

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/336288162>

On-Line Football Tickets Reservation System

Technical Report · June 2013

CITATIONS

0

READS

273

1 author:



[Hushang Jawzal](#)

Firat University

6 PUBLICATIONS 12 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Analyzing Breast Cancer using Thermography [View project](#)



On-Line Football Tickets Reservation System

*Project submitted in partial fulfilment of the
requirement for the award of the degree of
B.Sc. in Computer Science*

By
Hushang Jawzal

Under the esteemed guidance of
Mr. Qusay Idrees Sarhan
Department of Computer Science

2013



University of Duhok
Faculty of Science
Computer Science

1434 A.H

2713 K

2013 A.D

Certificate

This is to certify that the project entitled “On-line Football Tickets Reservation System” is being submitted by Hushang Jawzal, in partial fulfilment of the requirement for the award of the degree of BSc of Computer Science to the School Science.

The results presented in this documentation have been verified and are found to be satisfactory. The results embodied in this documentation have not been submitted to any other University for the award of any degree or diploma

Head of Department

Dr Ahmed A. Tahir

Supervisor

Mr. Qusay Idrees Sarhan

Acknowledgment

First of all, my deepest gratitude goes to Allah, the compassionate the merciful for everything.

I would like to express sincere thanks and gratitude to my supervisor . Mr. Qusay Idrees Sarhan for his assistance and fruitful suggestion through the period of preparing this project.

I would like also to express my thanks to staff members of Computer Science department.

Finally, special thanks to my family and to my nearest friends for their great and continuous support, patience and encouragement.

Contents

Chapter 1: INTRODUCTION

- 1.1 Introduction
 - 1.1.1 E-Ticketing
 - 1.1.2 History
 - 1.1.3 Advantages
 - 1.1.4 Disadvantage
- 1.2 Objective and scope
- 1.3 Methodology
- 1.4 Project Outlines

Chapter 2: Software Development Life Cycle

- 2.1 Introduction
- 2.2 Different phases of development process:
 - 2.2.1 Information gathering
 - 2.2.2 Requirement analysis
 - 2.2.3 Design
 - 2.2.4 Coding
 - 2.2.5 Testing
 - 2.2.6 Implementation & Maintenance

Chapter 3: Technologies Used

- 3.1. Introduction to .Net framework
 - 3.1.1 Features of the common language runtime
 - 3.1.2 .Net framework class library
 - 3.1.3 Client application development
- 3.2 ASP.NET
 - 3.2.1 Server Application Development
 - 3.2.2 Server-side managed code
 - 3.2.3 Active server pages.Net

3.3 C#.NET

3.3.1 Ado.net overview

3.4 sql server

3.4.1 Introduction.

3.4.2 Relational database.

3.4.3 Advantages of RDBMS.

3.4.4 Disadvantages of DBMS.

3.4.5 features of sql server (RDBMS).

Chapter 4: The proposed system

4.1 Introduction.

4.2 Requirements.

4.2.1 Software requirements:

4.2.2 Hardware requirements:

4.3 Design and specification.

4.5 Architecture.

4.6 Web Service Functionality

4.7 Software modelling.

4.8 Unified Modelling Language (UML)

4.8.1 Introduction

4.8.2 SEQUENCE DIAGRAMS

4.8.3 ACTIVITY DIAGRAM

4.9 Database Tables

Chapter 5: The experimental results

5.1 Introduction.

5.2 The software interface.

5.3 Scenario 1.

5.4 Scenario 2.

5.5 Project testing.

Chapter 6: conclusion and futures work

6.1 Conclusion.

6.1 Futures work.

List of Tables

Table 4.1: User

Table 4.2: Charge Card

Table 4.3: Payment

Table 4.4: Location

Table 4.5: Match info

Table 4.6: Seat details

Table 4.7: Reservation

Table 4.8: Admin

Table 5.1: Project Test

List of Figures

- Fig 2.1 Waterfall life cycle
- Fig 4.1 Architecture of proposed systems
- Fig 4.2 Database Design
- Fig 4.3 Sequence Diagram 1 -Admin
- Fig 4.4 Sequence Diagram 2 -User
- Fig 4.5 Activity Diagram Usecase:Login
- Fig 4.6 Activity Diagram Usecase:Payment
- Fig 4.7 Activity Diagram Usecase:Reservation
- Fig 4.8 Activity Diagram Usecase:Logout
- Fig 5.1 Snapshot of website
- Fig 5.2 User & Admin Log in
- Fig 5.3 User Registration
- Fig 5.4 Password Recovery
- Fig 5.5 User Account Information
- Fig 5.6 Charge Account
- Fig 5.7 Select Match
- Fig 5.8 Select Reservation Type
- Fig 5.9 Selection of Stadium Sections
- Fig 5.10 Payment Form
- Fig 5.11 Admin Registration
- Fig 5.12 Admin Account
- Fig 5.13 User Control



Fig 5.14 Add Events

Fig 5.15 Events Control

Fig 5.16 Admin Control

Fig 5.17 Count the Seat Number of the Match

Fig 5.18 Statistic of the Match

Fig 5.19 Contact Us

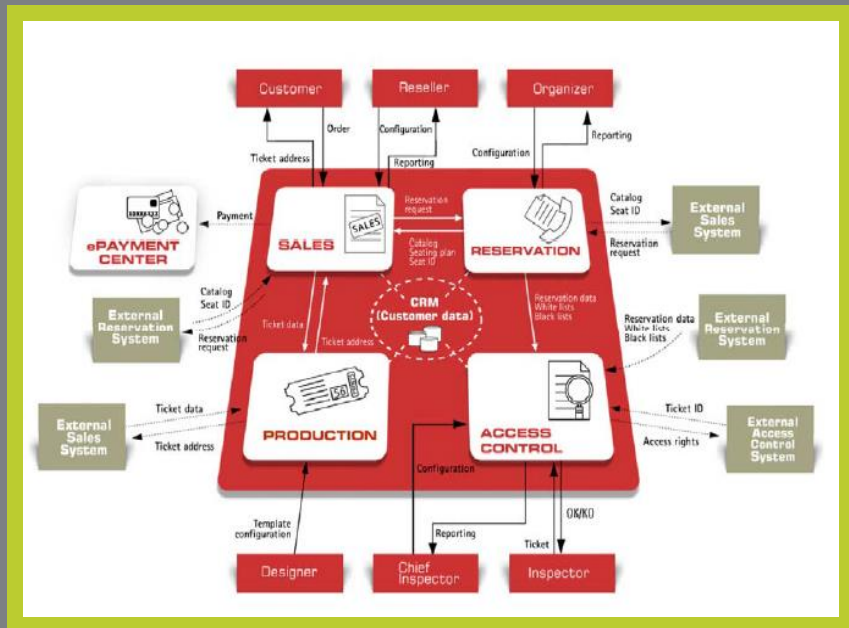
Fig 6.1 Application on iOS Smartphone.

Fig 6.2 Application on android Smartphone.

List of Abbreviations

No.	Abbreviation	Meaning
1	RDBMS	Relational database Management System
2	DBA	Data Base Administrator
3	ADMIN	Administrator
4	CLR	Common Language Runtime
5	NGWS	Next Generation Windows Services
6	SQL	Structured Query Language
7	ADO	Active Data Objects
8	LINQ	Language Integrated Query
9	DML	Data Manipulation Language
10	APPS	Applications
11	CLI	Common Language Infrastructure
12	GUI	Graphical User Interface
13	WPF	Windows Presentation Foundation
14	ASP	Active Server Page
15	IDE	Integrated Development Environment
16	PDA	Personal Digital Assistant
17	XML	Extensible Markup Language
18	WCF	Workflow Service Application
19	OLE DB	Object Linking and Embedding, Database

20	COM	Component Object Model
21	IP	Internet Protocol
22	OS	Operating System
23	PDA	personal digital assistant
24	SMTP	Simple Mail Transfer Protocol
25	HRD	Human Resource Development
26	O/RM	object - relational mapping
27	ERD	Entity Relationship Diagram
28	PK	Primary Key
29	FK	Foreign Key
30	OLE DB	Object Linking and Embedding, Database
31	UML	Unified Modelling Language
32	HTML	Hyper Text Mark-up Language
33	ASP	Active server pages



Chapter One

Introduction

This chapter give you an introduction on the “On-lineFootball Tickets Reservation System” and the need of them in these modern days.

1.1 Introduction

Ticketing is the process whereby customers can order, pay for, obtain and validate tickets from any location and at any time using mobile phones or other mobile handsets. Mobile tickets reduce the production and distribution costs connected with traditional paper-based ticketing channels and increase customer convenience by providing new and simple ways to purchase tickets. Agent ticketing is a prime example of horizontal telecommunication convergence. [1]

1.1.1 E-ticketing

We will try through this work, to understand and to explain what "E-Ticketing" means, the place of this system in Electronic Business, the different concrete applications of such a system in everyday life. Then we will explain schematically how it works. At the end, we will realize such a platform using a LAMP solution stack¹ to show in details how it works, we will present a method to generate a ticket which could be checked instantly at the turnstiles of an event. [2]

1.1.2 History

An electronic ticket or e-ticket is used to represent a purchase usually through a website or by phone. This form of ticket is rapidly replacing the old paper tickets.

Since its apparition in 1955, E-Ticket's use is growing very quickly in USA because of the considerable cost reductions it allows. E-Tickets are used particularly by airlines companies.

This is the case of American Airlines⁶ which sells now only E-Ticket for its national traffic and extends this service to international traffic since 2003, or charges \$50 for a traditional ticket.

One of its concurrent, Continental Airlines, is selling 95% of its flights by E- Ticket for its national traffic and 88% for its whole customers.

E- Ticket was introduced in Europe only in 1998 by Air France and the company is now selling 1 growing fast and in 2005, sales. [3]

1.1.3 Advantages

- Improved consumer convenience
- Increased revenue by increasing accessibility of tickets
- Reduced infrastructure costs.
- Reduced ticket printing/mailling cost

1.1.4 Disadvantages

- Can be forged.
- Many company phones block payment SMS messages.
- If the phone battery runs out, the mobile ticket is made unusable.

1.2 Objective

The purpose of our project is to make ticket reservation simple and easy and to provide a cheaper way of reserving the ticket. A user can reserve the tickets just by registering on the website and have payment card about the football company and the next step chose the match and clicking the submit button you will receive SMS an activation code about this match.

1.3 Methodology

Methodology has different meanings. As a branch of logic, it is about understanding the way that human beings knowledge is formed. It is also the combinations of best practices, procedures, rules and guidelines, in one word the methods, of the specific field of science and art by which professionals, specialists, and researchers can conduct their projects, research, development and study activities. [6]

The approach or methodology which we followed during the development of this project is described as below:

- We studied the real time system first and understood the workflow and involved technologies in the existing system. For ex: Initially, we talked about the stadium, the number of general Gates, emergency gate , ticket price , and period of time that requires for opening the gates before the match is started and VIP , press gate and technical aspects.
- After the problem definition, we moved on to the requirements gathering or requirements ,furnishing phase under which concrete requirements as to what exactly needs to be done was defined in terms of a set of objectives. For ex: After studying the existing systems, we have problem with online purchase on ticket to success this projects.
- Hence, after defining and understanding the problem within the existing systems, we defined the objectives as to what precisely would we plan to achieve in the application that we build which should also act as a solution to the devised problems.
- After that, we analysed the domain in which our application will be more secure and fast and easy to the user.

