a starting point to build more complicated features quickly. By doing these setup tasks automatically, boilerplate code makes it easier to develop apps consistently and lets developers spend more time on the unique parts of their apps—like the special things that make them different and unique for users.

**Problem Statement**

When developers are building a product, they **spend a lot of time building features that are not unique** to the application they are building. Also, **repeating code leaves room for errors and leads to overall poor quality code**..

## **Goals/Objectives**

The main objectives of the PHP Boilerplate is to ensure standardisation of the code produced by our PHP developers

It is also to save time spent building non-unique app features. reduce the time and effort required to set up a new product and get rid of repetitive configuration.

The key goal of the PHP boilerplate is to ensure consistency, boost the development of a product and get rid of repetitive configuration.

High-level objectives

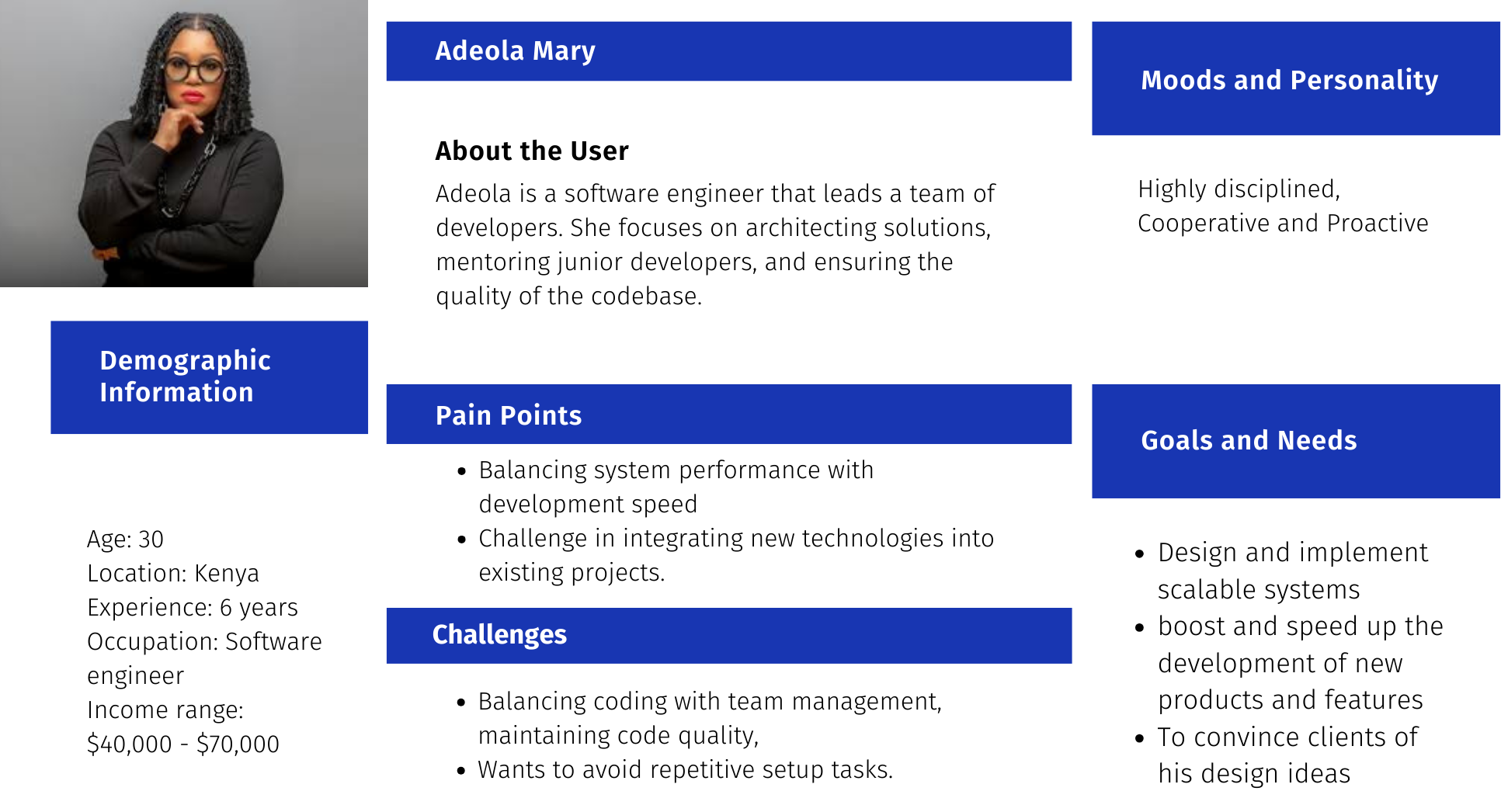
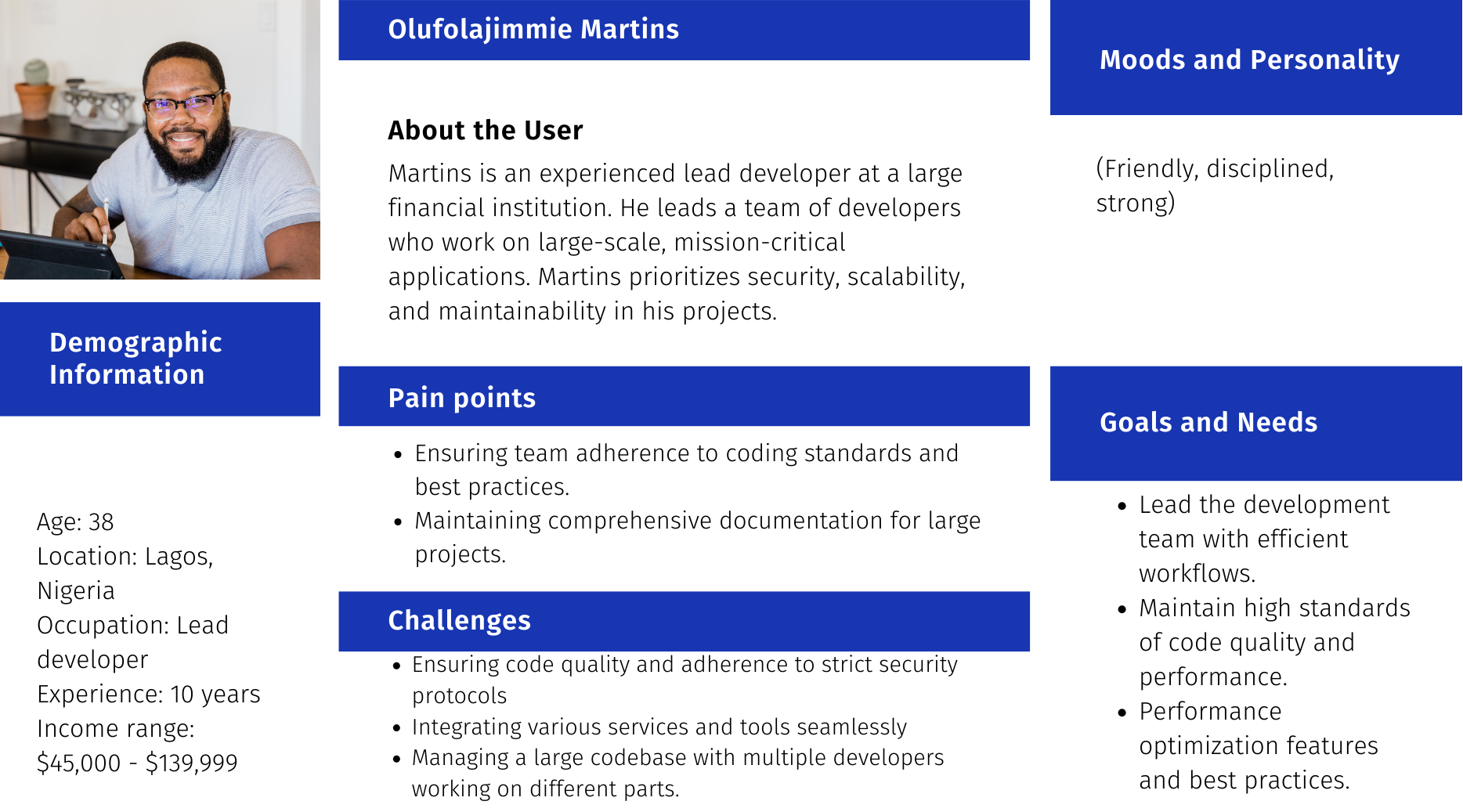
* Adoption and Integration: It would allow for a wide adoption within the development team, integrating into our workflow and speeding up project initiation.
* Improve efficiency: Developers should focus more on implementing unique features and enhancing user experience.
* Scalability: It should be able to scale applications smoothly as our user base grows, with the boilerplate providing a solid foundation for future expansions and feature enhancements.

