# Description:

This project is aimed at developing a sports management system. This system is an online web application that can be accessed throughout the country with proper login provided. The project helps in identifying the upcoming and ongoing events in an organization.

# 

# Functional Requirement

Functional specifications are statements of the services that the system should provide, how the system should respond to specific inputs, and how in specific circumstances the system should act. It defines the functionality of the app that developers must integrate into the product to allow users to perform their tasks.

* The system allows users to sign up with their mobile number and email id.
* The system validates the username and password for the authentication process.
* The system allows users to identifying the upcoming and ongoing events in an organization.
* The system allows users to identifying the team members for particular sports using the discussion forum by posting the requirements.

**Non-Functional Requirements**

Non-functional prerequisites, as the name suggests, are necessities that do not directly affect the particular administrations conveyed by the system to their customers. For instance, they can identify reliability, response time, and store inhabitants with emanant device properties. On the other hand, limitations to the use of the platform, such as the capacities of I/O gadgets or knowledge portrayals used in various interfaces of frameworks, can be characterized. In general, non-useful prerequisites, such as implementation, security or accessibility, describe or compel the features of the system. They are the

* + - * + Usability
        + Security
        + Performance
        + Availability

***Usability*:**

The device provides an aid and support menu for both interfaces for the user to connect with the system. The user can access the system after reading aid and support.

***Security:***

The framework requires a username and password to secure the computer from unauthorized entry. For staff, the password must be larger than eight characters.

***Performance:***

For each instruction performed by the operator, the system response time must not exceed a minimum of 10 seconds. The computer should have a high-performance rate when performing user input and should be able to respond within a limited time frame, normally 50 seconds for an extremely complicated task and 20 to 25 seconds for a less complicated task.

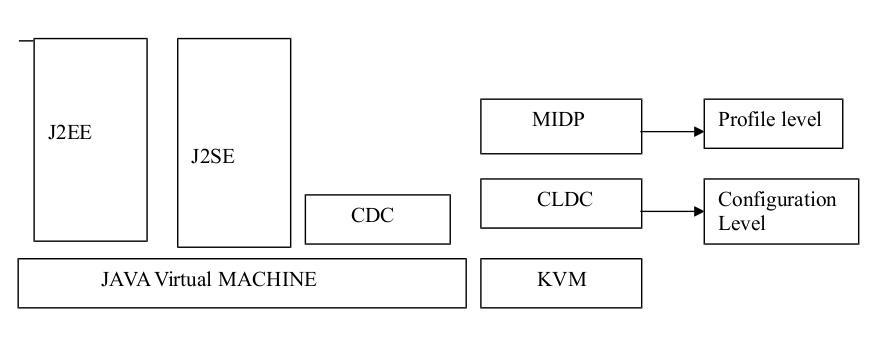
***Availability*:**

The device's entrance should also be available 24 hours a day, 7 days a week. In the event of any major system malfunction, the computer should also be accessible within 1 to 2 working days, ensuring that the business activity is not significantly affected.

**Interfaces**

**Configurations:**

To support the broad range of products that fit within the scope of J2ME, Sun introduced the Configuration.



J2ME Architecture

A Configuration defines a Java platform for a broad range of devices. A Configuration is closely tied to a JavaVirtualMachine (JVM). In fact, a Configuration defines the Java language features and the core Java libraries of the JVM for that particular Configuration.

**Connected Device Configuration (CDC)**

* 512 kilobytes (minimum) memory for running Java.
* 256 kilobytes (minimum) for runtime memory allocation.
* Network connectivity, possibly persistent and high bandwidth.

**Connected, Limited Device Configuration (CLDC)**

128 kilobytes memory for running Java.

* 32 kilobytes memory for runtime memory allocation.
* Restricted user interface.
* Low power, typically battery powered.
* Network connectivity, typically wireless, with low bandwidth and intermittent access.

**Java Virtual Machine:**

The engine behind any Java application (or applet, servlet, etc.) is the JVM. Once you've compiled your Java source code into a class file(s), and optionally included them in a Java Archive (JAR) file, the JVM translates the class files (more accurately, the byte code in the class files) into machine code for the platform running the JVM. The JVM is also responsible for providing security, allocating and freeing memory and managing threads of execution. It's what makes your Java programs go, so to speak.