- Thereafter, we moved to the design phase under which keeping the overall objectives into consideration, we designed the system in such a manner so that it is user-friendly and brings enjoyable experience to the user while ensuring the security, which is the prime aspect of this project. In the design phase, we firstly designed the overall architecture or calling sequence of the application and then, in the low level designing, we framed the interfaces of the individual modules.
- After design phase, we moved on to the coding phase keeping our objective into consideration and also ensuring that the coding standards like effective APIs, cohesion, etc. should be followed. During this phase, we also identified the classes, functions and constants to be used in the application and the whole logic formulation was done during this phase.
- After coding, we moved on to the installation and deployment phase under which we installed the application in an appropriate environment and also tested it during this period. During the testing phase, we also simultaneously debugged the system in order to remove any bugs or errors encountered. Hence, the installation and deployment phase also encompassed the sub-phases like testing and debugging.
- Finally, we tested the whole system once in order to check its compliance with the objectives and requirements that we furnished in the beginning.

1.5 Project outlines

This documentation contain six chapter , and each chapter present the required information about a part of the system ,and by reading the six chapter all aspect that include in this session well be identified .the chapter of this organized as follows:

Chapter 1: Introduction

This chapter give you an introduction on the “On-line Football Tickets Reservation System” and the need of them in these modern days.

Chapter 2: Software Development Life Cycle

In this chapter we explain Software Development Life Cycle (SDLC) and Different Phases of the Development Process, and Reliability of my software system.

Chapter 3: Technologies Used

In this chapter, all technologies, tools, and packages that used to achieve this proposed system are explained also this chapter gives a brief introduction to each technologies used, the advantages and functionalities of each one of them.

Chapter 4: The Proposed System

This chapter presents the proposed system for Football tickets agent to make ticket reservation in simple and cheaper way.

Chapter 5: The Experimental Results

In this chapter, the experimental result and analysis of the developed system are shown and also give a discussion on different types of results or cases that got from output of our system.

Chapter 6: Conclusion and Future Scope

This chapter summarize the overall work and indicating possible future work of the proposed system that can be down to improve the quality of the system in general.

2.1 Introduction

The basic objective of software engineering is to develop methods and procedures for software development that can scale up for large systems and that can be used to consistently produce high quality software at low cost and with a small cycle time. That is, the key objectives are consistency, low cost, high quality, small cycle time, and scalability.

The basic approach that software engineering takes is to separate the development process from the software. The premise is that the development process controls the quality, scalability, consistency, and productivity. Hence to satisfy the objectives, one must focus on the development process. Design of proper development process and their control is the primary goal of the software engineering. It is this focus on the process that distinguishes it from most other computing disciplines. Most other computing disciplines focus on some type of the product-algorithms, operating systems, databases etc. while software engineering focuses on the process for producing products. To better manage the development process and to achieve consistency, it is essential that the software development be done in phases

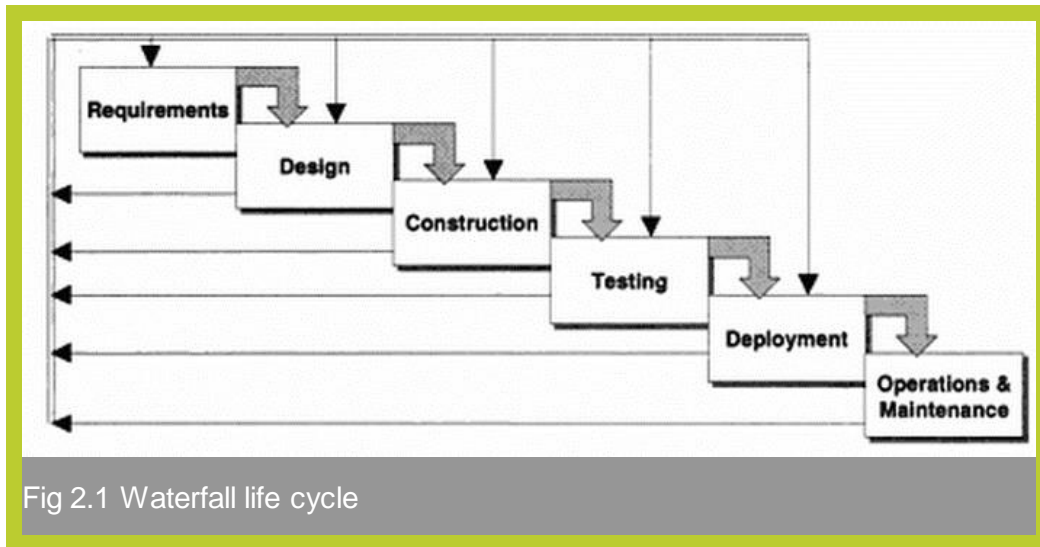
We have developed this system after duly spending time on each software development phase individually and freezing the status before we move on to the next phase. We have used Linear Sequential model in this application under which: [6]

1. In the analysis phase, we have attempted to understand the system completely in terms of its objectives & the problems faced. All the objectives were further subdivided into a set of smaller objectives which in turn were subdivided into the set of actions. The analysis also gave us an overview, about the individual expectations from each function and the challenges faced by that function due to which the ultimate objective is not achieved.

2. Before we moved to the designing phase, the objectives & challenges were clearly understood and we actually converted the set of objectives & actions into modules. The modules were designed in terms of their input, output, flow of information, storage of information & communication amongst each other, with the user and with the system. All the data objects were carefully designed and classified in terms of their inputs & outputs

3. After the analysis & design phases were over, we moved on to coding phase, where the implementation of tasks or functions on-paper were actualized. This was the phase where we actually became the use to have the look & feel of the application, where we actually thought from the user's perspective & company's perspective keeping all the objectives & challenges into consideration.

4. Then came the testing phase in which after developing the complete system, we rigorously tested it using all the testing types and checked every characteristic & attribute of the application of whether it coincides and is inline with the company's objectives and user's comfort.



2.2 Different phases of development process:

2.2.1 Information Gathering:

In this phase information is gathered from the customer, if some information are missing then we need to go to customer and take the required information. Mainly this is done by the Business Analyst as they are the one who are completely involved in Software Development life Cycle. The Document Prepared during this phase is: Business Requirement Specification (BRS) or Customer Required Specification (CRS) Or User Requirement Specification (URS).

4.2.2 Requirement Analysis

The analysis of the system was done rigorously because this is such a phase where all the loopholes had to be discovered keeping company's objectives & challenges in mind. We performed the analysis in 2 parts i.e. Feasibility Analysis & Requirements Analysis. [6]

➤ Feasibility Analysis

1. We studied the whole system & its objectives. Calculated the total time & resources incurred on every function being done manually.
2. Bifurcated the complete system into a list of functions & the users who operate on them.
3. Further subdivided all the functions into a list or source of requirements inputs & clearly defined the output/expectation from each function.
4. The interaction, communication & dependency of all the functions between each other were carefully analysed in terms of sequence & information.
5. The source & flow of the information was determined & how would it be processed & used was considered.

6. Finally, we visualized the complete system with automated functions & compared the total time & resources being incurred to check the feasibility & see whether it is fulfilling all the necessary objectives.

➤ Requirement Analysis

1. This was a subset of feasibility analysis in which we defined a set of objectives for the Complete system after thoroughly analysing it.
2. All the objectives were further subdivided into a set of function(s).
3. The inputs(s) required by each function & the expected output(s)/behaviour was/were clearly defined.
4. The source of information input to every function was determined & its corresponding processing, usage & storage were also taken into account.
5. After this the independency & communication was finalized.

2.2.3 Design:

➤ Architectural Design:

Under architectural design, after defining the whole system into set of objectives further subdividing them into function, we defined the basic dependency & communication between them.

This means that all the prime functions, their required inputs, expected output/behavior & interdependency between other functions were clearly defined. The corresponding interfaces for the user for each function were designed to ensure user-friendliness. We actually addressed the system-level problems here and made a conscious effort to build a robust design which can result in an effective communication within itself and with the system in terms of raw data or processed information. All the primary database design for data storage was also done in this phase.

➤ Detailed Design:

In this phase, we further subdivided every function into a set of modules & defined required inputs & expected behavior for each of them. All the minute correlations, interdependencies, communication between the modules were clearly defined. The source, usage & processing of data for every module was carefully done. The database design was also normalized at this stage to ensure that, the data is efficiently stored & retrieved. Detailed design helped us to exactly concretize every problem into inputs & outputs and visualize them in terms of their communication with each other. We focused on interdependency & interoperability between the broken modules here. It was this design phase where the factors like user-friendliness, ease of use, scalability and self-explanation of interfaces & outputs were actually realized. For all the modules, the placement of controls, passing of information, communication of different interfaces, user messages, data transfer to databases was defined.

2.2.4 Coding:

The goal of the coding phase is to translate the design of the system into code in a given programming language. Hence during coding, the focus should be on developing programs that are easy to read and understand, and not simply on developing programs that are easy to write.

2.2.5 Testing:

The testing is the major quality control measure used during software function is to detected error in software .Testing not only uncover errors introduced during coding, but also errors introduced during the previous phases. Thus, the goal of the testing is to uncover requirement, design and coding errors in the programs. Therefore, different levels of testing are used. Testing is an extremely critical and time consuming activity. It requires proper planning of the overall testing process. The output of the testing phase is the test report and the error report. Test report contains the set of test cases and the result of executing the code with these test cases. The error report describes the error encountered and the action taken to remove the errors.

We have used the below mentioned testing techniques to thoroughly test our application.

2.2.6 Implementation & Maintenance:

Software maintenance is a task that every development group has to face, when the software is delivered to the customer's site, installed and is made operational. Software maintenance is a very broad activity that includes error correction, enhancement of capabilities, deletion of obsolete capabilities and optimization [16].

- **Implementation:** Once testing is finished and the software is proven good for implementation, it is released to the public or will be removed from beta version. It's expected that on first days, developers will face serious challenge of fixing different bugs as they are discovered one by one by different users. The main difference of the implementation stage to the testing is the amount of bugs expected. Since they are implemented to the public or a wider audience, it's already expected that it should work properly.
- **Maintenance:** Software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment.

Software maintenance can best be described as a 6-step process mentioned below:

1. The implementation processes contains software preparation and transition activities, such as the conception and creation of the maintenance plan, the preparation for handling problems identified during development, and the follow-up on product configuration management.

2. The problem and modification analysis process, which is executed once" the application has become the responsibility of the maintenance group. The maintenance programmer must analyse each request, confirm it (by reproducing the situation) and check its validity, investigate it and propose a solution, document the request and the solution proposal, and, finally, obtain all the required authorizations to apply the modifications.
3. The process considering the implementation of the modification itself.
4. The process acceptance of the modification, by confirming the modified work with the individual who submitted the request in order to make sure the modification provided a solution.
5. The migration process (platform migration, for example) is exceptional, and is not part of daily maintenance tasks. If the software must be ported to another platform without any change in functionality, this process will be used and a maintenance project team is likely to be assigned to this task.
6. Finally, the last maintenance process, also an event which does not occur on a daily basis, is the retirement of a piece of software.



Chapter Three

Technologies Used

In this chapter, all technologies, tools, and packages that used to achieve this proposed system are explained also this chapter gives a brief introduction to each technologies used, the advantages and functionalities of each one of them.

3.1 Introduction To .Net Framework

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and Remoting , while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features.

The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, But with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

3.1.1 Features of the Common Language Runtime

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers.

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure

Enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

3.1.2 .Net Framework Class Library

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications.
- Scripted or hosted applications.
- Windows GUI applications (Windows Forms).
- ASP.NET applications.
- XML Web services.
- Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

3.1.3 Client Application Development

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs. For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your Applications can implement the features of a local application while being deployed like a Web page.

