**ABC RACING COMPANY**

**Problem Statement:**

ABC Racing Company is experiencing a downfall in the people showing interest in their sport. They have decided to increase the fanfare, by revamping their digital presence. They would like to recruit and increase their fan base by reaching out to fans wherever they are.

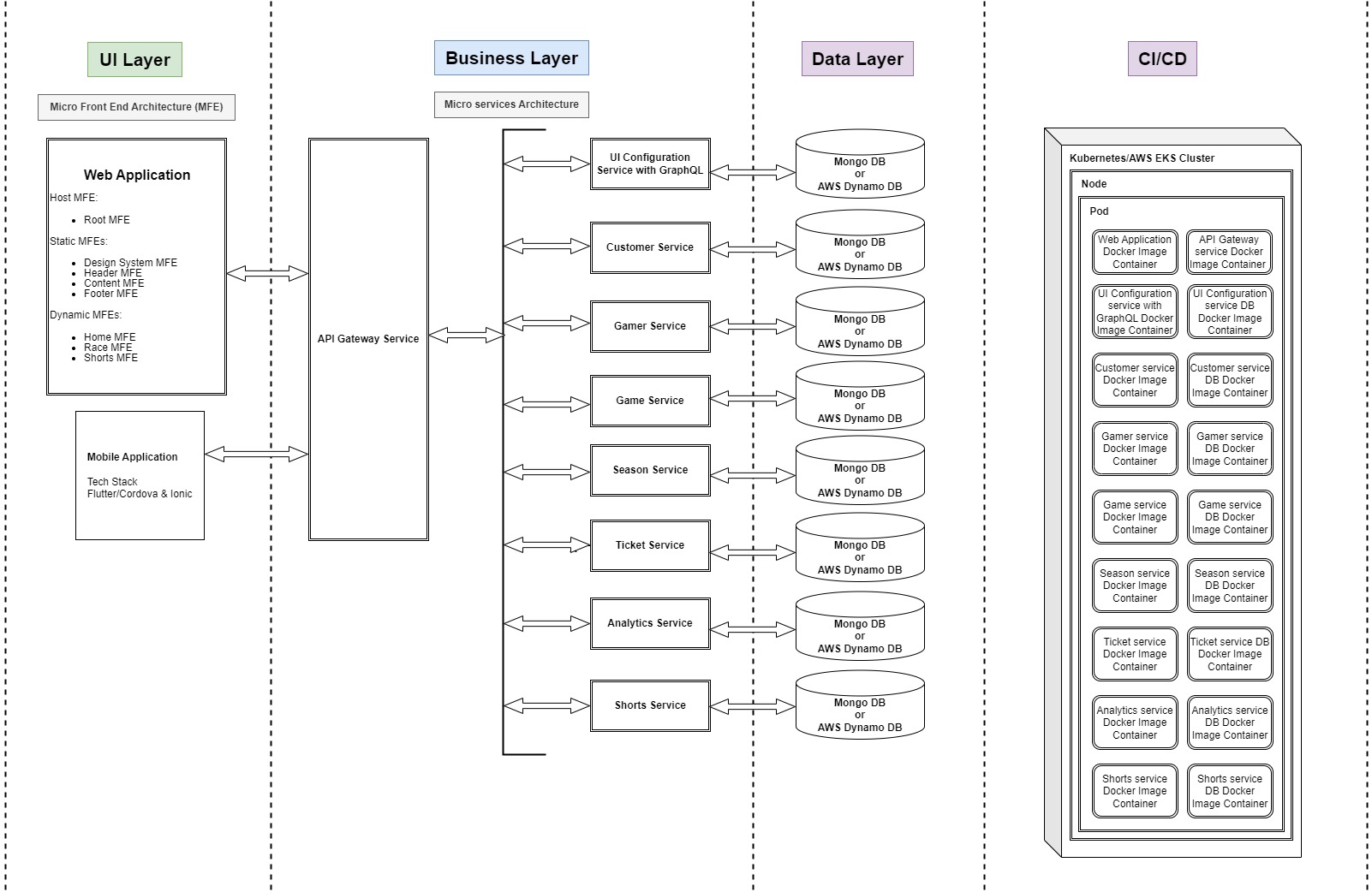
They still want to connect to their ageing fan base and provide access to their content (like fixtures, top 5 racers etc.) across all digital platforms.

**Proposing Solution:**

**Assumption:**

Assumed ABC Racing Company business is conducting races on sports like Car Race, Bike Race, etc., and wants to do Re-Engineering with some new features and Technology migration on their existing system. Based on the above assumption, I am proposing the architectural design and technical solution to build next gen app with tech stack of **MERN**.

