**Web Based Sports Stadium Information**

**Management System**

**By**

**Ugwueze, Stacey Adanna, U14/NAS/CSC/084**

Being a BSc project report submitted in partial fulfillment of the requirements for the award of a Bachelor’s degree in Computer Science of the Godfrey Okoye University.

**Supervisor: Associate Prof. Oguike O.E**

**Department of Computer Science and Mathematics**

**Godfrey Okoye University, Enugu.**

**JULY, 2018**

**Certification**

We hereby declare that the work presented herein was done by me, and not by a third party. Should I be convicted of having cheated in this work, I shall accept the verdict of the university.

**Ugwueze, Stacey Adanna, U14/NAS/CSC/084**

**Approval**

This project is approved for submission.

**Associate Prof. Oguike O.E Date**

Project Supervisor

**Dr. Monica Agu Date**

H.O.D

**Dedication**

I dedicate this work to God almighty who gave me wisdom and understanding from the beginning to the end and also to all those who are trying their best sincerely to solve little and mighty problems in the world using technology.

**Acknowledgement**

I would like to thank all those whose assistance proved to be a milestone in the accomplishment of my end goal. Thanks to God almighty who has been there from the very beginning until the very end. Thank you for the gift of knowledge and understanding, your provision and above all, the gift of life. I have seen the overall manifold wisdom of God made manifest during the entire process of working on this project.

My profound gratitude to my parents, Mr. & Mrs. Theophilus Ugwueze for their un-relented steadfastness in providing for my educational need without fear or grudges. Thanks so much, love and heart you plenty.

A very big thank you to my siblings, Francis Ugwueze, Kingsley Ugwueze, Valentine Ugwueze and Samson Ugwueze for their support, encouragement and understanding. Thanks for always calling in to ask how it’s been so far, you guys are the best.

To my supervisor, Prof. Oguike O.E. I deeply appreciate your time and energy in this work. Thanks a lot Sir, I appreciate your contributions beyond words.

To my lovely friends, Michael Cosmas (my heart throb), Etimbuk Udoekong, Peter John, Emmanuel Nwoye, Peace Makata, Dove, Clement, Golden Okhawere, Victor Chiedozie, Ogbe Blessing, Stella and Blessing. I appreciate you all for all your effort and understanding to seeing me through to the end of this project. Sometimes certain chapters were hard to figure out, but your ideas paved a way. Thanks a lot. No knowledge of mine is enough to see me through this work, but your prayers made available the presence of the wisdom of God in my life. Thanks lot and God bless you.

***Abstract***

*This work was centered on web based stadium information management system. It is worthy to note that the current process of information management is being operated manually and due to this procedure, numerous problems are encountered especially in the management of the booking of sports arena processing. The motivation of the software is to eliminate the manual procedures in the stadium and proper auditing is carried out in ensuring that those booking the arena are authorized personnel. The objective of the system is to successfully implement the computerized procedure; eliminate the clashing of two events and to reduce the work of stadium managers and operators. The new system which is the expected data provides management software and advisory services to the sports and entertainment sector. The new system was designed using OOAD methodology and implemented using sublime text and WAMP server IDE on a Windows 10 system using HTML, CSS and Jquery as the front end; PHP and MySQL server as the back end. This language was chosen because it is easy to read and understand. It is real time and user friendly.*

**Table of contents**

[1.0 Background of study 1](#_Toc487657646)

[1.1 Statement of the Problem 2](#_Toc487657647)

[1.2 Objectives 2](#_Toc487657648)

[1.3 Significance of the Project 3](#_Toc487657648)

[CHAPTER 2: LITERATURE REVIEW 4](#_Toc487657649)

[2.0 Introduction 4](#_Toc487657650)

[2.1 Theoretical Background 4](#_Toc487657651)

[2.2 Review Of Related Literature 10](#_Toc487657653)

[CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 16](#_Toc487657654)

[3.0 Introduction 16](#_Toc487657655)

[3.1 Description of the existing system](#_Toc487657656) 16

[3.2 Analysis of the Proposed System](#_Toc487657657) 17

[3.3.0 Design of the Proposed System 23](#_Toc487657658)

[3.3.1 Database Design](#_Toc487657659) 23

[3.3.2 Input Design](#_Toc487657662) 27

[3.3.3 Output Design](#_Toc487657663) 28

[3.3.4 System Architecture 30](#_Toc487657664)

[3.3.5 Algorithm Design 32](#_Toc487657665)

[CHAPTER 4: SYSTEM IMPLEMENTATION 33](#_Toc487657666)

[4.0 Introduction 33](#_Toc487657667)

[1.1 Choice of development Environment 33](#_Toc487657668)

[1.1.3 Choice of Programming Language used](#_Toc487657671) 34

[4.2 Implementation Architecture](#_Toc487657678) 38

[4.3 Software Testing 40](#_Toc487657679)

[4.4 Documentation](#_Toc487657683) 41

[4.4.1 User manual](#_Toc487657684) 42

[4.4.2 Source code listing](#_Toc487657688) 43

CHAPTER 5 SUMMARY AND CONCLUSION 45

5.0 Summary 45

5.1 Conclusion 45

5.2 Recommendation 45

Reference

Appendices

**List of Figures**

Fig 3.2.1 Use Case Diagram 19

Fig 3.2.2 Entity Relationship Diagram 20

Fig 3.2.3 Class Diagram 22

Fig 3.3.2 Input Design 27

Fig 3.3.3 Output Design 28

Fig 3.3.4 System Architecture 30

Fig 4.1.1 User Registration 34

Fig 4.1.2 User Login 34

Fig 4.1.3 Screenshot of the website page 35

Fig 4.1.4 Database Design Table 35

Fig 4.2 Implementation Architecture 38

Fig 4.4 Screenshot of the input form 40

Fig 4.5 Screenshot of a successfully space allocation 41

**List of Tables**

Table 1: Sports Arena table 24

Table 2: Spectator Event table 24

Table 3: Customer table 25

Table 4: Companies table 25

Table 5: Attendance table 26

Table 6: Maintenance table 26

**CHAPTER 1**

**INTRODUCTION**

**1.0 Background of Study**

A stadium is a structure that surrounds a central field or a stage in which a spectator event takes place. This project is developed for use by the public for the sole purpose of booking space for sporting activities. Although many stadiums built in the early 20th century were constructed for a narrowly defined use, the design of stadium evolved so that many modern structures can easily be used for a variety of events, also involving the seating capacity of stadiums which depends on event. Stadiums can be either multi-use for designated for one of several types of sporting events, including track, soccer and football. Washington State’s Qwest Field & Event Center, which is home to the Seattle Seahawks, provides a venue for job fairs, soccer games, trade shows and concerts. For example Rogers center, (formerly referred to as Sky Dome), in Toronto, Ontario, Canada, holds 50,600 spectators for baseball games, 53,000 for football games, and 67,000 for concerts, which had entertained manual manner of booking, and then the online in the recent times.

The project is hence, designed to speak volume on the assistance of the stadium managers for managerial and supervisory council skills on the stadium facilities like football court, basketball court, volleyball court, etc.

This project has been in use in some other developed countries like the United kingdom, United States of America, etc.,. Who have over the years shown strong sporting involvement. For example, siting Wembley stadium in the United Kingdom of which, their stadium resources/facilities for the past ten years has been greatly managed, and it has been of real help to the management team.

This software will help/provide solution in the clashes in booking of stadium arena and redundancy.

**1.1 Statement of the Problem**

1. Stadium managers allocate stadium arena/space for more than one person.
2. Users travel from a long distance to visit the stadium, only to find out that all the space has already been booked.
3. Poor file management is on the high side.
4. Fraud is encouraged.

**1.2 Objectives of Study**

1. To login to the website or register to book a space.
2. To view the available spaces, both the space in use and unused.
3. To enable users book a space for an event without stress.
4. To help stadium managers approve or decline orders.
5. To reduce the work load of stadium operators, managers and individuals.
6. To enable users complete the transaction by making payment.

**1.3 Significant of the Problem**

1. Booking of space in stadium will have central database.
2. It will be easy and convenient for the officials in charge of booking and space allocation.
3. It will also give room for easy accountability of events that take place in a particular stadium.
4. It will also help to eliminate the clashing of two events.
5. It will control fraud or deception rate on the stadium property and resource.