3.2 ASP.NET

3.2.1 Server Application Development

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code [12].

3.2.2 Server-Side Managed Code

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no

Longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description

Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS [15].

3.2.3 Active Server Pages.Net

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.
- **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
- **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
- **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
- **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to

deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.

- **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
- **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.
- **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

Language Support

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

What Is Asp.Net Web Forms?

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

- The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
- The ability for developers to cleanly structure their page logic in an orderly fashion.
- The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests .aspx Resources, the ASP.NET runtime parses and compiles the target file into a .NET Framework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form post back to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for `<% %>` code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

Code-Behind Web Forms

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method--enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

Introduction to Asp.Net Server Controls

In addition to (or instead of) using `<% %>` code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attributes value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp: adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both up level and down level HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

3.3 C#.NET

3.3.1 Ado.Net Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the **DataSet** as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.[9]

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections**. For connection to and managing transactions against a database.
- **Commands**. For issuing SQL commands against a database.

- **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
 - **DataSets**. For storing, remoting and programming against flat data, XML data and relational data.
 - **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.
- When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB.NET Data Provide (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

Connections

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and result sets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

Commands

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

DataReaders

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results of a search list in a web page.

Datasets and Data Adapters

DataSets

the **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made

to the **DataSet** they can be tracked and verified before updating the source data. The **Get Changes** method of the **DataSet** object actually creates a second **DataSet** that contains only

the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via Web Services. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion [13].

Data Adapters (OLEDB/SQL)

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the **Insert, Update or Delete** command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the **Insert, Update, And Delete** commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships

3.4 SQL Server

3.4.1 Introduction

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields. [11]

SQL Server Tables

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

Primary Key

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key.

The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

3.4.2 Relational Database

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the tables and enables you to define relationships between the tables. [10]

Foreign Key

When a field in one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

Referential Integrity

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

Data Abstraction

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

Physical level: This is the lowest level of abstraction at which one describes how the data are actually stored.

Conceptual Level: At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

View level: This is the highest level of abstraction at which one describes only part of the database.

3.4.3 Advantages of RDBMS

- Redundancy can be avoided.
- Inconsistency can be eliminated.
- Data can be shared.
- Standards can be enforced.
- Security restrictions can be applied.
- Integrity can be maintained.
- Conflicting requirements can be balanced.
- Data independence can be achieved.

3.4.4 Disadvantages of DBMS

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered [14].

3.4.5 Features of SQL Server (RDBMS)

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today's most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous

DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.

SQL Server RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application [14].

SQL Server with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

- The row level lock manager

Enterprise Wide Data Sharing

The unrivalled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

Portability

SQL Server is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database server platform that meets the system requirements.

OPEN SYSTEMS

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server's open architecture integrates SQL Server and non –SQL Server DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server's Open architecture provides transparent access to data from other relational database and even non-relational database.

Distributed Data Sharing

SQL Server's networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single

SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

Unmatched Performance

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.



Chapter Four

The proposed system

This chapter presents the proposed system for Football tickets agent to make ticket reservation in simple and cheaper way.

4.1 Introduction

Mobile ticketing offers a new channel of ticket distribution for football chains to the customer. Ticket-Solutions proposes a convenient alternative to traditional methods of purchasing football tickets. The benefits of this system apply to both the football and the customer. It is mutually beneficial relationship [4].

4.2 Requirements

The following system requirements are mandatory to implement our proposed system:

4.2.1 Software Requirements:

- Operating system: Microsoft Windows XP or later.
- Microsoft Visual Studio 2012 Ultimate.
- SQL Server 2012.

4.2.2 Hardware requirements:

- Processor Pentium 4 or higher.
- RAM- 1GB or higher.
- HDD-10GB or larger.
- Mobile phone to receive the SMS.

4.3 Design and Specification

This part present the design and specification that have been developed for our proposed system as follows: A customer will be having the certain requirements that must be provided by the system. Hence at the initial stage of the software (website) development, a requirement analysis is performed to identify the needs of the customer. Our software is ticket reservation software which can be used in stadium. To make it applicable in any of the stadium, we have taken into consideration the following requirements:

- Each stadium can have any number of classes.
- The price of the ticket is based on the class.
- Each class can have any number of rows.

There are two types of users of the system. One is the customer and the other is the administrator.

The **Customer** should be able to:

- Register with his username and credit card number (optional).
- Reserve the ticket.
- Make payment either through credit card.
- Unregister.

The **Administrator** should be able to

- Register himself by giving his details along with the user name and password
- Login into the system.
- Update his information.
- Change password.
- View seat status.
- Delete the old records.

4.4Architecture

Are several modules in our main system .The modules of this proposed system there are organized as in fig 4.1 .

Stage One:

The customer visits the Football website and has two options. He can become a member by registering his details once or he can remain a non-member but will have to enter his details each time he wants to purchase a ticket.

Stage Two:

There are three options for payment:

- They are via credit card (visa, master)
- Mobile payments.
- Credit card payment is via the football website and processed by the credit card companies.

Stage Three:

Delivery of tickets to mobile phones can be done in a variety of ways:

- Text messaging (SMS).
- Picture messaging (MMS) - usually uses a barcode.
- Dedicated Mobile application.

Stage Four:

When the customer goes to the stadium, an operator scans their mobile barcode. This will validate their mobile ticket.

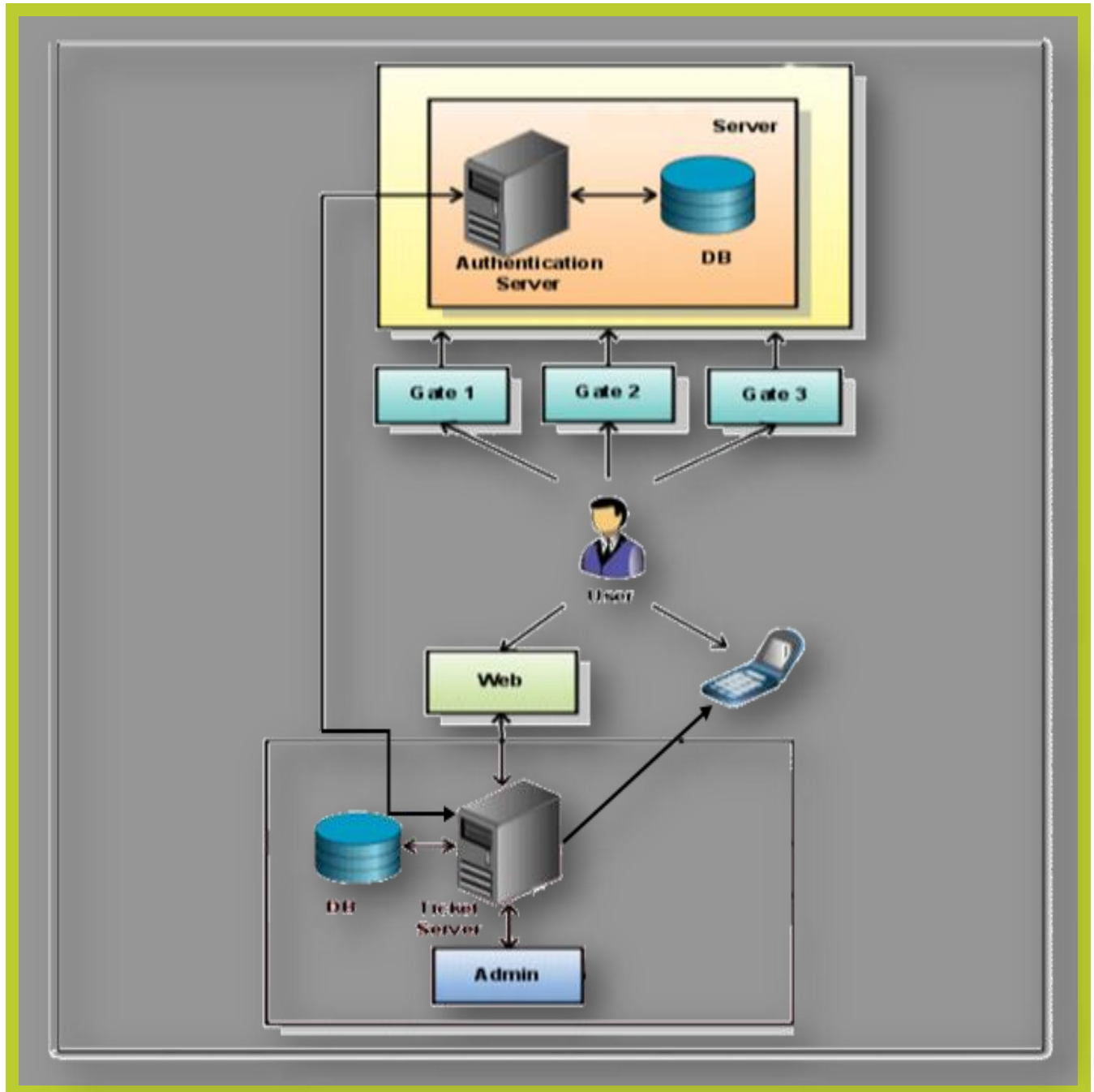


Fig 4.1 Architecture of proposed systems

4.5 Web Service Functionality

AJAX- web service technology has been developed to allow the web service client to get an updated image at regular intervals from the server side where the system works, where the client can dynamically make a call to the developed web service and get an image as a response of that call [7].

4.6 Software modelling

Software modelling is the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Modelling is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analysed, system modelling is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system -one that will be difficult to test, one whose quality cannot be assessed until the last stage. During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented.

