

1.INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The project brings the entire manual process of “**ONLINE SPORTS EVENT REGISTRATION**” which is built using JAVA as a front end and SQL Server as a backend. The main purpose of this project is to simplify the process of handling each sports event by providing a web interface for admin and teacher. The admin part consists of multiple modules to initiate with the sports event by adding the type of sport (indoor or outdoor), adding student who are interested in a particular sports activity, adding teachers who will conduct the particular sports activity which is allotted by the admin itself and lastly, viewing the results of sports event held in college. The teacher part has come up with handling all the sports related activity assigned by the admin. Teacher performs various task such as taking the attendance of the students who are registered for a particular sport event, viewing the list of students to mark the winner of each round, generating the results based on multiple rounds won by the student and also can view the 1st, 2nd and 3rd standings of student's name for the particular sport event.

1.2 SYSTEM SPECIFICATION

System Requirements Specification also known as Software Requirements Specification, is a document or set of documentation that describes the features and behavior of a software application.

1.2.1 HARDWARE SPECIFICATION

- Processor : P 4 700 GHz.
- RAM : 4 GB RAM
- Hard Disk Drive : 180 GB

1.2.2 SOFTWARE SPECIFICATION

- Operating System : Windows 7/8/10
- Front End : JAVA
- Back End : SQL

1.3 LANGUAGE DESCRIPTION

WINDOWS OS

Windows is a graphical operating system developed by Microsoft. It allows users to view and store files, run the software, play games, watch videos, and provides a way to connect to the internet. It was released for both home computing and professional works.

Microsoft introduced the first version as 1.0

It was released for both home computing and professional functions of Windows on 10 November 1983. Later, it was released on many versions of Windows as well as the current version, Windows 10.

In 1993, the first business-oriented version of Windows was released, which is known as Windows NT 3.1. Then it introduced the next versions, Windows 3.5, 4/0, and Windows 2000. When the XP Windows was released by Microsoft in 2001, the company designed its various versions for a personal and business environment. It was designed based on standard x86 hardware, like Intel and AMD processor. Accordingly, it can run on different brands of hardware, such as HP, Dell, and Sony computers, including home-built PCs.

Play Video

Editions of Windows

Microsoft has produced several editions of Windows, starting with Windows XP. These versions have the same core operating system, but some versions included advance features with an additional cost. There are two most common editions of Windows:

- Windows Home
- Windows Professional

Windows Home is basic edition of Windows. It offers all the fundamental functions of Windows, such as browsing the web, connecting to the Internet, playing video games, using office software, watching videos. Furthermore, it is less expensive and comes pre-installed with many new computers.

JAVA

Java is a high-level programming language developed by Sun Microsystems. It was originally designed for developing programs for set-top boxes and handheld devices, but later became a popular choice for creating web applications.

The Java syntax is similar to C++, but is strictly an object-oriented programming language. For example, most Java programs contain classes, which are used to define objects, and methods, which are assigned to individual classes. Java is also known for being stricter than C++, meaning variables and functions must be explicitly defined. This means Java source code may produce errors or "exceptions" more easily than other languages, but it also limits other types of errors that may be caused by undefined variables or unassigned types.

Unlike Windows executable (.EXE files) or Macintosh applications (.APP files), Java programs are not run directly by the operating system. Instead, Java programs are interpreted by the Java Virtual Machine, or JVM, which runs on multiple platforms. This means all Java programs are multiplatform and can run on different platforms, including Macintosh, Windows, and Unix computers. However, the JVM must be installed for Java applications or applets to run at all. Fortunately, the JVM is included as part of the Java Runtime Environment (JRE),

SQL

Structured query language (SQL) is a programming language for storing and processing information in a relational database. A relational database stores information in tabular form, with rows and columns representing different data attributes and the various relationships between the data values. You can use SQL statements to store, update, remove, search, and retrieve information from the database. You can also use SQL to maintain and optimize database performance.

Relational database management systems use structured query language (SQL) to store and manage data. The system stores multiple database tables that relate to each other. MS SQL Server, MySQL, or MS Access are examples of relational database management systems. The following are the components of such a system.

A SQL table is the basic element of a relational database. The SQL database table

consists of rows and columns. Database engineers create relationships between multiple database tables to optimize data storage space.

SQL statements, or SQL queries, are valid instructions that relational database management systems understand. Software developers build SQL statements by using different SQL language elements. SQL language elements are components such as identifiers, variables, and search conditions that form a correct SQL statement.

2.SYSTEM STUDY

2.1 EXISTING SYSTEM

In the existing Sports Event Management system, students are not able to get proper information about the games conducted in various colleges. The student needs to spend the time to get the information about the game. The student should attend the venue to get registered for the game which takes a lot of time.

2.1.1 DRAWBACKS

- No aware for outside tournaments
- Can't managing the team strength
- Can't generating the report for the tournament

2.2 PROPOSED SYSTEM

In the proposed Sports Event Management system student can get all the information of various games and the venue. The student can get registered from anywhere and at any time. By using this system student can save a lot of time and effort. The student can easily get the information from anywhere.