**Chapter 2**

**Literature Review**

**2.0 Introduction**

This chapter shows, explain and outlines the various technologies used during the development of my software. It presents the technologies used in my work and how I used them.

**2.1 Theoretical Background**

With the development of sport in the world, problems have arisen in the infrastructure of sports facilities and techniques used in the management of these facilities. Since we are in the era of information technology, any problem you are looking for a solution to it can be the use of information technology. The search for a solution to the problems facing the management of sports facilities is embedded in the use of information technology.

In this project, to find solutions to the problems facing the stadium management through the development of a reservation system. This study will convert the traditional system for the management of football stadiums to the electronic system so as to reduce problems for workers in football stadiums and football clubs and football audience.

The major technologies used in this work are basically web technologies (HTML, CSS, PHP, JavaScript, Bootstrap Materialize PHP framework) and relational database technology, MySQL. The web site developed is a collection of web pages, which are files written with HTML5, styled with CSS3. The web documents is rendered to users via the use of browsers e.g., UC Browser, Chrome, Internet Explorer, Firefox, Opera, Edge, Safari, Etc.

**HTML**

Each of our HTML documents is a sequence of elements [7]. An element consists of a start tag, content and a closing tag [7]. The following is an example of an HTML element in our web document to present the home page greetings if logged in:

<h1>

<center style="text-transform: capitalize;" >{{ auth.username }} welcome back to Welcome to Winsome Stadium</center>

</h1>

The major HTML features used were form and cascading style sheet. I also used forms to collect posted data from authors, comments from other users and during log in and when a user wants to update his/her account details. A form is a collection of data fields for the collection of input, e.g., user’s name, username, email and password and a submit button to send the collected data to a web document to act on the data. The form element has an ACTION which suppliers the address, better known as the universal resource locator (url) of this action document, and METHOD attributes which receives data using a POST HTTP method. The following is a fragment of an HTML form I used. It contains six data fields and the action document specified is register.php.

<form action="register.php" method="POST">

<div class="form-group">

<label for="email-reg" class="sr-only">Email</label>

<input type="email" class="form-control" id="email-reg" name="email-reg" placeholder="Email">

</div>

<div class="form-group col-sm-6 reg">

<label for="name" class="sr-only">First name</label>

<input type="text" class="form-control" id="name" name="name" placeholder="first name">

</div>

</form>

When the code above is rendered by the browser, it appears on the computer monitor as follows although some CSS files has been used to arrange and organize the form for a beautiful rendering.

Data received by our PHP document, register.php, from the form is loaded by register.php code into it corresponding fields in a record that is then stored in our relational database. I used a MySQL database management system to create and access the database.

The block of code below illustrates how the PHP document, register.php loads the data collected from collected fields, validates and store them in our relational database. This may not have the codes in details.

$request = $app->request;

$email = $request->post('email');

$username = $request->post('username');

$v = $app->validation;

$v->validate([

'email|Email' => [$email, 'required|email|uniqueEmail'],

'username|Username' => [$Username, 'required|max(20)'],

]);

if ($v->passes()) {

$identifier = $app->randomlib->generateString(128);

$user = $app->user->create([

'email' => $email,

'username' => $username,

]);

$user->permissions()->create(UserPermission::$defaults);

$app->mail->send('email/auth/register.php', ['user' => $user, 'identifier' => $identifier], function($message) use ($user) {

$message->to($user->email);

$message->subject('Thanks for registering.');

});

$app->flash('global', 'You have been registered.');

$app->response->redirect($app->urlFor('home'));

}

$app->render('auth/register.php', [

'errors' => $v->errors(),

'request' => $request,

]);

PHP and MySQL instructions are also used here in our code to open the database, establish a connection between it and our HTML code to insert data, retrieve data, update data, etc. for example:

<?php

'db' => [

'driver' => 'mysql',

'host' => '127.0.0.1',

'name' => 'educritique',

'username' => 'root',

'password' => '',

'charset' => 'utf8',

'collation' => 'utf8\_unicode\_ci',

'prefix' => ''

],

$app->configureMode($app->config('mode'), function() use ($app){

$app->config = Config::load(INC\_ROOT . "/app/config/{$app->mode}.php");

});

require 'database.php';

The overall presentation of information to the user by the browser is controlled in this site by means of cascading style sheet, CSS. CSS describes how HTML elements should be displayed or rendered in the browser. This is a collection (in a file name style.css) of styling rules or rule-sets. A CSS rule-set consists of a selector and a declaration block shown below [8]:

Declaration

Declaration

Selector

{color: #000; text-align: center;}

P

Value

Property

Value

Property

Fig 2.1 A CSS rule-set

The selector points to the element I want to style. The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon. [8]

A CSS declaration always ends with a semicolon and declaration blocks are surrounded by curly braces. The following rule has a selector <body> HTML element and four declarations. The second declaration has property, width, whose value is 100%.

body{

font: 13px Arial, Helvetica, sans-serif;

width: 100%;

margin: 0%;

color: #444;

}

**2.2 REVIEW OF RELATED LITERATURE**

The stadium management system is not a new project in computer science. There were people around the world that came across these projects, especially the electronic students. This section provides examples of relevant works which were found in the foreign universities.

According to Oparah and Oguike [3], they stated that computerization does not only involve computer technology consisting of only hardware and software but also the communication devices to interact and share data as well as transferring data/information from one location to another. Besides, computers can be used for keeping records and these records are always available whenever they are needed and the need of carrying office file from one place to another is eliminated; also there is no more loss of document or tempering with documents during transit as everything will be done electronically. Oparah and Oguike [3] also stated that computers have replaced manual technology because of its ability to process large volume of data or even handle complex work (processing cap ability) at a very high speed. It gives out accurate result at each time except when it is fed with incorrect data, Garbage-in-garbage-out. Hence, the need for computerization is certified.

Furthermore, French, states [4] that a file is a document stored in the computer individually by name and is organized in a particular way with a well-defined structure consisting of collection of records each of which are made up of fields. More so, Lucas, [6] commented that a typical organization has a large number of files, many of which may be stored on a computer device. These data are called machine readable because one can use computer to process them. Paper files on the other hand are much less accessible. A large organization related file as part of a database.

Oparah and Oguike, [3 ] defined a database as a single organization or collection of structured data stored with a minimum duplication of data items so as to provide consistent and controlled pool of data. The data is common to all users of the system, but is independent of programs that use the data. Databases are normally set up in order to meet the information needs of major parts of an organization. It is not possible to construct a database in a single operation; it is usually a built up section. During this process it is possible to:

i. Add new ―files of data.

ii. Add new fields to record already present in the database.

iii. Create relationship between the items in the database.

A database is required to be stored on large capacity direct access devices. The usual medium is the magnetic disk. For security purposes a copy of the database may be held on magnetic tape or disk. Although to the users, the database may appear as a collection of files, data in database is organized in a more complex way than data in conventional files. Database may be classified according to the approach taken to organize the database. The classes are relational, network, hierarchical and file inversions. But this project work discusses more on relational database. Data description must be standardized for this reasons a data description language (D.D.L) is provided which must be compared to the declarations and processing statement in a conventional programming language. Moreover, since complex files are processed in the database, a complex software system called Database Management System (DBMS) is required to construct, expands and maintain the database. It provides the controlled interface between the user and the data in the database. It also allocates storage of data. The Database Management System maintains indices so that any required data can be retrieved and separate items of data in the database can be cross-referenced. It also provides facilities for different types of file processing such as processing a complete file (serially or sequentially), processing required records (selective sequential or random) and retrieval of individual records. It has the function of providing security for the data in the database. Feingold, stressed [5] that users place more emphasis on meaningful things rather than the technicalities employed in information storage and retrieval. As records are entered and stored in the systems, their contents are automatically indexed by the software so that it will be very easy for the user to find every instance of selected record. A generally conclusion drawn from this is that, the provision of suitable information and storage retrieval system, in a manner suitable for the kind of data and the information need of the user or organization is very important. Also, the data to be processed by the computer must be collected. The process of data collection then involves getting the original data, converting it from one medium to another and finally getting it into the computer. Abudullahi defines data collection as the process involved in getting the data from its points of original.[1] Collection starts at the services of raw data and ends when valid data is within the computer in a form ready for processing. Therefore, in processing the stadium information record, data control measures should be involved. The following such as:

i. Manual controls

ii. Data collection controls

iii. Validation checks

iv. Batch controls

All these control measures have been taken to ensure that all processed data must preserve the integrity of maintained data, delete, corrected and all reprocess of error. Rajendran & Kamalanabhan, all state in their research that leadership and top management commitment is the most critical and crucial prerequisite for institutional success when implementing TQM. These researchers agree that leadership and top management commitment provide a focal point for the energies, hopes and aspirations of people in institutions where TQM is implemented.