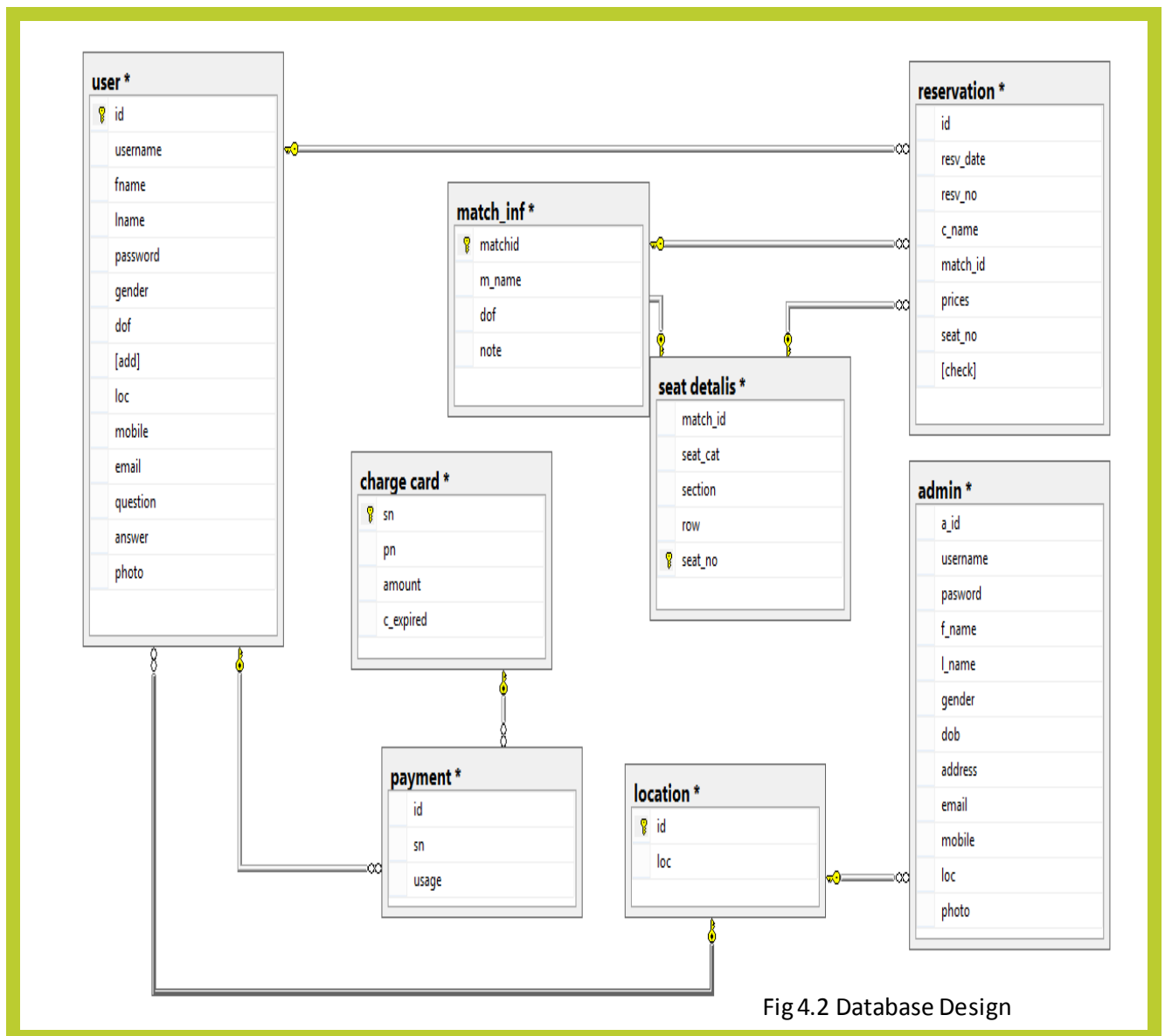


Fig 4.2 Database Design

Fig 4.2 is a database diagram for Ticketing Online System .the table user contain [id] primary key and have two foreign key ([add] reference for table location and [id] references for the table payment. The table match_inf. have primary key [matched]. The table seat details contain one primary key [seat _no] and one foreign key [match id] from the table match info .the table reservation contain three references key ([c _name] references for the table user, [seat _no] from seat details, [match _id] from match info.]. The table charge card contain one primary key [sn] And foreign key [sn]. The table location contain primary key [id].The table payment have two foreign key [id] from the table user and [sn] from the table charge card.

4.7 Unified Modelling Language (UML)

4.7.1 Introduction

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows. [8]

- User Model View
 - ✓ This view represents the system from the users' perspective.
 - ✓ The analysis representation describes a usage scenario from the end-users perspective.
- Structural model view
 - ✓ In this model the data and functionality are arrived from inside the system.
 - ✓ This model view models the static structures.
- Behavioural Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.
- Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.
- Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

- ✓ UML Analysis modelling, this focuses on the user model and structural model views of the system.
- ✓ UML design modelling, which focuses on the behavioral modelling, implementation modelling and environmental model views.

Use case Diagrams represent the functionality of the system from a user's point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of

the system. Use cases focus on the behavior of the system from external point of view. Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer ...etc., or another system like central database.

