System Design Document

Transferagentur

Hanna Lagerbauer, Henrik Marcussen, Jai Kamboj,
Eric Liebrecht, Maximillian Stabe, Leopold Lemmermann

December 12th 2024

1. Introduction

LaunchLab: From idea to startup success. This innovative iOS app guides users through every stage of their entrepreneurial journey, from idea generation to launching their own company. By digitizing the traditional consultation processes of Transferagentur, LaunchLab offers a dynamic and self-guided platform. At its core is Chatpreneur, an AI-powered co-founder designed to provide personalized insights and support throughout the process.

The app features an intuitive learning path, using gamification elements like a “rocket take-off” to symbolize user progress. Workshop-style learning modules are designed to be interactive, efficient, and engaging, enabling users to develop essential startup skills. Personalized outputs, such as elevator pitches, business plans, and pitch decks, ensure users have tangible results at every step.

LaunchLab emphasizes a user-friendly design, with no more than three actions required to complete primary tasks like onboarding or accessing features. Modules are structured to accommodate varied user experience levels, from beginners to advanced entrepreneurs, with functionality that allows users to pause and resume their learning without losing progress.

By integrating features such as artifact creation, expert consultation options, and a visually engaging startup journey, LaunchLab transforms the way entrepreneurs develop and launch their ideas. It bridges the gap between motivation and execution, empowering users to confidently turn their visions into successful ventures.

2. Design Goals

Simplicity v. Feature-Richness

Goal	Achieve a minimalistic and intuitive user interface while integrating features like a learning journey, personalized artifacts, consultation time, and AI Co-founder.
Trade-Off	Balancing simplicity with the inclusion of advanced features risks overwhelming users or complicating the interface.
Decision	Opt for a modular design that isolates complex features within specific contexts (e.g., a separate consultation module) while maintaining a streamlined primary interface for onboarding and learning paths.

Engagement v. Completion Time

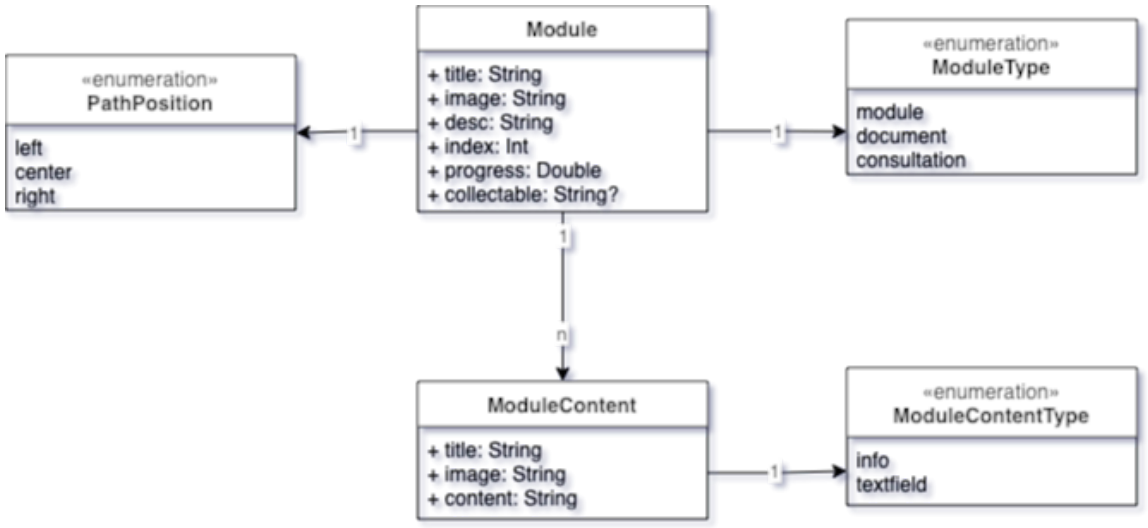
Goal	Design modules that are engaging yet concise, ensuring 75% of users can complete them within the allotted time (e.g., 30 minutes for beginner modules).
Trade-Off	Increasing engagement through interactive content could extend completion time, conflicting with usability requirements.
Decision	Limit module content to core elements, focusing on interactive but time-efficient elements like quizzes and animations while allowing users to pause and resume as needed.

Build v. Buy

Goal	Decide between custom development or integrating third-party tools for features like the AI Co-founder.
Trade-Off	Building custom solutions provides flexibility but increases development time and costs, whereas third-party tools may limit customization but reduce implementation effort.
Decision	Use third-party solutions for the AI Co-founder, but pair with custom logic to ensure the best user experience.