**TYPES OF STADIUM**

1. CLASSICAL STADIUM: The first Greek stadium was long and narrow in the shape of a ―U‖ or a horseshoe. They were sometimes cut into the side of a hill as an ephidamus and at Olympiad, the site of the Olympic Games, which began there in the 8th century BCE.

2. MODERN STADIUMS: The stadium for the first modern Olympiad in Athens was a reconstruction of the ancient marble stadium built by Heroes Atticus on the site of an even earlier stadium in Athens. The Olympic since that time have provided a major focal point for the development of the modern stadium concept. The stands were partly roofed and the stadium seated more than 50.000 people.

3. DESIGN INNOVATIONS: The shapes of stadium have varied widely according to the variety of use for which they were built. Some are rectangular with curved corner, whereas others are elliptical or U shaped. The building of large stadium in 20th century has been greatly facilitated by the use of reinforced concrete, this material has made possible the construction of daring new designs that would previously have been impossible to sustain structurally.

4. MULTI PURPOSE STADIUM: Are the type of stadium designed to be easily used by multiple types of events, while any stadium could potentially host more than one type of sports or event, this concept usually refers to a specific design philosophy that stores multi functionality over specificity. It is used most commonly in Canada and the United States where the two most popular outdoor team sports grind iron football and baseball require radically different facilities. Football uses a rectangular field while baseball is played on a diamond. This require a particular design to accommodate both usually an oval.

FUNCTIONS OF A STADIUM

* Corporate events
* Conferences
* Trade shows
* Wedding and receptions
* Catering Services
* Hospitality and Corporate partnership

**CHAPTER 3**

**SYSTEM ANALYSIS AND DESIGN**

**3.0 Introduction**

System Development Life Cycle is splitting of software development work into distinct phases (or stages) containing activities with the intent of better planning and management [12]. It is often considered a subset of the systems development life cycle. The methodology may include the pre-definition of specific deliverables and artifacts that are created and completed by a project team to develop or maintain an application.

**3.0.1 Methodology**

The methodology used in this research project is the Object- Oriented Analysis and Design Methodology (OOADM). Object-Oriented analysis and design is a technical approach used in the analysis and design of an application or system through the application of the object oriented paradigm and concepts including visual modeling. This is applied throughout the development life cycle of the system, fostering better product quality and even encouraging stakeholder participation and communication.

**3.1 Description of the existing system**

The existing stadium management system is mainly processed manually. The allocation of space happens when the sports coordinator or coach will go to the stadium complex for them to allocate a space for him/her for an event. The space allocation officers will then check whether the space the person is seeking to reserve have been allocated for someone else. If the date and the arena are free, the allocation officer will then issue the client a form to fill that will bear the date and arena booked and finally makes payment.

**3.2 Analysis of the Proposed System**

The proposed system is a web based stadium management system where the users have right to choose date which is convenient for them to carry out their own event. The system have different subsection which specifies events/sports that normally takes place in a stadium like soccer, basketball, volleyball table tennis and any social or religious event.

Every tab/menu in the system bears a form for adding with a date picker which will make the date at which the event will be carried out. The manager , which is also the administrator have the administrative access to approve all the date booked by the client to make sure the date, the client picked were not reserved for another client and also to check if payment has been made on the reserved space.

The payment for the space allocation is done after the client must have chosen a date convenient for the event to take place. He/she will then print a reference form and make payment in the bank.

In the proposed system, the processes include;

1. Users choose a sport to reserve a space.
2. Users choose a date/time and title of their sporting event.
3. Online payment at the end of online registration process.
4. The software sorts the issue of the two persons reserving a space in the same day.

In other to specify what the propose system does, I used some tools like use case diagram and class diagram to analyze the proposed system. The proposed system/software will proffer solutions to the problems listed earlier in the existing manual system.

**3.2.1 Use case diagram**

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.

A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.

The use case diagram below shows the functionality of the Stadium management system, being linked to their different actors.

The actors are represented with stick figures i.e. (Admin and Users), the use cases which shows the different functionalities are represented with oval shapes, while the arrows show their relationships to each other.

The use-case diagram below shows that both the admin and the user shares similar functions, which include (logging into the system, view the entire sports arena, and book an event), but on the contrary, only the Administrator has the functions of (creating new users, editing sport arena details, search for arena).

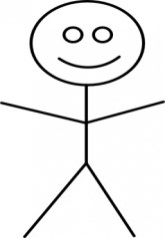
SPORTS STADIUM MANAGEMENT SYSTEM

Login

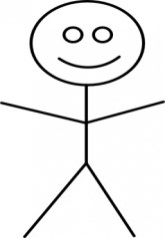
Add Event

Delete Event

SPACE



Approve Event



Edit Event

Search for Arena

Admin

User

**FIGURE 3.2.1: A USE-CASE DIAGRAM FOR THE SPORT STADIUM MANAGEMENT SYSTEM**

**3.2.2 Entity Relationship Diagram**

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information.

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract [data model](https://en.wikipedia.org/wiki/Data_modeling), that defines a data or information structure which can be implemented in a [database](https://en.wikipedia.org/wiki/Database), typically a [relational database](https://en.wikipedia.org/wiki/Relational_database).

Sports Arena

Events

Spectators

Customer

Companies

**Un-normalized Entity Relationship Diagram**

Events

Sports arena

Customers

Attendance

Maintenance

Spectators

Companies

**Normalized entity relationship diagram**

**3.2.3 Class diagram**

 Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) modeling. It is used for general [conceptual modeling](https://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](https://en.wikipedia.org/wiki/Programming_code). Class diagrams can also be used for [data modeling](https://en.wikipedia.org/wiki/Data_modeling). The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

Maintenance

Events

Arena

Int Event ID

Varchar name

Int Arena ID

Int Arena

Varchar Name

Int customer ID

Date Start Date

Int Company ID

Varchar Location

Int capacity

Int Arena ID

+ Add()

+Edit()

+Delete()

Date Finish Date

Int Amount

+ Add Arena ()

+ Delete Arena ()

+ Edit Arena ()

Int Paid Amount

Edit ()

Add ()

Delete ()

Attendance

Int event ID

Int Spectator ID

Date Date attended

Int Customer ID

Varchar Name

Varchar Address

Customers

Companies

Int company ID

Varchar Name

Varchar Address

Add()

Edit()

Delete()

+Add ()

+Edit ()

+Delete ()

Add Company()

Edit Company()

Delete Company()

Spectator

Int Spectator ID

Varchar Name

Int Phone No.

+Add()

+Edit()

+Delete()

**Fig 3.2.3 Class diagram of the stadium management system**

**3.3 Design of the Proposed System**

The proposed system is divided into different modules and subsystems. These modules perform a particular task. The whole modules and subsystems are integrated into one whole system in line with stated objectives.

Each end of the modules and subsystems are connected to the stadium management system. All the files, html documents, user forms, registration forms and all other components associated to the software will be joined together node to node. The design is intended to provide both manual and automated access to the database efficiently.