4.7.2 Sequence Diagrams

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

Sequence Diagram 1 Admin

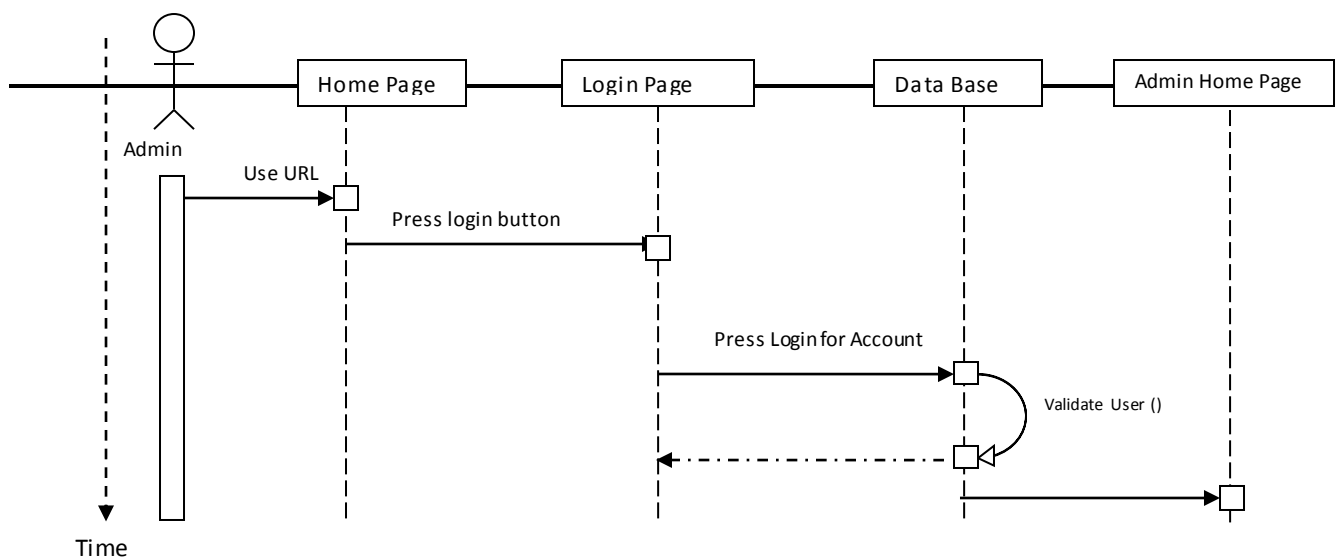


Fig 4.3 Sequence Diagram 1 -Admin

Sequence Diagram 2 User

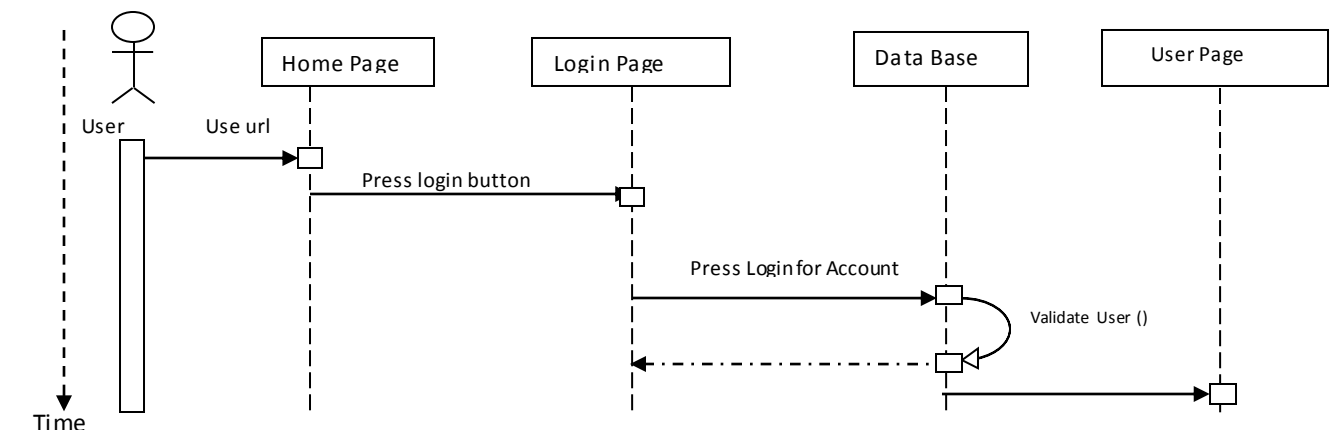


Fig 4.4 Sequence Diagram 2 -User

4.7.3 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Usecase: Login

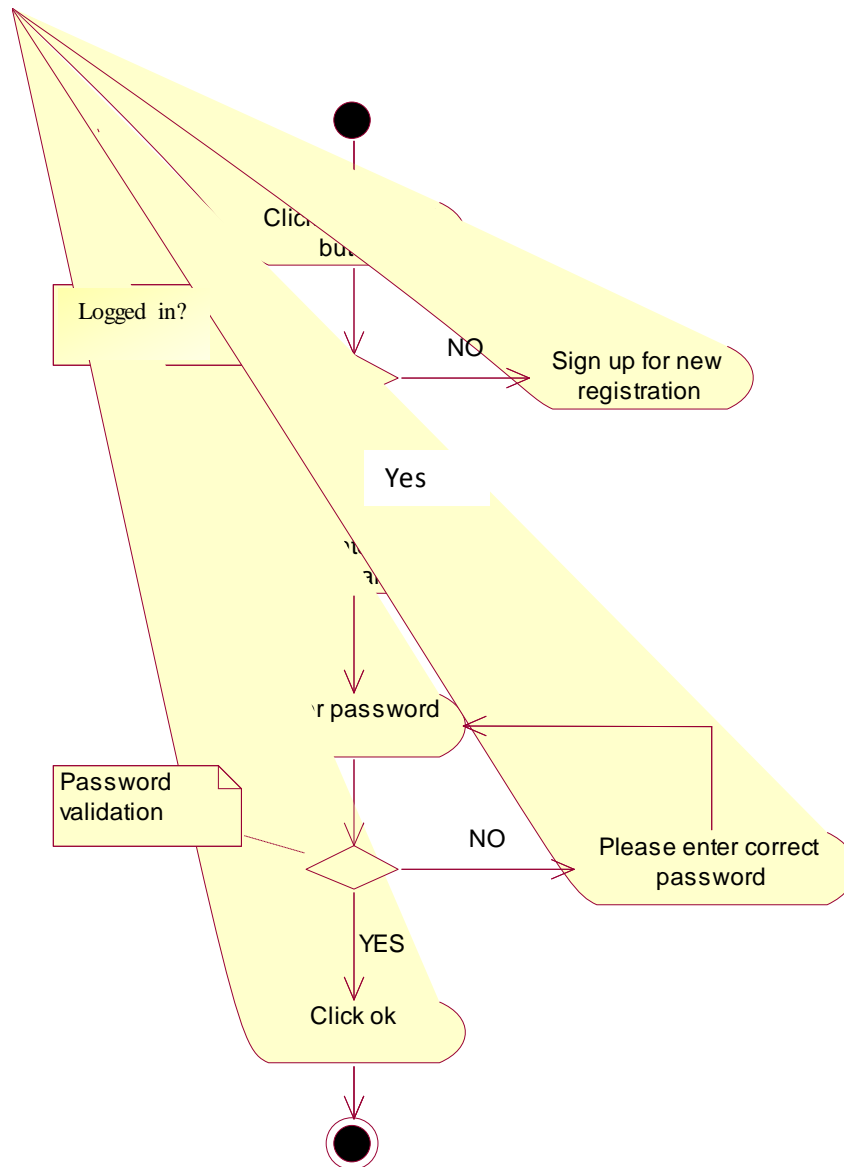


Fig 4.5 Activity Diagram Usecase: Login

Usecase: Payment

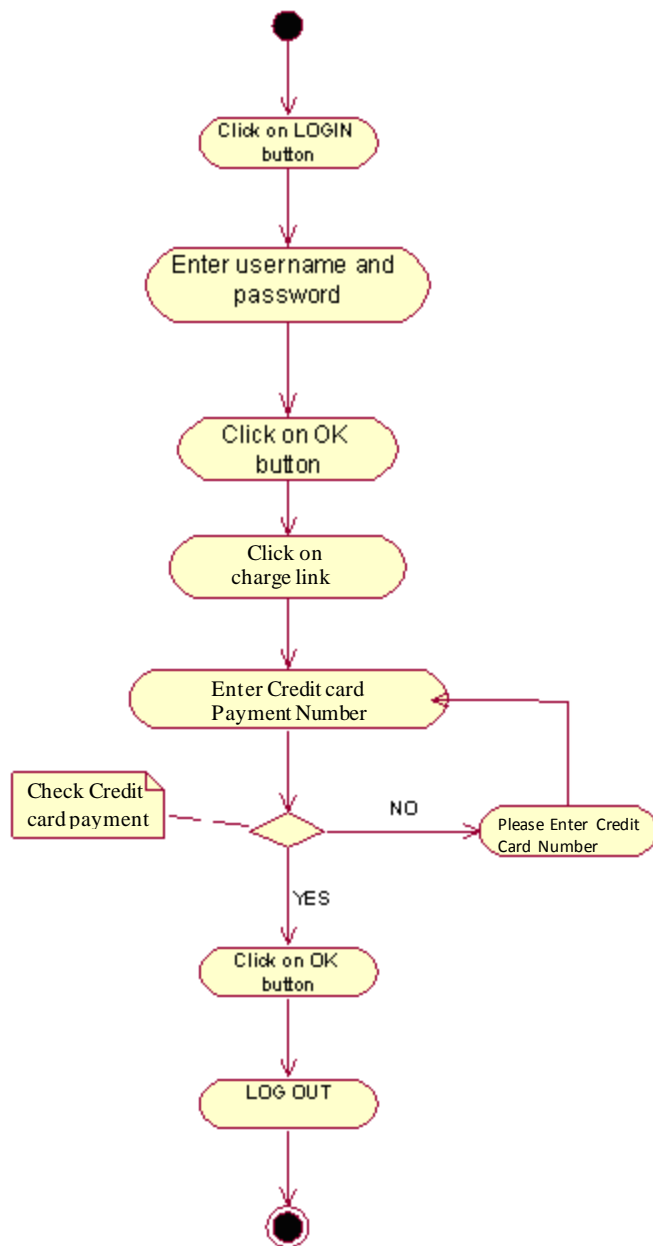


Fig 4.6 Activity Diagram Usecase: Payment

Usecase: Reservation

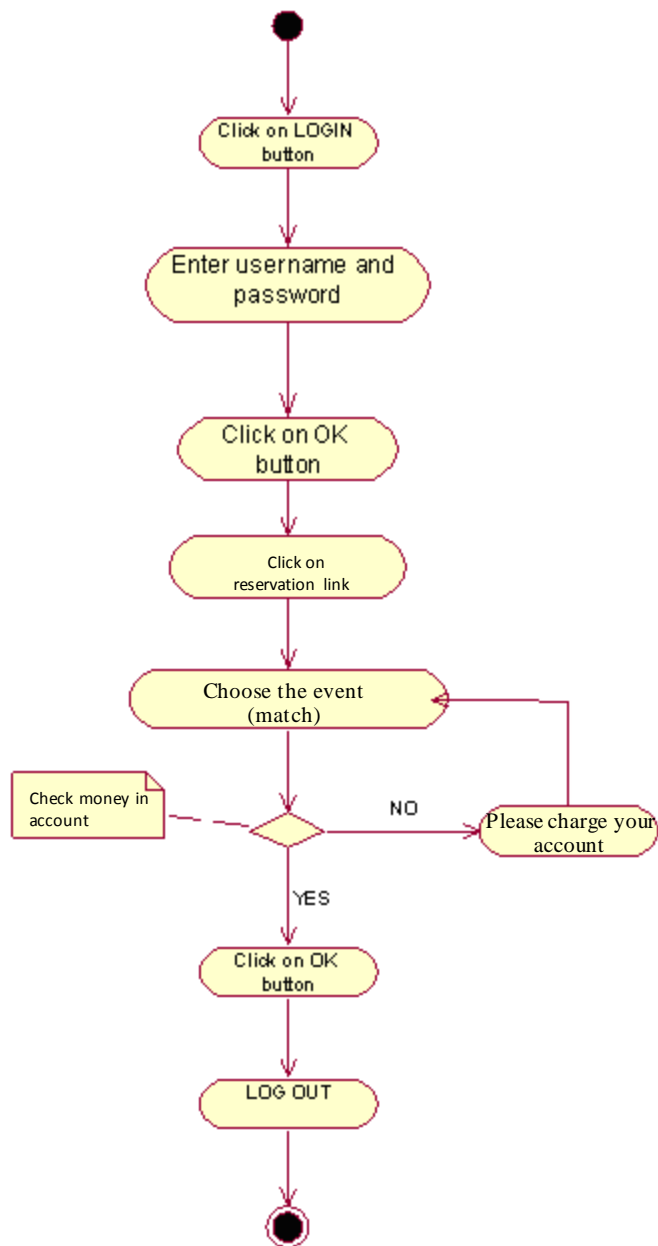


Fig 4.7 Activity Diagram Usecase: Reservation

Usecase: Logout

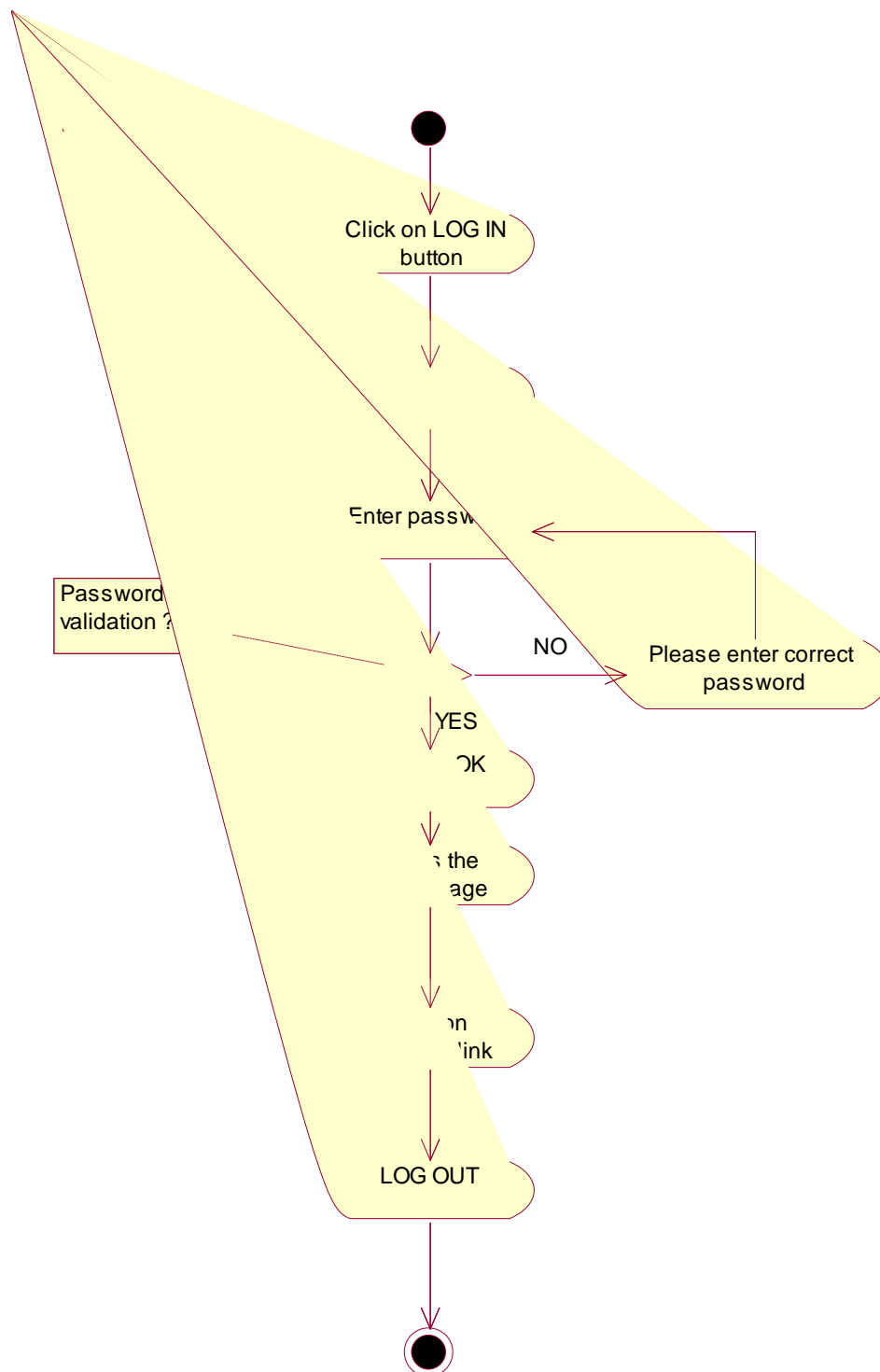


Fig 4.8 Activity Diagram Usecase: Logout

4.9 Database Tables:

Table 4.1: User

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>Id</i>	Number	Identifier of the user
<i>username</i>	Varchar	Username of the user
<i>password</i>	Varchar	Password of the user
<i>Photo</i>	Image	Photo of users
<i>Fname</i>	Varchar	First name of the user
<i>Lname</i>	Varchar	Last name of the user
<i>Gender</i>	Int	Gender of the user
<i>Dob</i>	Datetime	Date of birth of the user
<i>Address</i>	Varchar	Address of the user
<i>Loc</i>	Varchar	Location of the user
<i>Email _id</i>	Varchar	Email id of the user
<i>Mobile _no</i>	Varchar	Mobile number of the user
<i>question</i>	Char	Security question
<i>Answer</i>	Varchar	Answer for the security question

Table 4.2: Charge Card

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>Sn</i>	Number	Number of unique number
<i>Pn</i>	Number	Pin code of card
<i>Amount</i>	Money	Amount of many
<i>C _Expired</i>	Boolean	Have a money of credit card?