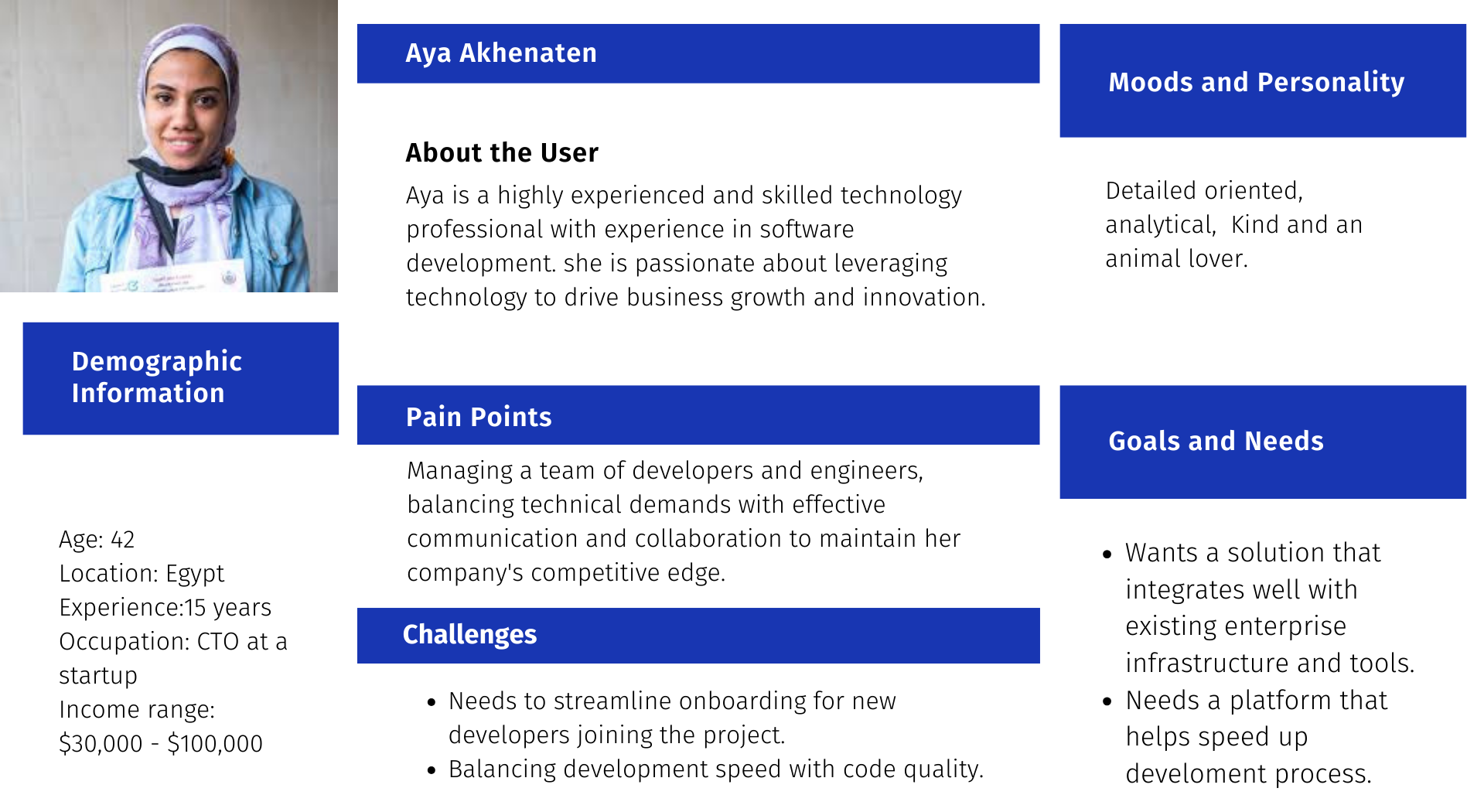
## **Assumptions/Hypothesis**

Our assumptions are that:

* Developers don’t have any aspect of the application software built and would have to start from scratch.
* There would be teams of developers collaborating remotely.
* Engineers will be joining the product at different stages.
* The boilerplate will be stored in a central location that is accessible to the development team.
* Every new product requires repetitive tasks which can be automated
* Developers might set up products in inconsistent ways leading to maintenance challenges. This assumption is true because developers have different preferences and practices which can result in varied product structures that are harder to maintain and understand
* The initial set-up of a product can be time-consuming and error-prone.

