Online Examination

CONTENT

INTRODUCTION TO PROJECT

ABOUT THE PROJECT

The ONLINE EXAMINATION is a web application to take online quiz in an efficient manner and no time wasting for checking the paper. The main objective of ONLINE EXAMINATION is to efficiently evaluate the candidate thoroughly through a fully automated system that not only saves lot of time but also gives fast results. Faculty give papers for users according to their convenience and time and there is no need of using extra thing like paper, pen etc. This can be used in educational institutions as well as in corporate world. Can be used anywhere any time as it is a web based application (user location doesn't matter). No restriction that examiner has to be present when the candidate takes the test. This Web Application provides facility to conduct online examination worldwide. It saves time as it allows number of users to give the exam at a time and displays the results as the test gets over, so no need to wait for the result. It is automatically generated by the server. Administrator has a privilege to add new questions . admin can create different tests for users. Users (student) can register, login and give the test and can see the results as well. It provides a competitive platform, where a student not only judges their knowledge/skill but also they can improve their knowledge/skill at the same time.

PROJECT SCOPE

Providing accessibility to the administrators who have a valid user id and password. Online Quiz also provides the following facilities such as:

- 'N 'number of users can participate in a quiz.
- Automation of scores of the users.

- This can be used in educational institutions as well as corporate world.

2. System Analysis

Existing System:

- Users from different areas have to visit university for every query regarding filling up of application forms, results and syllabus etc are collected by the user personally, thus wasting his/her precious time

- Users have to wait in long queues to take examination forms and to get know the status of their results. Even sometimes, they cannot take admission in higher studies outside the state.

- The manual examination system leads to errors, more time consumption, inefficient and wastage of valuable resources.

- There is repetition of work in the existing system, the same data is written again and again by different branches.

Proposed System:

To solve the problems faced with manual quiz writing, there is need focus computerized system to handle all the works.

- I propose a web based application that will provide a working environment that will be flexible and will provide easy of work and will reduce the time for report generation and other paper works.

- It will improve the efficiency of the workforce who in turn carries out many operations manually.

- It is reduces the time, easy to handle the multiple users and generating results automatically.

- After getting the results users can selected Or Deselected for certain company.
- If users was selected some message will shows like that (User carrying those educational original documents and passport size photos to particular Organization).

3. FEASIBILITY STUDY

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.1.1 ECONOMIC FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

3.1.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.1.3 TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

4. System Requirements Specification

4.1 Introduction

A **Software Requirements Specification** (**SRS**) – a requirements specification for a software system – is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- **Business requirements** describe in business terms *what* must be delivered or accomplished to provide value.
- **Product requirements** describe properties of a system or product (which could be one of Several ways to accomplish a set of business requirements.)

- **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles

PURPOSE

A systems engineering, a **requirement** can be a description of *what* a system must do, referred to as a Functional Requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Non-functional requirements, or 'performance requirements' or 'quality of service requirements.' Examples of such requirements include usability, availability, reliability, supportability, testability and maintainability.

A collection of requirements define the characteristics or features of the desired system. A 'good' list of requirements as far as possible avoids saying *how* the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called "implementation bias" or "solution engineering". However, *implementation constraints* on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality (e.g. of user interfaces) with other owned products.

In software engineering, the same meanings of requirements apply, except that the focus of interest is the software itself.

Modules:

- Admin
- User
- Security and Authencation

Admin:

Test Info:

- The Admin (faculties) create the quiz, admin can give the permission about that quiz to users.

New User:

- The Admin can add other users so that can make can enter into this web application login.

User:

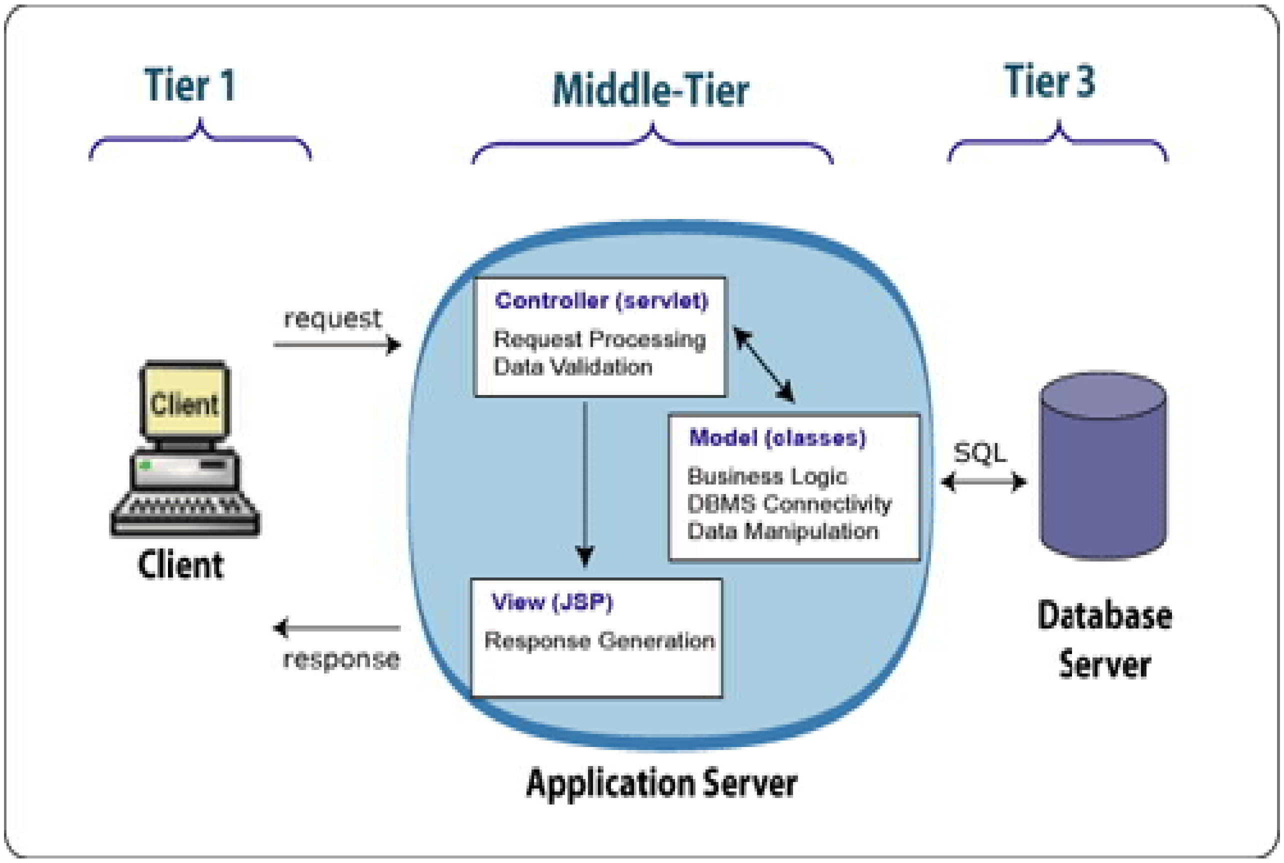
- The user can know the information about the quiz.
- Select the test enter your details if it is valid data user can attempt the quiz.
- Finally user got the result of the particular quiz.

SECURITY AND AUTHENCATION:

- The user details should be verified against the details in the user tables and if it is valid user, they should be entered into the system.
- Once entered, based on the user type access to the different modules to be enabled / disabled and individual user can change their default password or old password.

PROCESS FLOW

ARCHITECTURE DIAGRAM



1. THE PRESENTATION LAYER

Also called as the client layer comprises of components that are dedicated to presenting the data to the user. For example: Windows/Web Forms and buttons, edit boxes, Text boxes, labels, grids, etc.

2. THE BUSINESS RULES LAYER

This layer encapsulates the Business rules or the business logic of the encapsulations. To have a separate layer for business logic is of a great advantage. This is because any changes in Business

Rules can be easily handled in this layer. As long as the interface between the layers remains the same, any changes to the functionality/processing logic in this layer can be made without impacting the others. A lot of client-server apps failed to implement successfully as changing the business logic was a painful process

3. **THE DATA ACCESS LAYER**

   This layer comprises of components that help in accessing the Database. If used in the right way, this layer provides a level of abstraction for the database structures. Simply put changes made to the database, tables, etc do not affect the rest of the application because of the Data Access layer. The different application layers send the data requests to this layer and receive the response from this layer.

4. **THE DATABASE LAYER**

   This layer comprises of the Database Components such as DB Files, Tables, Views, etc. The Actual database could be created using SQL Server, Oracle, Flat files, etc.
   in an n-tier application; the entire application can be implemented in such a way that it is independent of the actual Database. For instance, you could change the Database Location with minimal changes to Data Access Layer. The rest of the Application should remain unaffected.

The **Systems Development Life Cycle (SDLC)**, or *Software Development Life Cycle* in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:

Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

WHAT IS SDLC?

A software cycle deals with various parts and phases from planning to testing and deploying software. All these activities are carried out in different ways, as per the needs. Each way is known as a Software Development Lifecycle Model (SDLC). A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. A descriptive model describes the history of how a particular software system was developed. Descriptive models may be used as the basis for understanding and improving software development processes or for building empirically grounded prescriptive models.
SDLC models * **The Linear model (Waterfall)** - Separate and distinct phases of specification and development. - All activities in linear fashion. - Next phase starts only when first one is complete. * **Evolutionary development** - Specification and development are interleaved (Spiral, incremental, prototype based, Rapid Application development). - Incremental Model (Waterfall in iteration), **-** RAD(Rapid Application Development) **-** Focus is on developing quality product in less time, - **Spiral Model** - We start from smaller module and keeps on building it like a spiral. It is also called Component based development. * **Formal systems development** - A mathematical system model is formally transformed to an implementation. * **Agile Methods.** - Inducing flexibility into development. * **Reuse-based development** - The system is assembled from existing components.

The General Model

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. There are tons of models, and many companies adopt their own, but all have very similar patterns. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced during implementation that is driven by the design. Testing verifies the deliverable of the implementation phase against requirements.

SDLC Methodology:

Spiral Model

The spiral model is similar to the incremental model, with more emphases placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A\ software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Requirements are gathered during the planning phase. In the risk analysis phase, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. Software is produced in the engineering phase, along with testing at the end of the phase. The evaluation phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral. In the spiral model, the angular component represents progress, and the radius of the spiral represents cost. Spiral Life Cycle Model.

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:
    1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
    2. Defining the requirements of the second prototype.
    3. Planning a designing the second prototype.
    4. Constructing and testing the second prototype.
- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

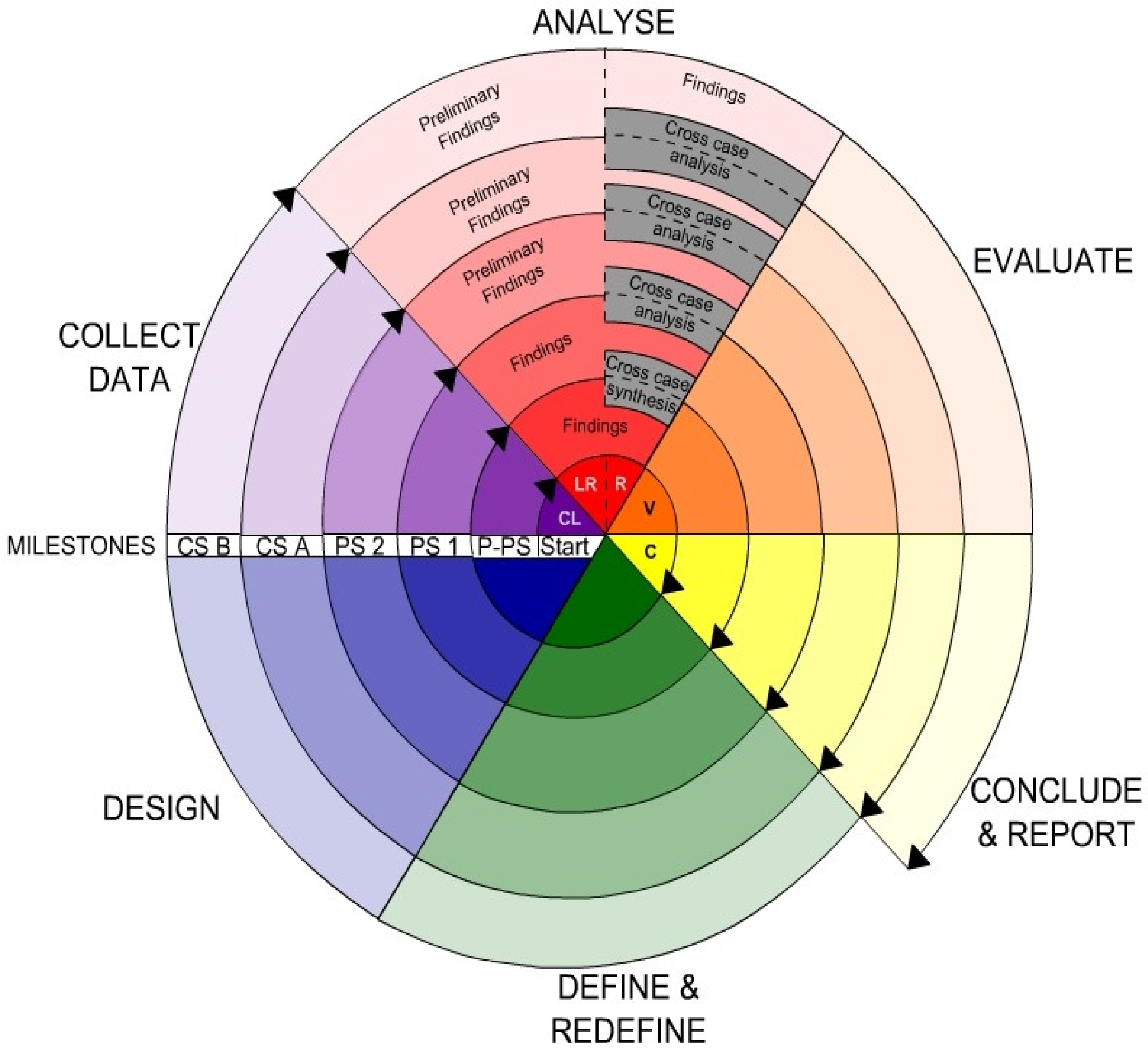


Fig -Spiral Model

Advantages of Spiral Model

- High amount of risk analysis
- Good for large and mission-critical projects.
- Software is produced early in the software life cycle.

Software Requirements:

Operating System	:	Windows/2003/ or Linux/Solaris
User Interface	:	HTML, CSS
Client-side Scripting	:	JavaScript
Programming Language	:	Java
Web Applications	:	JDBC, Servlets, JSP
IDE/Workbench	:	My Eclipse 8.0
Database	:	Oracle11g XE
Server Deployment	:	Tomcat

Hardware Requirements: (Minimum)

Processor	:	Core 2 Duo
Hard Disk	:	160GB
RAM	:	1 GB

5. System Design

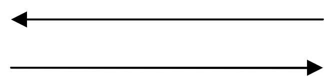
- **E-R Diagram:**

Data Flow Diagrams:

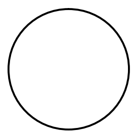
A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

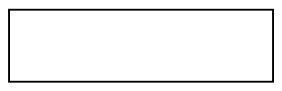
1. Dataflow: Data move in a specific direction from an origin to a destination.



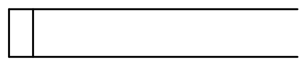
2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



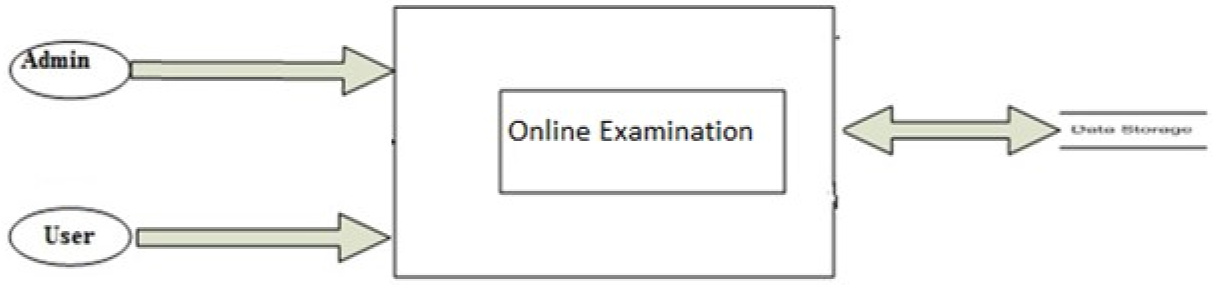
3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.