Table 4.3: Payment

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>Sn</i>	Number	Number of unique number
<i>Id</i>	Number	Identifier of the user
<i>Usage</i>	Money	Used amount

Table 4.4: Location

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>Id</i>	Number	Identifier of the user
<i>Loc</i>	Varchar	Location of the user

Table 4.5: Match Info

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>Match _id</i>	Number	Identifier of the match
<i>M _name</i>	Varchar	The name of both match teams
<i>Date _of _match</i>	Datetime	Date of match
<i>Note</i>	Varchare	Notes

Table 4.6: Seat Details

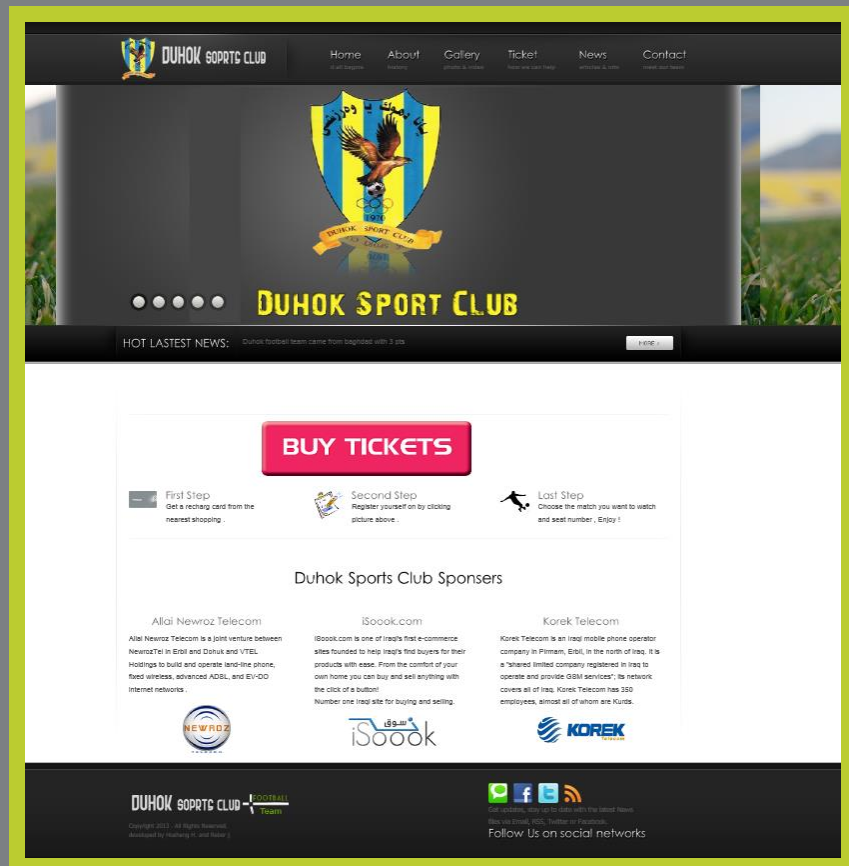
Field Name	Data Type	Description
<i>Match _id</i>	Number	Identifier of the match
<i>Seat _cat</i>	Varchar	Stadiums of category
<i>Section</i>	Number	Section of stadium
<i>Row</i>	Varchar	Numbers of row per section
<i>Seat _no</i>	Number	Numbers of seats sequence

Table 4.7: Reservation

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>id</i>	number	Identifier of the user
<i>Resv _date</i>	Datetime	Reservation date
<i>Resv _no</i>	number	Reservation number
<i>C _name</i>	Varchar	Customer? Name
<i>Match _id</i>	Number	Identifier of the match
<i>Price</i>	Real	Price
<i>Seat _no</i>	Int	Seat number
<i>Check</i>	Boolean	Check Reservation code

Table 4.8: Admin

<i>Field Name</i>	<i>Data Type</i>	<i>Description</i>
<i>A _id</i>	number	Identifier of the administrator
<i>Username</i>	Varchar	Username of the administrator
<i>Password</i>	Varchar	Password of the administrator
<i>Fname</i>	Varchar	First name of the administrator
<i>Lname</i>	Varchar	Last name of the administrator
<i>Gender</i>	Int	Gender of the administrator
<i>Dob</i>	Datetime	Date of birth of the administrator
<i>Address</i>	Varchar	Address of the administrator
<i>Email _id</i>	Varchar	Email id of the administrator
<i>Mobile _no</i>	Varchar	Mobile number of the administrator



Chapter Five

The experimental results

In this chapter, the experimental result and analysis of the developed system are shown and also give a discussion on different types of results or cases that got from output of our system.

5.1 Introduction.

In this chapter, the experimental results of the proposed system are discussed and some scenarios are given here to show how the system is really working with different cases and how the ticket reservation is done, then sending the required information using C# .NET language's in ASP.NET for sending Email, SMS using internet connection.

5.2 The Software Interfaces

The user can interact with the system throw GUI based system, where a user can run the system as shown in fig 5.1:

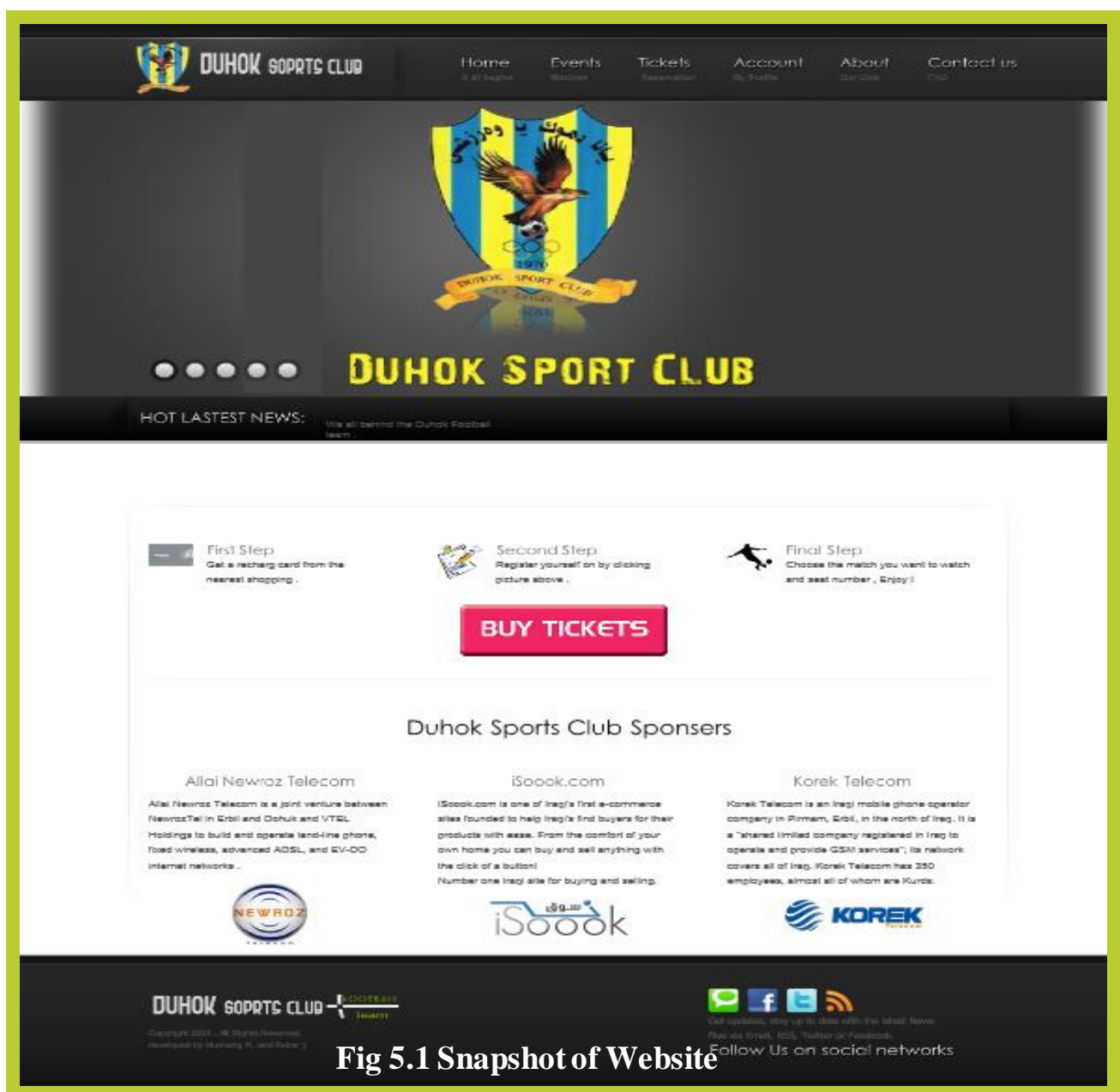


Fig 5.1 Snapshot of Website

5.3 scenario 1 (user Registration)



Home » Log in

Log in

User Name

Password

[Forgot My Password ?](#)

[Click Here to Registration](#)

This figure shows a login page with a header 'Home » Log in' and a title 'Log in'. It contains input fields for 'User Name' and 'Password', a 'Log in' button, and links for 'Forgot My Password ?' and 'Click Here to Registration'.

Fig 5.2 User & Admin Log in



Home » About » Registration

Registration

First Name

Last name

Username

Password

confirm Password

Gender ▼

Date of Birth

Address

Location ▼

Email

confirm Email

Mobile No.

Question

Answer

Photo

This figure shows a registration page with a header 'Home » About » Registration' and a title 'Registration'. It contains multiple input fields for personal information: 'First Name', 'Last name', 'Username', 'Password', 'confirm Password', 'Date of Birth', 'Address', 'Location' (with a dropdown menu showing 'Dohuk'), 'Email', 'confirm Email', 'Mobile No.', 'Question', and 'Answer'. There is also a 'Photo' field with a 'Browse...' button and a 'Registration' button at the bottom.

Fig 5.3 User Registration

Home » user » Forgot Password

Email	<input type="text"/>
Select Qusetion	<input type="button" value="v"/>
Answer	<input type="text"/>
	<input type="button" value="Send"/>

Fig 5.4 Password Recovery

Home » About » User

User Name jame
Mony 1000.0000

Reseviation
Charge Account



Fig 5.5 User Account Information

Home » user » charge money



Card Number

1	2	3
4	5	6
7	8	9
0		
Backspace		
Charge		

Fig 5.6 Charge Account

Home » Match

Match



Duhok Vs Erbil
Date :2013-5-9
Time: 3:30 PM

Fig 5.7 Select Match

Home » Tickets

Reservations type

No. of Seats

Next

Single
Friends
Family

1 ▼

Fig 5.8 Select Reservation Type

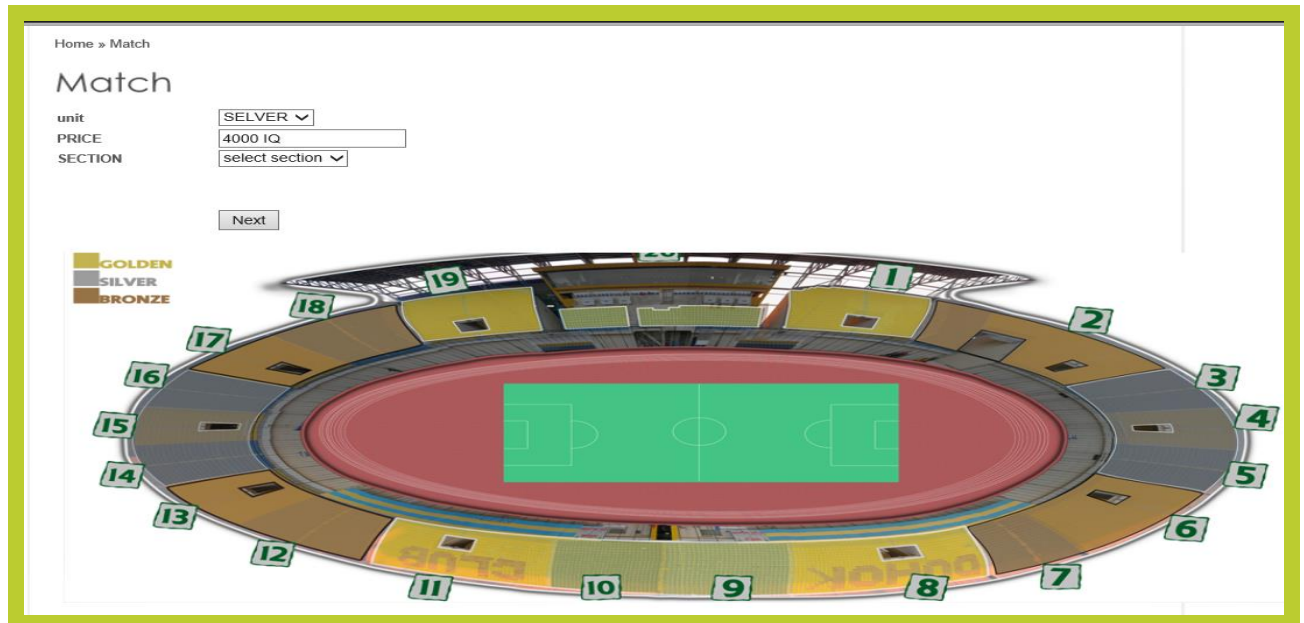


Fig 5.9 Selection of Stadium Sections

Unit:
Section:
Row:
Column:
Seat No.:

Stadium



Fig 5.10 Payment Form

5.3 Scenario 2 (Admin Registration)

Home » Admin Regesration

Admin Name

Password

Confirm Password

Email

[Back](#)

Fig 5.11 Admin Registration

Home » Admin Account

Adminstrator Name hushang

User Control

Add Match

Show Events

Add Admin.

