Department of Computer Science & Engineering
The University of Texas at Arlington

Architectural Design Specification
CSE 4316: Senior Design I
Spring 2024

Swift Start
Sprint O' Clock

Cesar Frayre
Bryant Eyum
Kosuke Satake
Gin Sang
Shaheen Nijamudheen

Revision History

Revision	Date	Author(s)	Description
0.1	04.23.2024	CF, BE, KS, GS, SN	document creation
0.2	04.25.2024	CF, BE, KS, GS, SN	complete draft
0.3	05.03.2024	CF, BE, KS, GS, SN	revise draft
0.4	05.03.2024	CF, BE, KS, GS, SN	official release

Contents

List of Figures

List of Tables

1 INTRODUCTION

Swift Start is a mobile application designed for runners and fitness enthusiasts, offering comprehensive platform to track, analyze, and improve running performance. The app caters to users on both iOS and Android platforms and provides a range of features designed to support runners of all levels, from beginners to experienced athletes. With a focus on usability, customization, and social interaction, Swift Start will have a front end User Login interface and UI interface. Back end will be more related into keeping track of the process, APIs, security, etc, more information will be provided ongoing this document. This application will be a user friendly that will be structured to ensure a smooth development process meeting users expectations.

2 System Overview

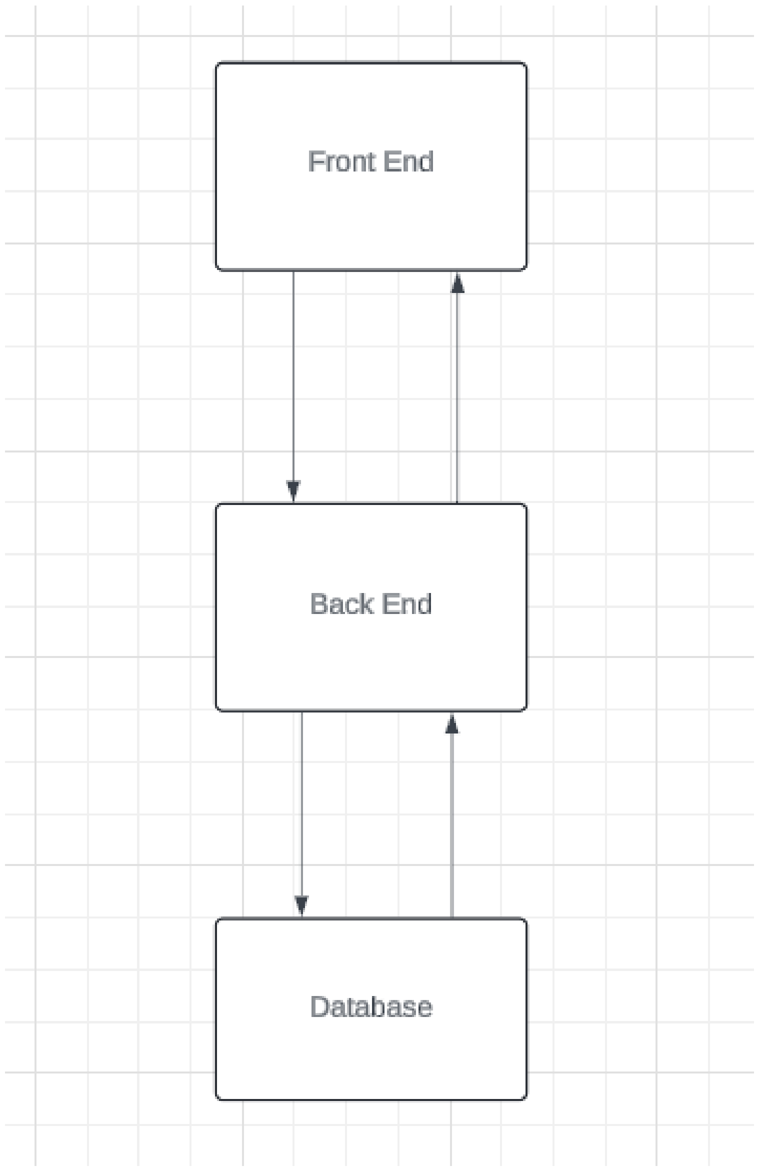

Figure 1: A high-level architectural layer diagram

2.1 Front-End Layer

The front-end layer is the visual interface through which users interact with the system. It is responsible for displaying information to the users and capturing their inputs, featuring a well-designed UI that enhances user experience. This layer communicates with the back-end layer through RESTful API calls served by FastAPI. It translates user actions into HTTP requests and updates the UI based on the responses received. Key services of the front-end include the Sprint O' Lock UI interface for secure user interactions and a User Login module that facilitates authentication and session management.