## **User Personas**





## **Key Features/Functionalities**

**These are the core features of the boilerplate:**

* **User Authentication:** Secure login and registration with social auth and magic links.
* **AI Integration**
* **Messaging**
* **Payments**
* **Users & Organisations**
* **Admin Interface:** Dashboard for managing users, content, and settings.
* **Landing Page, Settings Page**
* **Database:** Robust infrastructure for storing and managing data.
* **Waitlist and Invite Flow:** User access management
* **Migrations:** Ensuring data integrity and smooth updates.
* **Blogs:** Detailed guides on how to use each feature.

[This document](https://docs.google.com/document/d/1DBuzQHpp1zdgqiJNpGk3f36Z5wmX1e-ebC1baNHv9l0/edit?usp=drivesdk) contains a more detailed description, requirements and acceptance criteria of the product’s core features.

**User stories**

| **#** | **Feature** | **User Story** | **Acceptance criteria** |
| --- | --- | --- | --- |
| 1 | **AUTHENTICATION** | * As a user, I want to be able to log in to my account. * As a user, I want to be able to reset my password so that I can regain access to my account if I forget my credentials. * As a user, I want to be able to create an account so that I can save my progress and settings for future use. | * Easy to sign-up or login * Get an email to store password securely |
| 2 | **WAITLIST** | * As a potential user, I want to join the waitlist to be notified when the product becomes available. | * Able to receive update about the product via email * Can easily unsubscribe from waitlist when ever I want |
| 3 | **AI INTEGRATION** |  | * AI algorithms should analyze performances in real-time. |
| 4 | **SETTINGS** | * As a user, I want to be able to customise the application's appearance and behaviour so that it suits my needs. * As a user, I want to be able to update my account information. | It should:   * allows users to edit their account information (e.g., name, email, password). * be able to validate user input to ensure accuracy and completeness. * Save changes and update the user's account information successfully. |
| 5 | **LANDING PAGE** | * As a user, I want to see a clear and concise explanation of the app's benefits and features so that I can quickly understand what the app does. * As a user, I want to be able to easily navigate to the app's main features and functionality from the landing page so that I can get started right away.. | * Have clear, concise information about the purpose and benefits of the product.. |

## **Core UX Flow**

Here are user flows and wireframes for the product:

* This user authentication flow outlines the steps a user takes to log in to their account within a product.