**3.3.1 Database Design**

To achieve excellent software that can serve a number of users, serious attention and consideration must be given to the database design to ensure little or no database redundancy. The database management used in this work is MYSQL. The required entities (tables) include the following:

* sports arena,
* events,
* customer,
* attendance
* maintenance
* companies
* spectators

Below is the structure of all the tables in stadium’s database:

**Table 1: Sport\_Arena**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Attributes | Null | Default | Extra |
| Sport arena id | Int | 11 |  | No | None | AUTO\_INCREMENT |
| Name | Varchar | 25 |  | Yes | Null |  |
| Location | Varchar | 25 |  | Yes | Null |  |
| Capacity | Varchar | 255 |  | Yes | Null |  |

Table 3.1

**Table 2: Spectator Event Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Attributes | Null | Default | Extra |
| Event id | Int | 11 |  | No | None | AUTO\_INCREMENT |
| Spectator id | Int | 25 |  | Yes | Null |  |
| Seat no. | Int | 20 |  | Yes | Null |  |
| Date | Int | 20 |  | Yes | Null |  |
| Time | Int | 20 |  | Yes | Null |  |

**Table 3: Customer\_Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Attributes | Null | Default | Extra |
| Customer id | Int | 11 |  | No | None | AUTO\_INCREMENT |
| Name | Int | 25 |  | Yes | Null |  |
| Email | Int | 20 |  | Yes | Null |  |
| Password | Varchar | 20 |  | Yes | Null |  |
| Date booked | Int | 20 |  | Yes | Null |  |
| Time booked | Int | 20 |  | Yes | Null |  |

**Table 4: Companies\_Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Attributes | Null | Default | Extra |
| Company id | Int | 11 |  | No | None | AUTO\_INCREMENT |
| Name | Int | 25 |  | Yes | Null |  |
| Address | Varchar | 20 |  |  |  |  |
| Phone no. | Int | 20 |  | Yes | Null |  |
|  |  |  |  |  |  |  |

**Table 5: Attendance\_Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Attributes | Null | Default | Extra |
| Event id | Int | 11 |  | No | None | AUTO\_INCREMENT |
| Spectator id | Int | 25 |  | Yes | Null |  |
| Date attended | Int | 20 |  | Yes | Null |  |
| Phone no. | Int | 20 |  | Yes | Null |  |
|  |  |  |  |  |  |  |

**Table 6: Maintenance\_Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Data type | Size | Attributes | Null | Default | Extra |
| Arena id | Int | 11 |  | No | None | AUTO\_INCREMENT |
| Company id | Int | 25 |  | Yes | Null |  |
| Start date | Int | 20 |  | Yes | Null |  |
| Finish date | Int | 20 |  | Yes | Null |  |
| Amount paid | Int | 255 |  | Yes | Null |  |

**3.3.2 Input Design**

All data in the new system’s database are collected over via the software’s front end. The input design makes the collection of data possible using strokes on the keyboard. One of the input form designs takes the design below;

|  |
| --- |
| REGISTRATION FORM  RESIDENTIAL ADDRESS  FULL NAME  EMAIL ADDRESS  DESIGNATION  PHONE NUMBER  PASSWORD  SUBMIT |

**Fig 3.3.2 Registration form**

**3.3.3 Output Design**

The output design is to present information provided by users, and provided by me the date/time booked intended for the users to the users. The output design contains information collected from the different database tables as specified earlier.

The output produced can be printed on a hard copy or viewed on the screen. The following are generated output by the system:

1. Users bio data

2. Users choice of sport and title data

3. Date and time booked for the event

4. Users payment confirmation

5. Etc.

**EVENT PRINTOUT**

|  |  |
| --- | --- |
| ARENA BOOKED |  |
| DATE |  |
| TIME |  |
| REFERENCE NUMBER |  |

**Fig 3.3.3 Output Design**

**3.3.4 System Architecture**

The user interface has several forms that allow the feeding of data into the database through the PHP and some query languages. The user interface (UI) is a crucial part of this software. The users interface provides users of this software the ability to interact, feed the software with data, and see retrieved data. So, every operation of the users ends within the user interface. This aspect is achieved with HTML, CSS, and JavaScript. At the user interface, users enter their details for registration and login and choose their type of sport, and book their chosen event using form HTML element.

The business logic is achieved with PHP. The business logic collects those entered data by the user, validates it and dumps it into the appropriate database table as specified in the code. Example, Users table in my software is meant to store data like username, email, password, etc. PHP with SQL query helps insert data to the corresponding database table and fields. MYSQL: This is where all users’ bio data, choice of sports and title, number of event/space booked on the calendar are being stored. This is the integrity of the software.

Admin Login

Search

Reserve

Approve Event

Delete Event

Edit Event

Add Event

Data base

Admin Logout

Select Sport

Add Event

User Logout

Print Event Form

**Fig 3.3.4 System Architecture**

**3.3.5 Algorithm Design**

An algorithm is a set of steps of operations to solve a problem performing calculation, data processing, and automated reasoning tasks [15]. An algorithm is an efficient method that can be expressed within finite amount of time and space.

An algorithm is the best way to represent the solution of a particular problem in a very simple and efficient way. If we have an algorithm for a specific problem, then we can implement it in any programming language, meaning that the algorithm is independent from any programming languages [15].

The important aspects of my algorithm design include creating an efficient algorithm to solve a problem in an efficient way using minimum time and space.

To solve a problem, different approaches can be followed. I took processing speed and memory consumption into consideration while designing my algorithm. Although I had to bear in mind that both time consumption and memory usage may not be optimized simultaneously because to run in lesser time, I needed more memory and to run in lesser memory I would need more time. So I had to pay sufficient time to the memory size so that time consumption can be enhanced. I had to ensure my loops are not recursive thereby reducing time in processing, I put aside some blocks that are not necessary, merged functions and units sometimes to simplify operations and the overall time needed to run the software.

ALGORITHM FOR LOGIN PAGES

Step 1: Start

Step 2: Declare a variable username and password

Step 3: Read variable username and password

Step 4: if username= password

Open a new page

Else

Display error message

Step 5: stop

**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

**4.0 Introduction**

This chapter is to show the implementation processes undertook to accomplish the stadium management system. It shows the brief implementation, development, system platform, IDE used, choice of programming language, implementation architecture showing a block diagram that identifies the various components of the software and their respective linkages and software testing.

**4.1 Choice of Development Environment.**

The choice of programming used is the top-down approach. This approach was used because the project arrangement was in linear format and the linear format is always from top to-down approach to achieve a goal.

The Programming languages that were used to achieve this project are:

1. Hypertext preprocessor (PHP)
2. HTML (Hypertext Markup Language)
3. CSS(Cascading Style Sheet)
4. MYSQL Database.

The Integrated Development Environments used are:

* Sublime Text 3.
* Wamp Server.

