**JEE Mini Project**

Online Banking System.

PROJECT REPORT ON:

**Online Banking System (OBS)**

DEVELOPED BY

* Akhil Raina.
* Sandeep Pujari.
* Shalu Mishra.
* Subbulaxumi Yadhav.
* Sonali More.
* Sampada Sarfare.

**ONLINE Banking SYSTEM”**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Topic** | **Sub** **Topic** |  | **Content** | **Page** **No**. |
| **1.** |  |  | **Analysis Phase** | **9-11** |
|  | **1.1** |  | **Requirement Gathering** | **2** |
|  |  | 1.1.1 | Problem Statement | 9 |
|  |  | 1.1.2 | Fact Finding Statement | 9 |
|  |  | 1.1.3 | Existing System | 9 |
|  |  | 1.1.4 | Drawbacks Of Existing System | 10 |
|  |  | 1.1.5 | Proposed System | 10 |
|  |  | 1.1.6 | Advantages Of Proposed System | 11 |
|  | **1.2** |  | **Feasibility Study** | **12** |
|  |  | 1.2.1 | Technical Feasibility | 13 |
|  |  | 1.2.2 | Economic Feasibility | 13 |
|  |  | 1.2.3 | Behavior Feasibility | 14 |
|  | **1.3** |  | **Requirement Specification** | **15-17** |
|  |  | 1.3.1 | Technical Requirement | 16 |
|  |  | 1.3.2 | Functional Requirement | 16 |
|  |  | 1.3.3 | Software & Hardware Requirement Specification | 17 |
|  | **1.4** |  | **Tools & Techniques-Star UML** | **23** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2**. |  |  | **Design Phase** | 25-35 |
|  | **2.1** |  | **System Design** | **26** |
|  |  | 2.1.1 | Logical Design | 27 |
|  |  | 2.1.2 | Physical Design | 27 |
|  |  | 2.1.3 | Module Design | 27 |
|  |  | 2.1.4 | Input Design | 28 |
|  |  | 2.1.5 | Output Design | 28 |
|  | **2.2** |  | **Detailed Life Cycle Of Project** | **29** |
|  |  | 2.2.1 | Class Diagram | 30 |
|  |  | 2.2.4 | Usecase Diagram | 32 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **3.** |  |  | **Testing** | **35-43** |
|  | **3.1** |  | **Testing Introduction** | **36-37** |
|  |  | 3.1.1 | Test Plan | 38 |
|  |  | 3.1.2 | Unit Testing | 40 |
|  |  | 3.1.3 | Integration Testing | 41 |
| **4.** |  |  | **Data Dictionary** | **44** |
| **5.** |  |  | **List Of Abbreviation Figure & Table** | **44** |
| **6.** |  |  | **Bibliography** | **51** |

**Synopsis**

**1. Introduction about the project**

The project titled as “Online Banking System“ is a Web-based application. This software provides facility for doing online Banking Transactions .The Person who is using this software can create account and maintain entire details about their transcations with system .Any Number of clients can connect to the server. Each user first makes their login to sever to show their availability. The server can be any Web Server.The server name is Wildfly server.

**Modules**

* Customers
* Registered Users(Account Holders)
* Administrator
* Visitors

**The Visitor module includes:**

* View News: This module help the visitor to see the latest news which can updated by the admin.

**The Registered User module includes:**

* Login System :This module help the user to Login into the system using his/her credentials.
* View Mini/Detailed Statement: This module help the user to view mini or detailed statement of all the accounts.
* Request Cheque Book : This module help the user to request for new cheque book.
* Change Address : This Module help the user to request for change in communication address/mobile number for bank account.
* Track Service Request : This module helps the user to check the status of service.
* Fund Transfer : This module helps the user to transfer fund .
* Change password: This module helps the user to change the password of current account.

**The Administrator module includes :**

* View All Requests : This module help the Admin to receive requests from may customers.
* Create Account: This module help the Admin to create new account to customer

upon request.

* View Transactions : This module helps the admin to reports of overall transactions on daily,monthly,yearly basis.