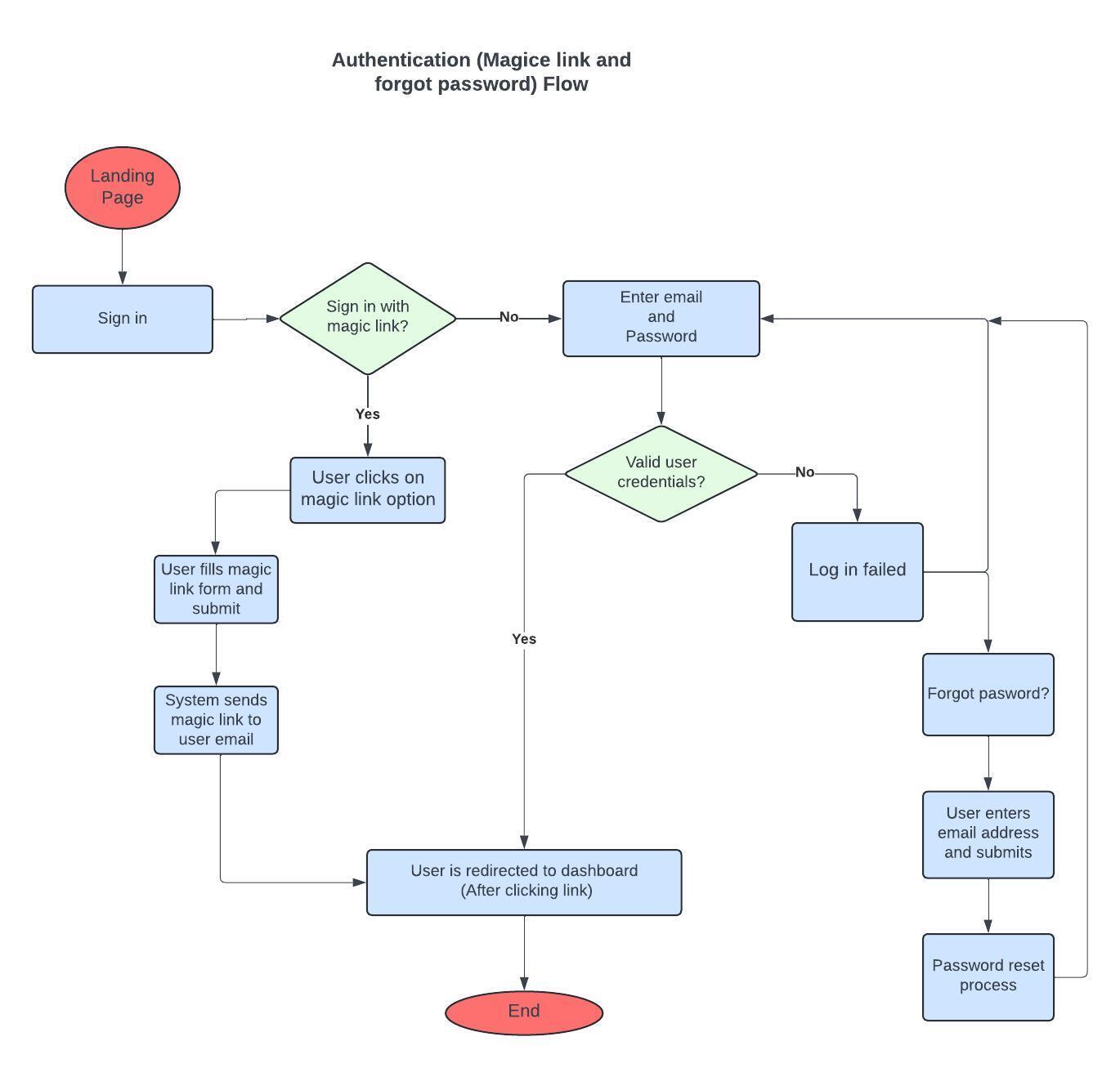
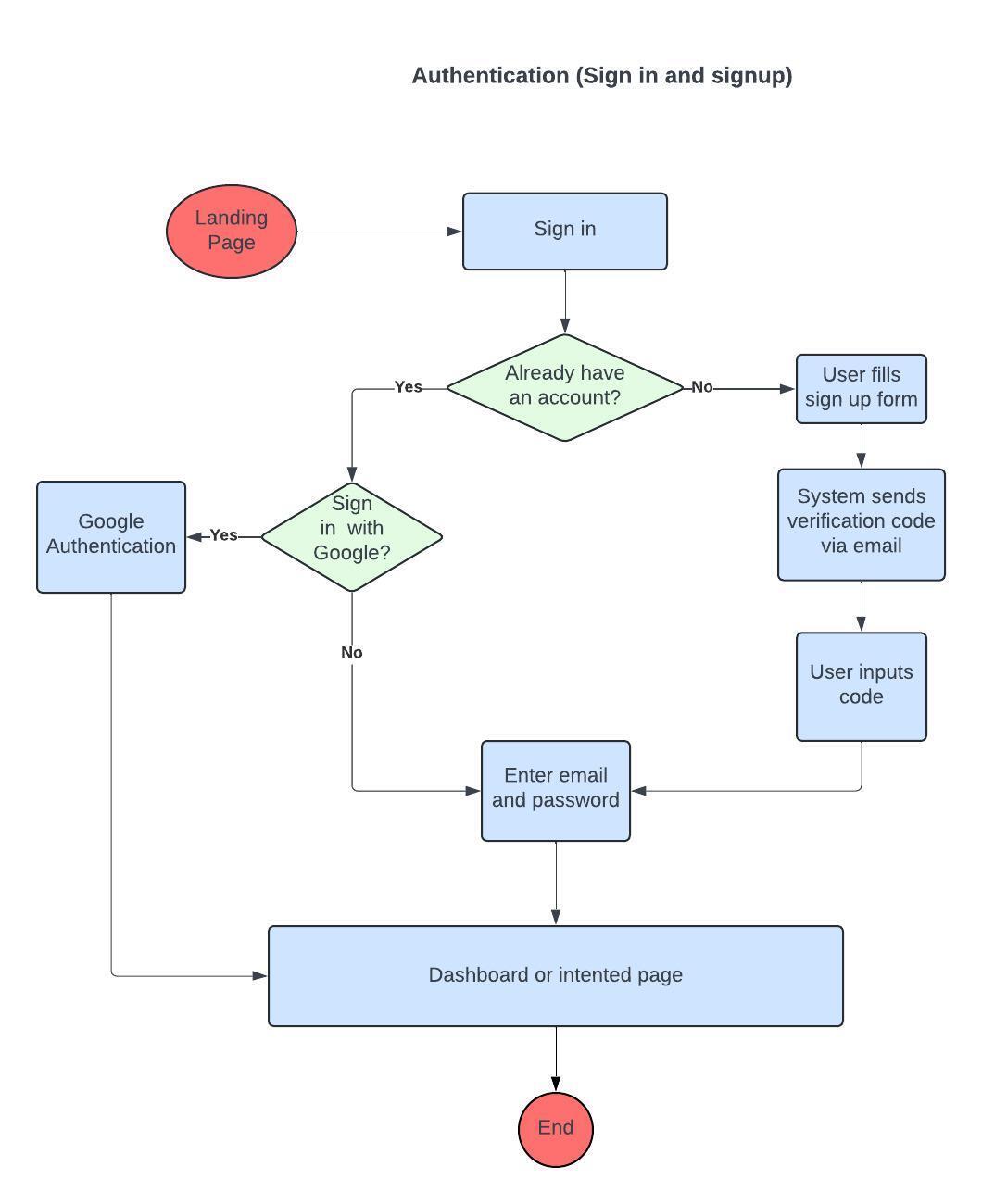
*Entry Point*