4. Data Store: Here data are stored or referenced by a process in the System.

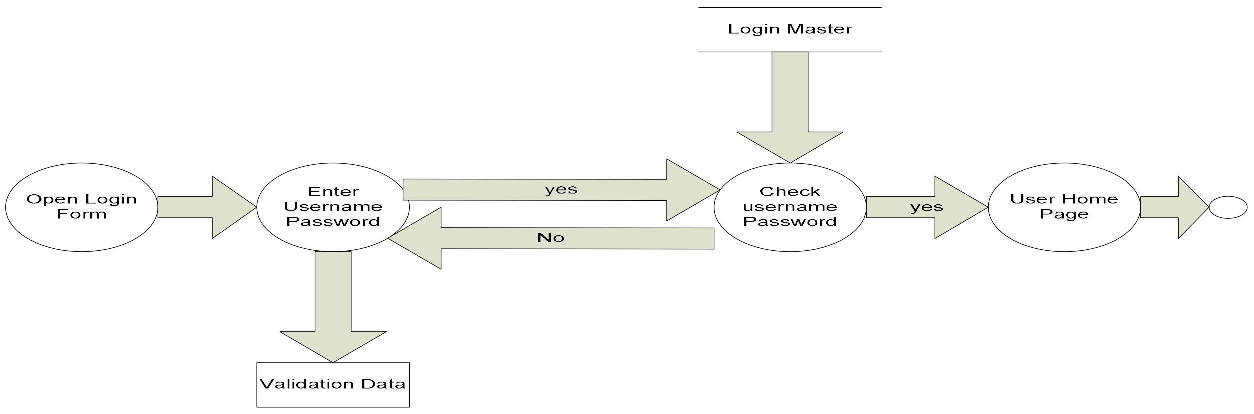


CONTEXT LEVEL 0 DIAGRAM:

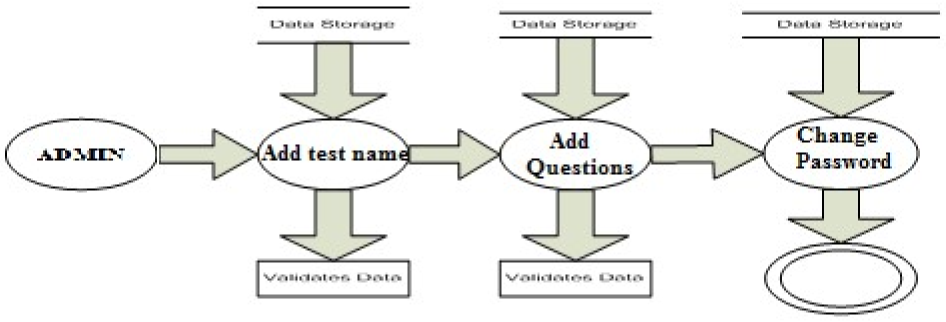


Context level1 Diagram:

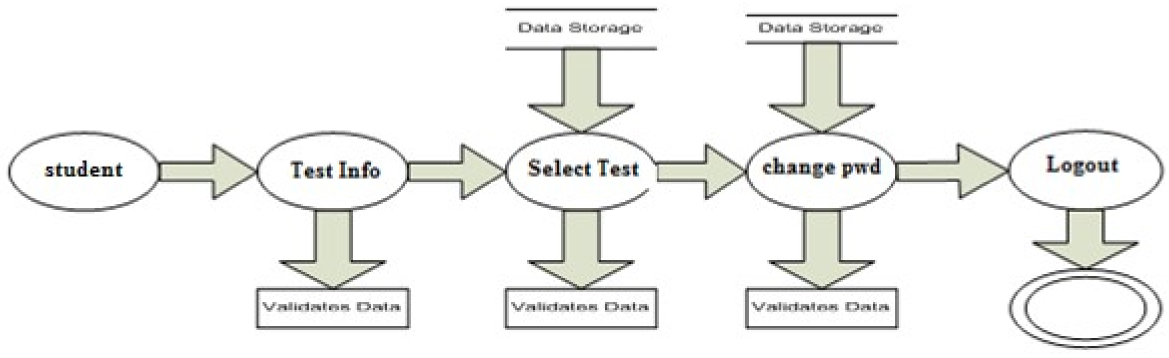
Login DFD



Context level 2:



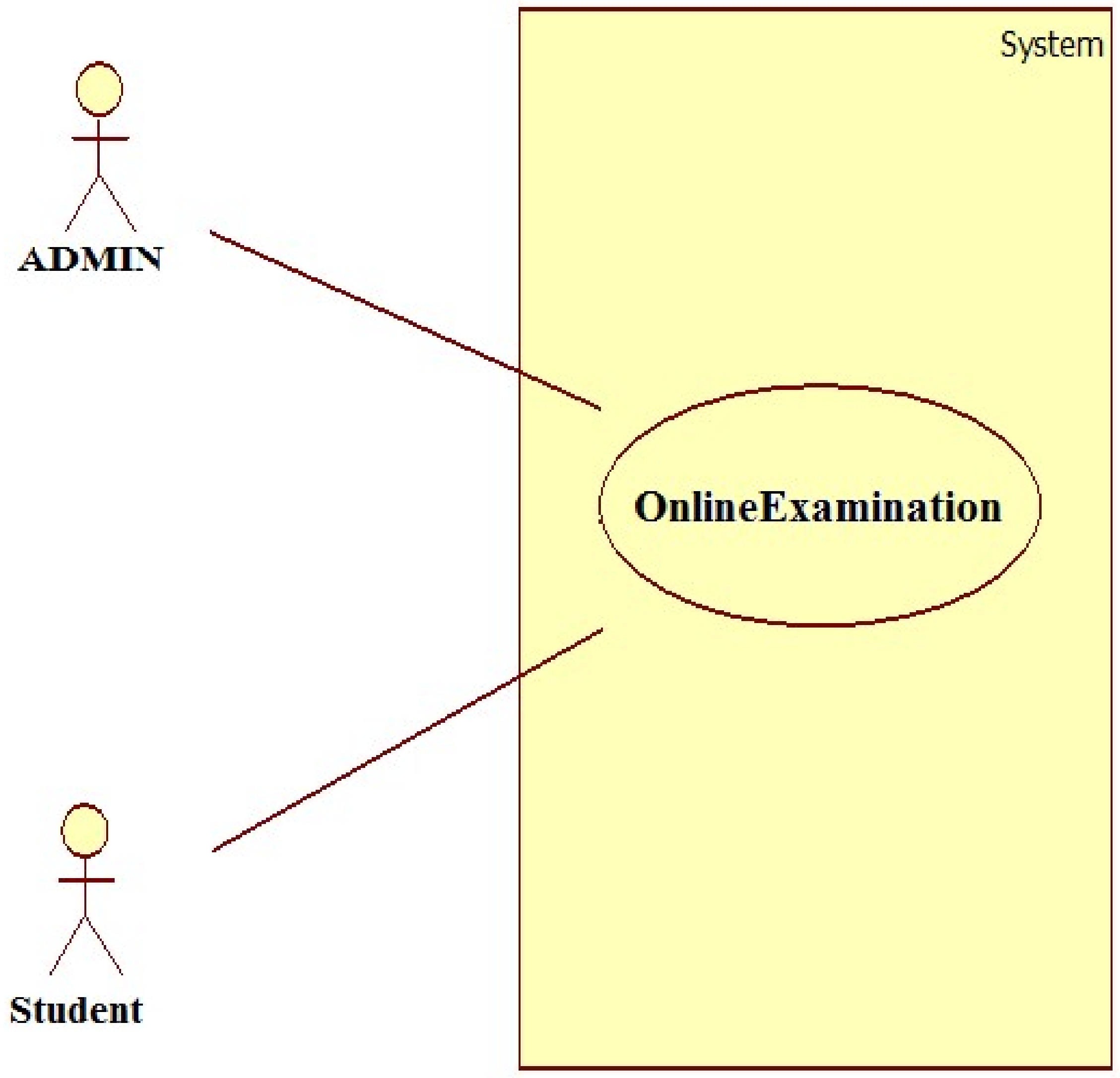
Context level 3:



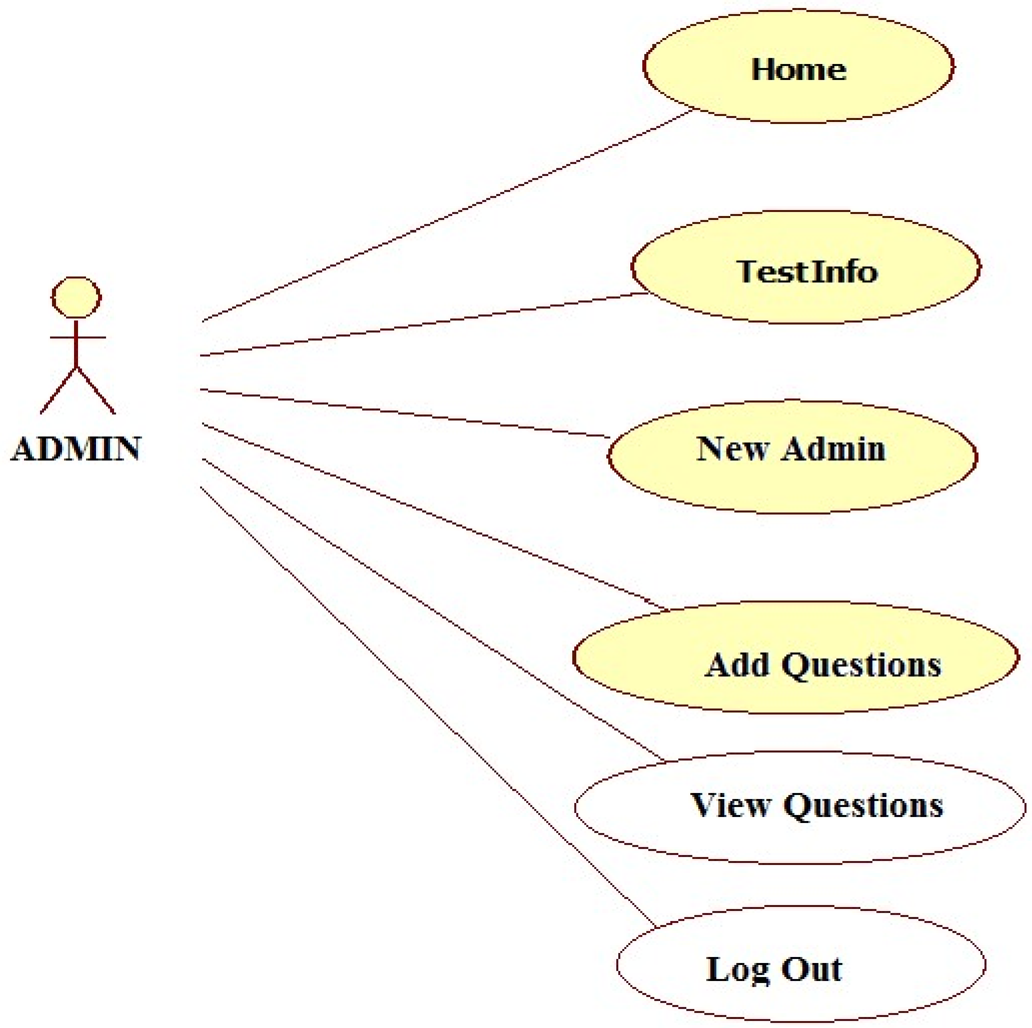
UML Diagrams

Class diagram:

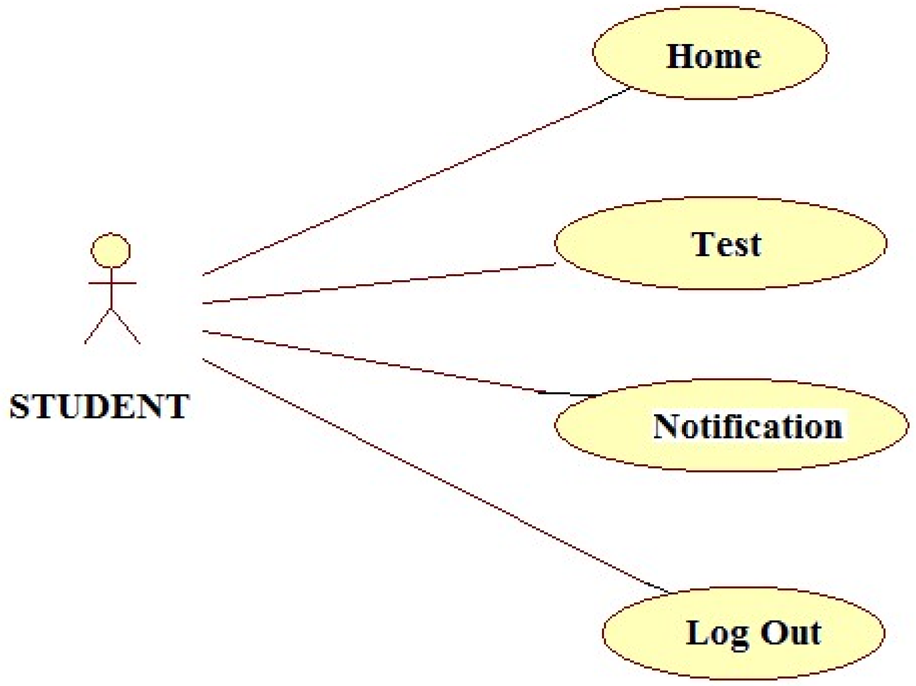
Use case diagrams:



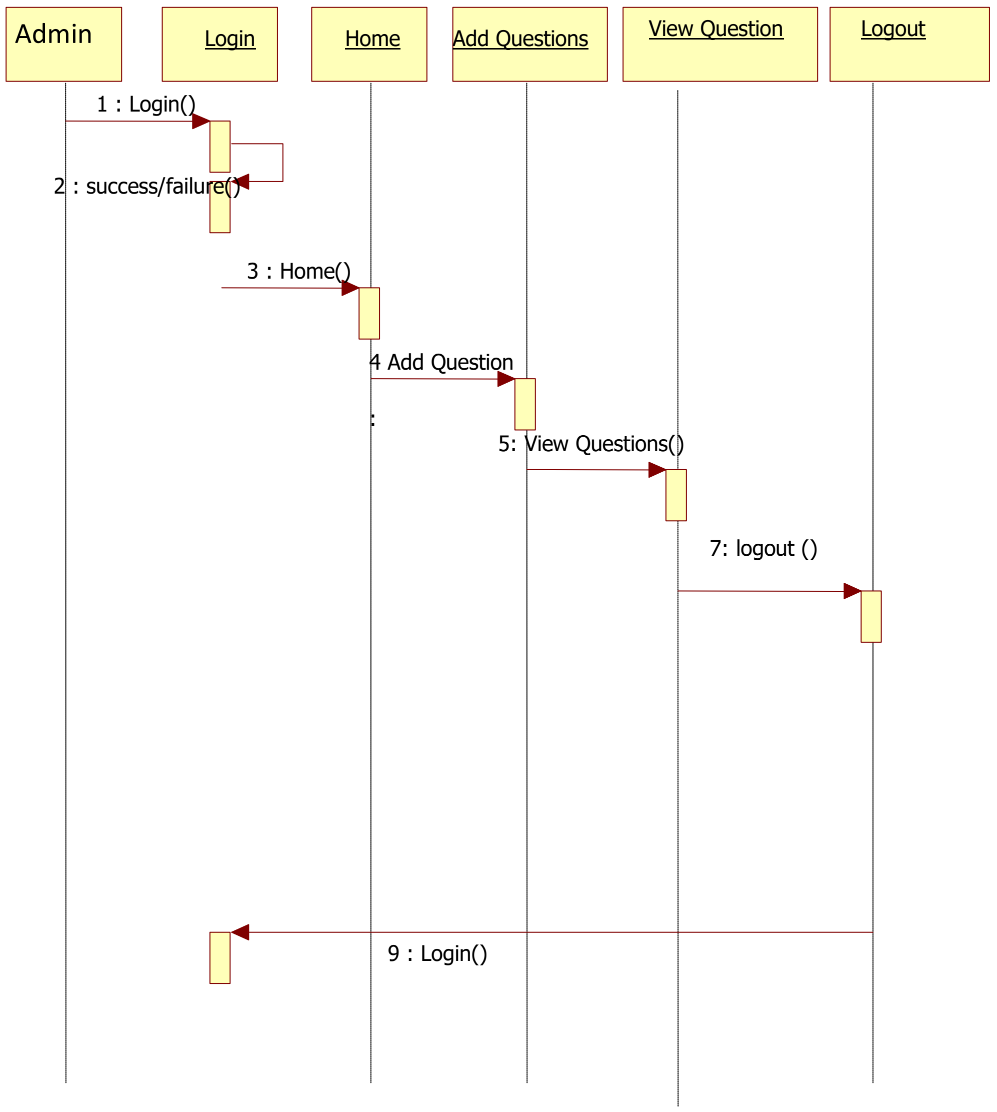
ADMIN


ADMIN
Home
TestInfo
New Admin
Add Questions
View Questions
Log Out

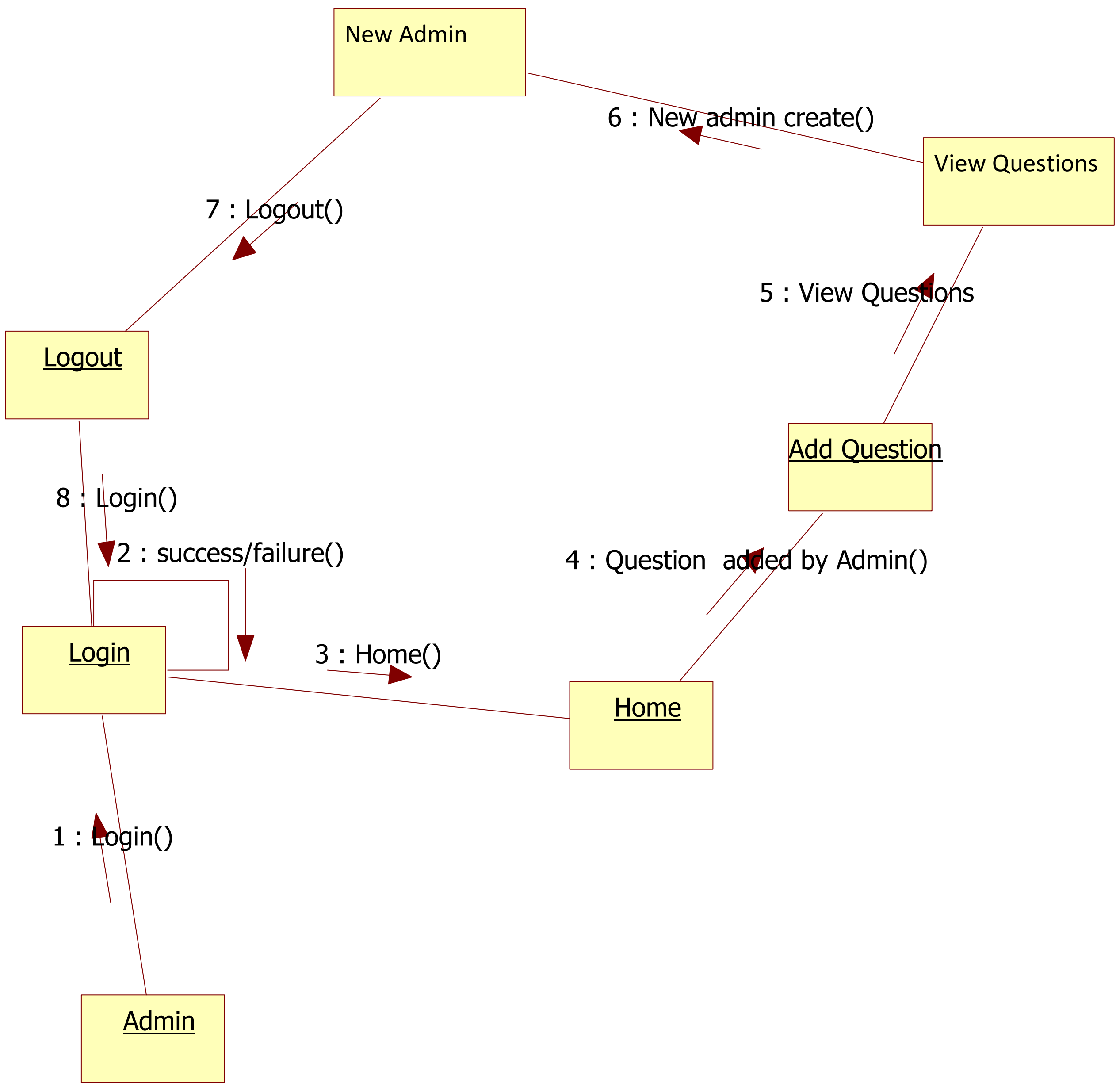
User:



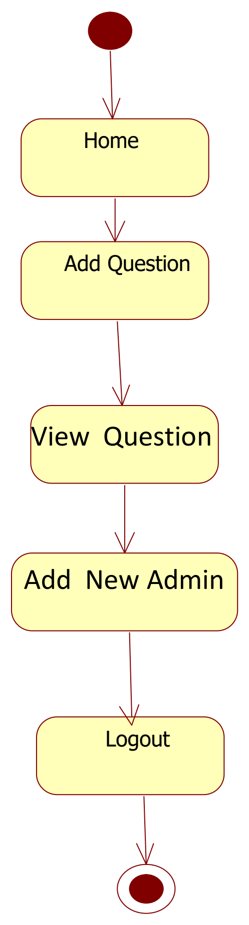
Admin Sequence Diagram:



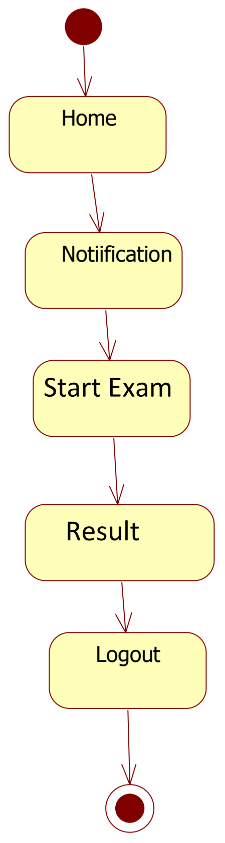
Admin Collaboration diagram:



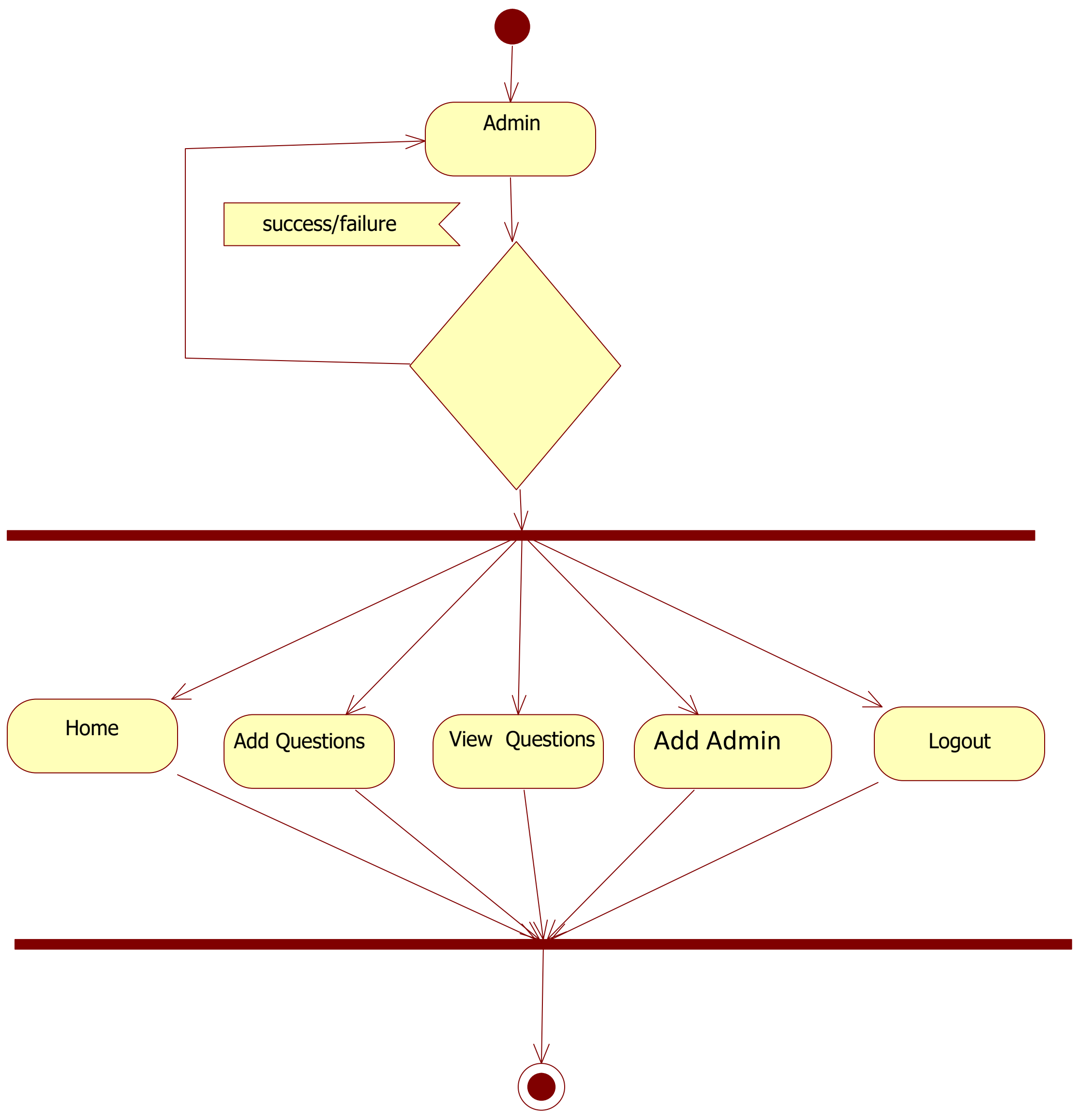
Admin State Chart:



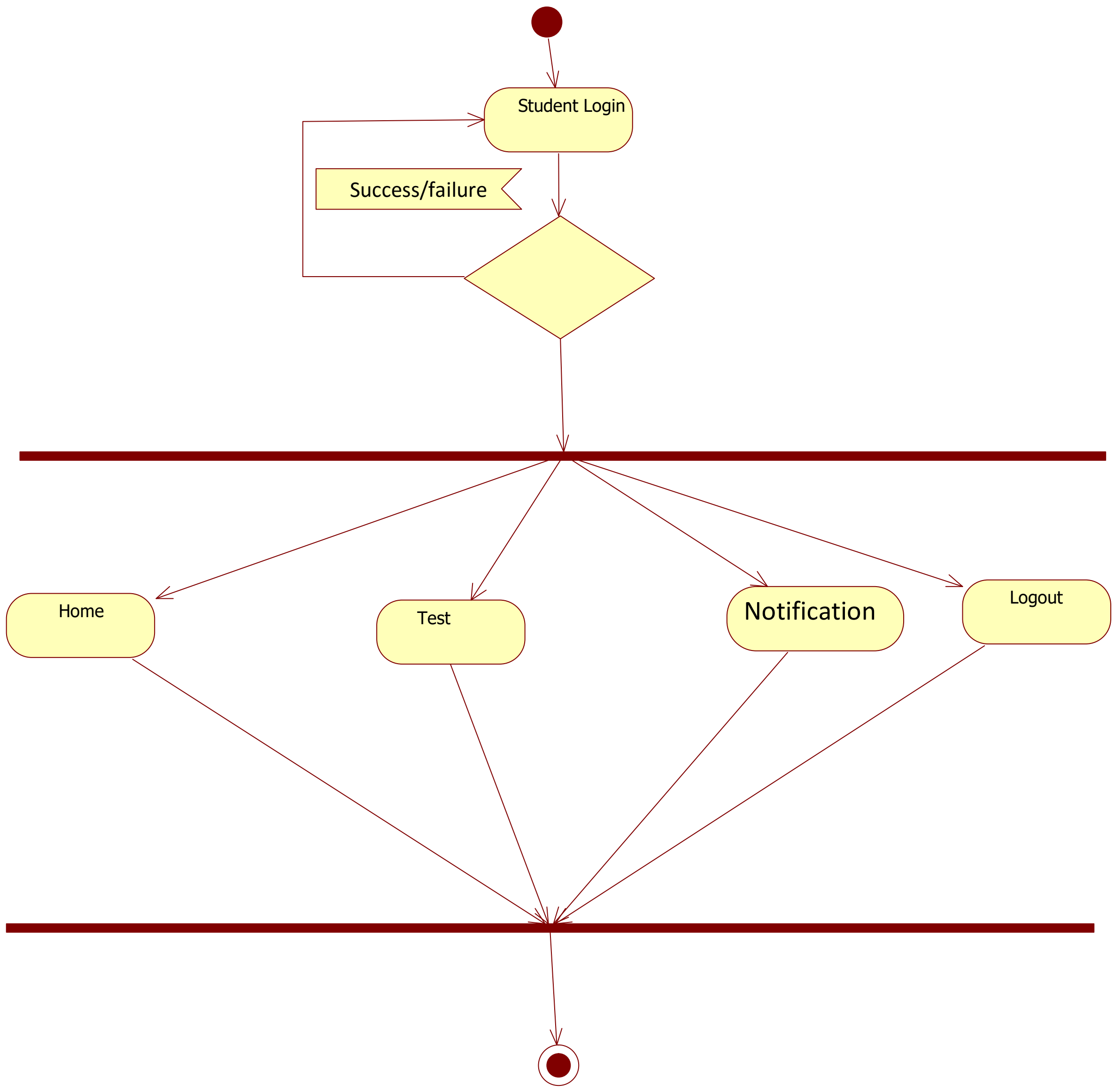
Student State Chart:



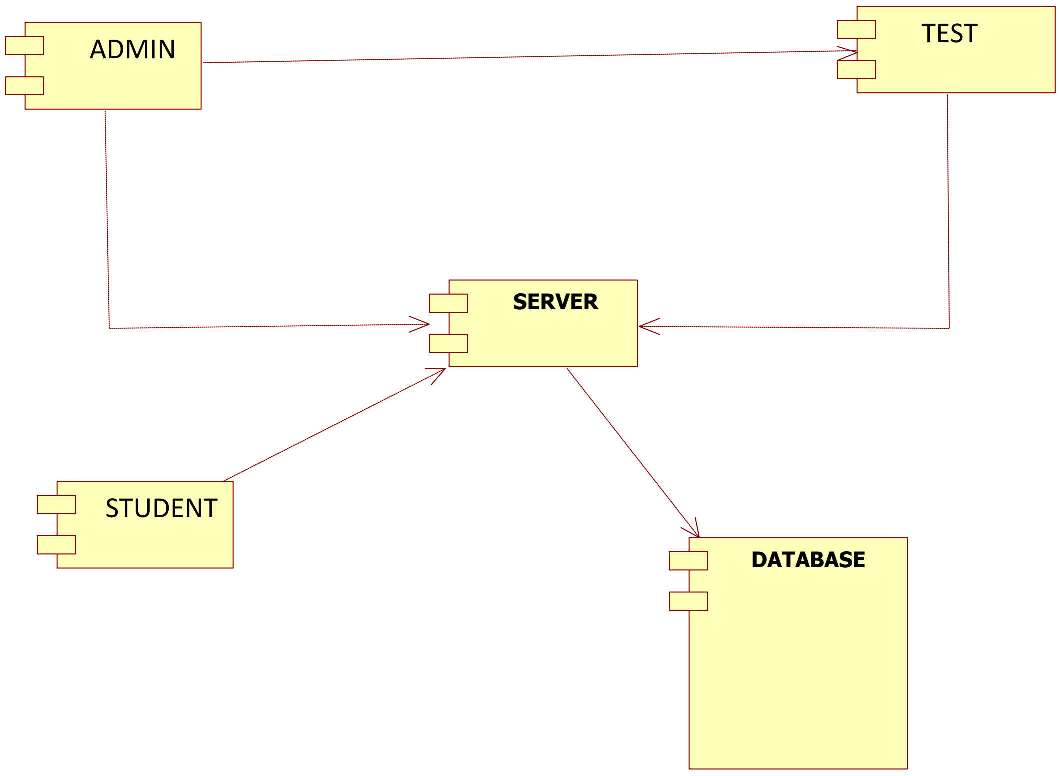
Admin Activity Diagram:



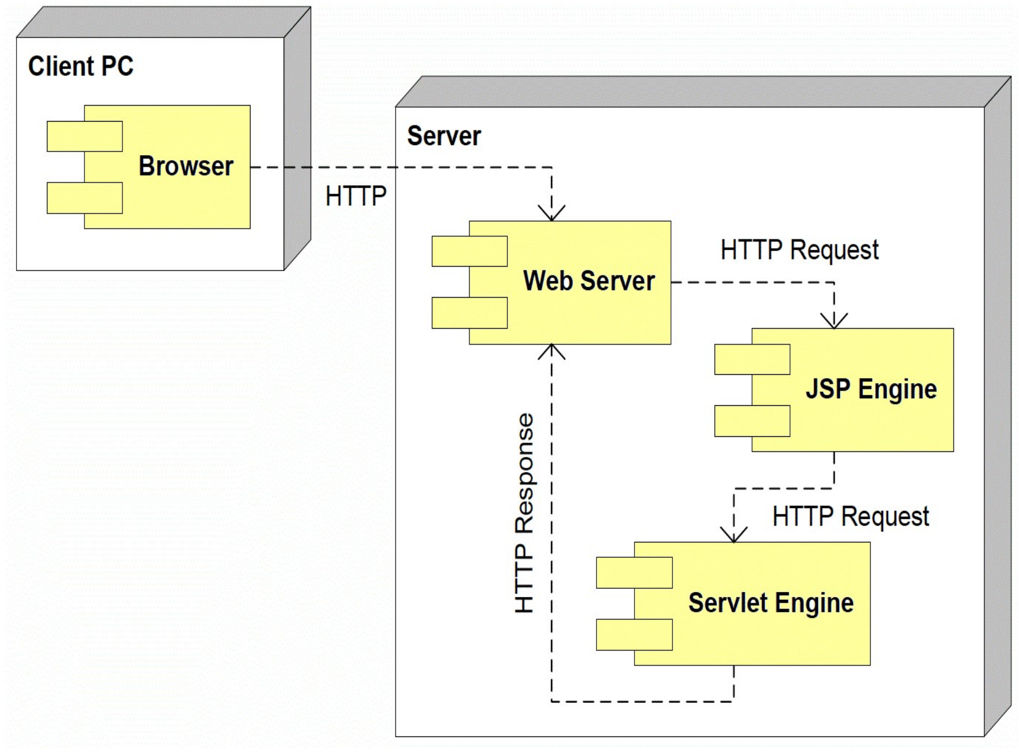
Student Activity Diagram:



Component Diagram:



Deployment diagram:



Data Dictionary

ADMIN REGISTRATION

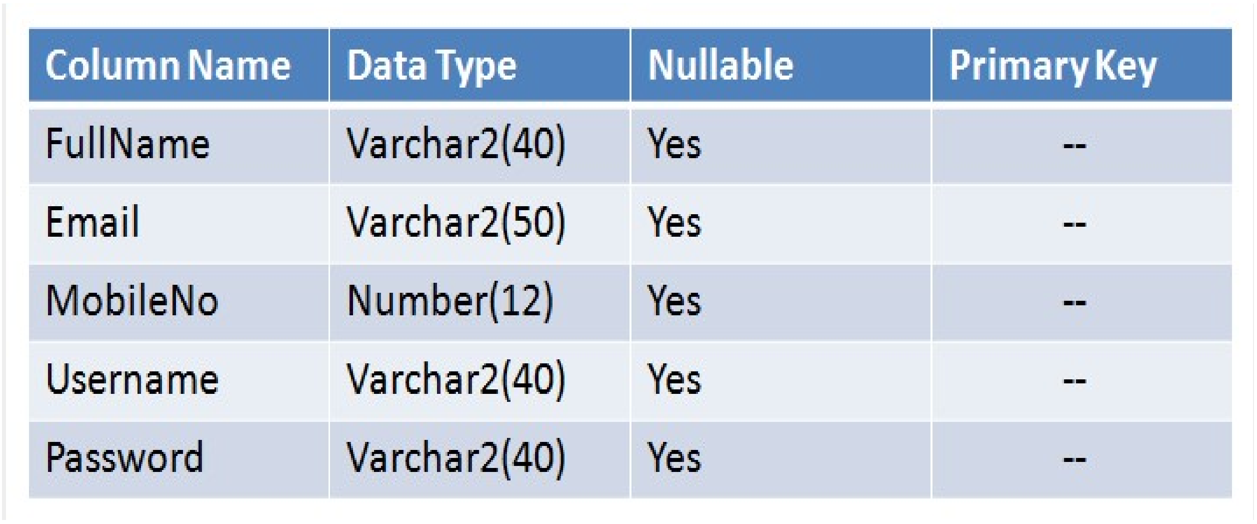
Column Name	Data Type	Nullable	Primary Key
Fullname	Varchar2(20)	Yes	--
MobileNo	Number (12)	Yes	--
Email	Varchar2(40)	Yes	--
SecurityQue	Varchar2(50)	Yes	--
Answer	Varchar2(50)	Yes	--
Username	Varchar2(50)	Yes	--
Password	Varchar2(50)	Yes	--

ADD QUESTIONS:

Column Name	Data Type	Nullable	Primary Key
Question	Varchar2(70)	Yes	--
OptionA	Varchar2(40)	Yes	--
OptionB	Varchar2(40)	Yes	--

OptionC	Varchar2(40)	Yes	--
OptionD	Varchar2(40)	Yes	--
Answer	Varchar2(40)	Yes	--

STUDENT RECORD:



Column Name	Data Type	Nullable	Primary Key
FullName	Varchar2(40)	Yes	--
Email	Varchar2(50)	Yes	--
MobileNo	Number(12)	Yes	--
Username	Varchar2(40)	Yes	--
Password	Varchar2(40)	Yes	--

HTML

HTML, an initialism of Hypertext Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document — by denoting certain text as headings, paragraphs, lists, and so on — and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of labels (known as tags), surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behavior of web browsers and other HTML processors.

HTML is also often used to refer to content of the MIME type text/html or even more broadly as a generic term for HTML whether in its XML-descended form (such as XHTML 1.0 and later) or its form descended directly from SGML

Hyper Text Markup Language

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

Basic HTML Tags:

<! -- -->	specifies comments
<A>……….</A>	Creates hypertext links
<B>……….</B>	Formats text as bold
<BIG>……….</BIG>	Formats text in large font.
<BODY>…</BODY>	Contains all tags and text in the HTML document
<CENTER>...</CENTER>	Creates text
<DD>…</DD>	Definition of a term
<DL>...</DL>	Creates definition list
<FONT>…</FONT>	Formats text with a particular font
<FORM>...</FORM>	Encloses a fill-out form
<FRAME>...</FRAME>	Defines a particular frame in a set of frames
<H#>…</H#>	Creates headings of different levels(1 – 6)
<HEAD>...</HEAD>	Contains tags that specify information about a document
<HR>...</HR>	Creates a horizontal rule
<HTML>…</HTML>	Contains all other HTML tags
<META>...</META>	Provides meta-information about a document
<SCRIPT>…</SCRIPT>	Contains client-side or server-side script
<TABLE>…</TABLE>	Creates a table

<TD>…</TD>	Indicates table data in a table
<TR>…</TR>	Designates a table row
<TH>…</TH>	Creates a heading in a table

Attributes

The attributes of an element are name-value pairs, separated by "=", and written within the start label of an element, after the element's name. The value should be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML).Leaving attribute values unquoted is considered unsafe.

Most elements take any of several common attributes: id, class, style and title. Most also take language-related attributes: lang and dir.

The id attribute provides a document-wide unique identifier for an element. This can be used by stylesheets to provide presentational properties, by browsers to focus attention on the specific element or by scripts to alter the contents or presentation of an element. The class attribute provides a way of classifying similar elements for presentation purposes. For example, an HTML document (or a set of documents) may use the designation class="notation" to indicate that all elements with this class value are all subordinate to the main text of the document (or documents). Such notation classes of elements might be gathered together and presented as footnotes on a page, rather than appearing in the place where they appear in the source HTML.

An author may use the style non-attributal codes presentational properties to a particular element. It is considered better practice to use an element's son- id page and select the element with a stylesheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach subtextual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tooltip. The generic inline span element can be used to demonstrate these various non-attributes.

The preceding displays as HTML (pointing the cursor at the abbreviation should display the title text in most browsers).

Advantages

- A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.
- HTML is platform independent.
- HTML tags are not case-sensitive.

JavaScript

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browser's display accordingly

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

```
<SCRIPTS>.. </SCRIPT>.

<SCRIPT LANGUAGE = "JavaScript">

JavaScript statements

</SCRIPT>
```

Here are a few things we can do with JavaScript:

- ➢ Validate the contents of a form and make calculations.
- ➢ Add scrolling or changing messages to the Browser's status line.
- ➢ Animate images or rotate images that change when we move the mouse over them.
- ➢ Detect the browser in use and display different content for different browsers.
- ➢ Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.

JavaScript Vs Java

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

Advantages

- JavaScript can be used for Sever-side and Client-side scripting.
- It is more flexible than VBScript.
- JavaScript is the default scripting languages at Client-side since all the browsers supports it.

Platform Independent Technologies:

These are the Technologies which will allow different Operating Systems to execute the applications. i.e. It is possible to use different Operating Systems for compilation & execution of its applications .

To execute the Compiled java program we have to convert neutral byte code to local Operating Systems native representations . To achieve this sun micro system has introduced a tool called as JVM(JAVA VIRTUAL MACHINE).Java is a platform independent Technology due to the availability of jvm's But jvm is platform dependent.

Architecture:

The hierarchy of H/w Components is called as Architecture. On the basis of architecture there are 2 types of Technologies.

1) Architectural Dependent Technologies: These are the Technologies ,which should require the same architecture for Compilation & for Execution.

Ex:c,c++,Pascal……..

2. Architectural Neutral Technologies:

These are the Technologies, which will allow different architectures for Compilation & for Execution of it's applications.

Ex:java.

Introduction to OOPs

Object Oriented Programming or OOP is the technique to create programs based on the real world. Unlike procedural programming, here in the OOP programming model programs are organized around objects and data rather than actions and logic. Objects represent some concepts or things and like any other objects in the real Objects in programming language have certain behavior, properties, type, and identity. In OOP based language the principal aim is to find out the objects to manipulate and their relation between each other. OOP offers greater flexibility and compatibility and is popular in developing larger application. Another important work in OOP is to classify objects into different types according to their properties and behavior. So OOP based software application development includes the analysis of the problem, preparing a solution, coding and finally its maintenance.

Java is a object oriented programming and to understand the functionality of OOP in Java, we first need to understand several fundamentals related to objects. These include class, method, inheritance, encapsulation, abstraction, polymorphism etc.

Class - It is the central point of OOP and that contains data and codes with behavior. In Java everything happens within class and it describes a set of objects with common behavior. The class definition describes all the properties, behavior, and identity of objects present within that class. As far as types of classes are concerned, there are predefined classes in languages like C++ and Pascal. But in Java one can define his/her own types with data and code.

Object - Objects are the basic unit of object orientation with behavior, identity. As we mentioned above, these are part of a class but are not the same. An object is expressed by the variable and methods within the objects. Again these variables and methods are distinguished from each other as instant variables, instant methods and class variable and class methods.

Methods - We know that a class can define both attributes and behaviors. Again attributes are defined by variables and behaviors are represented by methods. In other words, methods define the abilities of an object.

Inheritance –Getting the properties and behaviors from one class to another class is called Inheritance.

The Main Advantage of Inheritance is to provide the code Reusability.

Inheritance is also called IS-A Relationship.

By using extends keyword we can achieve the Inheritance.

Abstraction - Abstraction is used to hide certain details and only show the essential features of the object. In other words, it deals with the outside view of an object (interface).

Encapsulation – Encapsulation is a process of binding or wrapping the data and the codes that operates on the data into a single entity.

Encapsulation is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. For this reason, encapsulation is also referred to as data hiding.

Encapsulation can be described as a protective barrier that prevents the code and data being randomly accessed by other code defined outside the class. Access to the data and code is tightly controlled by an interface.

The main benefit of encapsulation is the ability to modify our implemented code without breaking the code of others who use our code. With this feature Encapsulation gives maintainability, flexibility and extensibility to our code.

Polymorphism –

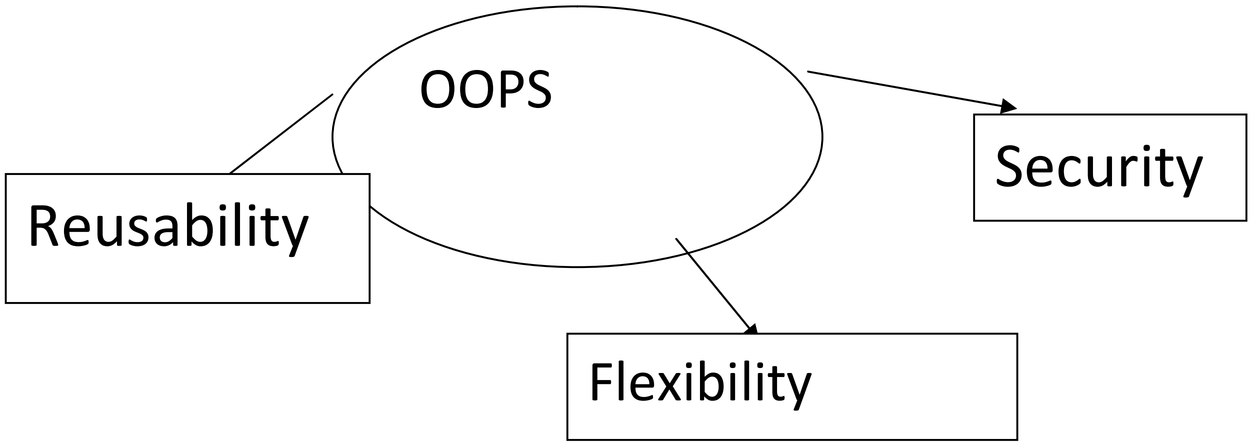
Polymorphism is a Greek word.Poly means many, morphism means forms.

Polymorphism means the ability to get the multiple forms.

The main advantage of Polymorphism is

To provide the Flexibility.

The Main Advantage of OOPS



What is an Abstract Class?

An abstract class is a special kind of class that cannot be instantiated. So the question is why we need a class that cannot be instantiated? An abstract class is only to be sub-classed (inherited from). In other words, it only allows other classes to inherit from it but cannot be instantiated. The advantage is that it enforces certain hierarchies for all the subclasses. In simple words, it is a kind of contract that forces all the subclasses to carry on the same hierarchies or standards.

What is an Interface?

An interface is not a class. It is an entity that is defined by the word Interface. An interface has no implementation; it only has the signature or in other words, just the definition of the methods without the body. As one of the similarities to Abstract class, it is a contract that is used to define hierarchies for all subclasses or it defines specific set of methods and their arguments. The main difference between them is that a class can implement more than one interface but can only inherit from one abstract class. Since C# doesn't support multiple inheritance, interfaces are used to implement multiple inheritance.

Feature	Interface	Abstract class
Multiple inheritance	A class may inherit several interfaces.	A class may inherit only one abstract class.
Default implementation	An interface cannot provide any code, just the signature.	An abstract class can provide complete, default code and/or just the details that have to be overridden.
Access Modfiers	An interface cannot have access modifiers for the subs, functions, properties etc everything is assumed as public	An abstract class can contain access modifiers for the subs, functions, properties
Core VS Peripheral	Interfaces are used to define the peripheral abilities of a class. In other words both Human and Vehicle can inherit from a IMovable interface.	An abstract class defines the core identity of a class and there it is used for objects of the same type.
Homogeneity	If various implementations	If various implementations

	only share method signatures then it is better to use Interfaces.	are of the same kind and use common behaviour or status then abstract class is better to use.
Speed	Requires more time to find the actual method in the corresponding classes.	Fast
Adding functionality (Versioning)	If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method.	If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly.
Fields and Constants	No fields can be defined in interfaces	An abstract class can have fields and constrants defined

<u>What is JDBC:-</u>

JDBC(Java DataBase Connectivity) IS API,IT IS A SPECIFICATION.

JDBC is a process of interacting with the database from a java application.

JDbc is a Service Technology from sun micro systems that enables any kind of java program to communicate with any kind of database in standard manner.Any kind of java program psvm program,servlet program,Ejb program, applet program etc Any kind of database oracle ,ms access,mysql,Sybase etc..

The JDBC API comes in the form of java.sql javax.sql packages having classes and interfaces.

java.sql package:-

Interfaces---> Connection

Driver

Statement

PreparedStatement

Result Set

Callable Statement etc...

Classes-----> DriverManager,Date,Time,Type etc........

javax.sql package -->

interfaces----> RowSet

DataSource

RowSetWriter etc........

Classes------> ConnectionEvent

RowSetEvent   etc......

In JDBC applications we must specify the complete database logic in java application as for the Java API representations, later on we need to send Java represented database logic to the database engine(DBE).

DBE must execute the database logic but it was configured as per the java representations but DBE able to understand only Query Language representations.

At the above situation if we want to execute our database logic, we need to use one interface in between java application and the database, that interface must convert java representations to query language representations and query language Representations to java representations. Now this interface is called as a "Driver".

- **What is Driver? How many Drivers are available in JDBC? What are the types?**

Driver:-

It is a software or an interface .Jdbc Driver acts as a bridge between java app and database s/w and convert java calls(instructions) to database calls,vice versa..

Initially sun Microsystems has provided "driver interface" to the market with this sun Microsystems has given an intimation to all the database vendors to have their own implementation as per their requirements for the Driver interface.

As a response all the database vendors are providing their own implementation for the Driver interface in order to interact with the respective databases from a java application.

Types of Drivers:

Type 1 Driver(JDBC-ODBC bridge Driver)

Type 2 Driver(Native Api/part java Driver(or) partNative java Driver)

Type 3 Driver(Middleware database access server driver)

Type 4 Driver(Pure java Driver)

steps to write a JDBC application:-

1. load and register the driver.
2. Establish a connection between java application and the database.
3. prepare either statement object or Prepared Statement object or CallebleStatement object as per the application requirements.
4. write and execute the sql queries.
5. terminate the connection which we have established.