**The Customer module includes :**

* Create account: This module helps to customer to create new account.
* Request Cheque Book : This module help the user to request for new cheque book.
* Change Address : This Module help the user to request for change in communication address/mobile number for bank account.

ANALYSIS PHASE

* **REQUIREMENT GATHERING**

**1.1.1 Problem Statement :**

The project is “ONLINE BANKING SYSTEM” .This is a website. In the computerized system the basic and the initial first steps is choose login type (i.e. login as users ) then it ask for id and password, thus it can also take care of data security, now after login if the user login as register user it has following options are perform transaction, view mini/detailed statements ,send request for new account, send request for cheque book, change communication address etc.

If user login as admin then its has two options one view all requests, second is view transactions.

**1.1.2 FACT FINDING STATEMENT :**

**Questions Are :**

a) What is the main objective of this project?

b) What help does this project gives after the release?

c) What are the requirements needed for this project to make it successful?

d) Who are the people who are going to uses this website?

e) How helpful this website will be for the people who are going to work on this particular website?

f) What are the benefits of this website?

**1.1.3 Existing System**

In earlier days every account holder or customer have to go in bank to perform any type of transactions. Lots of manual entries have to do It is time consuming . All Records stored in databases.if customer wants to do withdraw amount or transfer amount to other then customer needs to visit bank.

**1.1.4 Drawbacks of Existing System**

* More man power.
* Time consuming.
* Consumes large volume of paper work.
* Needs manual calculations.
* No direct role for the higher officials.
* Damage of machines due to lack of attention.

To avoid all these limitations and make the working more accurately the system needs to be computerized.

**1.1.5 Proposed System**

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system provides proper security and reduces the manual work. The existing system has several disadvantages and many more difficulties to work well. The proposed system tries to eliminate or reduce these difficulties Ferup to some extent. The proposed system will help the user to reduce the workload and mental conflict. The proposed system helps the user to work user friendly and he can easily do his jobs without time lagging.

**1.1.6 Advantages of Proposed System**

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features

* Ensure data accuracy.
* Proper control of the higher officials.
* Minimize manual data entry.
* Minimum time needed for the various processing.
* Greater efficiency.
* Better service.
* User friendliness and interactive.
* Minimum time required.

Feasibility Study Report

**1.2 FEASIBILITY STUDY**

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provide the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features:

**1.2.1 TECHNICAL FEASIBILITY**

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

* Does the existing technology sufficient for the suggested one
* Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project is developed within latest technology. Through the technology may become obsolete after some period of time, due to the fact that never version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using Java the project is technically feasible for development.

**1.2.2 ECONOMIC FEASIBILITY**

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

* The costs conduct a full system investigation.
* The cost of the hardware and software.
* The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

**1.2.3 BEHAVIORAL FEASIBILITY**

This includes the following questions:

* Is there sufficient support for the users?
* Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

Requirement

Specification

**1.3 Requirement Specification :**

**1.3.1 TECHNICAL REQUIREMENT :**

**Java:-**

Java is a small, simple, safe, object oriented, interpreted or dynamically optimized, byte coded, architectural, garbage collected, multithreaded programming language with a strongly typed exception-handling for writing distributed and dynamically extensible programs.

Java is an object oriented programming language. Java is a high-level, third generation language like C, FORTRAN, Small talk, Pearl and many others.

Java a supports this application and the follow features make it one of the best programming languages.

* It is simple and object oriented
* It helps to create user friendly interfaces.
* It is very dynamic.
* It supports multithreading.
* It is platform independent
* It is highly secure and robust.
* It supports internet programming

**Java Virtual Machine**

The heart of the Java Platform is the concept of a "virtual machine" that executes Java byte code programs. This byte code is the same no matter what hardware or operating system the program is running under. There is a JIT compiler within the *Java Virtual Machine*, or JVM. The JIT compiler translates the Java bytecode into native processor instructions at run-time and caches the native code in memory during execution.

The use of byte code as an intermediate language permits Java programs to run on any platform that has a virtual machine available. The use of a JIT compiler means that Java applications, after a short delay during loading and once they have "warmed up" by being all or mostly JIT-compiled, tend to run about as fast as native programs. Since JRE version 1.2, Sun's JVM implementation has included a just-in-time compiler instead of an interpreter.