- The user lands on the landing page

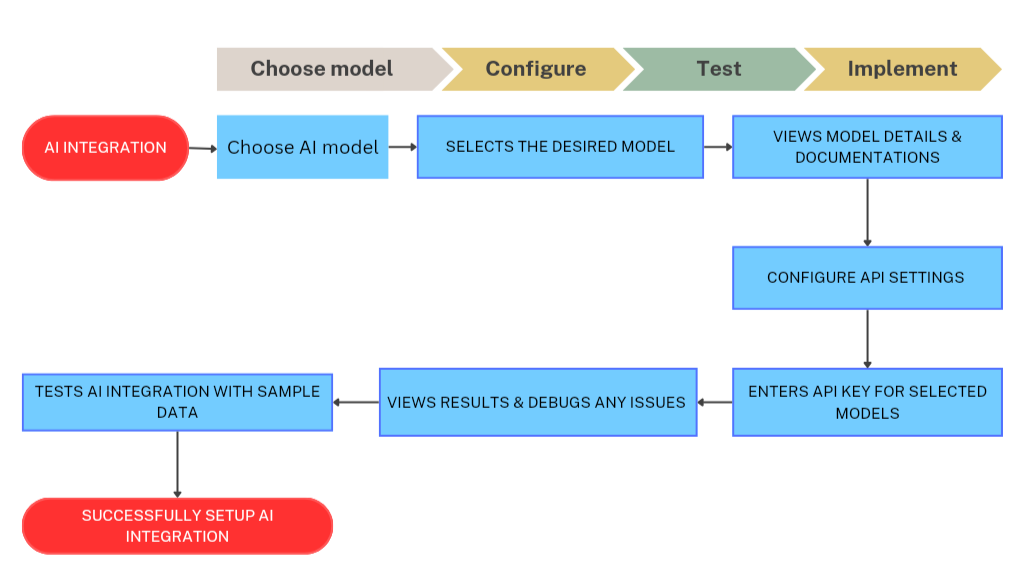
*Steps to Completion*

* 1. The user enters their username and password
* 2. The app validates the input credentials if it's an existing user
* 3. If credentials are valid, the app authenticates the user.