2.2 Back-End Layer

The back-end layer orchestrates the system's core logic, including processing data, authenticating users, and handling business logic. It acts as a bridge between the front-end and database layers. The back-end exposes RESTful API endpoints via FastAPI, through which it receives requests from and sends responses to the front-end. It communicates with the database layer using an Object-Relational Mapping (ORM) system, which abstracts and manages the database operations. It is responsible for user authentication, data retrieval through the database layer's ORM, executing business logic, managing security protocols, and processing events.

2.3 Database Layer

The database layer provides structured data storage and management solutions. It ensures that data is stored securely and is retrievable in an efficient manner, supporting complex queries and transactions. The database layer interfaces with the Back-End through an ORM, which allows for seamless data management and object-oriented access to the database. The ORM abstracts the complexity of direct database interactions, providing a high-level API for the Back-End to communicate with the database. The database layer offers services such as user profile management, enforcing security policy, cloud storage capabilities for scalability and data redundancy, and data storage that maintains data integrity and availability.

3 Subsystem Definitions & Data Flow

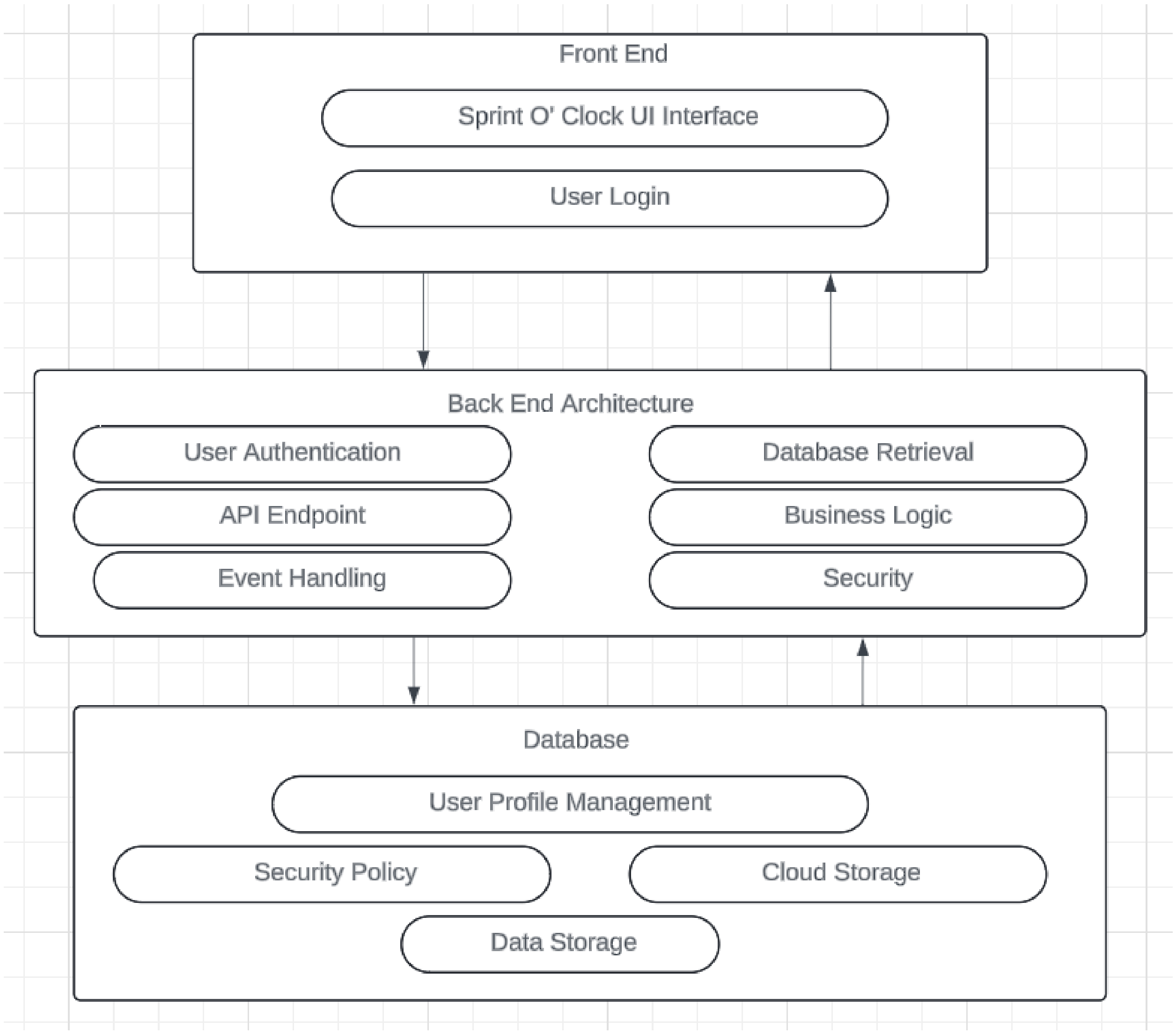

Figure 2: A high-level data flow diagram

4 Front-End Subsystems

In this section, we'll outline the subsystems of the front-end layer, which is responsible for managing the user interface and user interactions within the application. The front-end layer interacts directly with users and provides them with a seamless and intuitive experience.

4.1 User Login Subsystem

The User Login subsystem is responsible for managing the authentication process within the front-end layer of the application. It allows users to securely sign in to their accounts, providing access to personalized features and data. This subsystem communicates with authentication services or back-end APIs to verify user credentials and ensure data security. It also interacts with UI components to capture user input for email and password, handle authentication errors, and provide feedback to users upon successful login. Overall, the User Login subsystem ensures a smooth and secure authentication experience for users accessing the application.

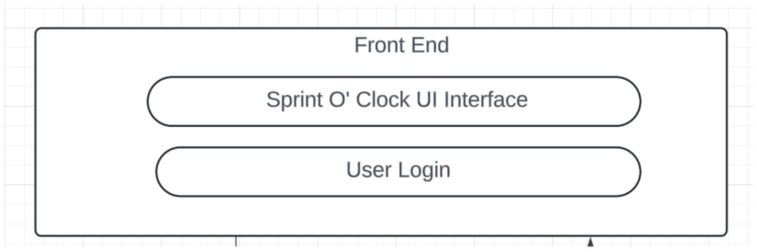

Figure 3: Front End Architecture diagram

4.1.1 Assumptions

The User Login subsystem assumes integration with authentication services or back-end APIs for user login and compatibility with various authentication methods, such as email/password and social login.