2.2.1 FEATURES

- Can track the team details
- Find the registration team details
- All the tournament details view in the single page

3.SYSTEM DESIGN AND DEVELOPMENT

3.1 FILE DESIGN

The selection of the file system design approach is done according to the needs of the developers what are the needed requirements and specifications for the new design. It allowed us to identify where our proposal fitted in with relation to current and past file system development. Our experience with file system development is limited so the research served to identify the different techniques that can be used. The variety of file systems encountered show what an active area of research file system development is. The file systems may be from one of the two fundamental categories. In one category, the file system is developed in user space and runs as a user process. Another file system may be developed in the kernel space and runs as a privileged process. Another one is the mixed approach in which we can take the advantages of both aforesaid approaches. Each development option has its own pros and cons. In this article, these design approaches are discussed.

A file system is the data structure designed to support the abstraction of the data blocks as an archive and collection of files. This data structure is unique because it is stored on secondary storage (usually the disk), which is a very slow device.

The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent upon the way it organizes files on storage devices.

File Design Information systems in business are file and database oriented. Data are accumulated into files that are processed or maintained by the system. The systems analyst is responsible for designing files, determining their contents and selecting a method for organizing the data.

The most important purpose of a file system is to manage user data. This includes storing, retrieving and updating data. Some file systems accept data for storage as a stream of bytes which are collected and stored in a manner efficient for the media.

3.2 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:’

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

- Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user
- will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

3.3 OUTPUT DESIGN

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

External Outputs

Manufacturers create and design external outputs for printers. External outputs enable the system to leave the trigger actions on the part of their recipients or confirm actions to their recipients.

Some of the external outputs are designed as turnaround outputs, which are implemented as a form and re-enter the system as an input.

Internal outputs

Internal outputs are present inside the system, and used by end-users and managers. They support the management in decision making and reporting.

Output Integrity Controls

Output integrity controls include routing codes to identify the receiving system, and verification messages to confirm successful receipt of messages that are handled by network protocol.

Printed or screen-format reports should include a date/time for report printing and the data. Multipage reports contain report title or description, and pagination. Pre-printed forms usually include a version number and effective date.

3.4 DATABASE DESIGN

Today's businesses depend on their databases to provide information essential for day-to-day operations, especially in case of electronic commerce businesses who has a definite advantage with up-to-date database access. Good design forms the foundation of any database, and experienced hands are required in the automation process to design for optimum and stable performance.

Software Solutions have been constantly working on these platforms and have attained a level of expertise. We apply proven methodologies to design, develop, integrate and implement database systems to attain its optimum level of performance and maximize security to meet the client's business model.

Business needs addressed:

- Determine the basic objects about which the information is stored
- Determine the relationships between these groups of information and the objects
- Effectively manage data and create intelligent information
- Remote database administration or on site administrative support
- Database creation, management, and maintenance
- Information retrieval efficiency, remove data redundancy and ensure data security

The most important consideration in designing the database is how the information will be used. The main objective of designing a database is Data Integration, Data Integrity and Data Independence.

Data Integration In a database, information from several files is coordinated, accessed and operated upon as through it is in a single file. Logically, the information is centralized, physically; the data may be located on different devices, connected through data communication facilities.

Data Integrity

Data integrity means storing all data in one place only and how each application accesses it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications. This leads to less data redundancy that is data items need not be duplicated.

Data Independence

Data independence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of application and allow modifications to application programs without reorganizing the physical data.

3.5 SYSTEM DEVELOPMENT

Systems development is the process of defining, designing, testing, and implementing a new software application or program. It could include the internal development of customized systems, the creation of database systems, or the acquisition of third party developed software.

Systems development life cycle phases include planning, system analysis, system design, development, implementation, integration and testing, and operations and maintenance.

3.5.1 DESCRIPTION OF MODULES

User Registration

This user registration will help to register the user, who interested to join this tournament. This will store into the user table, when to open the login this need will help to validate username and password.

Event Registration

Admin will focus the create the event name and details as well, once the registration has been completed, it will shows in the user screen. User easily figures out and follows the tournament details.

Team Registration

After the user login manually create the team, there is an number peoples can be added in the team, it may differ based on the sports.

Manage user details

Admin can see the user detail in his portal, the registered user can visible into the admin screen, the collect the user information and try to reach them.

Booking Events

When user need to join the tournament, should before booking an event is an mandatory option to fill. Here the admin can store some condition to promote the changes as an no of team can be participate.

4.TESTING AND IMPLEMENTATION

TESTING METHODOLOGIES

System testing is state of implementation, which is aimed at ensuring that the system works accurately and efficiently as expect before live operation commences. It certifies that the whole set of programs hang together.

System testing requires a test plan that consists of several key activities and step for run program, string, system and user acceptance testing. The implementation of newly designed package is important in adopting a successful new system

Testing is the important stage in software development. the system test in implementation stage in software development process. The system testing implementation should be confirmation that all is correct and an opportunity to show the users that the system works as expected. It accounts the largest percentage of technical effort in the software development process.

Testing phase in the development cycle validates the code against the functional specification testing is vital to achievement of the system goals. The objective of the testing is to discover errors to fulfill this objective a series of test step unit, integration. Validation and system tests were planned and executed the test steps are:

SYSTEM TESTING

Testing is an integral part of any system development life cycle. Insufficient and untested applications may tend to crash and the result is loss of economic and manpower investment besides user's dissatisfaction and downfall of reputation. Software testing can be looked upon as one among many processes, an organization performs, and that provides the lost opportunity to correct any flaws in the developed system. Software testing includes selecting test data that have more probability of giving errors.