**4.1.1 User registration Page**

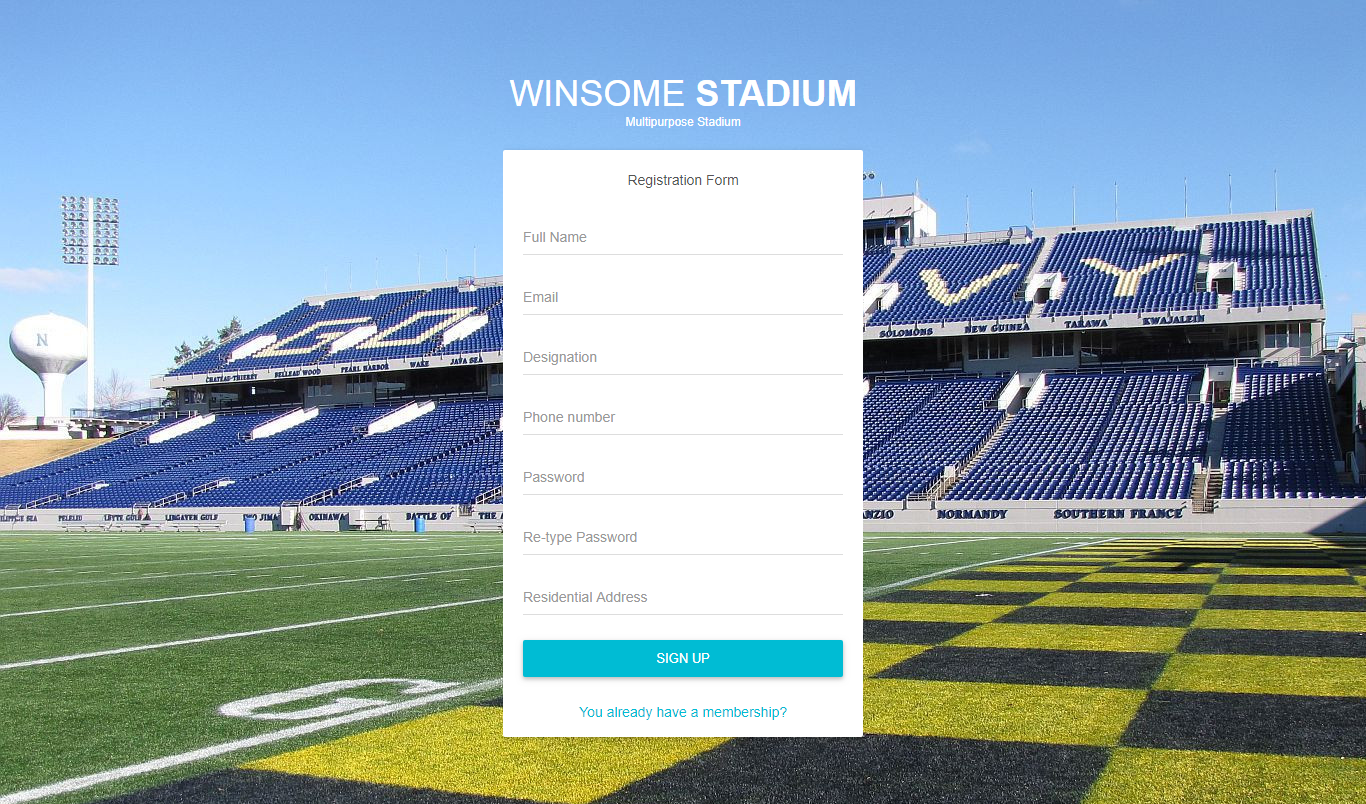
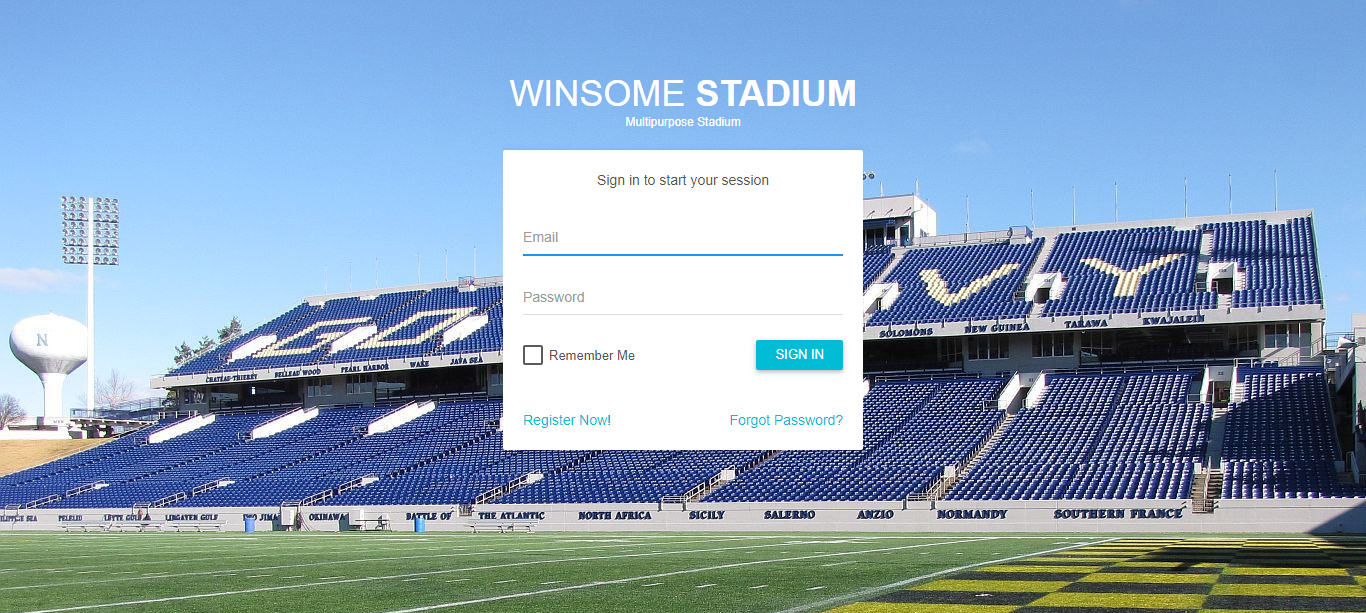


Figure 4.1: Screen shot of user registration page.