**Class libraries**

In most modern operating systems, a large body of reusable code is provided to simplify the programmer's job. This code is typically provided as a set of dynamically loadable libraries that applications can call at runtime. Because the Java Platform is not dependent on any specific operating system, applications cannot rely on any of the existing libraries. Instead, the Java Platform provides a comprehensive set of standard class libraries, containing much of the same reusable functions commonly found in modern operating systems.

**Java Runtime Environment**

The Java Runtime Environment, or *JRE*, is the software required to run any application deployed on the Java Platform. End-users commonly use a JRE in software packages and Web browser plugins. Sun also distributes a superset of the JRE called the Java 2 SDK (more commonly known as the JDK), which includes development tools such as the Java compiler, Javadoc, Jar and debugger.

One of the unique advantages of the concept of a runtime engine is that errors (exceptions) should not 'crash' the system. Moreover, in runtime engine environments such as Java there exist tools that attach to the runtime engine and every time that an exception of interest occurs they record debugging information that existed in memory at the time the exception was thrown (stack and heap values). These Automated Exception Handling tools provide 'root-cause' information for exceptions in Java programs that run in production, testing or development environments.

**1.3.2 Functional Requirement :**

**JSP architecture**

JSPs are built on top of SUN’s servlet technology. JSPs are essential an HTML page with special JSP tags embedded. These JSP tags can contain Java code. The JSP file extension is .jsp rather than .htm or .html. The JSP engine parses the .jsp and creates a Java servlet source file. It then compiles the source file into a class file; this is done the first time and this why the JSP is probably slower the first time it is accessed. Any time after this the special compiled servlet is executed and is therefore returns faster.

**Java Script**

JavaScript is a programming language that allows scripting of events, objects, and actions to create Internet applications. A website development environment that will allow the creation of Interactive Web Pages. The coding techniques capable of accepting a client's requests and processing these requests.

The web site development environment should also provide the facility for 'validating' user input. With JavaScript, forms are a consideration in nearly every page you design. Capturing user requests is traditionally done via a 'form'. So the web site needs to have facilities to create forms. Text fields and textareas can dynamically change in response to user responses.

**JSP (Java Server Pages):**

JSP is java technology that allows software developers to dynamically generate HTML,XML or other types of documents in response to a web client request.

**JSP pages typically comprise of:**

* Static HTML/XML components.
* Special jsp tags.
* Optionally, snippets of code written in the java programming language called “Scriplets”.

**Why JSP Technology?**

* Servlets tie up files(an HTML file for the static content and a java file for the dynamic content) to independently handle static presentation logic and dynamic business logic.
* JSP allows java to be embedded directly into an HTML page by using special tags. Any changes made to HTML content is automatically compiled and loaded onto the server.
* JSP requires less coding as compared to servlet.
* By the virtue of separate placement of static and dynamic content ,both web developers and web designers can work independently in JSP technolology.

**Advantage Of JSP:**

* JSP provide excellent server side scripting support for creating database driven web applications.
* JSP enable the developers to directly insert java code into jsp file, this makes the development process very simple and it’s maintenance also become very easy.
* Jsp provides excellent database connectivity in heterogeneous database environment.
* Jsp are compiled into java servlet by jsp compiler. A jsp compiler may generate a servlet in java code that is compiled by java compiler to generate byte code.
* JSP takes less time to reload changes ,respond to user querries from HTML form.

**Why HTML Technology?**

**HTML**, which stands for **Hyper Text Markup Language**, is the predominant [markup language](http://en.wikipedia.org/wiki/Markup_language) for [web pages](http://en.wikipedia.org/wiki/Web_page). It provides a means to create [structured documents](http://en.wikipedia.org/wiki/Structured_document) by denoting structural [semantics](http://en.wikipedia.org/wiki/Semantic) for text such as headings, paragraphs, lists etc as well as for links, quotes, and other items. It allows [images and objects](http://en.wikipedia.org/wiki/HTML_element) to be embedded and can be used to create [interactive forms](http://en.wikipedia.org/wiki/HTML_element). It is written in the form of [HTML elements](http://en.wikipedia.org/wiki/HTML_element) consisting of "tags" surrounded by [angle brackets](http://en.wikipedia.org/wiki/Brackets) within the web page content. It can include or can load [scripts](http://en.wikipedia.org/wiki/Scripting_language) in languages such as [JavaScript](http://en.wikipedia.org/wiki/JavaScript) which affect the behaviorof HTML processors like [Web browsers](http://en.wikipedia.org/wiki/Web_browser); and [Cascading Style Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets) (CSS) to define the appearance and layout of text and other material. The [W3C](http://en.wikipedia.org/wiki/W3C), maintainer of both HTML and CSS standards, encourages the use of CSS over explicit presentational markup.