The first step in system testing is to develop a plan that tests all aspects of the system. Completeness, correctness, reliability and maintainability of the software are to be tested for the best quality assurance that the system meets the specification and requirements for its intended use and performance. System testing is the most useful practical process of executing a program with the implicit intention of finding errors that make the program fails. System testing is done in three phases.

- Unit Testing
- Integration Testing
- Validation Testing

UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software the module. Using the detailed design and the process specification testing is done to registration by the user with in the boundary of the Login module. The login form receives the username and password details and validates the value with the database. If valid, the home page is displayed.

INTEGRATION TESTING

Integration Testing is the process of this activity can be considered as testing the design and hence module interaction. The primary objective of integration testing is to discover errors in the interfaces between the components. Login form and registration form are integrated and tested together. If the user is newly registered, the received details will be stored in the registration table. While logging in, the application will check for valid user name and password in the registration table and if valid the user is prompted for submitting complaints.

Data can be lost across an interface, one module can have adverse effect on another sub function when combined it may not produce the desired major functions. Integration testing is a systematic testing for constructing test to uncover errors associated within an interface.

The objectives taken from unit tested modules and a program structure is built for integrated testing. All the modules are combined and the test is made.

A correction made in this testing is difficult because the vast expenses of the entire program complicated the isolation of causes. In this integration testing step, all the errors are corrected for next testing process.

VALIDATION TESTING

Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it fulfills its in purpose the actual result from the expected result for the complaint process. Select the complaint category of the complaint by user. The input given to various forms fields are validated effectively. Each module is tested independently. It is tested that the complaint module fields receive the correct input for the necessary details such as complaint category, complaint id, reference name, complaint description, and email for further process.s

After the completion of the integrated testing, software is completely assembled as a package; interfacing error has been uncovered and corrected and a final series of software test validation begins.

Validation testing can be defined in many ways but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer. After validation test has been conducted, one of two possible conditions exists.

OUTPUT TESTING

The next process of validation testing, is output testing of the proposed system, since no system could be successful if it does not produce the required output in the specified format. Asking the user about the format required, list the output to be generated or displayed by the system under considerations.

Output testing is a different test whose primary purpose is to fully exercise the computer based system although each test has a different purpose all the work should verify that all system elements have been properly integrated and perform allocated functions.

The output format on the screen is found to be corrected as the format was designed in the system design phase according to the user needs for the hard copy also; the output testing has not resulted in any correction in the system.

SYSTEM IMPLEMENTATION

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about the working of the system. The aim of the system illustration was to identify any malfunction of the system.

After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly.

Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system; a demonstration was given to the end user about the working system.

This process is used to verify and identify any logical mess working of the system by feeding various combinations of test data. After the approval of the system by both end user and management the system was implemented.

System implementation is made up of many activities. The six major activities are as follows.

CODING

Coding is the process of whereby the physical design specifications created by the analysis team turned into working computer code by the programming team. A design code may be a tool which helps ensure that the aspiration for quality and quantity for customers and their requirements, particularly for large scale projects, sought by the water agency Design pattern are documented tried and tested solutions for recurring problems in a given context. So basically you have a problem context and the proposed solution for the same.

IMPLEMENTATION PROCEDURES

Implementation includes all the activities that take place to convert the old system to the new one. Proper implementation is essential to provide a reliable system to meet the organization requirements. Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

IMPLEMENTATION PROCEDURES

PILOT RUNNING

Processing the current data by only one user at a time called the pilot running process. When one user is accessing the data at one system, the system is set to be engaged and connected in network. This process is useful only in system where more than one user is restricted.

PARALLEL RUNNING:

Processing the current data by more than one user at a time simultaneously is said to be parallel running process. This same system can be viewed and accessed by more than one user at the time. Hence the implementation method used in the system is a pilot type of implementation.

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

The stage consists of,

- Testing the developed program with sample data.
- Detection's and correction of error.
- Creating whether the system meets user requirements.
- Making necessary changes as desired by the user.
- Training user personnel.

USER TRAINING

User Training is designed to prepare the user for testing & consenting the system. .

- User Manual.
- Help Screens.
- Training Demonstration.

USER MANUAL

The summary of important functions about the system and software can be provided as a document to the user.

HELP SCREENS

This features now available in every software package, especially when it is used with a menu. The user selects the “Help” option from the menu. The system accesses the necessary description or information for user reference.

TRAINING DEMONSTRATION:

Another User Training element is a Training Demonstration. Live demonstrations with personal contact are extremely effective for Training Users.

SYSTEM MAINTENANCE

Maintenance is actually the implementation of the review plan. As important as it is, many programmers and analysts are to perform or identify themselves with the maintenance effort. There are psychological, personality and professional reasons for this. Analysts and programmers spend far more time maintaining programs than they do writing them. Maintenance accounts for 50-80 percent of total system development

Maintenance is expensive. One way to reduce the maintenance costs are through maintenance management and software modification audits.