**4.1.2 User Login page**

****

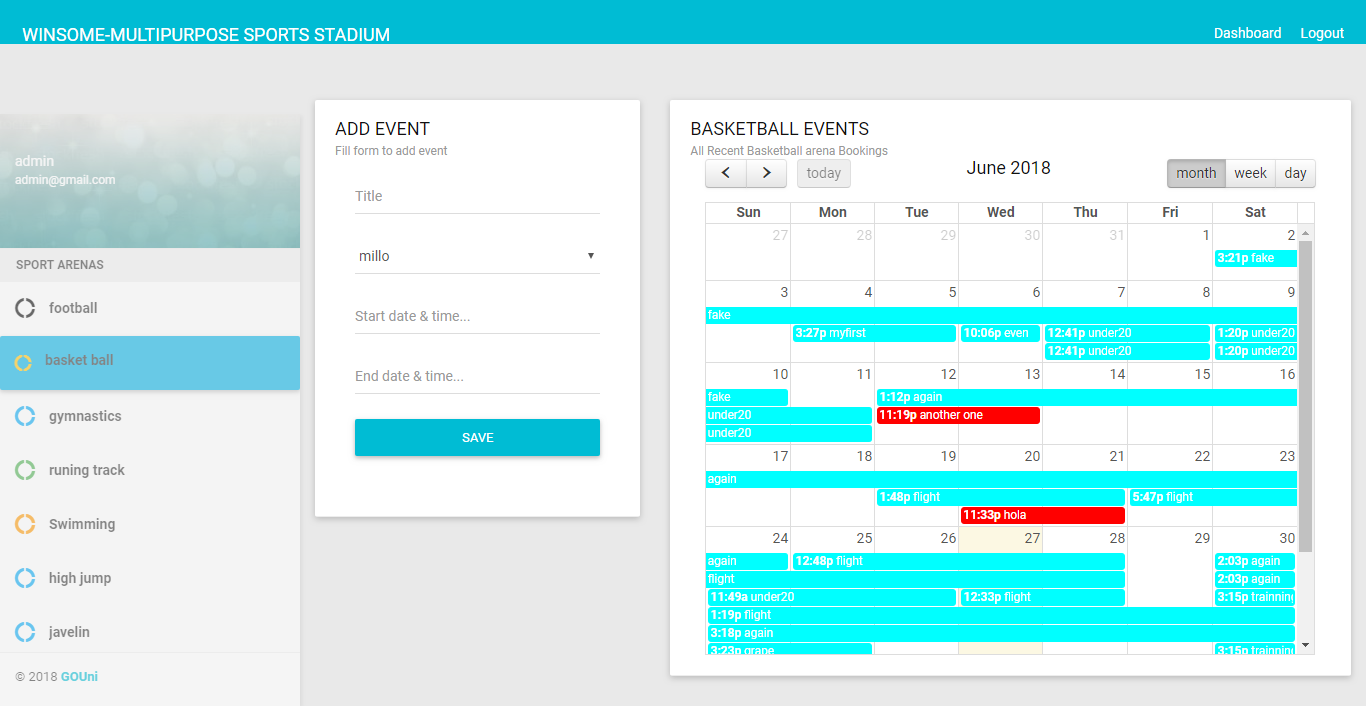
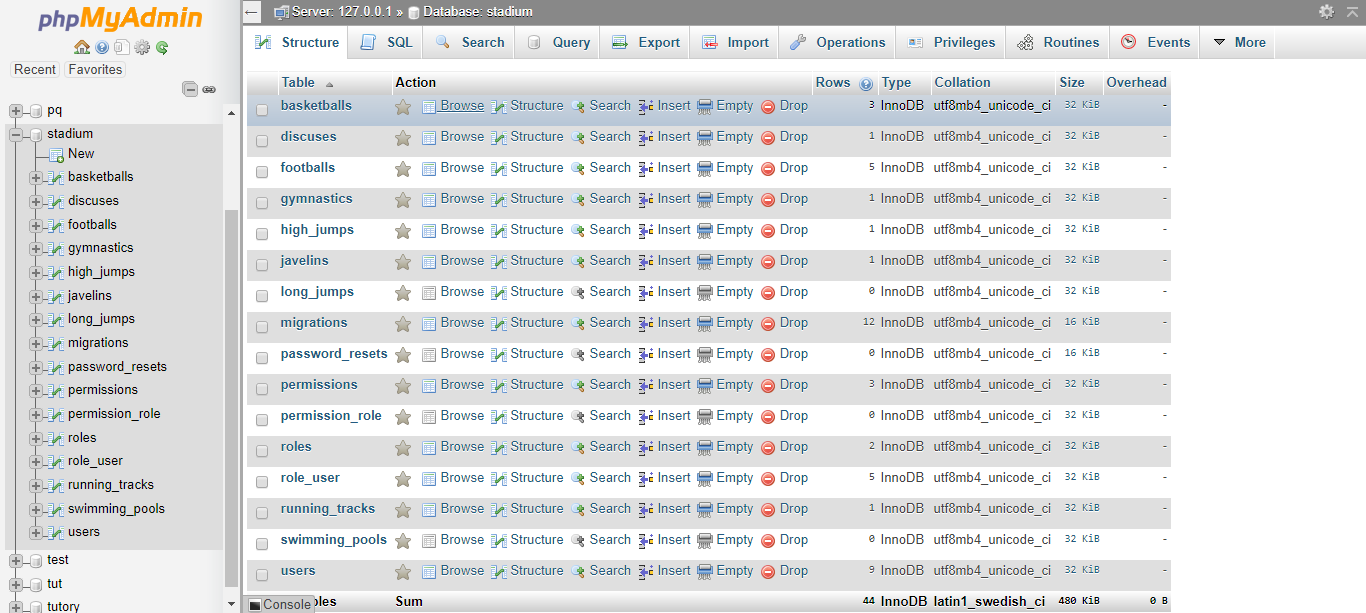
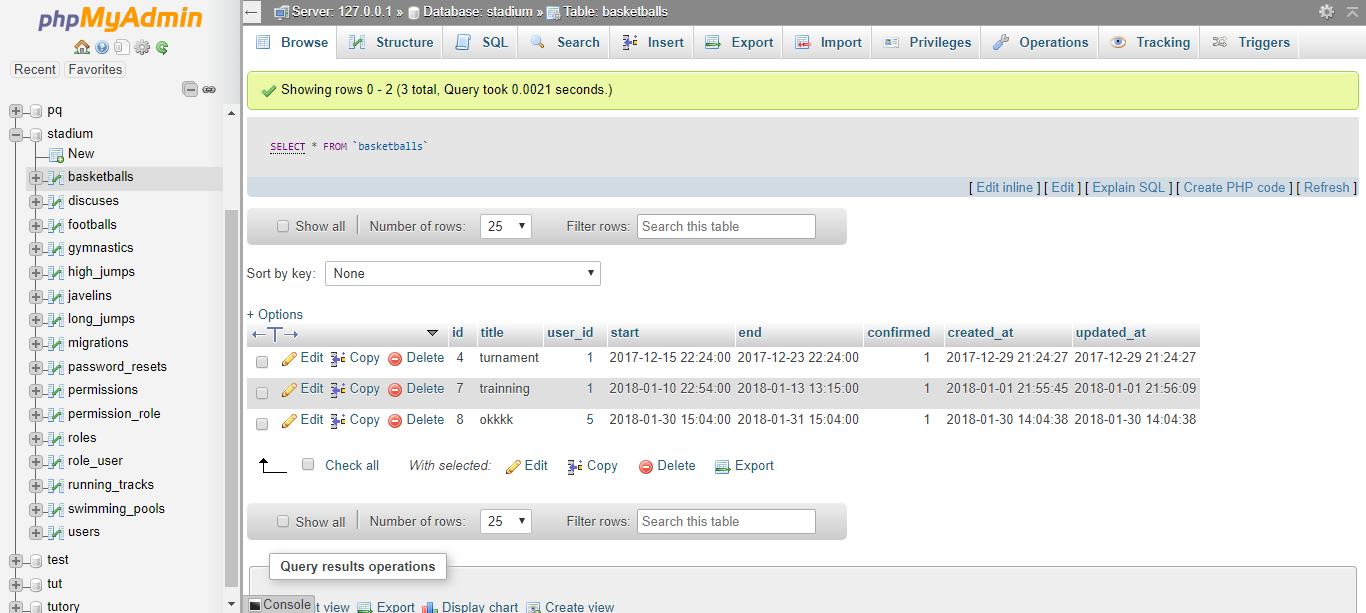
****

Figure 4.1.3: Screen shot of the Website Page.

****

**Fi11g 1**

****

**Fig 1 & 2: Database Design Table**

**Choice of Programming Language**

This system was built with basic HTML tags, PHP, CSS and MySQL language; they are open source program that allows modification to be done in future.

**HTML:** This is an acronym for Hyper Text Makeup Language; it is a language for describing the web pages. Html forms are elements that allow the user to enter information (like text fields, text area fields, drop-down menus etc.) in a form.

**CSS:** This is the acronym for cascading style sheets, these codes are used to style html elements, when html tags are written and viewed on a web browser, it doesn’t really appear in animated format, so we used CSS to style the html tags.

**PHP:** Hypertext preprocessor is a widely used scripting language that was originally designed for web development, to produce dynamic web pages; its codes can be embedded into html tags.

**MySQL:** My Structured Query Language; it uses a standard form of well-known SQL data language. Different data types were used in this project. It was used alongside with PHP to create and submit forms that are used in the software.

The developed system is packaged in a CD drive in a directory named stadium management system.

**4.2: Implementation Architecture**

Admin Login

Admin Dashboard

Admin Logout

Reserve

Approve Event

Delete Event

Edit Event

Add Event

Database

User Login

User Home Page

Select Sport

Add Event

User Logout

Print Event Form

**Fig 4.2 Implementation Architecture**

**4.3 Software Testing**

The system is tested at every stage of its development in other to be able to detect errors and remove them immediately. There are two phases of system testing:

Firstly, testing done during development phase. This testing includes:

* Syntax error testing: Syntax error testing is the method of error testing used to check all the code errors and structural errors during the development.
* Compatibility testing: This testing is used to test if all the languages used during the programming were compatible with each other during integration of modules.
* Logical testing: This includes all the argument stated during the programming. This will check whether the variable is accepted by the system or not.

Secondly, testing done by running the software on realistic data samples. This testing includes:

* Running the application in graphical user interface using a local server such as wamp server, xamp server lamp server that have local host which is using Apache as engine and MySQL as the database.
* Browser testing: This testing is used to test the appearance and shape of the software on the browser.