4.1.2 Responsibilities

This subsystem manages the user authentication process securely by designing and implementing UI components for user login, including input fields for email and password. It validates user credentials, handles authentication errors, integrates with authentication services or back-end APIs for user authentication, and provides clear feedback to users on the success or failure of the login attempt, ensuring transparency and user confidence in the authentication process.

4.1.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 2: Subsystem interfaces

ID	Interface Name	Inputs	Outputs
#1	Authentication Service	User credentials	Authentication status
#2	UI Components	User input (email, password)	UI feedback (success/failure)

4.2 Sprint O' Clock UI Interface Subsystem

This subsystem focuses on crafting the user interface elements for the "Sprint O' Clock" feature. It ensures seamless functionality and compatibility across iOS and Android platforms using React Native. By adhering to design guidelines, it maintains a visually appealing and user-friendly interface, optimizing user interaction during sprint sessions.

4.2.1 Assumptions

The Sprint O' Clock UI Interface assumes utilization of React Native for cross-platform development, ensuring compatibility with both iOS and Android platforms. It also assumes compatibility with various screen sizes and resolutions to allow the UI to adapt seamlessly to different devices. Additionally, adherence to design guidelines is assumed to maintain a visually appealing and user-friendly interface.

4.2.2 Responsibilities

This subsystem is dedicated to designing and implementing the user interface for the "Sprint O' Clock" feature. It focuses on fulfilling several key responsibilities, including designing and implementing UI components for the start button and "Start Reaction Timer" to facilitate the initiation of the sprint session and reaction time testing. It also synchronizes animation with audio cues for the "Ready... Set...Go!" sequence to provide users with a clear and engaging signal for the start of the sprint. Furthermore, it provides optional settings for phone vibration during the start, enhancing user experience by allowing customization according to preferences. Lastly, it handles user interactions for starting and stopping the sprint, ensuring smooth functionality and user control throughout the sprint session.

4.2.3 Subsystem Interfaces

Each of the inputs and outputs for the subsystem are defined here. Create a table with an entry for each labelled interface that connects to this subsystem. For each entry, describe any incoming and outgoing data elements will pass through this interface.