HTML is the basic technology for building the web pages. It is very simple technology but is more dynamic and very much effective.

**WildFly**

WildFly formerly known as Jboss AS, is an application server authored by JBoss.

Developed by Red Hat,WildFly is free and open source software.

Red hat acquired JBoss inc.in 2006.

WildFly 8 is the direct continuation to the JBoss AS project.WlidFly 8 was officially released on November 20,2014.

WildFly is a flexible. light weight, managed application runtime thathelps us build JEE applicaltions.

WildFly has become popular and is a safe choice for deploying JEE applications

because it provides a set of Java Enterprise products.

The following list of products are supported in WildFly

* WildFly Application Server – A server for deploying Enterprise applications
* Java Messaging Service – It is the asynchronous message queuing system
* Java Naming and Directory Interfaces (JNDI) – is a Java API for a directory service that
* allows Java clients to discover and look up data and objects via a name
* Hibernate-ORM (Hibernate in short) is an object relational mapping library for
* the Java language, providing a framework for mapping an object oriented domain
* model to a traditional relational database
* WildFly Web Server – Uses Tomcat internally, this web server supports Servlets/JSP,ASP.NET, PHP & CGI also.
* Also included are JBoss Community which consists of JBoss Plug-in for Eclipse IDE
* and JBoss developer studio.

Requirement Specification

**1.3.3 Software and Hardware Requirement Specifications**

**Hardware Specification**

Processor : Intel Pentium 90

RAM : 128 MB

Hard disk : 20 GB

FDD : 1.44MB

Monitor : 14 inch

Mouse : 3 Button scroll

CD Drive : 52 X

Keyboard : 108 keys

**Software Specification**

Operating System : Windows 2000,95,98,or NT 4.0,XP,Windows 7

Languages : java 2(EJB2.0, JDBC, JSP, Servlet, Java Mail)

Front End : HTML, JavaScript

Platform : JEE

Web Servers : WildFly

Backend : Oracle

Browser Program : Internet explorer/Mozilla Fireworks.

Tool : Eclipse Luna

Testing Tool : JUnit 4.0

**1.4 Tools And Technologies :**

**Star UML**

**Key Features**

|  |  |
| --- | --- |
| Feature | Description |
| Accurate UML standard model | Star UML™ strictly adheres to the UML standard specification specified by the OMG for software modeling. Considering the fact that the results of design information can reach 10 years or more into the future, dependence on vendor-specific irregular UML syntax and semantics can be quite risky. Star UML™ maximizes itself to order UML 1.4 standard and meaning, and it accepts UML 2.0 notation on the basis of robust meta model. |
| Open software model format | Unlike many existing products that manage their own legacy format models inefficiently, Star UML™ manages all files in the standard XML format. Codes written in easy-to-read structures and their formats can be changed conveniently by using the XML parser. Given the fact that XML is a world standard, this is certainly a great advantage, ensuring that the software models remain useful for more than a decade. |
| True MDA support | Star UML™ truly supports UML Profile. This maximizes extensibility of UML, making modeling of applications possible even in areas like finance, defense, e-business, insurance, and aeronautics. Truly Platform Independent Models (PIM) can be created, and Platform Specific Model (PSM) and executable codes can be automatically generated in any way. |
| Applicability of methodologies and platforms | Star UML™ manipulates the approach concept, creating environments that adapt to any methodologies/processes. Not only the application framework models for platforms like .NET and J2EE, but also basic structures of software models (e.g. 4+1 view-model, etc.) can be defined easily |
| Excellent  extensibility | All functions of the Star UML™ tools are automated according to Microsoft COM. Any language which supports COM (Visual Basic Script, Java Script,  VB, Delphi, C++, C#, VB.NET, Python, etc.) can be used to control Star UML™ or develop integrated Add-In elements. |