- Maintenance is not as rewarding as exciting as developing systems. It is perceived as requiring neither skill nor experience.
- Users are not fully cognizant of the maintenance problem or its high cost.
- Few tools and techniques are available for maintenance.
- A good test plan is lacking.
- Standards, procedures, and guidelines are poorly defined and enforced.
- Programs are often maintained without care for structure and documentation.
- There are minimal standards for maintenance.
- Programmers expect that they will not be in their current commitment by time their programs go into the maintenance cycle.

Corrective Maintenance

It means repairing, processing or performance failure or making changes because of previously uncovered problems or false assumptions. Task performed to identify, isolate, and rectify a fault so that the failed equipment, machine, or system can be restored to an operational condition within the tolerances or limits established for in-service operations.

Corrective maintenance can be subdivided into "immediate corrective maintenance" (in which work starts immediately after a failure) and "deferred corrective maintenance" (in which work is delayed in conformance to a given set of maintenance rules).

Perfective Maintenance

It means changes made to a system to add new features or to improve performance. Preventive maintenance is predetermined work performed to a schedule with the aim of preventing the wear and tear or sudden failure of equipment components. process or control equipment failure can have adverse results in both human and economic terms. In addition to down time and the costs involved to repair and/or replace equipment parts or components, there is the risk of injury to operators, and of acute exposures to chemical and/or agents.

Time-based or run-based Periodically inspecting, servicing, cleaning, or replacing parts to prevent sudden failure .On-line monitoring of equipment in order to use important/expensive parts to the limit of their serviceable life. Preventive maintenance involves changes made to a system to reduce the chance of future system failure.

An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends report information to a printer so that so that the system can adapt to changes in printer technology.

Preventive Maintenance

Changes made to a system to avoid possible future problems Perfective maintenance involves making enhancements to improve processing performance, interface usability, or to add desired, but not necessarily required, system features. The objective of perfective maintenance is to improve response time, system efficiency, reliability, or maintainability.

During system operation, changes in user activity or data pattern can cause a decline in efficiency, and perfective maintenance might be needed to restore performance. Usually, the perfective maintenance work is initiated by the IT department, while the corrective and adaptive maintenance work is normally requested by users.

CONCLUSION

Moreover, these systems enhance convenience by providing online registration forms that can be completed from anywhere, at any time. This eliminates the need for participants to visit physical registration centers and reduces the time and effort required to register for events. This can lead to increased participation and ensure that events are well-attended and successful.

Online sports event registration systems also increase participation by providing real-time updates on event progress, schedules, and participant information. This helps to promote excitement and engagement, and ensures that participants are informed and up-to-date on event details.

In conclusion, online sports event registration systems are powerful tools that enable athletes, teams, and organizers to manage the registration process for sports events efficiently and effectively. They offer several benefits, including improved accuracy, enhanced convenience, and increased participation. By leveraging these benefits, event organizers can attract more participants, ensure accurate and up-to-date information, and promote successful events.

BIBLIOGRAPHY

Book Reference:

- Bloch, Joshua. Effective Java: Programming Language Guide. Addison-Wesley, 2017.
- Eckel, Bruce. Thinking in Java. Prentice Hall, 2006.
- Freeman, Eric, and Elisabeth Robson. Head First Java. O'Reilly Media, 2005.
- Horstmann, Cay S. Core Java Volume I--Fundamentals. Prentice Hall, 2018.
- Lea, Doug. Concurrent Programming in Java: Design Principles and Patterns. Addison-Wesley, 2000.
- Naftalin, Maurice, and Philip Wadler. Java Generics and Collections. O'Reilly Media, 2006.

Website:

- Baeldung. "Java Tutorials and Articles." Baeldung, 2023, <https://www.baeldung.com/java-tutorials>.
- GeeksforGeeks. "Java Programming Language." GeeksforGeeks, 2023, <https://www.geeksforgeeks.org/java/>.
- Stack Overflow. "Questions tagged [java]." Stack Overflow, <https://stackoverflow.com/questions/tagged/java>.
- Tutorials Point. "Java Tutorial." Tutorials Point, 2023, <https://www.tutorialspoint.com/java/index.htm>.
- Vogella. "Java Tutorials." Vogella, 2023, <https://www.vogella.com/tutorials/java.html>