To load the driver's class byte code to the memory we will use the following method.

Public void forName(String class name)

Eg: Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Where forName() is a static method, which can be used to load the respective driver class byte code to the memory.Each and every driver class has already a static block included a method call DriverManager.registerDriver(Driver obj)

at the time of loading the respective Driver class byte codeto the memory automatically the static block could be executed, by this DriverManager.registerDriver(….) method will be executed as part of the static block.

By the execution of the registerDriver(….) method automatically the specified driver will be register to the jdbc application.

Note:- The best alternative for Class.forName(..) is

DriverManagre.registerDriver(new sun.jdbc.odbc.JdbcOdbcDriver());

To register the driver.

- **How to establish a Database connection between java application and Database?**

If we want to establish a connection between java application and the database we will the following piece of code.

Connection con=DriverManager.getConnection("jdbc:odbc:nag","nag","system","mana");

Where getConnection() is a static method from DriverManager class, it can be used to establish Connection b/w java app and database by calling connect() method as per the URL which we provided.public static connection getConnection(String url,String uname,String pwd)throws SqlException

Note:-

con=DriverManager.getConnection(-,-,-);

con is not the Object of java.sql Connection interface, it is the object of a java class supplied by underlying Jdbc driver implementing .java.sql.connection interface.

In general every jdbc driver should have their own driver class name and driver url.For type-I driver provided by the sun Microsystems.

Driver class name:- sun.jdbc.odbc.JdbcOdbcDriver

URL:- jdbc:odbc:dsn name

In general all the driver urls could be divided in to two parts.Main protocol which is fixed for every driver i.e. jdbc Sub protocol which is varied for every driver

- **What is the requirement to use Statement object?**

After establishing a connection between java application and the database we need to write the sql queries and we need to execute them.To execute the sql queries we will use some pre-defined library, which was defined in the form of Statement object, PreparedStatement object and CallableStatement object. As per the application requirements we need to create either Statement object or CallableStatement object and PreparedStatement object.

- **what is the difference between statement ,prepared statement,callable statement?**

When we have a requirement to execute the sql queries independently we should use stmt.When we have a requirement to execute same sql queries in the next sequence where to improve the performance of jdbc app we should use prepared stmt.When we have a requirement to access stored procedures and functions available of database from jdbc applications we should use callable stmt.

To create Statement object jdbc will use the following method from connection object.

public Statement createStatement()

Eg: Statement st = con.createStatement();

where createStatement () is non static method from Connection ,it can be used to return stmt object

- **How to execute SQL Queries from a java application?**

To execute the sql queries we will use the following methods from Statement object.

st.executeQuery(…)

st.executeUpdate(…)

st.execute(…)

- **What are the differences between executeQuery(…), executeUpdate(…) and execute(…) methods?**

where executeQuery() can be used to execute selection group sql queries to fetch the data from database.

When we use selection group sql query with executeQuery() then JVM will send that sql query to the database engine, database engine will execute it, by this database engine(DBE) will fetch the data from database and send back to the java application.

Java is a purely object oriented technology. That's why the jdbc application will maintain the fetched data from database, in the form of an object at heap memory, called as ResultSet object.

public ResultSet executeQuery(String sqlquery)

where executeUpdate() can be used to execute updation group sql query to update the database. When we provide updation group sql query as a parameter to executeUpdate(), then JVM will send that sql query to DBE, here DBE will execute it and perform updations on the database, by this DBE will identify the number of records got updated value called as "records updated count" and return back to the java application.

public int executeUpdate(String sqlquery)

where execute() can be used to execute either selection group sql queries or updation group queries.

When we use selection group sql

query with the execute() then we will get ResultSet object at heap

memory with the fetched data. But execute() will return “true” as a Boolean value.

When we use updation group sql query with execute() then we will get “ records updated count value” at jdbc application. But execute() will return “false” as a Boolean value.

public boolean execute(String sqlquery)

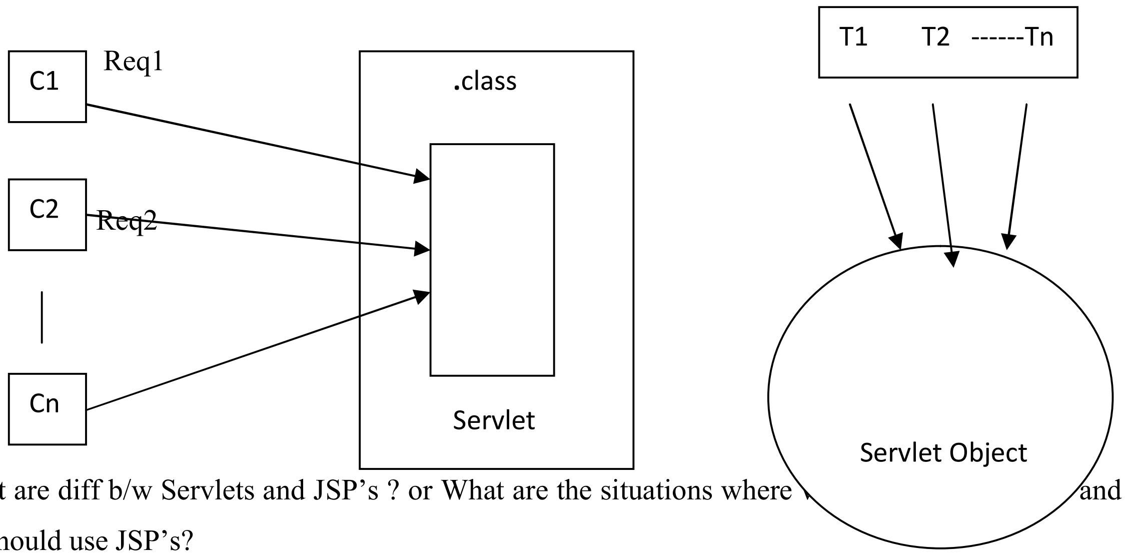
<u>Introduction to Servlets:</u>

Servlet is a server side technology which was designed on the basis of Java Technology. Here “Java Technology is a *thread based technology*” . So that servlet is a thread based technology.

If we deploy a servlet application at server machine for every new request from clients as a separate thread will be created on servlet object by the server.

Thread is the light weight component when compared to process, here if we increase no of requests to the same servlets, no of threads only we create on servlet object.

This approach will not increase over head to the server machine as a result performance of the server side application will be increased.



What are diff b/w Servlets and JSP’s ? or What are the situations where and where we should use JSP’s?

In case of the sevlets we are unable to separate both Presentation Logic and Business Logic.

But in case of JSP pages we are able to separate both Presentation Logic and Business Logic due to that clear cut separation b/w the tags which are for presentation logic and which are for business logic.

As past of the web application development sevlets are good at the time of pickup the request and process the request.

But JSP pages are good at the time of generating dynamic response to client machine with very good look and feel.

If we want to design any web application on the basis of MVC Architecture, where we should use a servlet as Controller and the set of JSP pages as View part.

If we made any modifications on the existed servlet then we should perform explicitly recompilation and reloading.

If we perform any modifications on the existed JSP's pages then we should not require to perform recompilation and reloading explicitly. Because JSP pages are Auto Compiled and Auto Loaded.

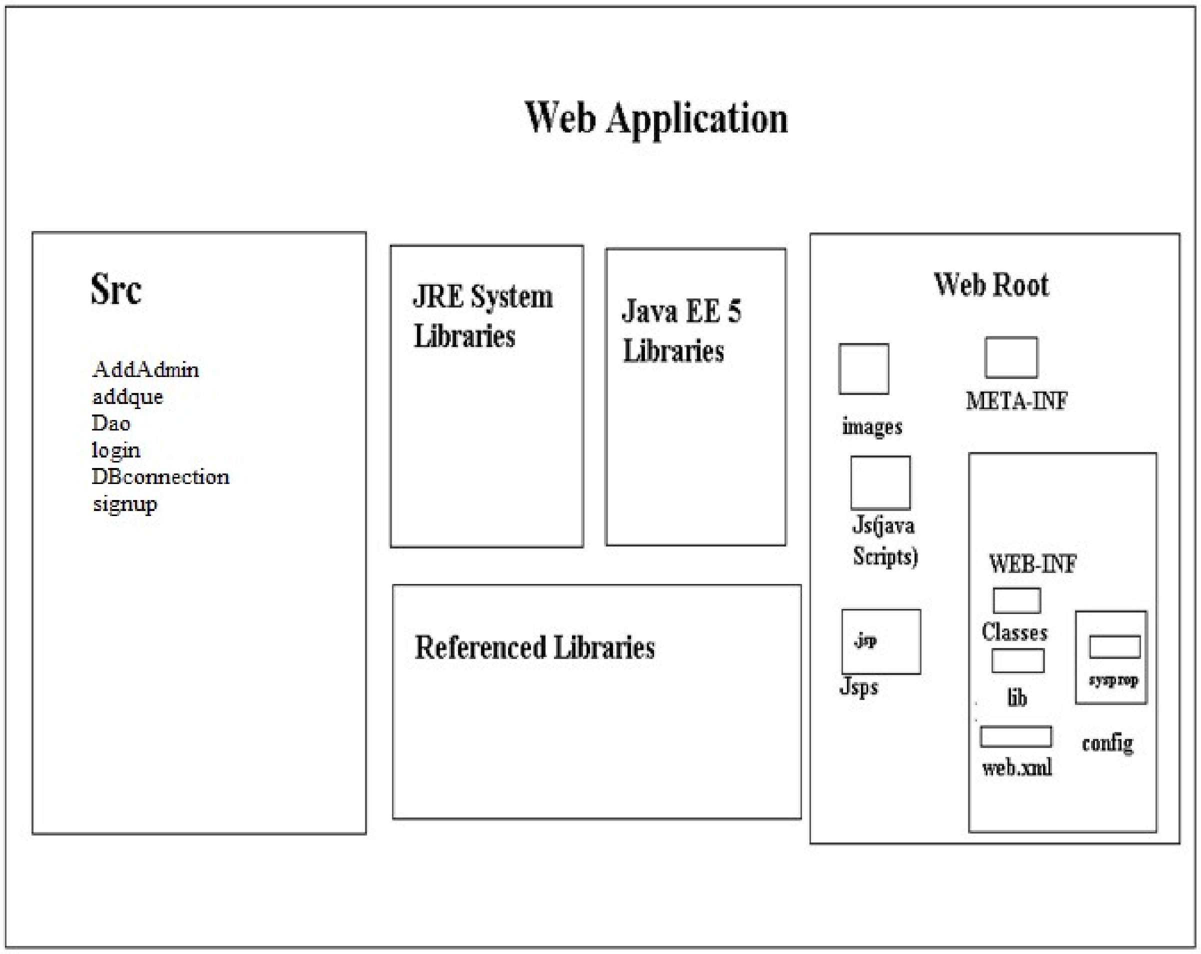
Steps to design First Web Application:

1. Design web application directory structure at server machine
2. Design deployment descriptor (ie : web.xml file)
3. Design web-resources which we require as for the application requirement
4. Start the server and access the web application from client

>Web application directory structure:

If we want to design any web application then we must use the following web application directory structure at server machine.

My Eclipse



When we install tomcat server on server machine then tomcat server will provide a folder like tomcat 6.0 called as "*Home-Directory*".

Home-Directory:

Tomcat home-directory includes the folders like

Bin

Lib

Conf

Logs

Work

Web-apps and -------etc

Where purpose of the web-apps folder is to accommodate all the user defined web applications

If we want to deploy any web application in server we have to prepare a folder with respect to our application under web-apps folder called as Application Folder (context root)

Where Application Folder includes

Themes -- > to store Cascading Style Sheets (CSS)

Images -- > to store .jpg, .gif

Literature -- > to store .doc files

Src -- > to store source files like .java and soon

Static resources -- > like .html

Dynamic resources -- > like .jsp

WEB-INF -- > to maintain a web application information

Where WEB-INF folder includes

web.xml files to provide meta data about application deployment.

Lib folder to store the application required .jar files

Classes folder to store .class files of servlets, filters, lateness

The above web application directory structure can be divided into the following two parts

<u>Public Area or Client Area</u>:

It is an area, come under outside of WEB-INF folder, and inside of Application Folder.

If we deploy any resource under public area then client able to access that resources by using their names directly.

<u>Private Area or Server Area:</u>

The area which come under inside WEB-INF folder is private area. If we deploy any resources under private area then client unable to access that resources with their names directly.

Example, In general we are able to deploy of servlets under classes folder i.e. private area, here if we want to access servlets then we must define on URL pattern in web.xml file through this URL pattern only client able to access the servlets.

- > Deployment Description:

Deployment description is a web.xml file, which will provide some description or meta data about the web application deployment.

In general web.xml file should include

Servlets Configuration

Filters Configuration

Listeniers Configuration

Welcome Files Definitions

Initialization Parameters

Context Parameters

Loader Startup Configuration

Session Constraints

Tag Library Definition and etc.

Web.xml file is deployment description which will includes the above specified configurations to execute the web application by the container

As part of the web applications web.xml file if optional, still if you want to provide deployment descriptor then whose name must be web.xml, it is not variable name

To define an URL pattern for a particular servlet and to provide mapping between servlets and URL patterns we should use the following XML Tags in web.xml file

```
<web-app>

<servlet>

<servlet-name>logical name </servlet>

<servlet-class>fully qualified name of servlet</servlet-class>

</serlvet>

<servlet-mapping>

<serlvet-name>logical name</servlet-name>

<url-pattern>/url pattern name</url-pattern>

</servlet-mapping>

---

----

</web-app>
```

To define URL pattern in web.xml file we should use the following three approaches.

Exact match method

Directory match method

Extension match method

Directory Match Method:

If we want to define URL Pattern in web.xml file with this approach then it must starts with '/' and ends with " * "

Ex: <url-pattern>/abc/*<url-pattern>

If we define URL Pattern in web.xml file with this approach then to access the respective servlet we must specify an URL Pattern at client URL, it must match with the prefix of the URL Pattern defined in web.xml file and may end with any thing.

Ex: http://localhost:8086/app1/abc - - >valid

http://localhost:8086/app1/abc /xyz- - >valid

Note:

As part of the web apps when we have a requirement to send group of requests to a particular server side component then we must use directory match method URL Pattern definition.

As part of the web apps in general we will use filters to provide pre processing of the request processing like authentication checks, security, data encryption, data compression. In this context in web apps it is required to use a single filter for many more no of web resources.

In the above context when we send requests to the web resources automatically all the requests must be gone through the respective filter. To achieve this requirement we must use directory match method to define URL Pattern in the respective filter configuration.

Extension Match Method:

In extension match method if we want to define any URL Pattern in web.xml file then it must starts with * and it must ends with a particular extension.

Ex: <url –pattern> * .xyz</url-pattern>

If we define URL Pattern with this approach in web.xml file to any specific resource then to access that resource we must specify an URL Pattern at client address bar, It may start with anything but it must end with the specified extension in web.xml file.

Ex: http://localhost:8086/abc.xyz - ->valid http;//localhost:8086/abc/xyz - ->invalid

http://localhost:8086/a.xyz - ->valid

http://localhost:8086/b.abc - ->invalid

Note;

As part of the web apps when we have a requirement to pass all the no of requests to a particular web resource, where to define an URL Pattern for the web resources we should use extension match method

In general in MVC based web apps we must send all the requests to the controller servlet, in this situation to trap all the no of requests t the controller servlet we must use extension match method to define URL Pattern for the controller

- >**Step3: Design Web Resources:**

As part of the web apps in general we are going to provide web app logic or business logic in the form of web resources like servlets, jsp's…

To prepare servlets in web apps servlet API has provided following predefined library as part of the packages like javax.servlet and java

>Design web Resources:

As part of the web apps in general we are going to provide web app logic or business logic in the form of web resources like servlets, jsp's,--

To prepare servlets in web apps servlet API has provided following predefined library as part of the packages like javax.servlet and javax.servlet.http

WHAT IS SERVLET? HOW MANY WAYS WE ARE ABLE TO PREPARE SERVLET:

Servlet is an object available at server machine, which was implemented either directly or indirectly servlet interface.

