System Design Document

Transferagentur

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

December 12th 2024

1. Introduction

1.1 Overview

LaunchLab. Starting up from nothing. It's the app to help you find ideas, refine them into a product, and start a company up around that product.
Currently, all of the consultation process is handled by Transferagentur's consultants. We want to make that consultation digital. One app to start you up from nothing. And what's even better: It comes with its own startup consultant AI, Chatpreneur.
The app shall combine a minimalistic, intuitive design with a Duolingo-style learning path and engaging, workshop-like modules, ensuring users can navigate easily and complete tasks, such as onboarding or accessing the consultation module, within three actions. Learning modules will be designed for efficient completion, allowing users to pause and resume without losing progress, and will include outputs like personalized artifacts. The chat system will provide seamless, unrestricted access on external pages, leveraging personalized context for startup guidance, and ensure compatibility across modern web browsers and devices to support a wide range of users.
The **LaunchLab** application is designed to digitally transform startup consultation services by offering a seamless and interactive platform for users at different stages of their entrepreneurial journey. It combines a **Duolingo-style learning path**, **workshop-style modules**, and an **AI-driven chat assistant (Chatpreneur)** to provide users with personalized guidance on startup ideation, development, and scaling. The system emphasizes minimalistic design and usability, ensuring users can perform key actions like onboarding and learning progression within three clicks.
To support scalability and adaptability, the system leverages a modular architecture with two key subsystems: the **Chat Subsystem** and the **Module Subsystem**. The chat subsystem facilitates AI-powered interactions for personalized startup advice, while the module subsystem dynamically adapts content based on user progress, ensuring an engaging learning experience. Core features include offline-first functionality through **CoreData**, context-aware interactions, and secure communication with the OpenAI API. By balancing innovative features with accessibility, LaunchLab aims to empower users to transform their ideas into successful ventures.

1.2 References

- Requirements Analysis Document
- …

2. Design Goals

Simplicity v. Feature-Richness

Goal	Achieve a minimalistic and intuitive user interface while integrating features like a Duolingo-style learning path, personalized outputs, and a GenAI-powered chat system.
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.

Scalability v. Immediate Availability

Goal	Ensure the chat system is accessible on external pages and supports large-scale usage without requiring user accounts.
Trade-Off	Prioritizing unrestricted access could increase server loads and security challenges due to unverified usage.
Decision	Implement lightweight APIs and serverless architecture for scalable, resource-efficient chat functionality, with safeguards like rate limiting and anonymized data handling for security.

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.