|  |  |
| --- | --- |
| Software model verification function | Users can make many mistakes during software modeling. Such mistakes can be very costly if left uncorrected until the final coding stage. In order to prevent this problem, Star UML™ automatically verifies the software model developed by the user, facilitating early discovery of errors, and allowing more faultless and complete software development. |
| Useful Add-Ins | Star UML™ includes many useful Add-Ins with various functionalities: it generates source codes in programming languages and converts source codes into models, imports Rational Rose files, exchanges modeling information with other tools using XMI, and supports design patterns. These Add-Ins offer additional reusability, productivity, flexibility and interoperability for the modeling information. |

Design

Phase

**2. DESIGN PHASE**

**2.1 System Design :**

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

**2.1.1 LOGICAL DESIGN:**

The logical flow of a system and define the boundaries of a system. It includes the following steps:

* Reviews the current physical system – its data flows, file content, volumes, Frequencies etc.
* Prepares output specifications – that is, determines the format, content and Frequency of reports.
* Prepares input specifications – format, content and most of the input functions.
* Prepares edit, security and control specifications.
* Specifies the implementation plan.
* Prepares a logical design walk through of the information flow, output, input, Controls and implementation plan.

**2.1.2 PHYSICAL DESIGN:**

Physical system produces the working systems by define the design specifications that tell the programmers exactly what the candidate system must do. It includes the following steps.

* Design the physical system.
* Specify input and output media.
* Design the database and specify backup procedures.
* Design physical information flow through the system and a physical design
* Plan system implementation.
* Prepare a conversion schedule and target date.
* Determine training procedures, courses and timetable.
* Devise a test and implementation plan and specify any new hardware/software.
* Update benefits , costs , conversion date and system constraints

**Design/Specification activities:**

* Concept formulation.
* Problem understanding.
* High level requirements proposals.
* Feasibility study.
* Requirements engineering.
* Architectural design.

**2.1.3 MODULE DESIGN**

* Customers
* Registered Users
* Administrator

**2.1.4 INPUT DESIGN**

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

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

**2.1.5 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

* Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
* Select methods for presenting information.
* Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**2.2 DETAILED LIFE CYCLE OF PROJECT:**

**2.2.1 Class Diagram** :

* Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
* The class diagram is the main building block of object oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code.
* Class diagram can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.
* The static view mainly supports the functional requirements of a system – the services the system should provide to the end users.

* A class diagrams shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams involve global system description, such as the system architecture, and detail aspects such as the attributes and operations within a class as well.
* The class diagram models the definition of resources of the system. The class diagram is the source for code generation and the target for reverse engineering.
* Basic class diagram symbols and notations are



* Classes represent an abstraction of entities with common characteristics. Associations represent the relationship between classes.
* Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (bolded, and capitalized), list the attributes in the second partition, and write operations into the third.

**Class Daigram:**



**2.2.2 USE CASE DIAGRAM**:

* A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service web site.

* Use case diagrams are employed in UML (Unified Modelling Language), a standard notation for the modelling of real-world objects and systems.
* The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.
* Because other four diagrams (activity, sequence, collaboration and state chart) are also having the same purpose.
* So we will look into some specific purpose which will distinguish it from other four diagrams.
* Use case diagrams are used to gather the requirements of a system including internal and external influences.
* These requirements are mostly design requirements. So when a system is analyst to gather its functionalities use cases are prepared and actors are identified.
* Now when the initial task is complete use case diagrams are modelled to present the outside view.
* The purposes of use case diagrams can be as follows:
* Used to gather requirements of a system.
* Used to get an outside view of a system.
* Identify external and internal factors influencing the system.
* Show the interacting among the requirements are actors.
* A use case diagram contains four components.
* The boundary, which defines the system of interest in relation to the world around it.
* The actors, usually individuals involved with the system defined according to their roles.
* The use cases, which is the specific roles played by the actors within and around the system.
* The relationships between and among the actors and the use cases.