**4.3.1 Input Specification**

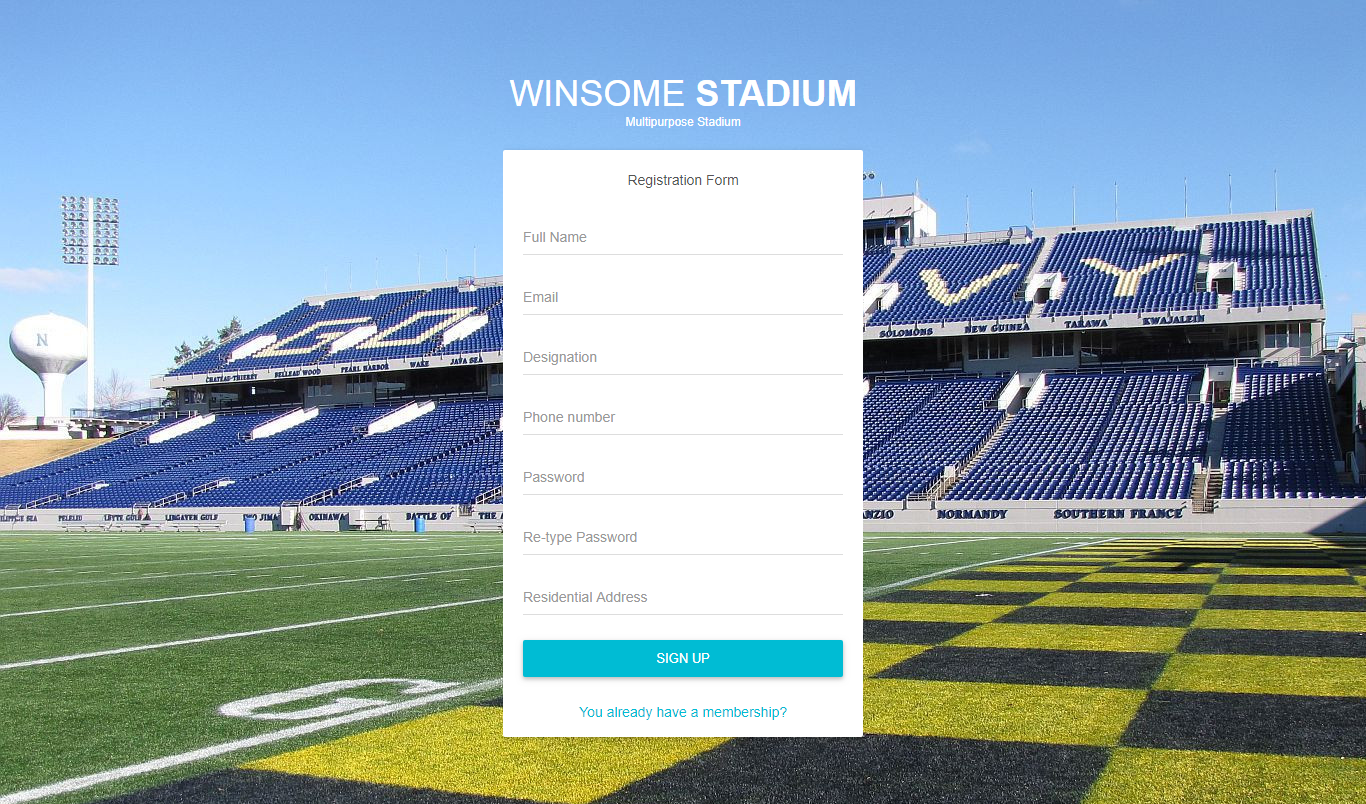
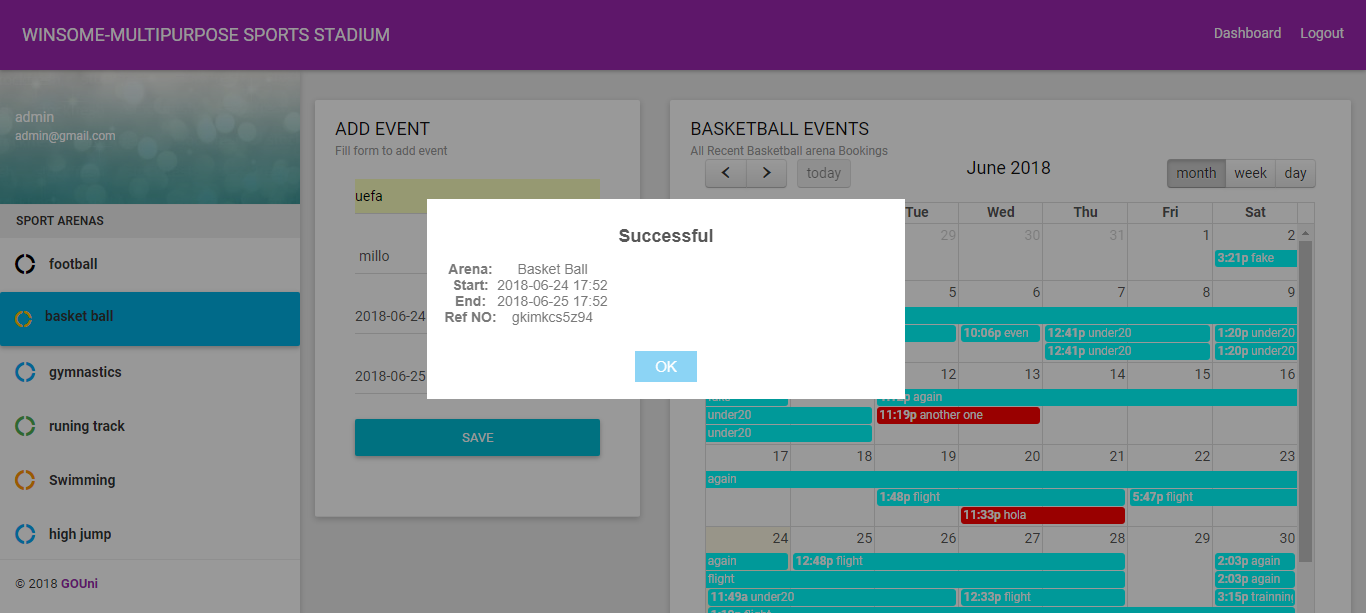


Figure 4.4: Screen shot of the input form,

**4.3.2 Output Specification**

  
Figure 4.5: Screenshot of the successful space allocation.

**4.4 Documentation**

**Firstly, Requirement documentation**

[Requirements](https://en.wikipedia.org/wiki/Requirement) documentation is the description of what this [software](https://en.wikipedia.org/wiki/Software) does. It is used to communicate how this software functions or how it is intended to operate. This software intends to allow users to book their choice of event at their convenient time and waits on the approval from the stadium events administrator.

The following are the requirements;

**4.4.1 Requirements**

* Php 5.5 + (server side)
* Mysql (server side)
* Browser (client side)
* Sublime text (developer)

**Software and tools**

* Laravel Framework
* Composer
* CSS
* HTML
* Apache server

**Database**

* Mysql 5.7

This software shall do the following;

* Register users
* Log users in
* Grant users permission as specified by the developer
* Shall allow users reserve a space at their convenience.
* Display relevant users information like name, designation, arena booked, and amount paid for the reservation, reference number as an output after making a reservation.

**4.4.1 User Manual**

1. Procedure on how to execute the program:
2. Boot the system to windows
3. Install sublime text and xamp server.
4. Launch sublime text and xamp server.
5. Install composer and laravel framework.
6. Open the index page with sublime text and run the software using a browser by locating the file using your server name local host and then point the index page.
7. The system has the following menu commands for the admin:

* Home: This is default for immediately logged in person.
* Add Event: The administrator also have the privilege to add an event that is introduced newly into the stadium complex.
* Delete Event. This menu is where the administrator deletes an event that its arena is under repair or no longer in existence.
* Approve Reservation: This is where the administrator approves the reservation made by the user before the space is finally booked.
* Reserve Event: This menu is where the administrators book a space for any user that request for it.

1. The system has the following menu commands for the user:

* Home: This is default for immediately logged in person.
* Reserve Space: This menu is where the user books a space.
* Print Event Form: This form is a confirmation that a user has reserved a space successfully.

**4.4.2 Source code listing**

A CD copy containing the entire source code has been attached herewith in the inside back cover of this report. All references to the code locations are contained in appendix A.

**CHAPTER 5**

**SUMMARY AND CONCLUSION**

**5.0 Summary**

It is justified that computer can be employed to process data related to stadium management. With this software, the management of the stadium can easily get achieve the following; total attendance in a match, accounting information staff records, maintaining security system, filling /access system updating /maintaining system for fans and staff (detection, update and insertion operation), records outputs in the form of soft copies and hard copies with respect to achieving this outline above, the paper narrows concentration down to the customers.

**5.2 Conclusion**

Computerization at large is ideal and effective towards solving today data processing problems with indent analysis of vast activities of computerization covered in this paper. Despite the limited time frame, the institution can discover it an easy task and processing records.

**5.3 Recommendation**

Therefore, computerization should be used in running the daily activities of data processing (as per records) in the stadium. In this order I recommend this new system of stadium management system to be used in when booking space for event.

**References**

[1] Abdullah, J.I (2004). Introduction to the computer, A management tool: Victory publisher Nigeria, No. 2 odor street Owerri.

[2] Abraham S., Henry F.K, & Surdarshan S. (2006). Database System Concepts:

(5th ed.). McGraw Hill Companies Inc, New York.

[3] Oparah, C.C & Oguike, O.E (2006). Management Information System, Shack Publisher Nigeria, Owerri.

[4] French, C.S. (2006). Computer Science, Book: Power publisher London.

[5] Feingold C. (2007). Introduction to data processing, (2nd ed.). USA, W.M.C. Brown Company publishers.

[6] Lucas, C.H.(2008). Information Technology for Management (6th ed.). New

York University, McGraw-Hill companies, New York.