APPENDICES

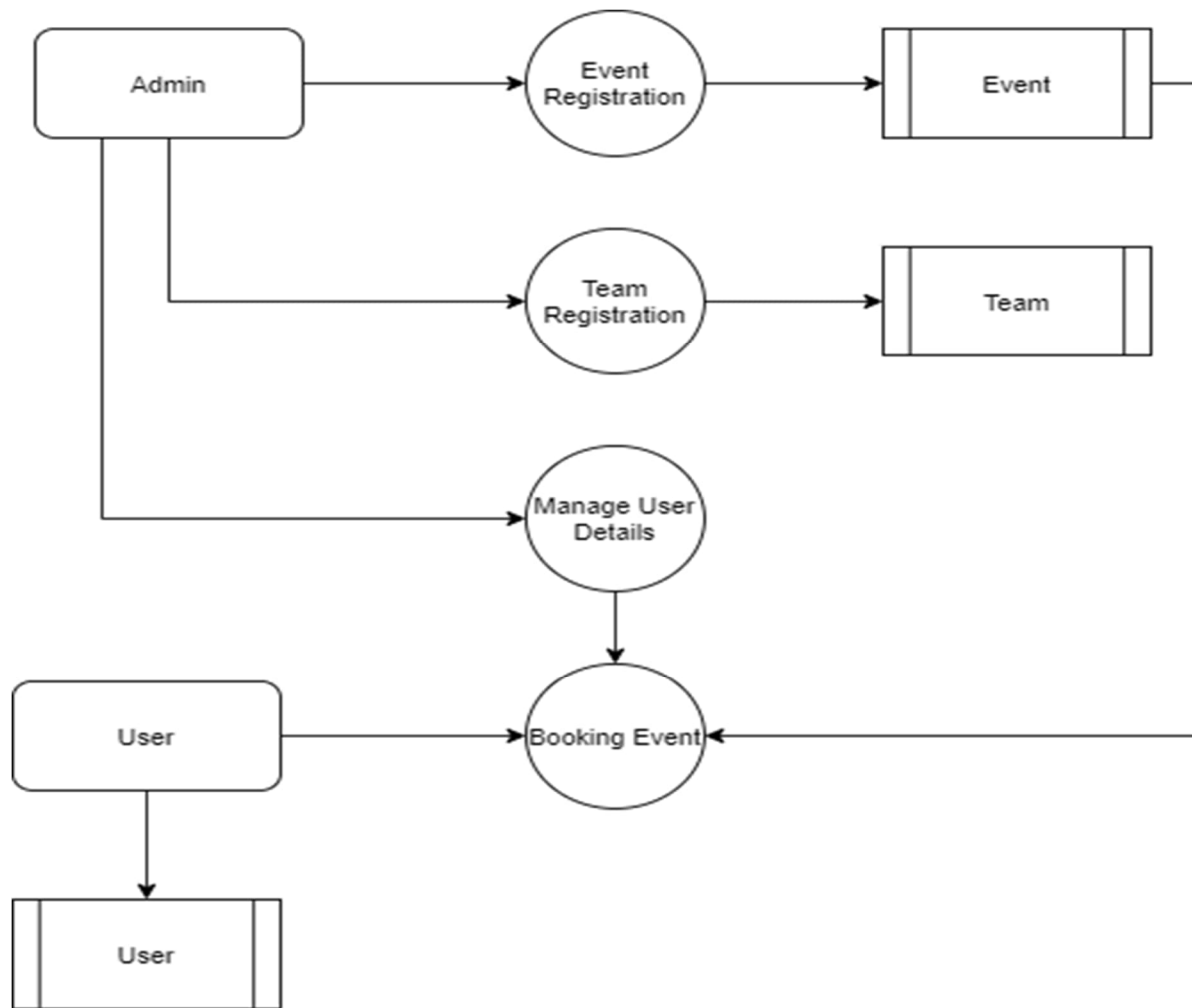
A. DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or system. The DFD also provides information about the outputs and inputs of each entity and process itself. A data-flow diagram is a part of structured-analysis modeling tools.

LEVEL 0:



LEVEL 1:



B. TABLE STRUCTURE

The table needed for each module was designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

TABLE NAME: ADMIN

PRIMARY KEY : Admin_id

FIELD	DATA TYPE	SIZE	CONSTRAINT
Id	Int	10	Primary key
Username	Varchar	20	Not null
password	Varchar	20	Not null

TABLE NAME: EVENT

PRIMARY KEY: Event_id

FIELD	DATA TYPE	SIZE	CONSTRAINT
Event id	Int	10	Primary key
Event name	Varchar	20	Not null
Event category	Varchar	20	Not null
Amount	Int	10	Not null
Total teams	Int	10	Not null
Location	Varchar	20	Not null
Contact person	Varchar	20	Not null
Mobile number	Int	10	Not null

TABLE NAME: USER

PRIMARY KEY: User_id

FIELD	DATA TYPE	SIZE	CONSTRAINT
User id	Int	10	Primary key
Username	Varchar	20	Not null
password	Varchar	20	Not null
Mobile number	Int	10	Not null

TABLE NAME: TEAM

PRIMARY KEY: Team_id

FIELD	DATA TYPE	SIZE	CONSTRAINT
Team id	Int	10	Primary key
Team name	Varchar	20	Not null
Total team members	Int	10	Not null
Captain name	Varchar	20	Not null
Mobile number 1	Int	10	Not null
Mobile number 2	Int	10	Not null
Full address	Varchar	30	Not null

TABLE NAME: BOOKING

PRIMARY KEY: Booking_id

FOREIGN KEY: Event_id

FIELD	DATA TYPE	SIZE	CONSTRAINT
Booking id	Int	10	Primary key
Event id	Int	10	Foreign key
Team id	Int	10	Foreign key
Booking status	Varchar	15	Not null