Table 3: Subsystem Interfaces

Interface	Description
Start Button	UI element triggering the start of the sprint session
Start Reaction Timer	UI component simulating the unpredictable start signal for testing reaction time
Audio Module	Syncs UI animations with audio cues for the start sequence
Settings	Optional settings for enabling/disabling phone vibration during the start

5 Back-End Subsystems

5.1 User Authentication Subsystem

This subsystem will manage user authentication and authorization, interacting with the Data Storage and Retrieval Subsystem to access user data. Additionally, it will facilitate front-end interface events, provide support for data visualization methods, and enable business logic capabilities.

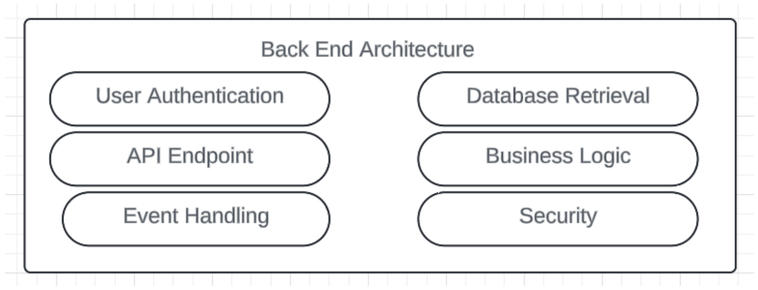

Figure 4: Back End Architecture diagram

5.1.1 Assumptions

It presupposes the presence of user data within the Data Storage and Retrieval Subsystem.

5.1.2 Responsibilities

The responsibility of the User Authentication Subsystem is to authenticate users, verifying their identities before granting access to the system or its resources.

5.1.3 Subsystem Interfaces

Table 4: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Data Storage and Retrieval Subsystem	User data	None

5.2 Database Retrieval

This subsystem will manage the storage and retrieval of data for the system. It will communicate with the User Authentication Subsystem to access user data and with the API Endpoint Subsystem to deliver data efficiently through APIs.

5.2.1 Assumptions

- Relies on the presence of a User Authentication Subsystem for accessing user data
- Relies on the existence of an API Endpoint Subsystem to deliver data efficiently via APIs

5.2.2 Responsibilities

- Manage system data storage and retrieval processes
- Fetch user data for the User Authentication Subsystem
- Deliver data efficiently via Fast APIs to the API Endpoint Subsystem

5.2.3 Subsystem Interfaces

Table 5: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	User Authentication Subsystem	None	User Data
#2	API Endpoint Subsystem	None	Data via APIs

5.3 API Endpoint Subsystem

This subsystem will manage API endpoints to facilitate communication with the front-end layer. It will collaborate with the Data Storage and Retrieval Subsystem to offer data via APIs and with the Business Logic Subsystem to handle request processing.

5.3.1 Assumptions

- Relies on the presence of a User Authentication Subsystem for accessing user data
- Relies on the existence of an API Endpoint Subsystem to deliver data efficiently via APIs

5.3.2 Responsibilities

- Manage system data storage and retrieval processes
- Fetch user data for the User Authentication Subsystem
- Deliver data efficiently via Fast APIs to the API Endpoint Subsystem

5.3.3 Subsystem Interfaces

Table 6: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	User Authentication Subsystem	None	User Data
#2	API Endpoint Subsystem	None	Data via APIs

5.4 Business Logic Subsystem

The Business Logic Subsystem manages processing the core functionality and rules of the application. It handles data processing, workflow management, decision making, system integration, and enforcement of business rules and validations. This subsystem ensures that all operations align with the business objectives and user requirements, effectively managing how the application behaves and responds to various inputs and conditions.

5.4.1 Assumptions

- Correct Inputs: Assumes that incoming data is already validated
- Reliable External Systems: Expects external dependencies like APIs to be stable and available
- Data Consistency: Assumes data in databases is consistent and adheres to expected formats
- Stable Business Rules: Operates under the assumption that business rules are clearly defined and seldom change
- Handled Security: Assumes that fundamental security measures are in place, though it also implements specific security checks

5.4.2 Responsibilities

- Implementing Business Rules: Applying and enforcing specific business rules and logic
- Data Manipulation: Handling database operations like CRUD (Create, Read, Update, Delete)
- Workflow Coordination: Managing the sequence and execution of business processes
- Decision Making: Automating decisions based on the logic defined
- System Integration: Facilitating communication and data exchange with other systems and APIs
- Security and Compliance: Ensuring data security and adherence to regulations
- Error Handling: Managing errors and exceptions within business processes effectively

5.4.3 Subsystem Interfaces

Table 7: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Processing Subsystem	Validated data	Processed data ready for use or further action
#2	Authorization Subsystem	User credentials and action requests	Authorization decisions, granting or denying access based on user permissions

5.5 Event Handling Subsystem

Event handling involves managing and responding to actions triggered by the front-end or other sources.