For CDC, the virtual machine has the same specification as J2SE. For CLDC, Sun has developed what is referred to as a reference implementation of a virtual machine, known as the KVirtualMachine, or simply KVM. This virtual machine was designed to handle the special considerations of resource-constrained devices. It's clear the KVM is not the "traditional" Java virtual machine:

* The virtual machine itself requires only 40 and 80 kilobytes of memory
* Only 20 to 40 kilobytes of dynamic memory (heap) are required
* Can run on 16-bit processors clocked at only 25MHz

**Generic Architecture:**

It begins with the host Operating System (OS) as the base (refer Figure 2.2), followed by the virtual machine (VM). The VM will take one of two forms:

* For systems complying with the CDC, it will be the "traditional” virtual machine; that is, the same feature set as in the Java 2 Standard Edition.
* For systems complying with the CLDC, it will be the KVMora

virtual machine that meets the specifications as required by the CLDC.

**Development Phase 1 :**

In the initial phase will develop the backend REST API using Spring Boot for the Sports Management project .

There will be total of 6 API’s needed for this project , will briefly explain all the API’s below :

1. **User Registration API** : This will be the POST API . In this api will get the user details from the screen as a request and store the details in the corresponding table and return a success code as the response .
2. **User Authentication API** : In this api will get the user login details and authenticate with by fetching the corresponding details from the backend database and comparing it and return a flag as response to check whether it is authenticated or not .
3. **Ongoing Events API** : This will be the GET API . In this api will fetch all the ongoing events which are stored in the corresponding table in the database and return a JSON object containing all the details of the ongoing events .
4. **Incoming Events API** : This will be the GET API . In this api will fetch all the incoming events which are stored in the corresponding table in the database and return a JSON object containing all the details of the incoming events .
5. **Discussion Forum POST API** : In this api will post the details related to the discussion forum to the corresponding table stored in the database and return a success code as the response .
6. **Discussion Forum GET API :** In this api will get all the details related to the discussion forum from the corresponding table stored in the database as a JSON object.

**Development Phase 2 :**

In the second phase of the project will develop the front end screens in REACT JS for the Sports Management project.There will be total of 6 screens needed for this project , will briefly explain all the screen below :

1. **Login Page** : This will be the initial screen when we open the application . In this screen if the user already exists then they give their details and authenticate themself or if they are a new customer then there will be a register link , on clicking it will navigate to the register screen .
2. **User Registration Page** : In this page the new customer will fill in the details to register himself . The page contains all the necessary fields that are needed to register the new customer .
3. **Dashboard Page** : This page will open after the user successfully authenticates themself . This will be the main page containing all the information about the ongoing events , incoming events and discussion forum . So it consists of 3 sub screens .
4. **Ongoing Event Page** : This will be the sub page of the dashboard page. It will display all the information related to the ongoing events by rendering the data received from the backend api .
5. **Incoming Event Page** : This will be the sub page of the dashboard page. It will display all the information related to the incoming events by rendering the data received from the backend api .
6. **Discussion Forum Page** : This will be the sub page of the dashboard page. It will display all the information related to the discussion forum by rendering the data received from the backend api .

**Development Phase 3 :**

In the third phase of the project will integrate the front end pages with the backend api for the Sports Management project.

This will be the final phase of the project . When the user register or login or navigate to any screen then we need to call the corresponding backend api from the front end and get the corresponding response and render it in the page .

While calling the api we need to handle the fallback received from the backend api by notifying the customers through rendering the corresponding component in the front end page .

**Member’s contribution:**

| Members | Contribution description | Overall contribution | Note |
| --- | --- | --- | --- |
| Abhiram | system structure | 25% | system structure |
| Anudeep | non-functional requirements | 25% | non-functional requirements |
| Aditya | functional requirements | 25% | functional requirements |
| Chaitanya | Interfaces | 25% | Interfaces |
| Charan | development phases - 1 | 25% | development phases - 1 |
| Nitesh | development phases - 2 | 25% | development phases - 2 |
| Sai Krishna | development phases - 3 | 25% | development phases - 3 |
| Manvitha | MOM and Contribution | 25% | MOM and Contribution |