[7]Refsnes Data. (2017, June). Introduction to HTML. W3schools. Norwegian. [Online]. Available: <Https://Www.W3schools.Com/Html/Html_Intro.Asp>

[8]Refsnes Data. (2017, June). CSSSyntax and Selectors. W3schools. Norwegian. [online]. Available: https://www.w3schools.com/css/css\_syntax.asp

[9]Refsnes data. (2017, June). PHP 5 tutorial. W3schools. Norwegian. [Online]. Available: https://www.w3schools.wom/php/

[10] Wikipedia. (2017, July 2). User Guide. Wikipedia Foundation. United States. [Online]. Available: https://en.wikipedia.org/wiki/user\_guide

[11] Wikipedia. (2017, June 30). Software Documentation. Wikipedia Foundation. United States. [Online]. Available: https://en.wikipedia.org/wiki/software\_documentation

[12] Wikipedia. (2017, July 13). Software Development Process. Wikipedia Foundation. United States. [Online]. Available: https://en.wikipedia.org/wiki/software\_development\_process

[13] Mark Otto, Jacob Thornton. (2011, August 19). CSS - Bootstrap. Bootstrap. United States. [Online]. Available: http://getbootstrap.com/css/

[14]PHP. (2017). WHAT IS PHP?.PHP Group. United States. [Online]. Available:http://php.net/manual/en/intro-whatis.php

[15]Tutorialspoint. (2017). DAA - Introduction. Tutorialspoint. Telangana. [Online]. Available: https://www.tutorialspoint.com/design\_and\_analysis\_of\_algorithms/design\_and\_analysis\_of\_algorithms\_introduction.htm

[16]Tutorialspoint. (2017). Software Testing Levels. Tutorialspoint. Telangana. [Online]. Available: https://www.tutorialspoint.com/software\_testing/software\_testing\_levels.html

[17]Tutorialspoint. (2017). MySQL – Introduction. Tutorialspoint. Telangana. [Online]. Available: https://www.tutorialspoint.com/mysql/mysql-introduction.htm

[18]Tutorialspoint. (2017). Software Requirements. Tutorialspoint. Telangana. [Online].Available: https://www.tutorialspoint.com/software\_engineering/software\_requirements.html

**APPENDIX**

**Source code: For loginpage**

@extends('layouts.app')

@section('title')

SignIn

@endsection

@section('content')

<form class="form-horizontal" method="POST" action="{{ route('login') }}">

{{ csrf\_field() }}

<div class="msg">Sign in to start your session</div>

<div class="input-group">

<div class="form-line{{ $errors->has('email') ? ' error' : '' }}">

<input id="email" type="email" placeholder="Email" class="form-control" name="email" value="{{ old('email') }}" required autofocus>

</div>

@if ($errors->has('email'))

<span class="help-info">

<strong>{{ $errors->first('email') }}</strong>

</span>

@endif

</div>

<div class="input-group">

<div class="form-line{{ $errors->has('password') ? ' error' : '' }}">

<input id="password" placeholder="Password" type="password" class="form-control" name="password" required>

</div>

@if ($errors->has('password'))

<span class="help-info">

<strong>{{ $errors->first('password') }}</strong>

</span>

@endif

</div>

<div class="row">

<div class="col-xs-8 p-t-5">

<input type="checkbox" name="remember" {{ old('remember') ? 'checked' : '' }} class="filled-in chk-col-cyan">

<label for="rememberme">Remember Me</label>

</div>

<div class="col-xs-4">

<button class="btn btn-block bg-cyan waves-effect" type="submit">SIGN IN</button>

</div>

</div>

<div class="row m-t-15 m-b--20">

<div class="col-xs-6">

<a href="{{ route('register') }}">Register Now!</a>

</div>

<div class="col-xs-6 align-right">

<a href="{{ route('password.request') }}">Forgot Password?</a>

</div>

</div>

</form>

@endsection

**Source Code: For registration code**

@extends('layouts.app')

@section('title')

SignUp

@endsection

@section('content')

<form class="form-horizontal" method="POST" action="{{ route('register') }}">

{{ csrf\_field() }}

<div class="msg">Registration Form</div>

<div class="input-group">

<div class="form-line{{ $errors->has('name') ? ' error' : '' }}">

<input id="name" type="name" placeholder="Full Name" class="form-control" name="name" value="{{ old('name') }}" required autofocus>

</div>

@if ($errors->has('name'))

<span class="help-info">

<strong>{{ $errors->first('name') }}</strong>

</span>

@endif

</div>

<div class="input-group">

<div class="form-line{{ $errors->has('email') ? ' error' : '' }}">

<input id="email" placeholder="Email" type="email" class="form-control" name="email" placeholder="Email" value="{{ old('email') }}" required>

</div>

@if ($errors->has('email'))

<span class="help-block">

<strong>{{ $errors->first('email') }}</strong>

</span>

@endif

</div>

<div class="input-group">

<div class="form-line">

<input id="designation" placeholder="Designation" type="text" class="form-control"name="designation" required>

</div>

</div>

<div class="input-group">

<div class="form-line">

<input id="phone" placeholder="Phone number" type="text" class="form-control" name="phone" required>

</div>

</div>

<div class="input-group">

<div class="form-line{{ $errors->has('password') ? ' error' : '' }}">

<input id="password" placeholder="Password" type="password" class="form-control" name="password" required>

</div>

@if ($errors->has('password'))

<span class="help-info">

<strong>{{ $errors->first('password') }}</strong>

</span>

@endif

</div>

<div class="input-group">

<div class="form-line">

<input id="password-confirm" placeholder="Re-type Password" type="password" class="form-control" placeholder="Confirm Password" name="password\_confirmation" required>

</div>

</div>

<div class="input-group">

<div class="form-line">

<input id="address" placeholder="Residential Address" type="text" class="form-control" name="address" required>

</div>

</div>

<button class="btn btn-block btn-lg bg-cyan waves-effect" type="submit">SIGN UP</button>

<div class="m-t-25 m-b--5 align-center">

<a href="{{route('login')}}">You already have a membership?</a>

</div>

</form>

@endsection

{{ csrf\_field() }}

</form>

</li>

@endguest

**Source Code: For basketball**

@extends('admin.skeleton')

@section('title')

Basketball

@endsection

@section('content')

<div class="row clearfix">

<div class="col-lg-4 col-md-4 col-sm-12 col-xs-12">

<div class="card">

<div class="header">

<h2>

ADD EVENT

<small>Fill form to add event</small>

</h2>

<div class="body">

<div class="row clearfix">

<div class="col-sm-12">

<form method="POST">

<input type="hidden" name="confirmed" id="confirmed" value=1>

<div class="form-group">

<div class="form-line">

<input type="text" class="form-control" placeholder="Title" id="title" name="title">

</div>

</div>

<div class="form-group">

<div class="form-line">

<select class="form-control show-tick" name="user\_id" id="user\_id">

@forelse ($users as $user)

<option value="{{ $user->id }}">{{ $user->name }}</option>

@empty

<option>No users</option>

@endforelse

</select>

</div>

</div>

<div class="form-group">

<div class="form-line">

<input type="text" class="datetimepicker form-control" name="start" id="start" placeholder="Start date & time...">

</div>

</div>

<div class="form-group">

<div class="form-line">

<input type="text" class="datetimepicker form-control" name="end" id="end" placeholder="End date & time...">

</div>

</div>

<button id="addevent" class="btn bg-cyan btn-block btn-lg waves-effect">SAVE</button>

</form>

</div>

</div>

</div>

</div>

</div>

</div>

<div class="col-lg-8 col-md-8 col-sm-12 col-xs-12">

<div class="card">

<div class="header">

<h2>

BASKETBALL EVENTS

<small>All Recent Basketball arena Bookings</small>

</h2>

<div class="container-fluid">

<div id='calendar'></div>

</div>

</div>

</div>

</div>

</div>

@endsection

@section('eventsScript')

@foreach($basketballEvents as $basketballEvent)

{

title : '{{ $basketballEvent->title }}',

start : '{{ $basketballEvent->start }}',

end : '{{ $basketballEvent->end }}',

id : '{{ $basketballEvent->id }}',

@if($basketballEvent->confirmed==1)

backgroundColor: 'cyan',

borderColor: 'cyan',

@else

backgroundColor: 'red',

borderColor: 'red',

@endif

},

@endforeach