5.5.1 Assumptions

- Valid Inputs: Inputs from the front-end are valid or pre-validated
- Reliable Network: Network connections are stable and reliable
- Resource Availability: Server and database resources are sufficiently available to handle requests
- Security Measures: Adequate security protocols are in place to protect data and prevent attacks
- Error Handling: Effective error management mechanisms are implemented
- Asynchronous Support: The system can handle time-consuming operations asynchronously to maintain responsiveness

5.5.2 Responsibilities

The responsibilities of event handling include processing requests, executing business logic, interacting with databases and external services, managing security, providing responses, handling errors, and managing asynchronous tasks to ensure efficient and secure operation.

5.5.3 Subsystem Interfaces

Table 8: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	User Interface	User data and actions	Data and responses for display
#2	Database Management System	Queries and data manipulation command	Data and operation status
#3	External APIs	Requests for external data or services	Data and responses from external services
#4	Security Services	Authentication and authorization requests	Security statuses and tokens
#5	Logging and Monitoring Systems	Application and user activity data	Logs, alerts, and performance reports

6 Database Subsystems

The database layer subsystems will manage all data operations such as User Profile Management, Security Policy, Cloud Storage, and Data Storage for JavaScript.

6.1 User Profile Security Management System

This subsystem manages all user profile data and ensures the data is inaccessible to anyone but the user.

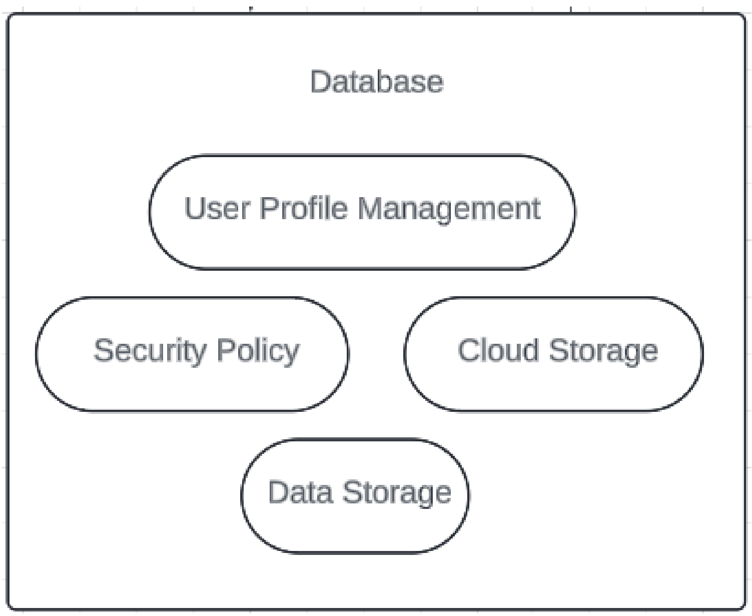

Figure 5: Database Subsystem Description diagram

6.1.1 Assumptions

- This subsystem assumes that user profile information is in the database
- This subsystem assumes that the User Profile data is being contained and that all data pertaining to the user is secured and inaccessible to the developers

6.1.2 Responsibilities

- Manages operations related to users profile information
- Makes sure inaccessible users don't access the user's data

6.1.3 Subsystem Interfaces

Table 9: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	User Profile Management System	Back up data	Restore data
#2	Security Policy System	Data Validation	Auditing and Logging

6.2 Data Cloud Storage System

This subsystem oversees data storage, including cloud storage, within the JavaScript database.

6.2.1 Assumptions

- This subsystem assumes that the running data is being logged into the Data Storage
- This subsystem assumes that cloud storage is being managed

6.2.2 Responsibilities

- Oversee all data within the database
- Maintain the data schema to align with system requirements
- Update the data schema as needed

6.2.3 Subsystem Interfaces

Table 10: Subsystem interfaces

ID	Description	Inputs	Outputs
#1	Data Persistence Subsystem	Handles storage	Retrieval of user and app data
#2	Cloud Synchronization Subsystem	Syncs data to cloud	Stores data to cloud