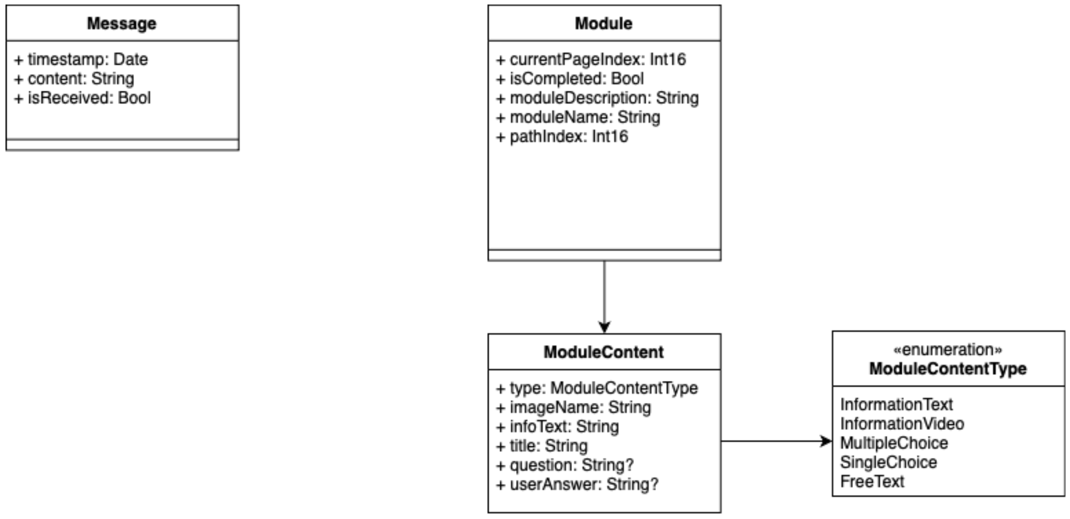
Build v. Buy

Goal	Decide between custom development or integrating third-party tools for features like the chat system or learning path.

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 chat system, integrating it with custom-built modules for tailored functionality, maintaining a balance between customization and efficiency.

3. Subsystem Decomposition

The system is designed with a modular architecture, where each subsystem is responsible for specific functionalities to ensure a scalable and maintainable implementation. The structural UML class diagram outlines the relationships between key components of the system.



Modules subsystem

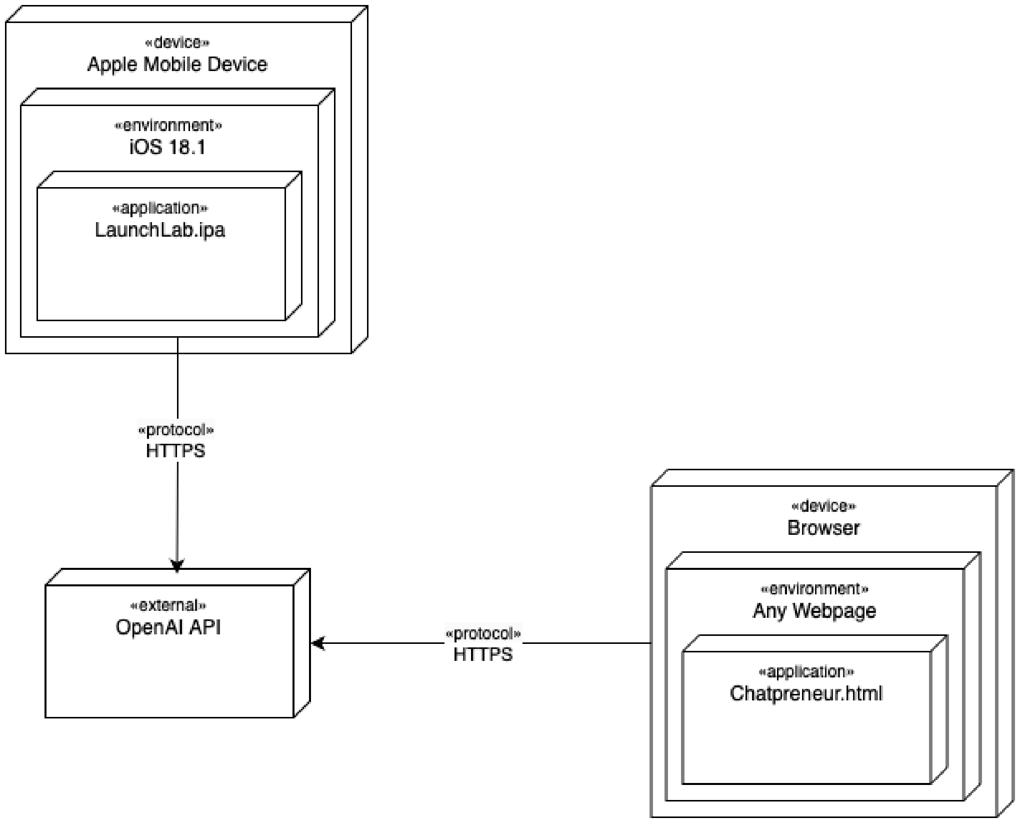
Description	This subsystem handles the core learning modules, including their structure, progression, and content.
Classes	**Module**: Represents an individual learning module, including its name, description, current page index, completion status, and position within the learning path (pathIndex). **ModuleContent**: Represents the content of a module, supporting multiple content types such as information text, videos, and interactive questions. **ModuleContentType**: Enumerates the different types of content supported by a module (e.g., InformationText, MultipleChoice, FreeText).
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.