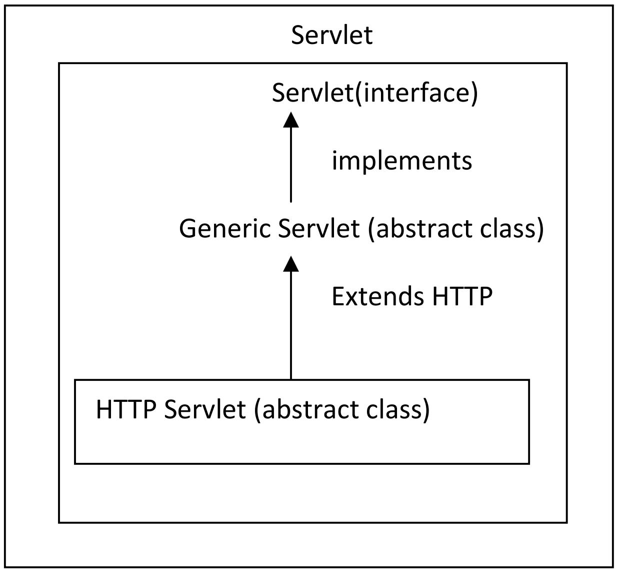
As per the predefined library provided by servlet API are three ways to prepare servlets

Implementing Servlet Interface

Extending Generic Servlet Abstract Class

Extending HTTP Servlet

Javax


Servlet
Servlet(interface)
implements
Generic Servlet (abstract class)
Extends HTTP
HTTP Servlet (abstract class)

Implementing Servlet Interface:

In this approach to prepare servlet we must take an userdefined class, which must implement servlet interface.

```
public class MyServlet implements Servlet
{


}
```

Extending Generic Servlet Abstract Class:

To prepare servlets with this approach we must define a class which must be a subclass to generic servlet abstract class

public class MySerVlet extends Generic Servlet {
}

Extending HTTP Servlet:

In this approach to design servlet we must take an user defined class, which must be extended from http servlet abstract class.

Public class MyServlet extends HTTP Servlet

{

}

>Start the Server and Access the Web Application:

After designing the web app at server side we have to start the server to execute web app

To start the server we should use the following approaches

Execute either tomcat6.0 exe file or startup. Batch file as per the availability, which could be available in bin folder of tomcat home directory

Execute system service tomcat Start - - >Run- - >services.msc

Select tomcat6 - - >select the icons start service, stop service….

Execute system program apache tomcat's start tomcat

Start - - >All Programs - ->Apache Tomcat 6 - - >Monitor Tomcat/Start Tomcat

The following example demonstrates how to design a servlet by using servlet interface and how to access it from client

<u>JSP:</u>

Jsp is a server side technology which can be used to generate dynamic response from server machine.

The main intention of introducing jsp technology is to reduce java code as much as possible.

→In general in web app we are able to utilize JSp pages

To generate dynamic response to the clinets with very good look and feel.

If u made any modifications on existed JSp pages,then

It is not required to perform recompilation and reloading because JSP pages are autocompiled and autoloaded.

DEPLOYMENT:

In general in web applications it is possible to deploy jsp pages at any location of the web.

Application directory structure but it is suggestable to deploy the jsp pages under Application floder.

If we deploy the jsp pages under application floder then we are able to access that jsp pages from client with respect with it's name directly.

If we deploy any jsp page under private area that is under web-inf folder (or)classes folder then we are able to acess respect to jsp page by defining on url-pattern in web.xml file.

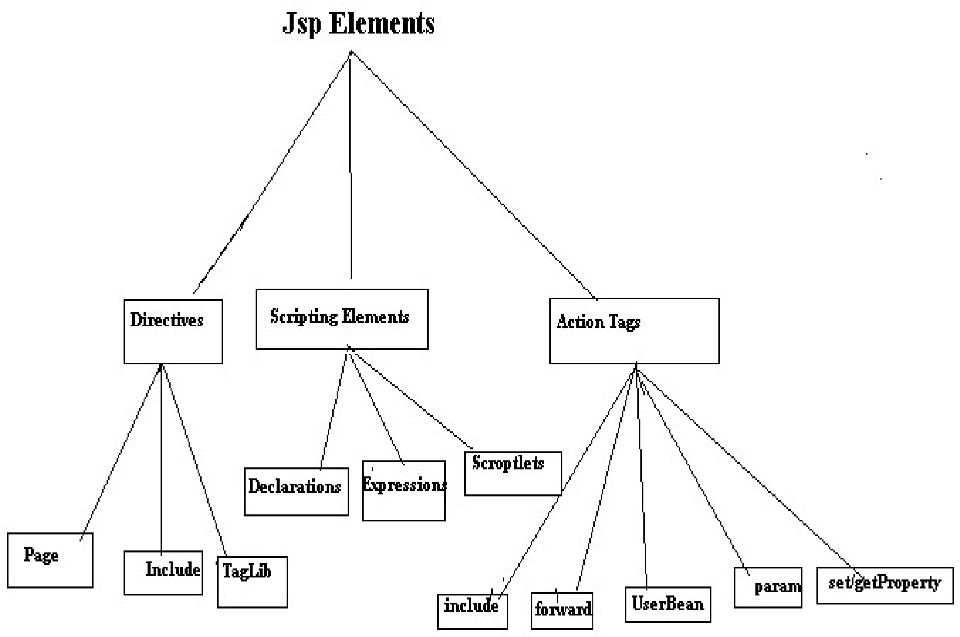
To define url pattern and to provide mapping between url-pattern and the respect to jsp-page we should use the following tags in web.xml.

```
<web-app>

<servlet>

<servlet-name>logicalname</servlet-name>

<jsp-file>/context relation path& jsp page</jsp-file>

</servlet>

<servlet-mapping>
```

```
<servlet-name>logical servlet</servlet-mapping>

<url-pattern>/pattern name</url-pattren>

</servlet-name>

</web-app>
```

EX:

```
<web-app>

<servlet>

<servlet-name>f</servlet-name>

<jsp-file>/web-INF/classes/first.jsp</jsp-file>

</servlet>

<servlet-mapping>

<servlet-name>f</servlet-name>

<url-pattern>/first</url-pattren>

</servlet-mapping>

</web-app>
```



JSP ACTIONS:

→In jsp technology there are two types of actions .

1)standard Actions

2) custom Actions

1)standard Actions:

These are the actions which could be provided by the jsp technology along with the software.

→jsp technology has provided all the standard

Actions in the form of a set of predefined tags called as Action tags.

<jsp : useBean----->

<jsp: setProperty…..>

<jsp:getProperty……>

<jsp:include…>

<jsp:forward….>

<jsp:param…>

<jsp:plugin…>

<jsp:fallback..>

<jsp:params…>

<jsp:declarations…>

<jsp:scriptlet….>

Jsp:useBean Tag:

→the main purpose of this Tag is to interact with object from a jsp page to set and get object state.

→syntax:

<jsp:useBean id :"----"class:"……" scope:"….."/>

→where id attribute can be used to specify a variable to hold the generated BEAN OBJECT REFERNCE.

Jsp:getProperty:
this tag can be used to getproperties of bean object.

<jsp:getProperty> contains the following 2 Attributes.

Name:

The name of the bean instance from which the required property is obtained.

It is equal to Id Attribute value of the <jsp:useBean>

Property:

The name of the java Bean property which has to retrieve.

<jsp:setProperty>

This Action can be used to set the properties of a bean object.

→we can use <jsp:setProperty> in the following way.

EX:
<jsp:setProperty name=”e” property=”ename” param=”ename”/>

1.name →it refers name of the Bean object whose property has to set.

->this is exactly same as id Attribute of<jsp:useBean>

→it is mandatory.

2.property :

The name of the java Bean property which has to be set.

it is mandatory.

3.value: it specifies the value which has to set to the javaBean property.

it is optional.

4. param:

This attribute specifies the name of request parameter whose value has to set to Bean property.

it is optional.

Database:

A database management system (DBMS) is computer software designed for the purpose of managing databases, a large set of structured data, and run operations on the data requested by numerous users. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, Firebird, PostgreSQL, MySQL, SQLite, FileMaker and Sybase Adaptive Server Enterprise. DBMSs are typically used by Database administrators in the creation of Database systems. Typical examples of DBMS use include accounting, human resources and customer support systems.

Originally found only in large companies with the computer hardware needed to support large data sets, DBMSs have more recently emerged as a fairly standard part of any company back office.

Description

A DBMS is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. A DBMS includes:

- ✓ A modeling language to define the schema of each database hosted in the DBMS, according to the DBMS data model.
    - The four most common types of organizations are the hierarchical, network, relational and object models. Inverted lists and other methods are also used. A given database management system may provide one or more of the four models. The optimal structure depends on the natural organization of the application's data, and on the application's requirements (which include transaction rate (speed), reliability, maintainability, scalability, and cost).
    - The dominant model in use today is the ad hoc one embedded in SQL, despite the objections of purists who believe this model is a corruption of the relational model, since it violates several of its fundamental principles for the sake of practicality and performance. Many DBMSs also support the Open Database Connectivity API that supports a standard way for programmers to access the DBMS.
- ✓ Data structures (fields, records, files and objects) optimized to deal with very large amounts of data stored on a permanent data storage device (which implies relatively slow access compared to volatile main memory).

- ✓ A database query language and report writer to allow users to interactively interrogate the database, analyze its data and update it according to the users privileges on data.
    - It also controls the security of the database.
    - Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called subschemas. For example, an employee database can contain all the data about an individual employee,

but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data.

- If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases. However, it may not leave an audit trail of actions or provide the kinds of controls necessary in a multi-user organization. These controls are only available when a set of application programs are customized for each data entry and updating function.

✓ A transaction mechanism, that ideally would guarantee the ACID properties, in order to ensure data integrity, despite concurrent user accesses (concurrency control), and faults (fault tolerance).

- It also maintains the integrity of the data in the database.
- The DBMS can maintain the integrity of the database by not allowing more than one user to update the same record at the same time. The DBMS can help prevent duplicate records via unique index constraints; for example, no two customers with the same customer numbers (key fields) can be entered into the database. See ACID properties for more information (Redundancy avoidance).

The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

When a DBMS is used, information systems can be changed much more easily as the organization's information requirements change. New categories of data can be added to the database without disruption to the existing system.

Organizations may use one kind of DBMS for daily transaction processing and then move the detail onto another computer that uses another DBMS better suited for random inquiries and analysis. Overall systems design decisions are performed by data administrators and systems analysts. Detailed database design is performed by database administrators.

Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with RAID disk arrays used for stable storage. Connected to one or more servers via a high-speed channel, hardware database accelerators are also used in large volume transaction processing environments.

DBMSs are found at the heart of most database applications. Sometimes DBMSs are built around a private multitasking kernel with built-in networking support although nowadays these functions are left to the operating system.

SQL

Structured Query Language (SQL) is the language used to manipulate relational databases. SQL is tied very closely with the relational model.

In the relational model, data is stored in structures called relations or tables.

SQL statements are issued for the purpose of:

Data definition: Defining tables and structures in the database (DDL used to create, alter and drop schema objects such as tables and indexes).

Data manipulation: Used to manipulate the data within those schema objects (DML Inserting, Updating, Deleting the data, and Querying the Database).

A schema is a collection of database objects that can include: tables, views, indexes and sequences

List of SQL statements that can be issued against an Oracle database schema are:

- **ALTER** - Change an existing table, view or index definition (DDL)
- **AUDIT** - Track the changes made to a table (DDL)
- **COMMENT** - Add a comment to a table or column in a table (DDL)
- **COMMIT** - Make all recent changes permanent (DML - transactional)
- **CREATE** - Create new database objects such as tables or views (DDL)
- **DELETE** - Delete rows from a database table (DML)

- **DROP** - Drop a database object such as a table, view or index (DDL)
- **GRANT** - Allow another user to access database objects such as tables or views (DDL)
- **INSERT** - Insert new data into a database table (DML)
- **No AUDIT** - Turn off the auditing function (DDL)
- **REVOKE** - Disallow a user access to database objects such as tables and views (DDL)
- **ROLLBACK** - Undo any recent changes to the database (DML - Transactional)
- **SELECT** - Retrieve data from a database table (DML)
- **TRUNCATE** - Delete all rows from a database table (can not be rolled back) (DML)
- **UPDATE** - Change the values of some data items in a database table (DML)

Normalization:

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

What is an "inconsistent dependency"? While it is intuitive for a user to look in the Customers table for the address of a particular customer, it may not make sense to look there for the salary of the employee who calls on that customer. The employee's salary is related to, or dependent on, the employee and thus should be moved to the Employees table. Inconsistent dependencies can make data difficult to access because the path to find the data may be missing or broken.

There are a few rules for database normalization. Each rule is called a "normal form." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

The following descriptions include examples.

First Normal Form :

Eliminate repeating groups in individual tables.

Create a separate table for each set of related data.

Identify each set of related data with a primary key.

Do not use multiple fields in a single table to store similar data. For example, to track an inventory item that may come from two possible sources, an inventory record may contain fields for Vendor Code 1 and Vendor Code 2.

What happens when you add a third vendor? Adding a field is not the answer; it requires program and table modifications and does not smoothly accommodate a dynamic number of vendors. Instead, place all vendor information in a separate table called Vendors, then link inventory to vendors with an item number key, or vendors to inventory with a vendor code key.

Second Normal Form :

Create separate tables for sets of values that apply to multiple records.

Relate these tables with a foreign key.

Records should not depend on anything other than a table's primary key (a compound key, if necessary). For example, consider a customer's address in an accounting system. The address is needed by the Customers table, but also by the Orders, Shipping, Invoices, Accounts Receivable, and Collections tables. Instead of storing the customer's address as a separate entry in each of these tables, store it in one place, either in the Customers table or in a separate Addresses table.

Third Normal Form :

Eliminate fields that do not depend on the key.

Values in a record that are not part of that record's key do not belong in the table. In general, any time the contents of a group of fields may apply to more than a single record in the table, consider placing those fields in a separate table.

For example, in an Employee Recruitment table, a candidate's university name and address may be included. But you need a complete list of universities for group mailings. If university information is stored in the Candidates table, there is no way to list universities with no current candidates. Create a separate Universities table and link it to the Candidates table with a university code key.

EXCEPTION: Adhering to the third normal form, while theoretically desirable, is not always practical. If you have a Customers table and you want to eliminate all possible interfield dependencies, you must create separate tables for cities, ZIP codes, sales representatives, customer classes, and any other factor that may be duplicated in multiple records. In theory, normalization is worth pursing. However, many small tables may degrade performance or exceed open file and memory capacities.

It may be more feasible to apply third normal form only to data that changes frequently. If some dependent fields remain, design your application to require the user to verify all related fields when any one is changed.

Other Normalization Forms :

Fourth normal form, also called Boyce Codd Normal Form (BCNF), and fifth normal form do exist, but are rarely considered in practical design. Disregarding these rules may result in less than perfect database design, but should not affect functionality.

Eclipse IDE

Eclipse is an open-source software framework written primarily in Java. In its default form it is an Integrated Development Environment (IDE) for Java developers, consisting of the Java Development Tools (JDT) and the Eclipse Compiler for Java (ECJ). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs are available for over a dozen languages.

Architecture

The basis for Eclipse is the Rich Client Platform (RCP). The following components constitute the rich client platform:

- ✓ OSGi - a standard bundling framework
- ✓ Core platform - boot Eclipse, run plug-ins
- ✓ the Standard Widget Toolkit (SWT) - a portable widget toolkit
- ✓ JFace - viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors
- ✓ the Eclipse Workbench - views, editors, perspectives, wizards

Eclipse's widgets are implemented by a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also leverages an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the rich client platform, in contrast to some other applications where functionality is typically hard coded. This plug-in mechanism is a lightweight software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX, networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK.

The key to the seamless integration of tools with Eclipse is the plugin. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that a plug-in you develop integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are created equal.

The Eclipse SDK includes the Eclipse Java Development Tools, offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filespace allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards. The Visual Editor project allows interfaces to be created interactively, hence allowing Eclipse to be used as a RAD tool.

The following is a list of notable projects and plugins for the Eclipse IDE.

These projects are maintained by the Eclipse community and hosted by the Eclipse Foundation.

TESTING

Software Testing is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors. Quality is not an absolute; it is value to some person. With that in mind, testing can never completely establish the correctness of arbitrary computer software; testing furnishes a criticism or comparison that compares the state and behavior of the product against a specification. An important point is that software testing should be distinguished from the separate discipline of Software Quality Assurance (SQA), which encompasses all business process areas, not just testing.

There are many approaches to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating and following routine procedure. One definition of testing is "the process of questioning a product in order to evaluate it", where the "questions" are operations the tester attempts to execute with the product, and the product answers with its behavior in reaction to the probing of the tester[citation needed]. Although most of the intellectual processes of testing are nearly identical to that of review or inspection, the word testing is connoted to mean the dynamic analysis of the product—putting the product through its paces. Some of the common quality attributes include capability, reliability, efficiency, portability, maintainability, compatibility and usability. A good test is sometimes described as one which reveals an error; however, more recent thinking suggests that a good test is one which reveals information of interest to someone who matters within the project community.

Introduction:

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

Software testing may be viewed as a sub-field of Software Quality Assurance but typically exists independently (and there may be no SQA areas in some companies). In SQA, software process specialists and auditors take a broader view on software and its development. They examine and change the software engineering process itself to reduce the amount of faults that end up in the code or deliver faster.