Admin control

Stadium _Res.

Chart

Fig 5.12 Admin Account

Home » User Control

	id	username	fname	lname	password	gender	dof	add	loc	mobile	email	question	answer	money
Edit Delete Select5	monk	mashod	ali	halima		<input type="checkbox"/>								
Edit Delete Select8	shango	hushang	hussien	123		<input type="checkbox"/>	01/01/1990 12:00:00 AM	zaxo	1	0750412	hoshanksendy@yahoo.com	fine	how are you	
Edit Delete Select9	dgvdff	fsf	fsf	123		<input type="checkbox"/>	01/01/1990 12:00:00 AM	zaxo	1	0750412	hoshanksendy@yahoo.com	545	fine	
Edit Delete Select10	q	dff	dfsdf	12		<input type="checkbox"/>	01/01/1990 12:00:00 AM	zaxo	3	0750412	hoshanksendy@yahoo.com	545	dd	
Edit Delete Select13	zozo	hushang	zaxoy	zozo		<input type="checkbox"/>	01/01/1990 12:00:00 AM	aa	1	0020	hos@yahoo.com			
Edit Delete Select14	ss	hushang	zaxoy	ss		<input type="checkbox"/>	01/01/1990 12:00:00 AM	ff	2	5555	hos@yahoo.com	s	s	10000.0000
Edit Delete Select15	ddd	ccc	ccs	ddd		<input type="checkbox"/>	01/01/1990 12:00:00 AM		1		hos@yahoo.com			
Edit Delete Select16	gdfg	fvv	ff	11		<input type="checkbox"/>	01/01/1990 12:00:00 AM	s	1	0020	hos@yahoo.com	55		
Edit Delete Select17	gdfg	fvv	ff	11		<input type="checkbox"/>	01/01/1990 12:00:00 AM	s	1	0020	hos@yahoo.com	55		
Edit Delete Select18	fff	fff	fff	fff		<input type="checkbox"/>	01/01/1990 12:00:00 AM	aa	2		hos@yahoo.com	55	54	1999000.0000
Edit Delete Select19	zaxo	hushang	ghussien	zaxo		<input type="checkbox"/>	01/01/1990 12:00:00 AM	dohuk1	0750		hoshanksendy@yahoo.com	cars	kia	
Edit Delete Select20	jame	james	rony	jame		<input type="checkbox"/>	01/01/1990 12:00:00 AM	kro	1	0750000	jame@yahoo.com	first car	ford	1000.0000

[Back](#)

Fig 5.13 User Control

On-line Football Tickets Reservation Systems

Home » Add Event

Mach Name	<input type="text"/>
Notes	<input type="text"/>
Date	<input type="text"/>
	<input type="button" value="add"/>

Fig 5.14 Add Events

Home » Show the Events

	ID_Of_Match	Match_Name	Date_Of_Match	Note
Edit Delete Select1		duhok vs zakho		tiket price 1000
Edit Delete Select2		duhok vs kerkuk		iraq league
Edit Delete Select3	3			dd
Edit Delete Select4			2003	

Fig 5.15 Events Control

Home » Admin Control

admin	password	email
ad	aa	
fd	fd	dff
hushang	aaaa	hoshanksendy

[Back](#)

Fig 5.16 Admin Control

Home » Match

Match Count

Mach_ID	<input type="text" value="1"/>
Gold	11
Silver	5
Bronz	8
Sum	24

[Back](#)

Fig 5.17 Count the Seat Number of the Match

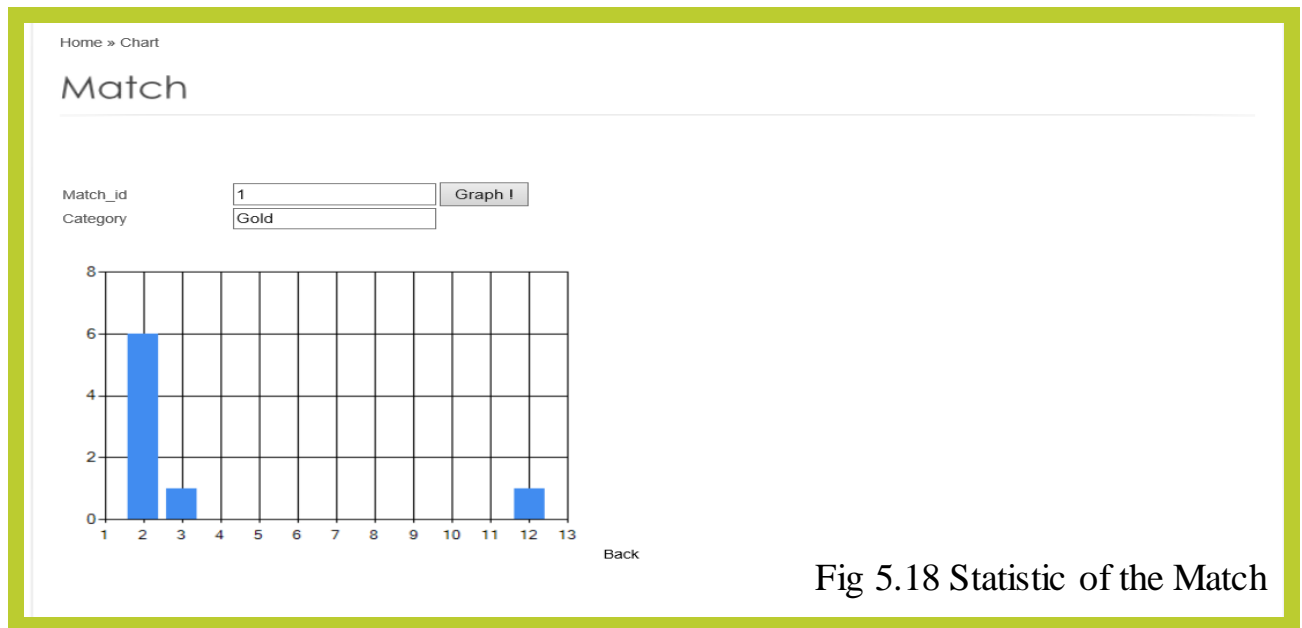


Fig 5.18 Statistic of the Match

Home » Contact us

Contact us

Meet Duhok SC

28. Reza Road: Duhok / Kurdistan Region - Iraq

Call us via Phone

Tel. +964 #### #### ##
Tel. +964 #### #### ##

Emails

General: info@Duhoksports.com
Support: support@Duhoksports.com

Duhok SC address via Settelite Maps

In addition to online resources like the forums and mailing lists a great way to get ailing lists a great way to get involved withIn addition to online resources like the forums and mailing lists a great way to get ailing listsIn addition to online resources like the forums and mailing lists a great way to get ailing lists

Contact Form

Name (* required)

Email Address (* required)

Subject (* required)

Message (* required)

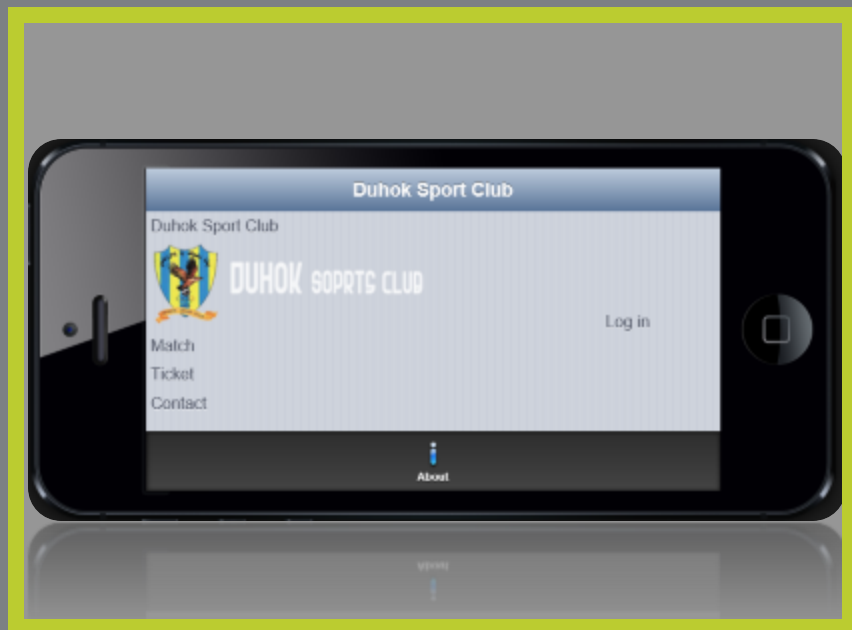
Fig 5.19 Contact Us

5.5 Project Testing.

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way. Normally the former is considered a better practice since it, allows interface issues to be localized more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

Table 5.1: Project Test

<i>SL.NO</i>	<i>Test case</i>	<i>Condition being Checked</i>	<i>Expected Result</i>	<i>Obtained Result</i>
1.	Administrator login invalid	Are username and the password valid?	Error message printed "Invalid Password, Try Again".	Same as expected
2.	Administrator login valid	Is the username and the password valid?	Main window displayed.	Same as expected
3.	Insert event	Is the detail present?	IF the detail is present then it exits ELSE, Insert details.	Same as expected
4.	Delete the user	Is entry present?	IF found delete the details ELSE, display message "ENTRY NOT PRESENT"	Same as expected
5.	Update the user information	Is record present?	IF present update	Same as expected
6	Delete event information	Are the event information present for the day?	If present deletion of the event cannot be done. Else event is deleted	Same as expected
7	Payment	Is the reservation details present for the specified reservation date and reservation code?	Payment is done	Same as expected



Chapter Six

Conclusion and Future Scope

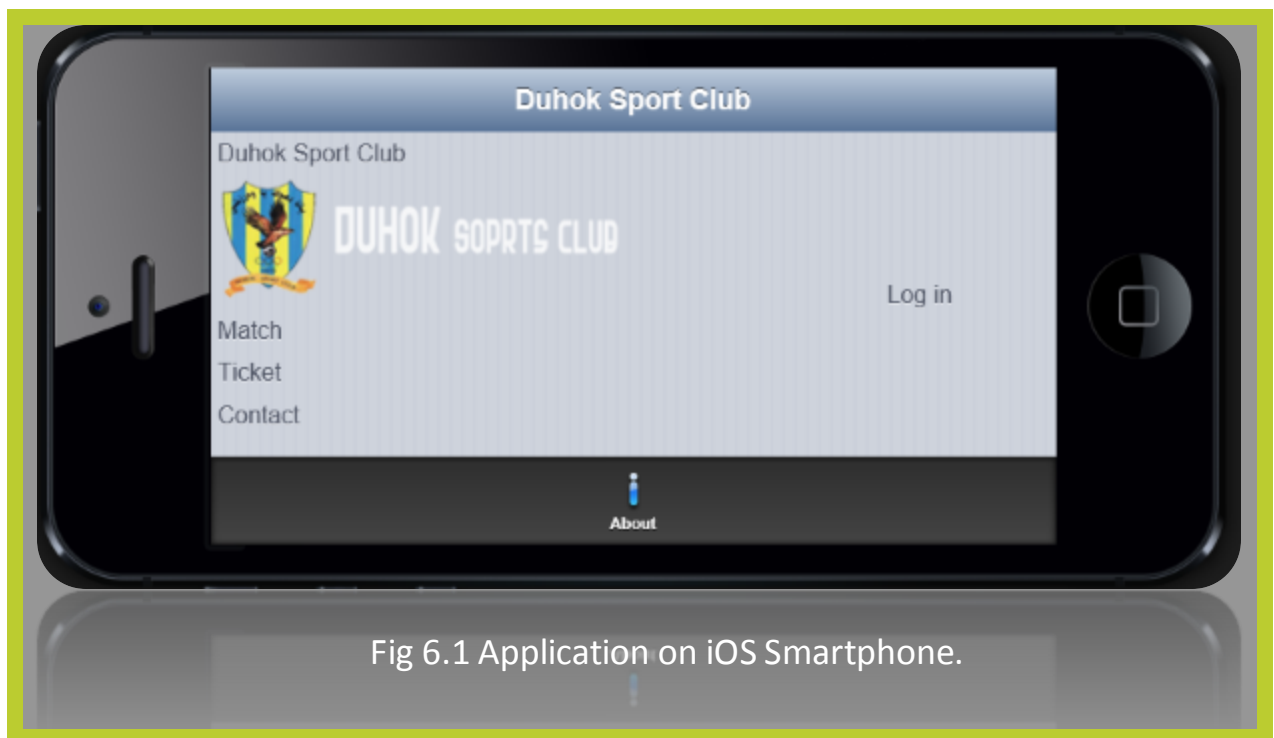
This chapter summarize the overall work and indicating possible future work of the proposed system that can be down to improve the quality of the system in general.