**Component of**  **USECASE**

**1. USECASE**



* A use case represents a user goal that can be achieved by accessing the system or software application.
* In Visual Paradigm, you can make use of the sub-diagram feature to describe the interaction between user and system within a use case by creating a sub-sequence diagram under a use case.
* You can also describe the use case scenario using the Flow of Events editor.

**2. ASSOCIATION**



* Actor and use case can be associated to indicate that the actor participates in that use case.
* Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

**3. ACTOR**



* Actors are the entities that interact with a system.
* Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system.
* So, an actor may be people, computer hardware, other systems, etc.
* **INCLUDE**



* An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

* Basically it is use to represent the internal process of system such as validation in my project
* **SYSTEM**



* The scope of a system can be represented by a system (shape), or sometimes known as a system boundary.
* The use cases of the system are placed inside the system shape, while the actor who interact with the system are put outside the system.
* The use cases in the system make up the total requirements of the system.
* **EXTEND**



* · An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.
* **DEPENDENCY**



* A dependency relationship represents that a model element relies on another model element for specification and/or implementation.

**Use case :**



TESTING

**3. TESTING :**

**3.1 Testing Introduction**

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified?. Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Validation **:** Are we doing the right job?

Verification **:** Are we doing the job right?

Software testing should not be confused with debugging. Debugging is the process of analyzing and localizing bugs when software does not behave as expected. Although the identification of some bugs will be obvious from playing with the software, a methodical approach to software testing is a much more thorough means for identifying bugs. Debugging is therefore an activity which supports testing, but cannot replace testing.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis

looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are

* Testing is a process of executing a program with the intend of findingan error.
* A good test case is one that has high possibility of finding an undiscovered error.
* A successful test is one that uncovers an undiscovered error.

     If a testing is conducted successfully according to the objectives as stated above, it would uncovered errors in the software also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

* For correctness
* For implementation efficiency
* For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

**3.1.1 TEST PLAN**

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

* Unit testing
* Integration Testing
* Data validation Testing
* Output Testing

**3.1.2 UNIT TESTING**

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified.

**Unit testing at User Side:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No. | Unit | Input | Expected  Output | Observed  Output | Status |
| 1 | Registration | Give all the details | Check with the database for duplication | Details inserted | Success |
| 2 | Login | UserName& Password | Check with the database for their relevance &  Redirect to homepage | Database is verified & application redirected to homepage | Success |
| 4 | Send Request | Give all the details | Inserted into database | Data inserted into database | Success |
| 5 | Edit Profile | Give all the registration details | Check with the database | Data is verified & details updated | Success |
| 6 | Feedback | description | Inserted into database | Data inserted into database | Success |

**Unit testing at Administrator Side:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Unit | Input | Expected Output | Observed Output | Status |
| 1 | Add Account | Generate account id | Inserted into database | Id and details added into database | Success |
| 2 | Show different reports  of transaction | Select type | record should be displayed | record displayed | Success |

**3.1.3 INTEGRATION TESTING**

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop.

After unit testing in Sell-Soft System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

**Integration Testing of all modules:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Module | Input | Expected Output | Observed Output | Status |
| 1 | Customer | Ask for new account | success message with new account Id | Success Message | Success |
| 2 | Registor User | Registertion details | Perform transaction | Success Message | Success |
| 3 | Administrator | Customer details | New account id should be generated | Account id is generated | Success |

**VALIDATION TESTING OR SYSTEM TESTING**

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

**Validation Checking:**

At the culmination of integration testing, software is completely assembled as a package; interfacing errors have been uncovered and corrected, and a final series of software test-validation checks may begin. Validation can be defined in many ways, but a simple definition is that validation succeeds when software functions in a manner that can be reasonably expected by a customer. Software validation is achieved through a series of black-box tests to be conducted and a test procedure defines specific test cases that will be used in attempt to uncover errors in conformity with requirements.

In the proposed system, if the clients click the send button after selecting a file from his file list, then the system will show the confirmation message for sending files. Similarly if a client makes an attempt to download a file from the server file list, then also the system will show the confirmation message for downloading. This is how the data validations were made in the proposed system.