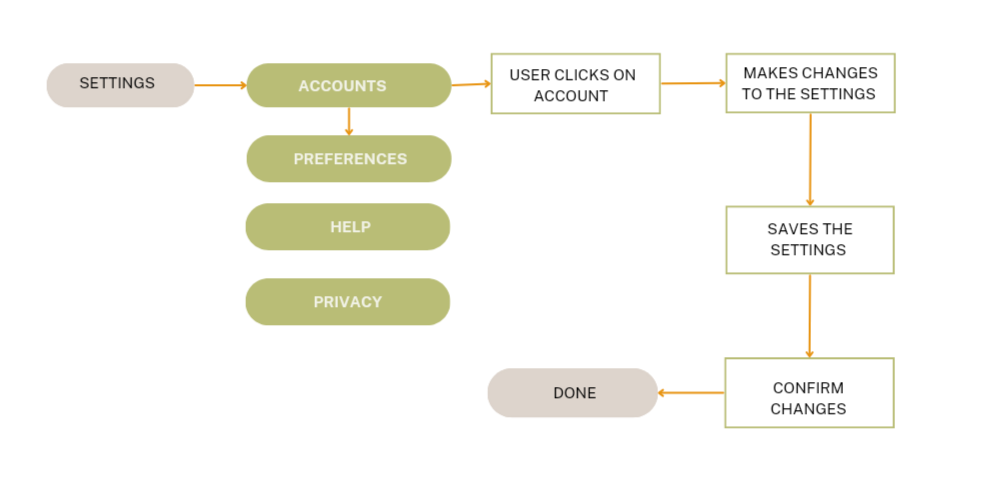
*Final Step/Interaction*

- The app grants access to the user's account, and the user can now land on the intended page

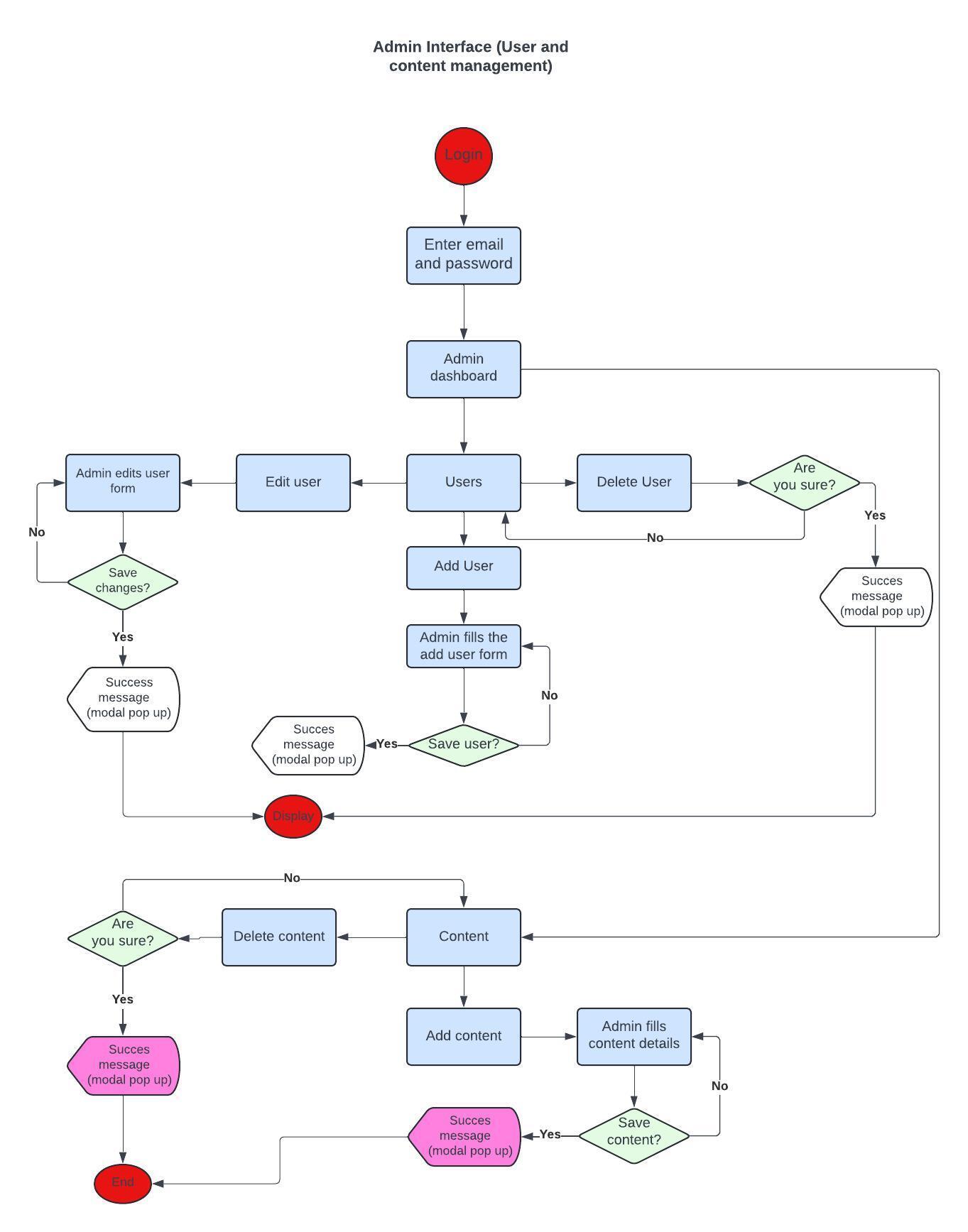
* This user flow illustrates the key steps involved in integrating AI capabilities into a product.