Regardless of the methods used or level of formality involved the desired result of testing is a level of confidence in the software so that the organization is confident that the software has an acceptable defect rate. What constitutes an acceptable defect rate depends on the nature of the software. An arcade video game designed to simulate flying an airplane would presumably have a much higher tolerance for defects than software used to control an actual airliner.

A problem with software testing is that the number of defects in a software product can be very large, and the number of configurations of the product larger still. Bugs that occur infrequently are difficult to find in testing. A rule of thumb is that a system that is expected to function without faults for a certain length of time must have already been tested for at least that length of time. This has severe consequences for projects to write long-lived reliable software.

A common practice of software testing is that it is performed by an independent group of testers after the functionality is developed but before it is shipped to the customer. This practice often results in the testing phase being used as project buffer to compensate for project delays. Another practice is to start software testing at the same moment the project starts and it is a continuous process until the project finishes.

Another common practice is for test suites to be developed during technical support escalation procedures. Such tests are then maintained in regression testing suites to ensure that future updates to the software don't repeat any of the known mistakes.

It is commonly believed that the earlier a defect is found the cheaper it is to fix it.

Unit tests are maintained along with the rest of the software source code and generally integrated into the build process (with inherently interactive tests being relegated to a partially manual build acceptance process).

The software, tools, samples of data input and output, and configurations are all referred to collectively as a test harness.

History

The separation of debugging from testing was initially introduced by Glen ford J. Myers in his 1978 book the "Art of Software Testing". Although his attention was on breakage testing it illustrated the desire of the software engineering community to separate fundamental development activities, such as debugging, from that of verification. Drs. Dave Gelperin and William C. Hetzel classified in 1988 the phases and goals in software testing as follows: until 1956 it was the debugging oriented period, where testing was often associated to debugging: there was no clear difference between testing and debugging. From 1957-1978 there was the demonstration oriented period where debugging and testing was distinguished now - in this period it was shown, that software satisfies the requirements. The time between 1979-1982 is announced as the destruction oriented period, where the goal was to find errors. 1983-1987 is classified as the evaluation oriented period: intention here is that during the software lifecycle a product evaluation is provided and measuring quality. From 1988 on it was seen as prevention oriented period where tests were to demonstrate that software satisfies its specification, to detect faults and to prevent faults. Dr. Gelperin chaired the IEEE 829-1988 (Test Documentation Standard) with Dr. Hetzel writing the book "The Complete Guide of Software Testing". Both works were pivotal in to today's testing culture and remain a consistent source of reference. Dr.

Gelperin and Jerry E. Durant also went on to develop High Impact Inspection Technology that builds upon traditional Inspections but utilizes a test driven additive.

Testing Concepts

- ***Testing***

- ***Testing Methodologies***

  - Black box Testing:
  - White box Testing.
  - Gray Box Testing.

- ***Levels of Testing***

- Unit Testing.
- Module Testing.
- Integration Testing.
- System Testing.
- User Acceptance Testing.

- ***Types Of Testing***

  - Smoke Testing.
  - Sanitary Testing.
  - Regression Testing.
  - Re-Testing.
  - Static Testing.
  - Dynamic Testing.
  - Alpha-Testing.
  - Beta-Testing.
  - Monkey Testing.
  - Compatibility Testing.
  - Installation Testing.
  - Adhoc Testing.
  - Ext….

TCD (Test Case Documentation)

- ***STLC***
  - Test Planning.
  - Test Development.
  - Test Execution.
  - Result Analysis.
  - Bug-Tracing.

- Reporting.

- ***Microsoft Windows – Standards***
- ***Manual Testing***
- ***Automation Testing (Tools)***
    - Win Runner.
    - Test Director.

Testing:

- The process of executing a system with the intent of finding an error.
- Testing is defined as the process in which defects are identified, isolated, subjected for rectification and ensured that product is defect free in order to produce the quality product and hence customer satisfaction.
- Quality is defined as justification of the requirements
- Defect is nothing but deviation from the requirements
- Defect is nothing but bug.
- Testing --- The presence of bugs
- Testing can demonstrate the presence of bugs, but not their absence
- Debugging and Testing are not the same thing!
- Testing is a systematic attempt to break a program or the AUT
- Debugging is the art or method of uncovering why the script /program did not execute properly.

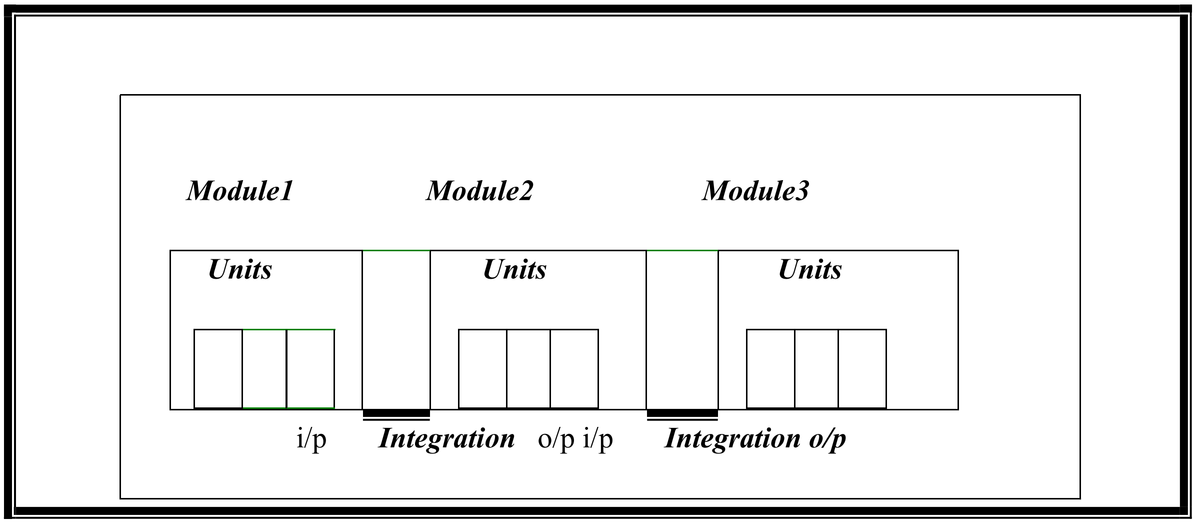
Testing Methodologies:

- **Black box Testing**: is the testing process in which tester can perform testing on an application without having any internal structural knowledge of application.
  Usually Test Engineers are involved in the black box testing.

- **White box Testing**: is the testing process in which tester can perform testing on an application with having internal structural knowledge.
  Usually The Developers are involved in white box testing.

- **Gray Box Testing**: is the process in which the combination of black box and white box tonics' are used.

Levels of Testing:



System Testing: Presentation + business +Databases

UAT: user acceptance testing

STLC (SOFTWARE TESTING LIFE CYCLE)

Test Planning:

1.Test Plan is defined as a strategic document which describes the procedure how to perform various testing on the total application in the most efficient way.

2.This document involves the scope of testing,

3. Objective of testing,

4. Areas that need to be tested,

5. Areas that should not be tested,

6. Scheduling Resource Planning,

7. Areas to be automated, various testing tools

Used….

Test Development:

1. Test case Development (check list)

2. Test Procedure preparation. (Description of the Test cases).

1. Implementation of test cases. Observing the result.

Result Analysis: **1.** Expected value: is nothing but expected behavior Of application.

2. Actual value: is nothing but actual behavior of application

Bug Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

Types Of Testing:

> **Smoke Testing**: is the process of initial testing in which tester looks for the availability of all the functionality of the application in order to perform detailed testing on them. (Main check is for available forms)

> **Sanity Testing:** is a type of testing that is conducted on an application initially to check for the proper behavior of an application that is to check all the functionality are available before the detailed testing is conducted by on them.

> **Regression Testing:** is one of the best and important testing. Regression testing is the process in which the functionality, which is already tested before, is once again tested whenever some new change is added in order to check whether the existing functionality remains same.

>**Re-Testing:** is the process in which testing is performed on some functionality which is already tested before to make sure that the defects are reproducible and to rule out the environments issues if at all any defects are there.

Static Testing: is the testing, which is performed on an application when it is not been executed.ex: GUI, Document Testing

Dynamic Testing: is the testing which is performed on an application when it is being executed.ex: Functional testing.

Alpha Testing: it is a type of user acceptance testing, which is conducted on an application when it is just before released to the customer.

Beta-Testing: it is a type of UAT that is conducted on an application when it is released to the customer, when deployed in to the real time environment and being accessed by the real time users.

- **Monkey Testing:** is the process in which abnormal operations, beyond capacity operations are done on the application to check the stability of it in spite of the users abnormal behavior.

- **Compatibility testing:** it is the testing process in which usually the products are tested on the environments with different combinations of databases (application servers, browsers…etc) In order to check how far the product is compatible with all these environments platform combination.

- **Installation Testing:** it is the process of testing in which the tester try to install or try to deploy the module into the corresponding environment by following the guidelines produced in the deployment document and check whether the installation is successful or not.

- **Adhoc Testing:** Adhoc Testing is the process of testing in which unlike the formal testing where in test case document is used, with out that test case document testing can be done of an application, to cover that testing of the future which are not covered in that test case document. Also it is intended to perform GUI testing which may involve the cosmotic issues.

TCD (Test Case Document:

Test Case Document Contains

- **Test Scope (or) Test objective**
- **Test Scenario**
- **Test Procedure**
- **Test case**

This is the sample test case document for the Acadamic details of student project:

Test scope:

- Test coverage is provided for the screen " Acadamic status entry" form of a student module of university management system application
- Areas of the application to be tested

Test Scenario:

- When the office personals use this screen for the marks entry, calculate the status details, saving the information on student's basis and quit the form.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively

Test Cases:

- Template for Test Case

T.C.No	Description	Exp	Act	Result
1	Enter user nam and password Enter valid date to store in the	True/false Accurate/Valid	True Valid date	Home page Data stored

2	database	data		successfully

Guidelines for Test Cases:

1. GUI Test Cases:

- Total no of features that need to be check
- Look & Feel
- Look for Default values if at all any (date & Time, if at all any require)
- Look for spell check

Example for Gui Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Check for all th features in the screen	The screen mus contain all th features		

2	Check for the alignmen of the objects as per th validations	The alignmen should be in prope way		

2. Positive Test Cases:

- The positive flow of the functionality must be considered
- Valid inputs must be used for testing
- Must have the positive perception to verify whether the requirements are justified.

Example for Positive Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Check for the dat Time Aut Display	The date and tim of the systen must be displayed		
2	Enter the vali Roll no into th student roll n field	It should accept		

3. Negative Test Cases:

- Must have negative perception.
- Invalid inputs must be used for test.

Example for Negative Test cases:

T.C.No	Description	Expected value	Actual value	Result
1	Try to modify Th information in dat and time	Modification should not b allow		
2	Enter invalid dat in to the studen details form, clicl on save	It should no accept invali data, save shoul not allow		

Login Page Test Case

Test Case Name	Test Case Description	Test Steps		
		Step	**Expected**	**Actual**
Login	Validate Login	To verify that Login name on login page must be greater than 1 characters	enter login name less than 1 chars (say a) and password and click Submit button	an error message "Login not less than 1 characters" must be displayed
			enter login name 1 chars (say a) and password and click Submit button	Login success full or an error message "Invalid Login or Password" must be displayed
Pwd	Validate Password	To verify that Password on login page must be greater than 1 characters	enter Password less than 1 chars (say nothing) and Login Name and click Submit button	an error message "Password not less than 1 characters" must be displayed

Pwd02	**Validate Password**	**To verify that Password on login page must be allow special characters**	**enter Password with special characters(say !@hi&*P) Login Name and click Submit button**	**Login success full or an error message “Invalid Login or Password” must be displayed**
Llnk	**Verify Hyperlinks**	**To Verify the Hyper Links available at left side on login page working or not**	**Click Sign Up Link**	**Home Page must be displayed**
			Click Sign Up Link	**Sign Up page must be displayed**
			Click New Users Link	**New Users Registration Form must be displayed**

Registration Page Test Case

Test Case Name	Test Case Description	Test Steps		
		Step	Expected	Actual
Registration	Validate User Name	To verify that User name on Registration page must be Declared	enter User name click Submit button	an error message User Name Must be Declared
	Validate Password	To verify that Password on Registration page must be Declared	enter Password click Submit button	an error message Password Must be Declared
	Validate First Name	To verify that First Name on Registration page must be Declared	enter First Name click Submit button	an error message First Name Must be Declared
	Validate Last Name	To verify that Last Name on Registration page must be Declared	enter Last Name click Submit button	an error message Last Name Must be Declared
	Validate Address	To verify that Address on Registration	enter Address click Submit	an error message Address Must

		page must be Declared	button	be Declared
	Validate Phone number	To verify that Phone number on Registration page must be Declared	enter Phone number click Submit button	an error message Phone number Must be Declared
	Validate Phone number is giving characters	To verify that Phone number (say abc) Registration page must be Declared	enter Phone number is only numeric values click Submit button	an error message Phone number Must be numeric Declared
	Validate Phone number valid number	To verify that Phone number (say 1234) Registration page must be Declared	enter Phone number is Valid values click Submit button	an error message Phone number Must be Valid value Declared

LIMITATIONS AND SCOPE FOR FUTURE ENHANCEMENTS:

Limitations of the system:.

- System works in all platforms and its compatible environments.
- Advanced techniques are not used to check the authorization.

Future Enhancements:

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
- Because it is based on object-oriented design, any further changes can be easily adaptable.
- Based on the future security issues, security can be improved using emerging technologies.
- Attendance module can be added
- sub admin module can be added