**Table Structure**

**1. DATA DICTIONARY**

All the information required by a specific system is accomplished by a central repository called a **Data Dictionary.** The objective of creating data dictionary is to centralize all the information so that it becomes easy to understand the system the information contained in it.

A conventional Data Dictionary describes all the entities in the system along with all its attributes, the various process, methods and modules in which this attributes are used and their aliases.

Table which are used to store the database

* **Account Master**
* **User Table**
* **Customer**
* **Transactions**
* **Service Tracker**
* **Fund Transfer**
* **Payee Table**
* **Registration Table**

**DATABASE DESIGN**

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design

**RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS):**

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

**RELATIONS, DOMAINS & ATTRIBUTES:**

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values. Every value in a relation is atomic, that is not decomposable.

RELATIONSHIPS:

* Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
* Entity Integrity enforces that no Primary Key can have null values.
* Referential Integrity enforces that no Primary Key can have null values.
* Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.
* Relationships have been set between every table in the database. This ensures both Referential and Entity Relationship Integrity.

NORMALIZATION:

As the name implies, it denoted putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These includes:

* Normalize the data.
* Choose proper names for the tables and columns.
* Choose the proper name for the data.

**Table Name: Account Master**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| Account Id | Number | 10 | Primary key |
| Account Type | Varchar2 | 25 | - |
| Account Balance | Number | 15 | - |
| Open Date | Date | - | - |

**Table Name: User Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| User Id | Number | 10 | Primary key |
| Account Id | Number | 10 | Foreign key |
| Login Password | Varchar2 | 10 | - |
| Secrete Question | Varchar2 | 50 | - |
| Secrete Answer | Varchar2 | 50 | - |
| Lock Status | Varchar2 | 1 | - |

**Table Name: Customer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| Customer ID | Varchar2 |  | Primary key |
| Customer Name | Varchar2 | 20 | - |
| Email | varchar2 | 30 | - |
| Address | Varchar2 | 50 | - |
| Pancard | varchar2 | 15 | - |

**Table Name: Transactions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| Transaction Id | Number | 20 | Primery key |
| Transaction Description | Varchar2 | 20 | - |
| Date of Transaction | Date |  | - |
| Transaction Amount | Number | 10 | - |
| Account Id | Number | 10 | Foreign key |

**Table Name: Service Tracker**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| Service Id | Number | 10 | Primary Key |
| Service Description | Varchar2 | 20 | - |
| Account Id | Number | 10 | Foreign Key |
| Service Raised Date | Date | - | - |
| Service Status | varchar2 | 20 | - |

**Table: Fund Transfer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| FundTransfer Id | Number | 10 | Primary Key |
| Account Id | Number | 10 | Foreign Key |
| Payee Account Id | Number | 10 | - |
| Date Of fund transfer | Date | - | - |
| Transfer Amount | Number | 10 | - |

**Table: Payee Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| Payee Account Id | Number | 10 | Primary Key |
| Account Id | Number | 10 | Foreign Key |
| Nick Name | varchar2 | 20 | - |

**Table : Registration Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Data Type** | **Length** | **Key** |
| Customer Id | Number | 10 | Primary Key |
| Account Type | Varchar2 | 20 | - |

**2.List of Abbreviations :**

* JSP:-Java Server Pages
* SQL:-Structured Query Language
* ID:- Identity
* i.e.:-that is
* SDLC:- Software Development Life Cycle
* URL:-Uniform Resource Locator

**BIBLIOGRAPHY**

**ONLINE REFERENCE:**

* [www.theserverside.com](http://www.theserverside.com/)
* [www.java.sun.com](http://www.java.sun.com/)
* [www.w3schools.com](http://www.w3schools.com/)
* <http://www.tutorialspoint.com/java/index.html>
* <http://www.java-suresh.com/search/label/java>
* [http://www.java](http://www.java/)-suresh.com/search/label/SQL%20Server
* http://www.tutorialspoint.com/sql/sql-sub-queries.htm
* http://www.w3schools.com/
* http://www.sitepoint.com/article/sql-server-2008R2-database
* http://www.wikipedia.com/