Chat subsystem

Description	This subsystem supports user communication with a GenAI-powered chat assistant for personalized startup guidance.
Classes	**Message**: Manages individual chat messages, including the timestamp, content, and whether the message was received or sent by the user.
Services	➔ Send and receive messages between the user and the assistant. ➔ Store and retrieve chat history for context-aware interactions.

4. Hardware/Software Mapping

The system is distributed across multiple hardware and software components, as illustrated in the UML deployment diagram.



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. **Web-Based Chat**: For lightweight, browser-accessible guidance.
3. **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. The web-based chat system does not persist any data, ensuring a stateless and lightweight operation.

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, Message) 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. In contrast, the browser-based chat system is stateless, persisting no data and focusing solely on real-time interactions without additional storage overhead.

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. The **Message** entity stores chat messages locally, capturing details such as timestamp, content, and isReceived to maintain conversation history. This data is stored within a lightweight SQLite database managed by Core Data, ensuring fast read/write operations and maintaining data consistency.

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:

```
{
        "user_id": "device_identifier",
        "message": "How do I create a startup pitch deck?",
        "context": {
                "module_name": "Pitch Deck Basics",
                "current_progress": 0.75
        }
}
```

```
{
        "response": "To create a startup pitch deck, focus on key slides such as problem,
solution, market, and financials...",
        "suggested_action": "Review the 'Pitch Deck Basics' module for further details."
}
```

---

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

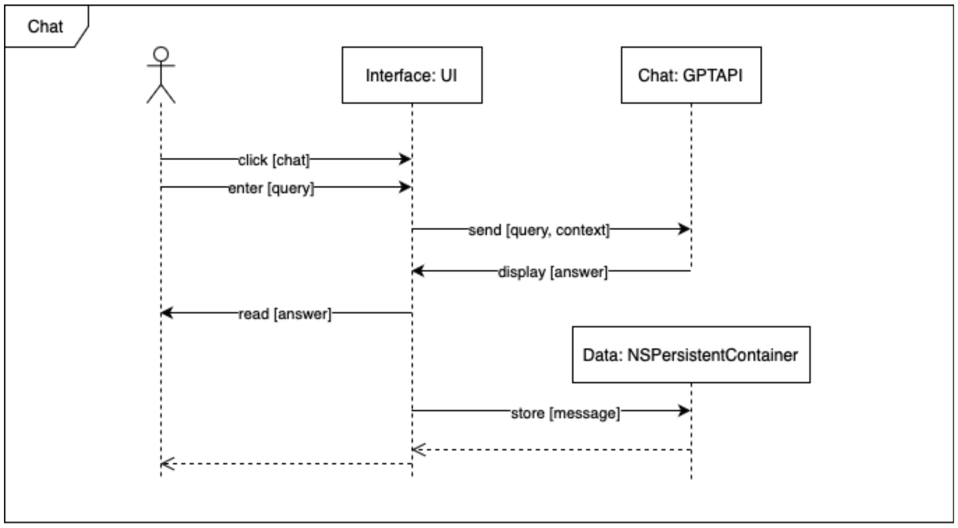
The system deliberately avoids implementing authentication mechanisms such as login credentials or third-party integrations like OAuth. This decision prioritizes ease of use and local data storage, as user data is stored locally on the device without syncing across devices, removing the need for account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