* This settings user flow outlines the steps a user takes to log in to their account within a product.



* This admin interface user flow outlines the steps for administrators to effectively manage the application, starting with login and authentication, to accessing and managing user accounts.



## **Success Metrics**

**Success Metrics for Boilerplate Development**

**Developer Productivity:**

1. **Development Time:** Measure the average time it takes developers to complete a project using the boilerplate compared to traditional methods.
2. **Number of Lines of Code (LOC) Reduction:** Track the reduction in code written due to code reuse and pre-built functionalities.
3. **Deployment Frequency:** Analyse if using the boilerplate accelerates deployment cycles.
4. **Memory Usage**: Tracks the amount of memory consumed by the boilerplate code. Lower memory usage is beneficial.
5. **Comment Clarity**: Ratings on the clarity and usefulness of comments in the code. Clear comments enhance code comprehension.

**Boilerplate Usability:**

1. **Adoption Rate:** Monitor how many developers within your team or community are actively using the boilerplate.
2. **User Feedback:** Conduct surveys or gather feedback from users to understand their satisfaction level with the boilerplate's functionalities and ease of use.
3. **Number of Issues Reported:** Track the number of bugs and usability issues reported for the boilerplate. A low number indicates a well-functioning system.

**Project Quality:**

1. **Defect Rate:** Monitor the number of bugs discovered in projects built with the boilerplate. A lower defect rate signifies a solid foundation for development.
2. **Code Maintainability:** Evaluate the maintainability of the boilerplate code. Metrics like code complexity can help here.
3. **Documentation Completeness:** Assess the comprehensiveness of the documentation and its effectiveness in guiding users.
4. **Code Duplication**: Tracks duplicated code segments across the project. Lower duplication improves maintainability.

**Additional Metrics:**

* **Cost Savings:** If applicable, estimate the cost savings achieved due to increased developer productivity.
* **Community Engagement:** For publicly shared boilerplates, track metrics like downloads, forks (in version control systems), or forum activity.

## **Roadmap**

**Boilerplate Roadmap**

**Phase 1: Planning Phase**

1. **Identify Use Cases:** Clearly identify common scenarios where boilerplate code can be beneficial.
2. **Target Audience:** Identify who will be using the boilerplate. Beginners, experienced developers, or a specific team?
3. **Functionality and Usability:** Gather requirements regarding the boilerplate's functionality and usability.

**Phase 2: Design Phase**:

1. **Architecture Design**: Define the structure and components of the boilerplate code.
2. **API Design**: Specify interfaces and APIs to ensure integration ease. Include code snippets, configuration files, directory structure, or documentation.

**Phase 3 : Development & Testing**

1. **Boilerplate Development**: Develop the core functionalities and optional features based on the design specifications.
2. **Code Documentation**: Write comprehensive comments and documentation within the codebase to explain functionality and usage.
3. **Unit Testing**: Implement unit tests to ensure individual components of the boilerplate function as expected.
4. **Integration Testing**: Test the boilerplate's integration with other development tools and libraries used.

**Phase 4: Deployment Phase**

1. **Comprehensive Documentation:** Create clear and concise documentation (user guides, tutorials, and other forms) that explains how to use the boilerplate effectively. Include examples and troubleshooting tips.
2. **Version Control:** Use a version control system (e.g., Git) to track changes, collaborate with others, and maintain different versions.
3. **Distribution Strategy:** Choose a distribution method based on our target audience. Options include public repositories, internal shared drives, or package managers. Release the boilerplate codebase and documentation through the designated channel.
4. **Initial Feedback Mechanism**: Establish channels for developers to provide feedback on the boilerplate's usability, functionality, and documentation.

**Phase 5: Maintenance and Evolution Phase**

1. **Regular Updates**: Address bug fixes, security vulnerabilities, and compatibility issues with new technologies.
2. **Feature Enhancements**: Implement prioritised features and functionalities based on developer feedback and evolving project needs.
3. **Maintain Documentation**: Update documentation to reflect changes and new features.
4. **Stay Updated with Technologies**: Monitor advancements in relevant technologies and consider boilerplate updates to stay compatible.
5. **Success Metric Monitoring**: Continuously track the defined success metrics to assess the boilerplate's impact and identify areas for further improvement.