3. Subsystem Decomposition

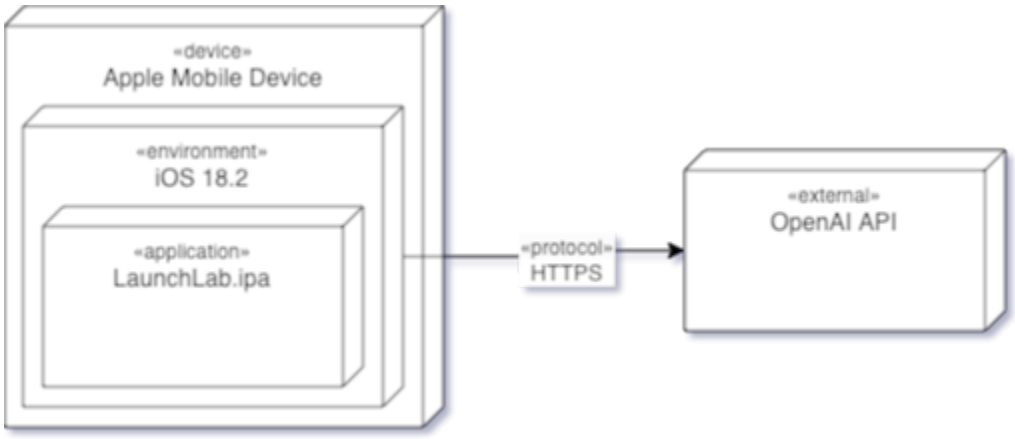
The system is designed with the goal of simplicity. It takes a module-centric perspective, which is clear and understandable in the context of the app's requirements. The learning path is implicitly handled in the module type. The structural UML class diagram outlines the relationships between key components of the system.



Description	The module system handles learning modules, including their structure, progression, and content.
Classes	**Module**: Represents an individual learning module, including its name, description, current page index, completion status, and position within the learning path (index and pathPosition). **ModuleType:** Enumerates different types of modules: (learning) module, consultation, document (export). **ModuleContent**: Represents the content of a module, supporting multiple content types such as information text, and interactive questions. **ModuleContentType**: Enumerates the different types of content supported by a module: info or textfield).
Services	➔ Load and display modules in the user's learning path. ➔ Track progress within modules (e.g., current page and completion status). ➔ Provide module-specific content in a structured format.

4. Hardware/Software Mapping

The system is local to the iOS device, and makes calls to the OpenAI API for the AI Co-Founder feature.



This mapping ensures a seamless and secure user experience by leveraging:

1. **Native Mobile App**: For structured, offline-first learning modules with integrated chat features.
2. **OpenAI API**: Centralized AI-driven intelligence layer, accessible via HTTPS for secure and responsive interactions.

This setup ensures high availability, device compatibility, and robust performance while prioritizing user accessibility and data security.

5. Persistent Data Management

The persistent data management strategy is tailored to the application's needs, leveraging **Core Data** for the mobile app to store user-related data locally while avoiding cloud-based solutions like CloudKit.

5.1 Rationale

Core Data was chosen for its seamless integration with iOS, offering robust local data persistence while maintaining high performance. It provides an object graph management framework, making it efficient for mapping the app's entity objects (e.g., Module, ModuleContent) to the local file system. This approach is ideal for offline-first functionality, ensuring that users can continue using the app without internet connectivity. CloudKit is intentionally excluded to keep data entirely local, simplifying implementation and avoiding concerns about syncing or managing user accounts.

5.2 Storage Scheme

Core Data maps key entities to its SQLite-backed persistent store, ensuring efficient local data management. The **Module** entity stores user progress and information for individual learning modules, including attributes such as currentPageIndex, isCompleted, and pathIndex. The **ModuleContent** entity holds content within a module, including its type (e.g., MultipleChoice, InformationVideo), title, and user interactions like userAnswer.

5.3 OpenAI API call

Messages are exchanged with the OpenAI API over HTTPS. An example request and payload for sending user queries to the API is as follows:

TODO: add payload

TODO: add response

---

6. Access Control and Security

The system is designed with simplicity and ease of use as primary goals, which extends to its access control and security mechanisms. To ensure a seamless user experience, no special authentication is required for accessing the app or its features. This decision is aligned with the stateless nature of the web-based chat and the offline-first design of the mobile app. The lack of authentication is a deliberate design decision to enhance usability and accessibility for all users. By removing the need for accounts or logins, the system ensures a frictionless experience while maintaining reasonable security measures through local encryption and secure communication protocols. This approach aligns with the goal of providing an intuitive and engaging system without compromising the user's privacy or data integrity.

6.1 Access Control

The system does not implement role-based access control (RBAC), providing all users with equal access to the app's features, including learning modules, progress tracking, and chat functionality. The browser-based chat system is open to all users, requiring no login or registration, ensuring unrestricted access to startup resources and personalized guidance. Similarly, the mobile app allows immediate usage upon installation, with no authentication required.