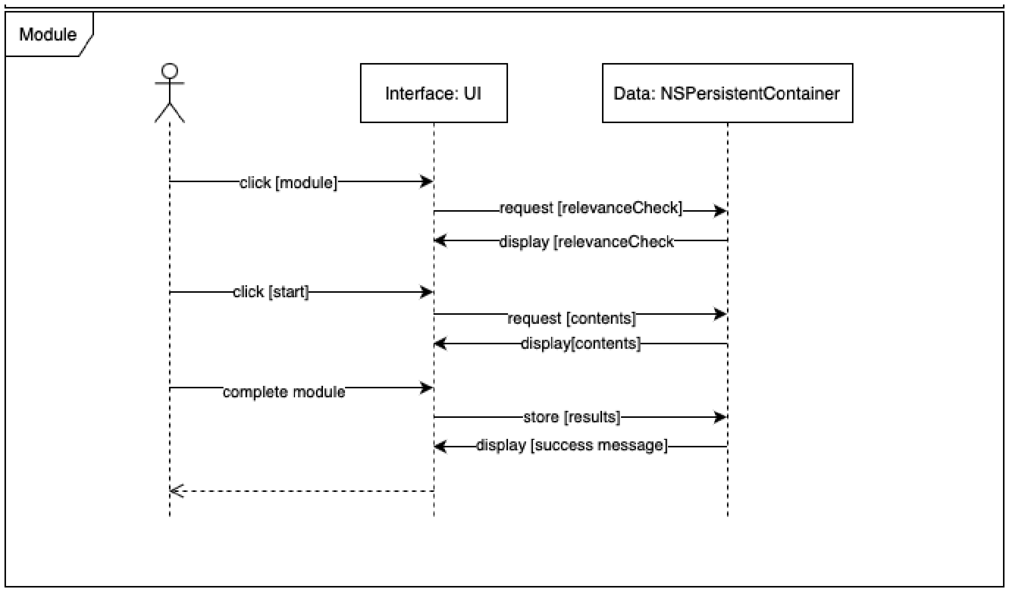
7. Global Software Control

7.1 Chat subsystem



The chat subsystem facilitates user interaction with the **Chat: GPTAPI** to provide personalized startup guidance. Users initiate queries through the UI, which are sent along with context (e.g., user profile, progress) to the GPTAPI for processing. The AI generates responses that are displayed in real-time via the user interface. Additionally, the system stores the conversation history in the **NSPersistentContainer**, enabling context-aware, seamless interactions across sessions. This local storage approach ensures privacy and supports a stateless server model.

7.2 Module subsystem



The module subsystem 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—whether the mobile app or the web-based chat—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.

8.1 Mobile App (iOS)

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.

8.2 Chat System (Web)

The web-based chat system is hosted on the marketing and Transferagentur pages, automatically loading in the browser without any installation. It communicates with the OpenAI API over HTTPS, with error handling for connection failures. Since the chat is stateless, no shutdown procedure is needed, and the session ends when users close the browser tab. In the event of failures, the chat system displays error messages or retry options. If there’s an issue with the OpenAI API, users are informed and offered a retry option. Prolonged issues lead to providing an alternative contact method, such as email or phone, for user support.

Acronyms and Abbreviations

AI - Artificial Intelligence
BMC - Business Model Canvas
USP - Unique Selling Proposition
API - Application Programming Interface
HTTPS - Hypertext Transfer Protocol Secure
RBAC - Role-Based Access Control
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. It includes interactive modules and an AI assistant called Chatpreneur.

Chatpreneur - An AI-powered digital assistant integrated into the LaunchLab app and web platforms, providing personalized guidance and answers to startup-related queries.

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.

Business Model Canvas (BMC) - A strategic tool used to develop and visualize the essential elements of a business model, including value propositions, customer segments, and revenue streams.

Lean Canvas - A simplified version of the Business Model Canvas, specifically tailored for startups, focusing on solving customer problems and defining unique solutions.

Persona Creation - A method of building user archetypes to better understand target audiences, typically employed in marketing and product development.

Duolingo-style Learning Path - A gamified structure within the app that organizes learning modules in a visually engaging, sequential format to track progress and encourage completion.

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.