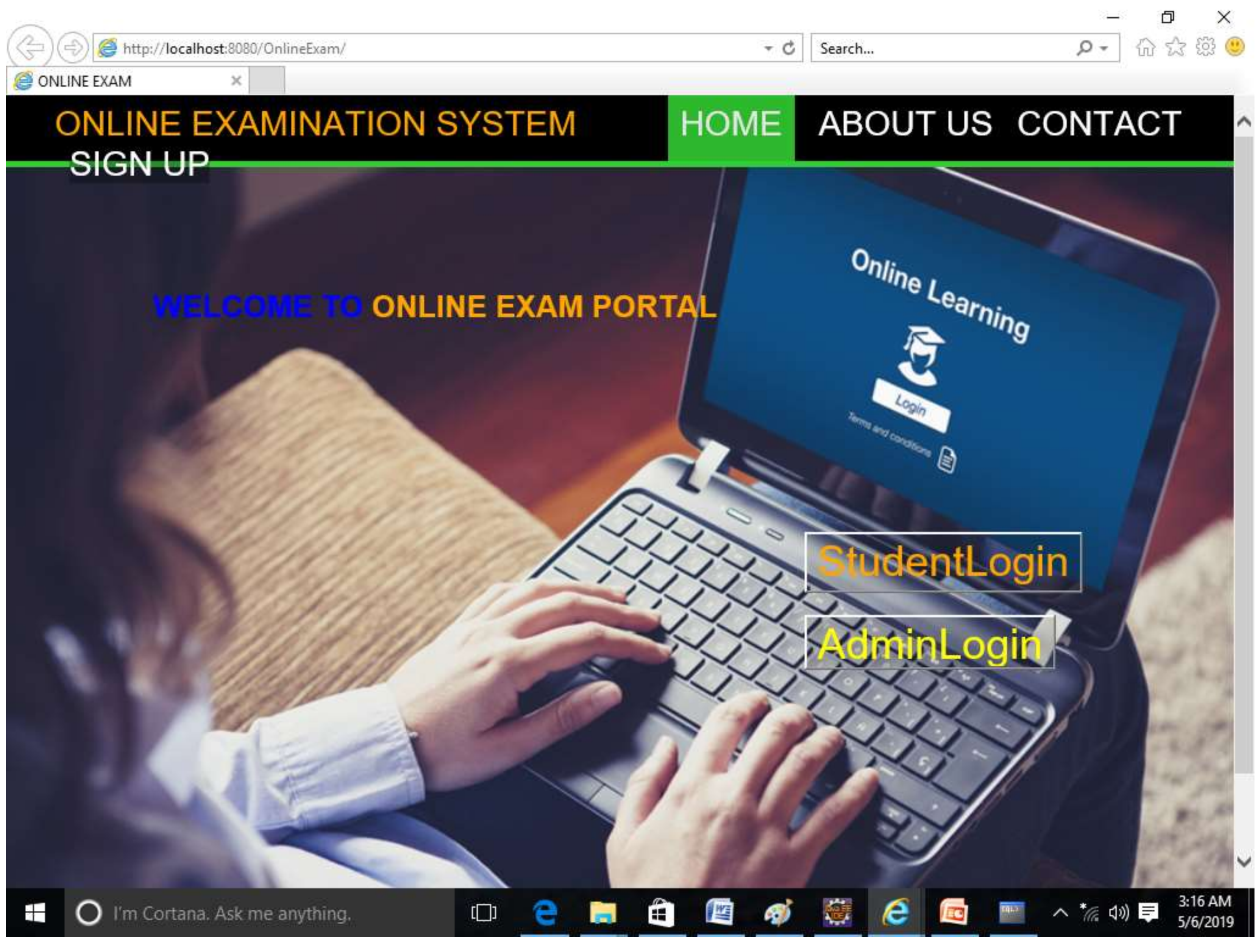
PROJECT SUMMARY

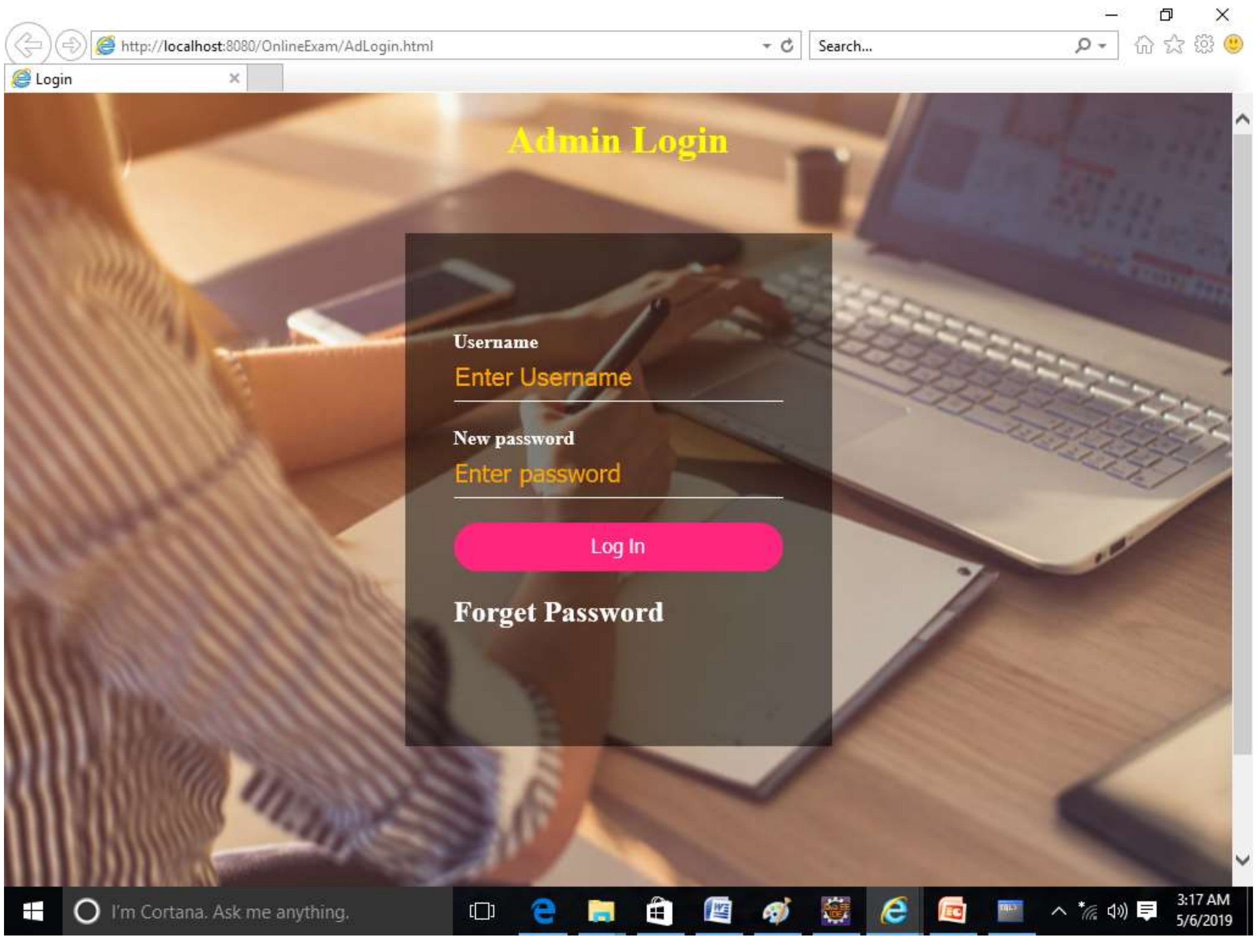
This application software has been computed successfully and was also tested successfully by taking “test cases”. It is user friendly, and has required options, which can be utilized by the user to perform the desired operations.

The software is developed using Java as front end and Oracle as back end in Windows environment. The goals that are achieved by the software are:

- ✓ Optimum utilization of resources.
- ✓ Efficient management of records.
- ✓ Simplification of the operations.
- ✓ Less processing time and getting required information.
- ✓ User friendly.
- ✓ Portable and flexible for further enhancement.

OUTPUT SCREENS:




http://localhost:8080/OnlineExam/AdLogin.html
Search...
Login
I'm Cortana. Ask me anything.
3:17 AM
5/6/2019

Admin Login

Username

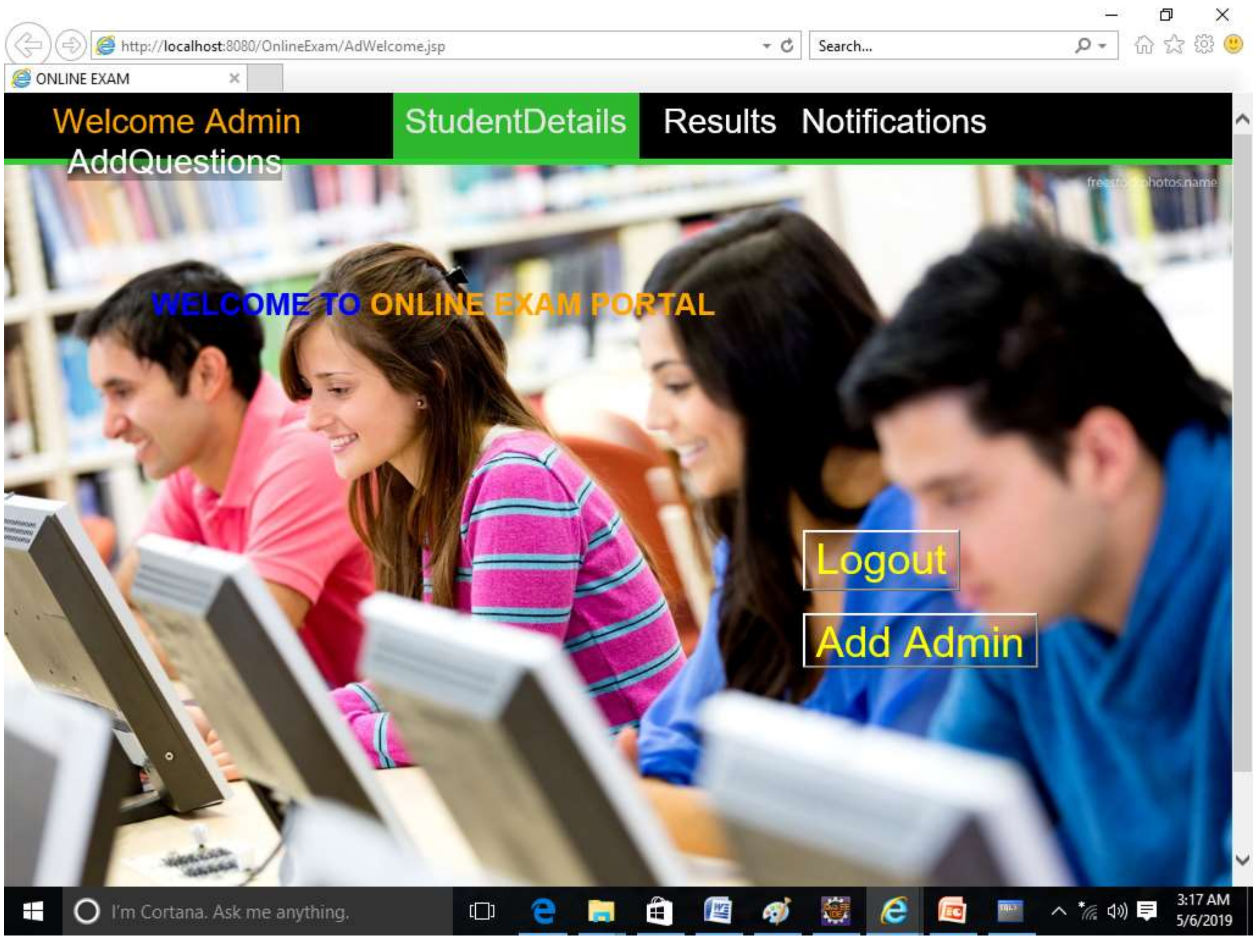
Enter Username

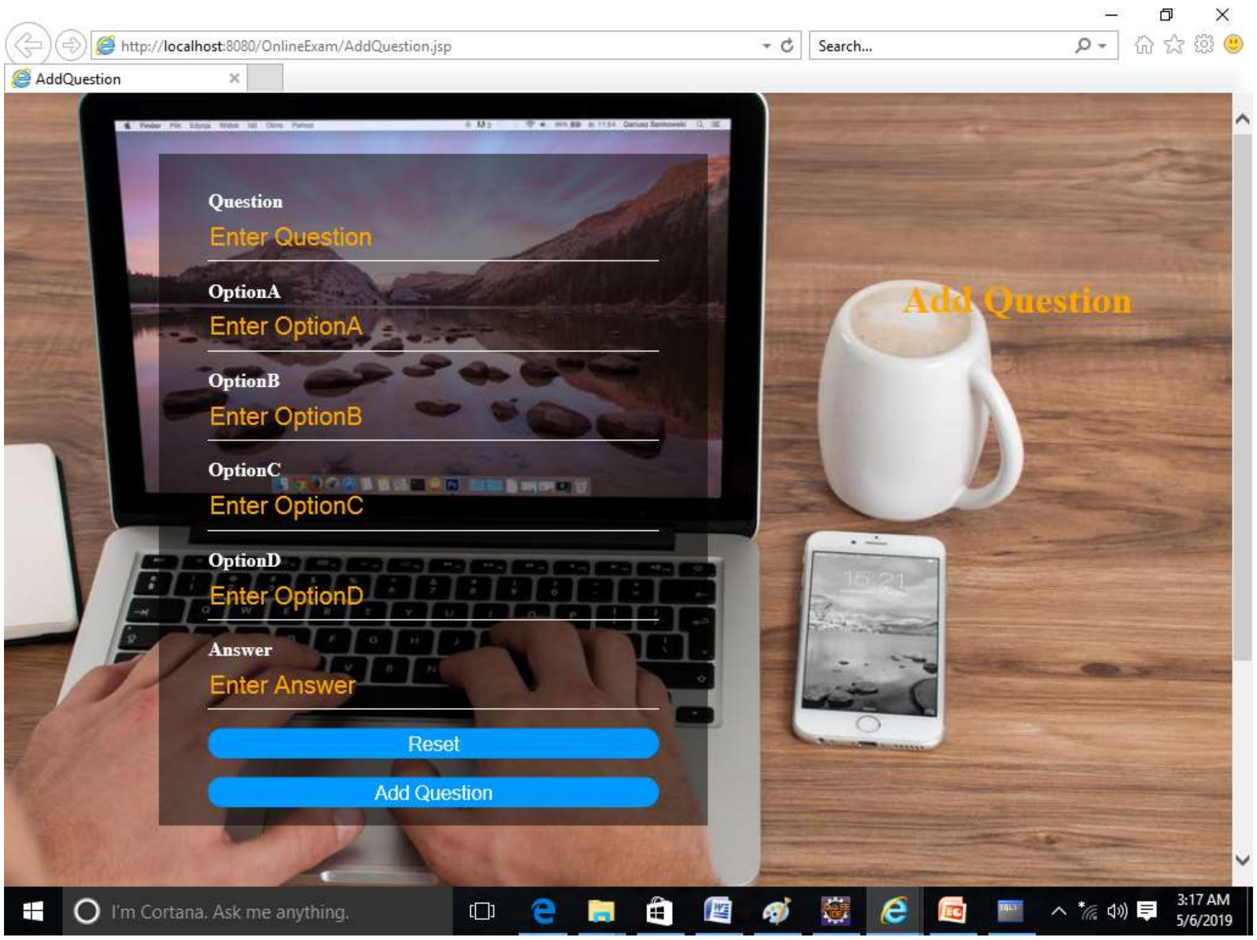
New password

Enter password

Log In

Forget Password


http://localhost:8080/OnlineExam/AdWelcome.jsp
Search...
ONLINE EXAM
Welcome Admin
StudentDetails
Results
Notifications
AddQuestions
freestockphotos.name
WELCOME TO ONLINE EXAM PORTAL
Logout
Add Admin
I'm Cortana. Ask me anything.
3:17 AM
5/6/2019



http://localhost:8080/OnlineExam/AddQuestion.jsp

AddQuestion

Add Question

Question

Enter Question

OptionA

Enter OptionA

OptionB

Enter OptionB

OptionC

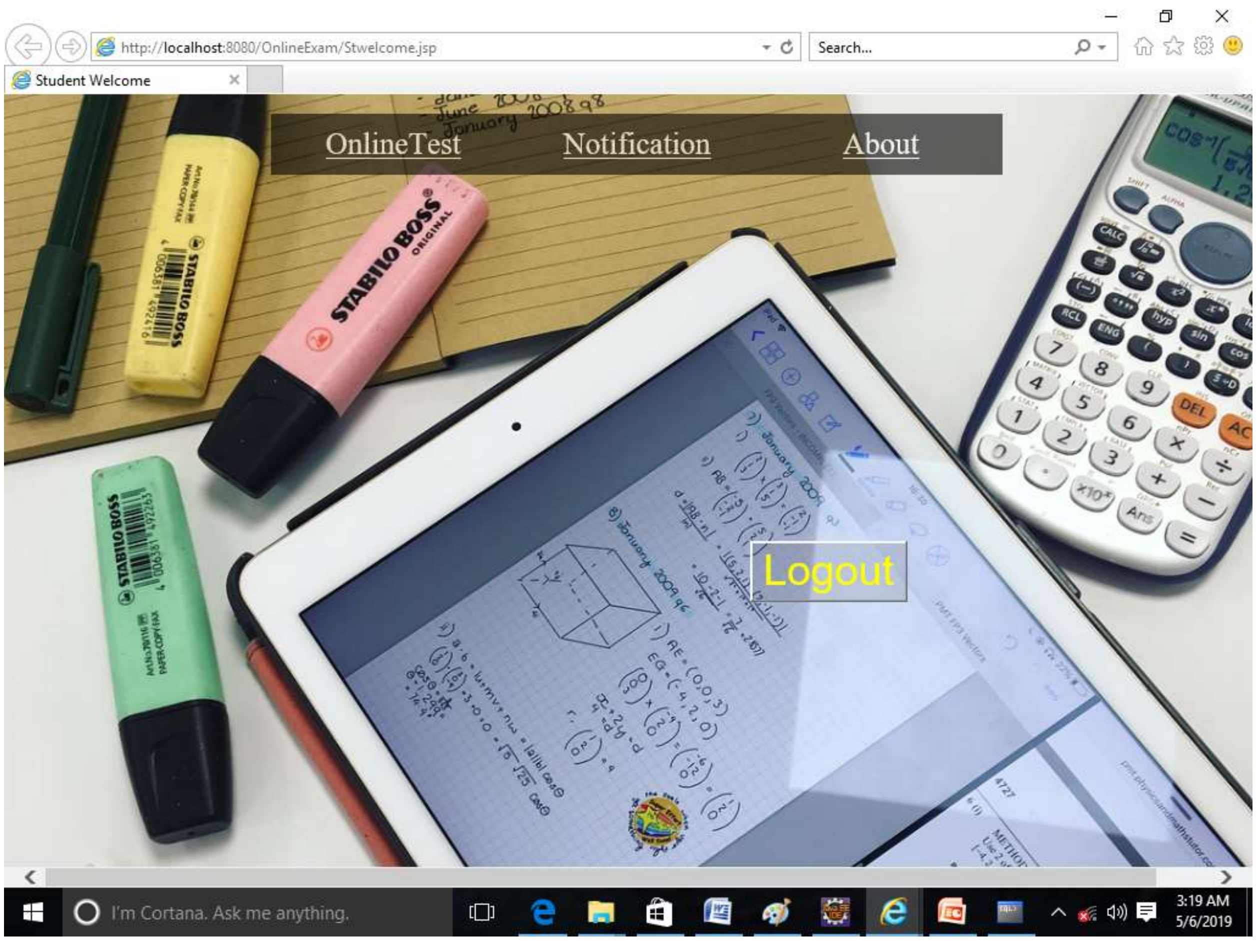
Enter OptionC

OptionD

Enter OptionD

Answer

Enter Answer


http://localhost:8080/OnlineExam/Stwelcome.jsp
Student Welcome
OnlineTest
Notification
About
Logout
I'm Cortana. Ask me anything.
3:19 AM
5/6/2019