This is the Table of contents form our skeleton document.  
This document gives a sample to refine 2 parts of our Chapter 4 (4.2 to 4.3).  
Please consider this document as a basic intro to write a draft for these two parts.   
A draft with following information will help the team:  
1. Necessary info about methodology of programming and design  
2. Necessary info about tools be chosen

* We use JIRA to manage requirements and do agile development.
* We use Client-Server architecture, Java for backend and React for frontend. Database is H2 and persisted in files.
* Server side is on Springboot framework and React is on Antd. Maven is used to manage project dependencies and Git is used for code repository.
* We deploy the application on AWS EC2 and apply CICD deployment method that allows us to automatize the work flow.
* In order to satisfy different traffics in different times we use AWS elastic load balancer in front of backend cluster. And the cluster size can be scaled up or down automatically according to the traffic.
* In terms of development IDEs we use Eclipse for java programming and Visual Studio for React

3. Any screenshots or records that might be useful (can be independent package)  
A paper with text on it

Description automatically generated  
  
This Part   
\*\*Example: Football Data Collection App\*\*

Let's explore how the methodology and program architecture can be combined for developing a Football Data Collection App.

\*\*1. Choosing the Methodology:\*\*

For the development of the Football Data Collection App, we consider the scope and requirements. Since data collection apps often require frequent updates and may involve user feedback for improvements, we opt for an Agile development methodology. Agile will allow us to be responsive to changing data requirements and continuously improve the app based on user needs.

\*\*2. Methodology Guides Architecture Design:\*\*

With Agile methodology in mind, we start by understanding the data collection requirements for the app. It could involve recording player statistics, match results, goals, assists, and other relevant football data. Based on this analysis, we design the program architecture for the app. A suitable architecture could be a Client-Server architecture where the app's clients (users) can submit data to a central server. The server processes and stores the collected data securely.

\*\*3. Methodology Impacts Development Process:\*\*

The Agile development process involves iterative sprints, where we develop specific features in each sprint. In this context, we may prioritize features like player statistics collection and match results recording in early sprints, allowing us to release a functional version of the app sooner. The iterative nature of Agile also permits us to gather feedback from users during each sprint and make necessary adjustments to the data collection process based on their inputs.

\*\*4. Iterative Improvements and Continuous Optimization:\*\*

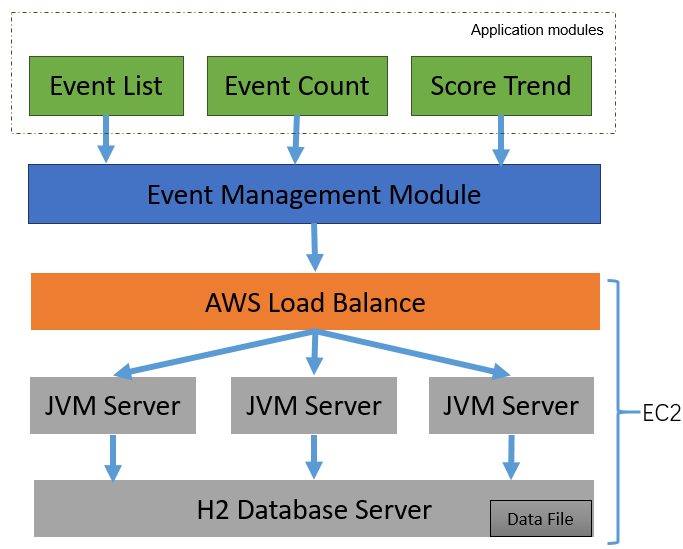
Throughout the development of the Football Data Collection App, we continuously gather user feedback. Based on this feedback and any emerging data trends, we may identify the need to enhance the app's architecture. For instance, if the user base grows significantly, we might need to consider scaling the server infrastructure or optimizing data processing to handle increased data volume effectively. Agile allows us to make these improvements iteratively, ensuring that the app remains responsive and up-to-date.

\*\*Summary:\*\*

By adopting the Agile methodology, we can quickly develop and release a functional version of the Football Data Collection App. With a Client-Server architecture, we ensure that data collected by users is securely managed and stored. The iterative nature of Agile enables us to prioritize essential data collection features, gather user feedback, and make continuous improvements based on the user's needs and data trends. Throughout the development process, we can optimize and refine the app's architecture to handle any future scalability requirements effectively.

5.3 Application Delivery

**5.3.1 High-level architecture**



**Event Management Module：**Intake event data and do query for application modules

**Event List:** Show all event data of the two teams

**Event Count:** Count every event amount and show them with average data

**Score Trend:** Display the score trend in lines

**AWS Load Balance:** Ensure application availability and scale up/down EC2 machines according to the traffic requirement.

**JVM Server:** Java runtime environment, start up by springboot.

**H2 Database Server:** In-Memory database, persist data in files.

**5.3.2 APIs**

1. **Save Event “/api/createEvent”**

**Request：**

{

"team": "Leinster",

"event": "Wide",

"minutes": 5,

"seconds": 18.38154800000001

}

**Response:**

{

"id":270,

"team":"Leinster",

"event":"Wide",

"minutes":5,

"seconds":18,

"score":null,

"updateTime":1690230338629

}

1. **Load data “/api/loadData”**

**Request：**

{

"homeTeam": "Leinster"

"awayTeam": "Munste"

}

**Response:**

