# Activity: Determining Architecture & Requirements

## Introduction

In this activity, you will learn the process of critically evaluating the requirements and architecture of a mobile app.

## Instructions

1. The instructor will divide the class into groups of 2-3 students.
2. Each group will be assigned a set of requirements and architecture for a hypothetical mobile app.
3. The groups are to critique the requirements and architecture, focusing on the following aspects:

* Completeness: Are all the necessary requirements included? Are there any missing requirements?
* Clarity: Are the requirements clear and unambiguous?
* Feasibility: Are the requirements achievable within the given constraints, such as time, budget, and resources?
* Scalability: Can the architecture handle increasing amounts of users and data?
* Maintainability: Is the architecture easy to maintain and modify over time?
* Security: Are the necessary security measures in place to protect sensitive user data?

1. Summarizing your findings and recommendations for improving the requirements and architecture.
2. Be prepared to present your findings and recommendations to the class and receive feedback from peers and instructors.

**Scenario: Fitness Tracking App**

**Requirements**:

1. User registration and login: Users should be able to create an account and log in to access their fitness data.
2. Physical activity tracking: Users should be able to log and track their daily physical activity, including exercises and steps taken.
3. Goal setting: Users should be able to set and track their fitness goals, such as steps taken or calories burned.
4. Progress history: Users should be able to view a history of their physical activity and progress towards their goals.
5. Social connectivity: Users should be able to connect with friends and compare their fitness progress.
6. Personalized tips and recommendations: Users should receive personalized tips and recommendations for reaching their fitness goals.
7. User interface: The app should have a user-friendly interface that is easy to navigate.

**Architecture**:

1. Front-end: The front-end of the app should be built using React Native, a popular framework for building cross-platform mobile apps. The interface should be responsive and user-friendly.
2. Back-end: The back-end of the app should be built using Node.js and a NoSQL database such as MongoDB. The database will store user data, including their physical activity, goals, location, and progress history.
3. Data analytics: The app should use data analytics to provide insights into the user's fitness progress and identify areas for improvement.
4. Social connectivity: The app should use the Firebase social networking API to connect with friends and compare progress.
5. Machine learning: The app should use machine learning algorithms to provide personalized tips and recommendations for reaching fitness goals.

**Scenario: Shopping List App**

**Requirements**:

1. User registration and login: Users should be able to create an account and log in to access their shopping list.
2. Item management: Users should be able to add, edit, and delete items on their shopping list.
3. Item categorization: Users should be able to categorize items into different categories, such as groceries, household items, or personal items.
4. List sharing: Users should be able to share their shopping list with others, such as family or friends.
5. Item recommendations: Users should receive recommendations for items based on their shopping history and past purchases.
6. User interface: The app should have a user-friendly interface that is easy to navigate.

**Architecture**:

1. Front-end: The front-end of the app should be built using Cordova, a framework for building cross-platform mobile apps. The interface should be responsive and user-friendly.
2. Back-end: The back-end of the app should be built using Node.js and a relational database such as MariaDB. The database will store user data, including their shopping list, past purchases, and payment information.
3. Data analytics: The app should use data analytics to provide insights into the user's shopping habits and make recommendations.
4. Sharing functionality: The app should use a real-time database, such as Firebase, to allow users to share their shopping list in real-time with others.
5. Machine learning: The app should use machine learning algorithms to provide personalized recommendations for items based on the user's shopping history.

**Scenario: Music Player App**

**Requirements:**

1. User registration and login: Users should be able to create an account and log in to access their music library.
2. Music library management: Users should be able to add, edit, and delete music from their library.
3. Playlist creation: Users should be able to create and manage playlists of their favorite songs.
4. Music streaming: Users should be able to stream music directly from the app.
5. Music recommendations: Users should receive recommendations for music based on their listening habits and past playlists.
6. User interface: The app should have a user-friendly interface that is easy to navigate.

**Architecture:**

1. Front-end: The front-end of the app should be built using Xamarin, a framework for building cross-platform mobile apps. The interface should be responsive and user-friendly.
2. Back-end: The back-end of the app should be built using Node.js and a NoSQL database such as MongoDB. The database will store user data, including their music library and playlists.
3. Music streaming API: The app should use a music streaming API, such as Spotify, to provide access to a vast library of music.
4. Data analytics: The app will use data analytics to provide insights into the user's music habits and make recommendations.
5. Machine learning: The app should use machine learning algorithms to provide personalized recommendations for music based on the user's listening history.