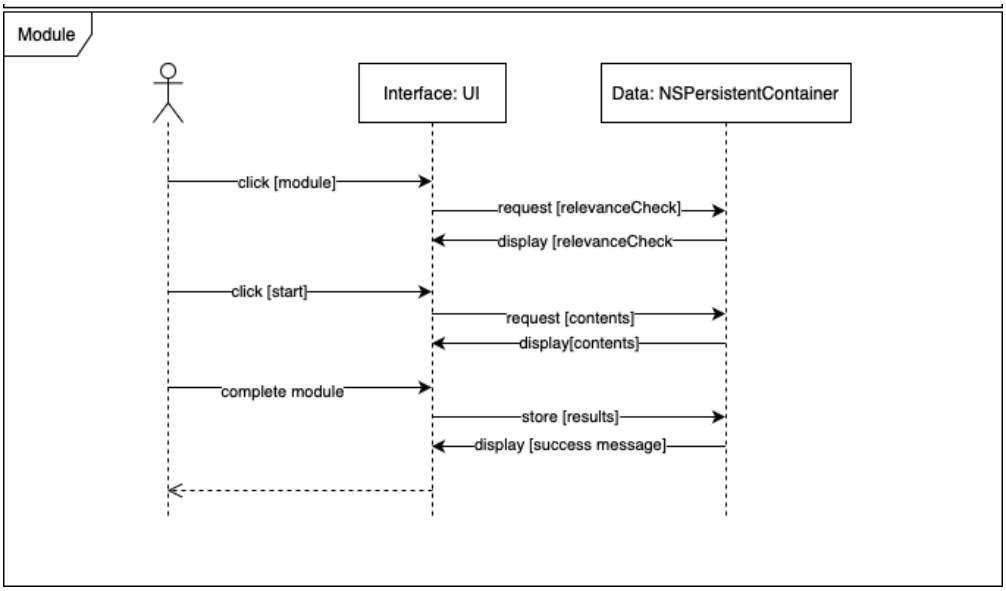
6.2 Authentication

The system deliberately avoids implementing authentication mechanisms such as login credentials or third-party integrations like OAuth. This decision prioritizes ease of use and local data storage, as user data is stored locally on the device without syncing across devices, removing the need for account management or authentication.

6.3 Data Security

Sensitive user data stored locally, such as progress, module interactions, and chat history, is protected through encryption mechanisms provided by iOS. The app utilizes iOS's built-in encryption for SQLite databases to ensure data security. Additionally, all communication between the app (both mobile and web chat) and the OpenAI API is encrypted via HTTPS. The system minimizes privacy risks by keeping all data local and avoiding cloud-based storage, while the web chat does not persist any data, further safeguarding user privacy.

7. Global Software Control



The module system manages the user's learning experience, starting with a relevance check to ensure the selected module aligns with their progress. If relevant, the module content is retrieved and displayed, enabling users to engage interactively. Upon completion, progress and results are stored in the **NSPersistentContainer**, ensuring offline functionality and continuity in the user's learning path. The system reinforces engagement with visual feedback, such as success messages, while dynamically adapting to user needs.

8. Boundary Conditions

The deployment plan ensures that the system components are robust and resilient. By leveraging iOS's built-in management tools and implementing error-handling mechanisms for the chat system, the plan supports a smooth user experience even in the event of failures. Regular monitoring and recovery strategies will minimize downtime and ensure that users have access to essential features at all times.

The mobile app is distributed via the Apple App Store and initializes automatically upon first launch, starting the onboarding process and setting up local data management through Core Data for offline functionality. Users can close the app without any formal shutdown process, and the system preserves their state using Core Data, allowing them to resume seamlessly on re-launch. In case of failures, iOS handles recovery, such as app re-launching or notifying the user, while Core Data ensures that data corruption is addressed by prompting users to restore or reset local data. Network-related failures, such as OpenAI API access issues, are handled with error messages and retry mechanisms.

Acronyms and Abbreviations

AI - Artificial Intelligence
BMC - Business Model Canvas
USP - Unique Selling Proposition
API - Application Programming Interface
LLM - Large Language Model
iOS - Apple's Mobile Operating System
SQLite - Structured Query Language Lite (Lightweight database engine)

Glossary

LaunchLab - A mobile application aimed at guiding users through the process of building and refining startup ideas.
Transferagentur - The organization responsible for the consultation processes and development of the LaunchLab app to digitize startup consultation.
Learning Modules - Interactive, workshop-style resources within the app that educate users on various aspects of startups, such as ideation, market analysis, and business planning.
Onboarding - A guided setup process within the app that personalizes user experience by assessing their knowledge and current status in the startup process.
Pitch Deck - A presentation document aimed at summarizing a business or product concept, typically used to attract investors or funding.
Core Data - Apple's object graph and persistence framework used for data storage and management in iOS applications.
OpenAI API - A programming interface provided by OpenAI, enabling access to AI-driven functionalities like natural language processing and content generation.
Value Proposition Canvas - A tool used to ensure that a product or service is positioned to meet customer needs effectively by mapping the value provided to the customer.
GenAI - General Artificial Intelligence; typically refers to advanced AI systems capable of general problem-solving, such as large language models.
Minimalistic Design - A user interface design principle that emphasizes simplicity and functionality by avoiding unnecessary complexity.