C.SAMPLE CODEING

```
package com.example.demo.controller;
package com.example.demo.configuration;

import java.util.Properties;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
public class HibernateConfiguration {

    @Value("${db.driver}")
    private String DB_DRIVER;

    @Value("${db.password}")
```

```

private String DB_PASSWORD;

@Value("${db.url}")
private String DB_URL;

@Value("${db.username}")
private String DB_USERNAME;

@Value("${hibernate.dialect}")
private String HIBERNATE_DIALECT;

@Value("${hibernate.show_sql}")
private String HIBERNATE_SHOW_SQL;

// @Value("${hibernate.hbm2ddl.auto}")
private String HIBERNATE_HBM2DDL_AUTO;

@Value("${entitymanager.packagesToScan}")
private String ENTITYMANAGER_PACKAGES_TO_SCAN;

@Bean
public LocalSessionFactoryBean sessionFactory() {
    LocalSessionFactoryBean sessionFactory = new LocalSessionFactoryBean();
    sessionFactory.setDataSource(dataSource());
    sessionFactory.setPackagesToScan(ENTITYMANAGER_PACKAGES_TO_SCAN);
    Properties hibernateProperties = new Properties();
    hibernateProperties.put("hibernate.dialect", HIBERNATE_DIALECT);
    hibernateProperties.put("hibernate.show_sql", HIBERNATE_SHOW_SQL);
//    hibernateProperties.put("hibernate.hbm2ddl.auto", HIBERNATE_HBM2DDL_AUTO);
    sessionFactory.setHibernateProperties(hibernateProperties);
    return sessionFactory;
}

@Bean
public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName(DB_DRIVER);
    dataSource.setUrl(DB_URL);
    dataSource.setUsername(DB_USERNAME);
    dataSource.setPassword(DB_PASSWORD);
}

```

```

        return dataSource;
    }

    @Bean
    public HibernateTransactionManager transactionManager() {
        HibernateTransactionManager txManager = new HibernateTransactionManager();
        txManager.setSessionFactory(sessionFactory().getObject());
        return txManager;
    }
}

package com.example.demo.configuration;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

@Configuration
@EnableWebMvc
public class WebConfig extends WebMvcConfigurerAdapter {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/*");
    }
}

package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dao.ApiDao;

```

```

import com.example.demo.response.GetCitizenResponse;
import com.example.demo.response.GetComplaintResponse;
import com.example.demo.service.ApiService;

@RestController
@RequestMapping(value = { "/sports/api" })
public class ApiController {

    @Autowired
    ApiService service;

    @Autowired
    ApiDao dao;

    @GetMapping("/login/{username}/{password}")
    public String login(@PathVariable String username, @PathVariable String password) {
        return service.login(username, password);
    }

    @GetMapping("/add_user/{mobile}/{email}/{username}/{password}")
    public String add_user(@PathVariable String mobile,
        @PathVariable String email,
        @PathVariable String username,
        @PathVariable String password) {
        dao.add_user(mobile, email, username, password);
        return "User Register Sucessfully";
    }

    @GetMapping("/test")
    public String test() {
        return "test";
    }

    @GetMapping("/add_event/{name}/{category}/{amount}/{total}/{location}/"
        + "{contactperson}/{mobile}/{eventdate}")
    public String add_event(@PathVariable String name,
        @PathVariable String category,
        @PathVariable String amount,
        @PathVariable String total,
        @PathVariable String location,

```



```

        @PathVariable String contactperson,
        @PathVariable String mobile,
        @PathVariable String eventdate) {
    dao.add_event(name,category,amount,total,location,contactperson,mobile,eventdate);
    return "Event Register Sucessfully";
}

@GetMapping("/add_team/{userid}/{teamname}/{total}/{captain}/{mobile1}/{mobile2}/{address}")
public String add_team(@PathVariable Integer userid,
        @PathVariable String teamname,
        @PathVariable String captain,
        @PathVariable String mobile1,
        @PathVariable String mobile2,
        @PathVariable String address,
        @PathVariable String total) {
    dao.add_event(userid,teamname,total,captain,mobile1,mobile2,address);
    return "Team Register Sucessfully";
}

@GetMapping("/book_event/{eventid}/{teamid}/{category}/{amount}/{total}/{location}/{contactperson}/{mobile}/{eventdate}")
public String book_event(@PathVariable Integer eventid,
        @PathVariable Integer teamid,
        @PathVariable String category,
        @PathVariable String amount,
        @PathVariable String total,
        @PathVariable String location,
        @PathVariable String contactperson,
        @PathVariable String mobile,
        @PathVariable String eventdate) {
    dao.add_booking(eventid,teamid,category,amount,total,location,contactperson,mobile,eventdate);
    return "Team Register Sucessfully";
}

@GetMapping("/get_events")
public List<Object[]> get_event() {
    return dao.get_event();
}

```

```

    }

    @GetMapping("/get_teams/{id}")
    public List<Object[]> get_teams(@PathVariable Integer id) {
        return dao.get_teams(id);
    }

    @GetMapping("/get_booking")
    public List<Object[]> get_booking() {
        return dao.get_booking();
    }
}

}
package com.example.demo.dao;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;

import javax.transaction.Transactional;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.NativeQuery;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

@Repository
@Transactional
public class ApiDao {
    @Autowired
    SessionFactory sf;

```

```

public List<Object[]> get_complaints(Integer id) {
    // TODO Auto-generated method stub
    Session session = sf.getCurrentSession();
    String sql = "SELECT * FROM `complaint` as c left join citizen as cz on
(cz.id=c.citizen_id) where c.citizen_id="+id;
    NativeQuery nq = session.createNativeQuery(sql);
    return nq.list();
}