6.1 Conclusion

In this project, we present the implementation of “On-line Football Tickets Reservation System”. This system is an effort to develop a software that can be accessed through the internet connection via smart phone or computer. Our project makes the ticket reservation simple and easy.

6.2 Future Work

The system use web browser to view the website .the future work for this project is to create an application for smart phone (Android, iOS, Windows RTM) operating system which is easy for user.



The system uses PC and mobile to access the developed site through the internet.

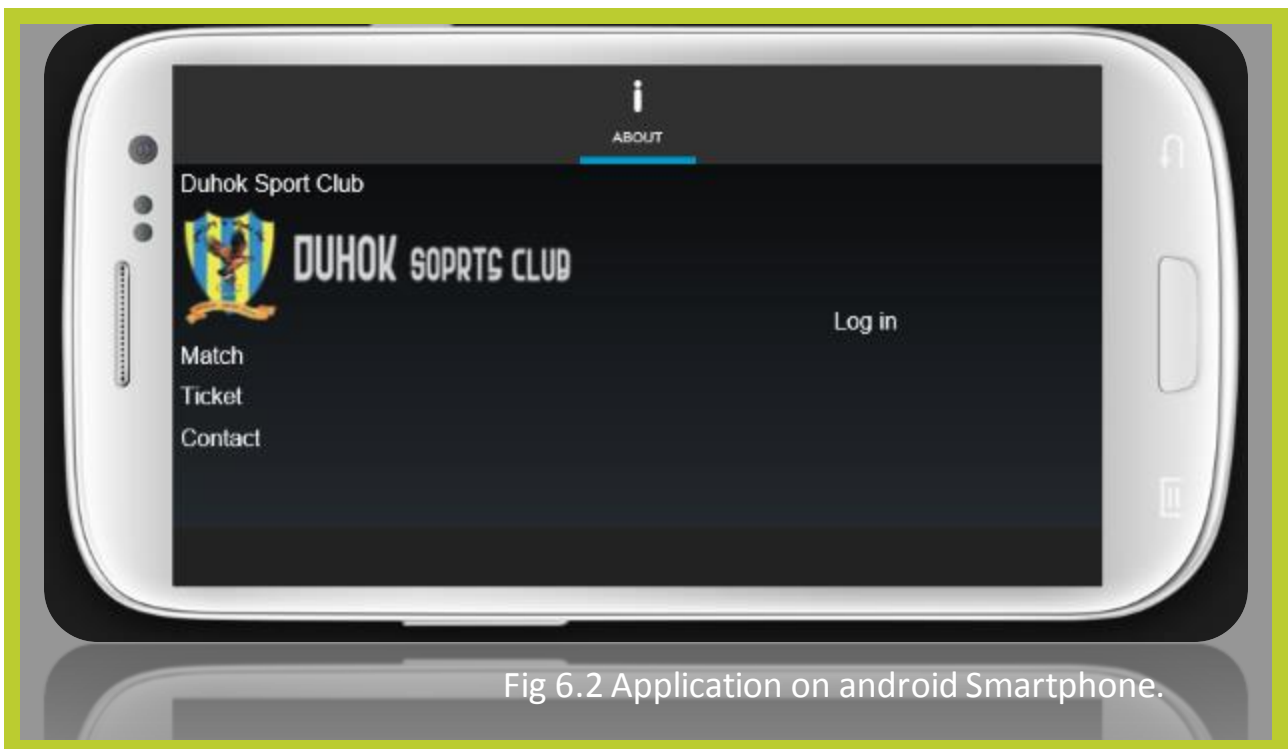


Fig 6.2 Application on android Smartphone.

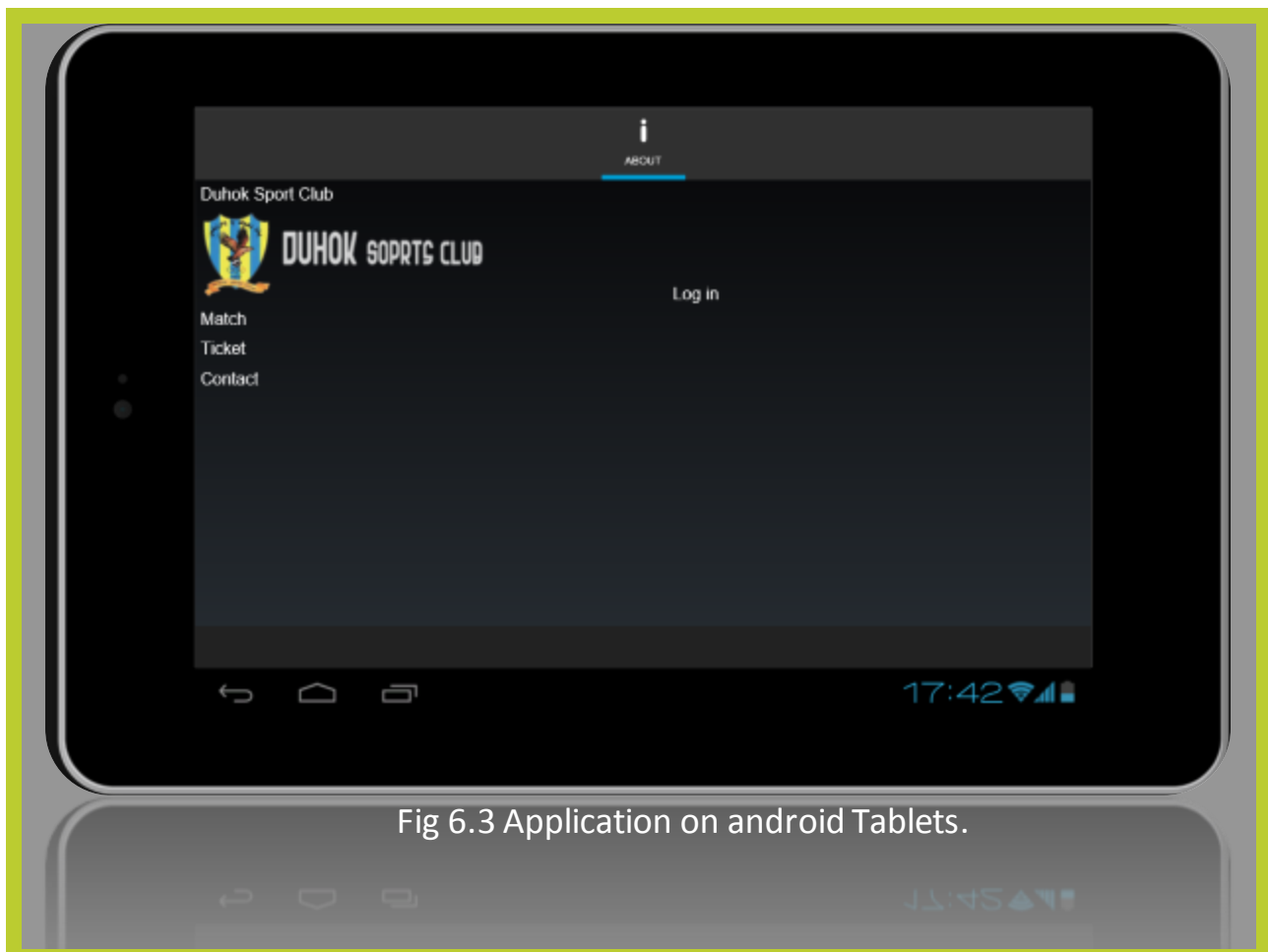


Fig 6.3 Application on android Tablets.

References

- [1] http://en.wikipedia.org/wiki/Mobile_ticketing
- [2] http://en.wikipedia.org/wiki/Electronic_ticket
- [3] <http://www.mobility.siemens.com/mobility/global/SiteCollectionDocuments/en/complete-mobility/e-ticketing-en.pdf>
- [4] <http://electrofriends.com/projects/computer-programming/mticket-mobile-booking/>
- [5] www.w3school.com
- [6] Swapna Kishore, Rajesh Naik, 2007, Software Requirement and estimation, Tata McGraw Hill
- [7] Qusay Idrees Sarhan, 2011, Vision Based Home Monitoring System Using Java Technology
- [8] Bennett, s., McRobb, s., Farmer, R & Mosman, k., 2006. Object oriented analysis and design using uml. 3rd ed. London, England: McGraw hill.
- [9] Bell, D. & Parr, M., 2004. C# for students. Harlow, England: Addison-Wesley.
- [10] Barzdins, J. & Caplinskas, A., 2001. Databases and information systems. Dordrecht, The Netherlands: Kluwer Academic Publishers.
- [11] Balter, A. & Kanouse, G., 2006. Microsoft SQL server 2005 express in 24 hours. Indianapolis, USA: Sams Publishing.
- [12] Darie, G, Ruvalcaba, Z. & Laidlaw, G., 2006. Build your own ASP.NET 2.0 web site using C# & VB. 2nd ed. ed. Collingwood, Australia: Sitepoint Pty Limited.
- [13] Deitel, P.J. & Deitel, H.M., 2009. Visual C# 2008. 3rd International ed. ed. Upper Saddle River, USA: Pearson Prentice Hall.
- [14] Date, GJ. 2004. An introduction to database systems. 8th International ed. ed. Boston, USA: Pearson Addison Wesley.
- [15] Deitel, P., Deitel, H., Hart & Hirsch, M., 2011. Visual C# 2010. 4th International ed. ed. Boston, USA: Pearson.
- [16] Somerville, I., 2011. Software engineering. 9th International ed. Boston, USA: Pearson.
- [17] <http://www.1000projects.org/sr1p/dn1p/jo-po-pr.zip>
- [18] <http://www.maxartists.com/images/media/Mobileticketing.pdf>
- [19] <http://www.computing.dcu.ie/~asmeaton/MECPracticums2005/MSolutions.pdf>
- [20] www.msdn.microsoft.com
- [21] www.asp.net
- [22] <http://www.mticket.com>
- [23] www.codeproject.com