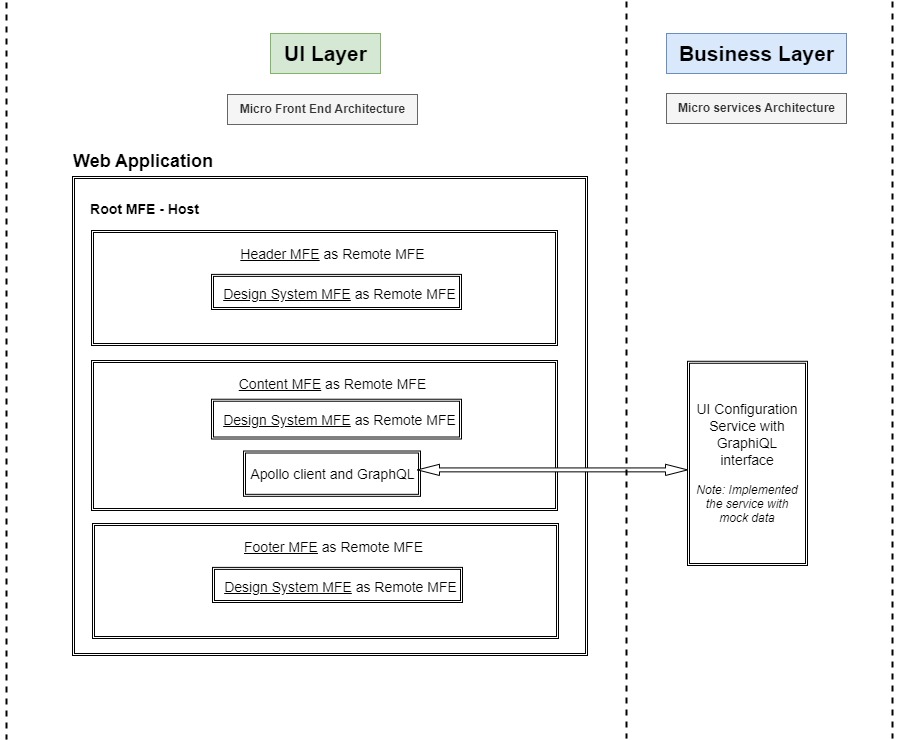
**High Level End to End Architectural design:**



**UI Layer:**

Recommending Micro Front End (MFE) Architecture and Configurable UI Driven model to increase **Productivity, and able to achieve Separation of concerns, Maintainability, Extensibility, Scalability and Performance.**

**Elaborated Front-end Architecture**



|  |
| --- |
| **Note:**  As part of coding, I have implemented the above front end architecture design as working code. Please refer the project folder and screen shots.  As part of implementation, I have demonstrated the following concepts.   1. Project setup for MFE architecture using Web pack Module Federation plugin 2. How to load Static MFEs and how to do Communication between MFEs using web API called Broad cast channel 3. How we can effectively use the Design system as common remote MFE and Implementation of Styled components 4. Implementation of Configurable UI driven approach using Appollo client and GraphQL 5. Rendering pages using Configurable UI JSON which is received through GraphQL using useQuery to achieve both Horizontal and Vertical Extensibility   I have fully focused to implement the above Front-end architecture design rather than focusing on CSS work but achieved the bare minimum responsive design. |

**Front End Architecture Design details:**

1. Host MFE – Responsible to load/host other remote MFEs
   1. Root MFE
      1. Implemented the Root MFE to host the following static remote MFEs.
         1. Header MFE
         2. Content MFE
         3. Footer MFE
2. Static MFEs – MFEs which are loaded statically through the remote object of **web pack module federation** 
   1. Design System MFE
      1. Recommended React Material UI as Design system for maintainability
      2. Implemented this MFE as common MFE which is responsible to maintain the theming object, design system components, **custom components, common components and Localization** as well.
   2. Header MFE
      1. Added logo and implemented the user location dropdown.
      2. By selecting the user location respective location theme will be applied to both Header MFE and Footer MFE through Design system MFE theming object. Here, I havedemonstrated how MFEs are communicating each other usingweb API called **Broad cast channel**.
   3. Content MFE
      1. Implemented Home page and its sections to demonstrate how to achieve both Vertical and Horizontal extensibility and modularity by rendering a page using configurable UI JSON with the help of Apollo client and GraphQL.
   4. Footer MFE
3. Dynamic MFEs – I am proposing this activity as Enhancements/Improvements
   1. Home MFE
   2. Race MFE
   3. Shorts MFE

**Mobile App:**

Recommending Flutter or Cordova as tech stack for mobile app

**Business Layer:**

Recommending Microservice Architecture to increase **Productivity, and able to achieve Separation of concerns, Maintainability, Extensibility, Scalability and Performance.**

Microservices:

1. API Gateway Service –
   1. Responsible for Authentication, Authorization, Service Discovery and Load balancing
2. UI Configuration Service with GraphQL –
   1. Responsible to serve the UI configuration JSON for all MFE, modules, pages, sections and components
   2. Mocked data in this service instead of connecting to Mongo DB.
3. Customer Service – Responsible to serve Customer related concerns.
4. Gamer Service – Responsible to serve Racer related concerns.
5. Game Service – Responsible to serve Race related concerns.
6. Ticket Service – Responsible to serve Ticket related concerns.
7. Season Service – Responsible to serve Season related concerns.
8. Analytics Service – Responsible to serve Analytics.
9. Shorts Service – Responsible to serve Shorts videos. **(My new innovation idea to bring more fans. I have explained the details in Innovation document)**

**Database Layer:**

Databases:

1. MongoDB – Recommending MongoDB because it’s a best DB for processing Image, video related data, to achieve scalability and consistency.
2. AWS DynamoDB - In case of Preferring public cloud, I would recommend AWS because its very cost effective like **Pay as Go**.

**CI/CD Layer:**

Deployment tools:

1. Jenkins – Responsible for build process and to maintain code quality using test cases coverage and sonarqube
2. Docker – Responsible for containerization of application
3. Kubernetes – Responsible for Container orchestration and to **achieve scalability on race time and seasonal race time to manage the traffic and load.**
4. Terraform – Responsible to maintain multiple public cloud platform
5. AWS - In case of Preferring public cloud, I would recommend AWS because its very cost effective like **Pay as Go**.