```

```

public String login(String username, String password) {
    // TODO Auto-generated method stub
    Session session = sf.getCurrentSession();
    String sql = "select * from admin where username='"+username+"' and
password='"+password+"'";
    NativeQuery nq = session.createNativeQuery(sql);
    if (nq.list().size() != 0) {
        return "admin";
    } else {
        String sql1 = "select * from user where username='"+username+"' and
password='"+password+"'";
        NativeQuery nq1 = session.createNativeQuery(sql1);
        if (nq1.list().size() != 0) {
            List<Object[]> a = nq1.list();
            return "id="+a.get(0)[0];
        } else {return "Invalid";}
    }
}

```

```

public void add_event(String name, String category, String amount, String total, String
location,
    String contactperson, String mobile, String eventdate) {
    // TODO Auto-generated method stub
    Session session = sf.getCurrentSession();
    String sql = "INSERT INTO `event` (`id`, `name`, `category`, `amount`, `total`,

```

```

        `location`, `contactperson`, `mobile`, `eventdate`) VALUES "
            + "(NULL, '"+name+"', '"+category+"', '"+amount+"', '"+total+"',
        '"+location+"', '"+contactperson+"', '"+mobile+"', '"+eventdate+"');";
        session.createSQLQuery(sql).executeUpdate();
    }

    public void add_event(Integer userid, String teamname, String total, String captain, String
mobile1, String mobile2,
        String address) {
        // TODO Auto-generated method stub
        Session session = sf.getCurrentSession();
        String sql = "INSERT INTO `team` (`id`, `user_id`, `teamname`, `totalmembers`,
        `captain`, `mobile1`, `mobile2`, `address`) VALUES "
            + "(NULL, "+userid+", '"+teamname+"', '"+total+"', '"+captain+"',
        '"+mobile1+"', '"+mobile2+"', '"+address+"');";
        session.createSQLQuery(sql).executeUpdate();
    }

    public void add_booking(Integer eventid, Integer teamid, String category, String amount,
String total,
        String location, String contactperson, String mobile, String eventdate) {
        // TODO Auto-generated method stub
        Session session = sf.getCurrentSession();
        String sql = "INSERT INTO `book` (`id`, `eventid`, `teamid`, `category`, `amount`,
        `total`, `location`, `contactperson`, `mobile`, `eventdate`) VALUES "
            + "(NULL, "+eventid+", "+teamid+", '"+category+"', '"+amount+"',
        '"+total+"', '"+location+"', '"+contactperson+"', '"+mobile+"', '"+eventdate+"');";
        session.createSQLQuery(sql).executeUpdate();
    }

    public void add_user(String mobile, String email, String username, String password) {
        // TODO Auto-generated method stub
        Session session = sf.getCurrentSession();
        String sql = "INSERT INTO `user` (`id`, `mobile`, `email`, `username`, `password`)
VALUES (NULL, '"+mobile+"', '"+email+"', '"+username+"', '"+password+"');";
        session.createSQLQuery(sql).executeUpdate();
    }

    public List<Object[]> get_event() {
        // TODO Auto-generated method stub

```

```

        Session session = sf.getCurrentSession();
        String sql = "select * from event";
        NativeQuery nq = session.createNativeQuery(sql);
        return nq.list();
    }

    public List<Object[]> get_teams(Integer id) {
        // TODO Auto-generated method stub
        Session session = sf.getCurrentSession();
        String sql = "select * from team where user_id="+id;
        NativeQuery nq = session.createNativeQuery(sql);
        return nq.list();
    }

    public List<Object[]> get_booking() {
        // TODO Auto-generated method stub

        Session session = sf.getCurrentSession();
        String          sql          =          "SELECT
book.eventdate,event.name,team.teamname,team.captain,team.mobile1,team.address FROM `book`
left join event on(event.id=book.eventid) LEFT JOIN team on(team.id=book.teamid)";
        NativeQuery nq = session.createNativeQuery(sql);
        return nq.list();
    }

}

package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAutoConfiguration;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;
import org.springframework.web.bind.annotation.GetMapping;

@SpringBootApplication
@EnableAutoConfiguration(exclude = HibernateJpaAutoConfiguration.class)
public class SampleApplication extends SpringBootServletInitializer {

```

```

@Override
protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
    // TODO Auto-generated method stub
    return builder.sources(SampleApplication.class);
}

public static void main(String[] args) {
    SpringApplication.run(SampleApplication.class, args);
}

@GetMapping("/hello")
public String hello() {
    return "Hello";
}

}

logging.level.org.springframework.web=DEBUG
# Database
db.driver: com.mysql.jdbc.Driver
db.url: jdbc:mysql://localhost:3306/sports?useLegacyDatetimeCode=false&serverTimezone=UTC
db.username: root
db.password:
db.databases: sql

# Hibernate
hibernate.dialect: org.hibernate.dialect.MySQL5Dialect
hibernate.show_sql: true
#hibernate.hbm2ddl.auto: update
entitymanager.packagesToScan: com
package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

```

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dao.ApiDao;

import com.example.demo.response.GetCitizenResponse;
import com.example.demo.response.GetComplaintResponse;
import com.example.demo.service.ApiService;

@RestController
@RequestMapping(value = { "/sports/api" })
public class ApiController {

    @Autowired
    ApiService service;

    @Autowired
    ApiDao dao;

    @GetMapping("/login/{username}/{password}")
    public String login(@PathVariable String username, @PathVariable String password) {
        return service.login(username,password);
    }

    @GetMapping("/add_user/{mobile}/{email}/{username}/{password}")
    public String add_user(@PathVariable String mobile,
        @PathVariable String email,
        @PathVariable String username,
        @PathVariable String password) {
        dao.add_user(mobile,email,username,password);
        return "User Register Sucessfully";
    }

    @GetMapping("/test")
    public String test() {
        return "test";
    }

    @GetMapping("/add_event/{name}/{category}/{amount}/{total}/{location}/")

```

```

        + "{contactperson}/{mobile}/{eventdate}")

public String add_event(@PathVariable String name,
        @PathVariable String category,
        @PathVariable String amount,
        @PathVariable String total,
        @PathVariable String location,
        @PathVariable String contactperson,
        @PathVariable String mobile,
        @PathVariable String eventdate) {
    dao.add_event(name,category,amount,total,location,contactperson,mobile,eventdate);
    return "Event Register Sucessfully";
}

@GetMapping("/add_team/{userid}/{teamname}/{total}/{captain}/{mobile1}/{mobile2}/"
        + "{address}")
public String add_team(@PathVariable Integer userid,
        @PathVariable String teamname,
        @PathVariable String captain,
        @PathVariable String mobile1,
        @PathVariable String mobile2,
        @PathVariable String address,
        @PathVariable String total) {
    dao.add_event(userid,teamname,total,captain,mobile1,mobile2,address);
    return "Team Register Sucessfully";
}

@GetMapping("/book_event/{eventid}/{teamid}/{category}/{amount}/{total}/{location}/{c
ontactperson}/{mobile}/{eventdate}")
public String book_event(@PathVariable Integer eventid,
        @PathVariable Integer teamid,
        @PathVariable String category,
        @PathVariable String amount,
        @PathVariable String total,
        @PathVariable String location,
        @PathVariable String contactperson,
        @PathVariable String mobile,
        @PathVariable String eventdate) {

    dao.add_booking(eventid,teamid,category,amount,total,location,contactperson,mobile,eventd
ate);

```



```

        return "Team Register Sucessfully";
    }

    @GetMapping("/get_events")
    public List<Object[]> get_event() {
        return dao.get_event();
    }

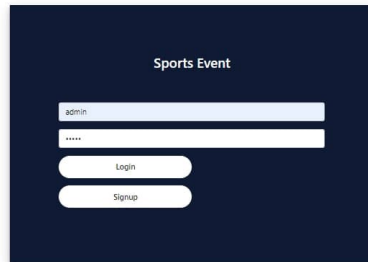
    @GetMapping("/get_teams/{id}")
    public List<Object[]> get_teams(@PathVariable Integer id) {
        return dao.get_teams(id);
    }

    @GetMapping("/get_booking")
    public List<Object[]> get_booking() {
        return dao.get_booking();
    }
}

```

D.SAMPLE INPUT & OUTPUT DESIGN

Admin login



The image shows a dark blue rectangular box representing a login interface. At the top center, the text "Sports Event" is displayed in white. Below this, there are two white input fields. The first field contains the text "admin". The second field contains six asterisks "*****". Below the input fields, there are two white buttons with rounded corners. The top button is labeled "Login" and the bottom button is labeled "Signup", both in dark blue text.

Event Registration

Event Registration	Booking History	Logout
Name		
Event 2023		
Category		
Cricket		
Amount		
2500		
Total Teams		
15		
Location		
Tropur		
Contact Person		
Gukul		
Mobile Number		
8847216589		
Event Date		
05-05-2024		
Register		

User Registration

User Registration

Mobile

18880467678

Email

shahin@gmail.com

Username

shahin

Password

shahin

Register

User Registration

Mobile

98989876543

Email

shafiq@gmail.com

Username

shafiq

Password

shafiq

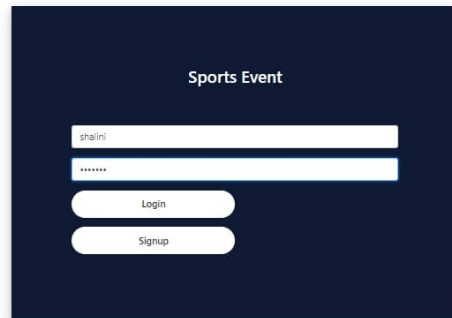
Register

localhost says

User Register Sucessfully

OK

Event login page



A dark blue rectangular card with rounded corners and a subtle drop shadow. At the top center, the text "Sports Event" is displayed in a white sans-serif font. Below this, there are two white input fields. The first field contains the text "shahin" in a small, light gray font. The second field contains a series of dots representing a password. Below the input fields are two white, rounded rectangular buttons. The top button is labeled "Login" and the bottom button is labeled "Signup", both in a small, light gray font.

Sports Event

shahin

.....

Login

Signup

Team registration

Team Registration

Team Name

My Team

Total members

11

Captain Name

Shubini

Mobile Number 1

9898987878

Mobile Number 2

8545456578

Address

Polladam

Register

localhost says
Team Register Sucessfully

OK

View registration status

Event Registration		Booking History		Logout		
#	Event Date	Event Name	Team Name	Contact Person	Mobile Number	Address
1	05-05-2023	Event 2023	My Team	Shalini	9896967678	Palladam

