|  |  |
| --- | --- |
| **Table of Contents** | **Page No** |
| 1. Title of the Project |  |
| 1. Introduction and Objectives of the Project |  |
| 1. Project Category (RDBMS/OOPS/Networking/Multimedia/Artificial Intelligence/Expert Systems etc.) |  |
| 1. Tools/Platform, Hardware and Software Requirement specifications |  |
| 1. Goals of Implementation |  |
| 1. SDLC Process Applied |  |
| 1. Data Model |  |
| 1. Functional Requirements (Use Case Diagram) |  |
| 1. Nonfunctional Requirements |  |
| 1. Feasibility Study |  |
| 1. Project Planning |  |
| 1. Project Scheduling |  |
| 1. Software Engineering Paradigm applied |  |
| * 1. Data Flow Diagram (DFD) |  |
| * 1. Structure Chart |  |
| * 1. Class Diagram |  |
| 1. System Design |  |
| 1. Database design |  |
| 1. User Interface Design |  |
| 1. Coding |  |
| 1. Testing |  |
| 1. System Security measures (Implementation of security for the project developed) |  |
| 1. Database/Data security |  |
| 1. Creation of User profiles and access rights |  |
| 1. Cost Estimation of the Project along with Cost Estimation Model |  |
| 1. Reports (sample layouts should be placed) |  |
| 1. Future scope and further enhancement of the Project |  |
| 1. Bibliography |  |

1. **Title of the Project**

Development of a feature-rich, practical online intranet knowledge mgmt system for the college (KMS)

1. **Introduction and Objectives of the Project**

This project is aimed at developing an online intranet knowledge mgmt system that is of importance to either an organization or a college. The system (KMS) is an Intranet based application that can be accessed throughout the organization. This system can be used as a knowledge/information mgmt system for the college. Students logging should be able to upload any kind of technical information. Students logging in may also access/search any information put up by others.KMS should have super users for approval of documents uploaded by others.

1. A person should be able to
   * Access/ Search documents/information from the first page (only read access).
   * login to the system through the first page of the application
   * See/change his/her details (like how many reports/technical documents/submitted)
   * get help about the Knowledge mgmt system on how to use the different features of the system
2. An admin login should be present who can read, approve as well as remove any uploads.
3. **Project Category**

Web Application

1. **Tools/Platform, Hardware and Software Requirement specifications.**

**Tools**

1. Adobe Dreamweaver
2. XAMPP
3. Dia
4. Ms-Office

**Platform**

1. Microsoft Windows 7/8

**Hardware Requirement Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Machine** | | **Server Machine** | |
| **HDD** | 200 MB | **HDD** | 320 GB |
| **Processor** | Pentium 4 or newer processor that supports SSE2 | **Processor** | Dual Core or newer processor |
| **Memory** | 512 MB | **Memory** | 2 GB |

**Software Requirement Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Client Machine** | | **Server Machine** | |
| **Browser** | Any standard browser with Javascript interpreter | **Software** | PHP-5 |
| **Client side mark up / scripting languages** | HTML, Javascript | **Database Management System Software** | MySQL |
|  |  | **Specification** | MySQL 4.1 |

**5. Goals of Implementation**

The implementation aims at seamless document sharing across the institution.

**6. SDLC Process Applied**



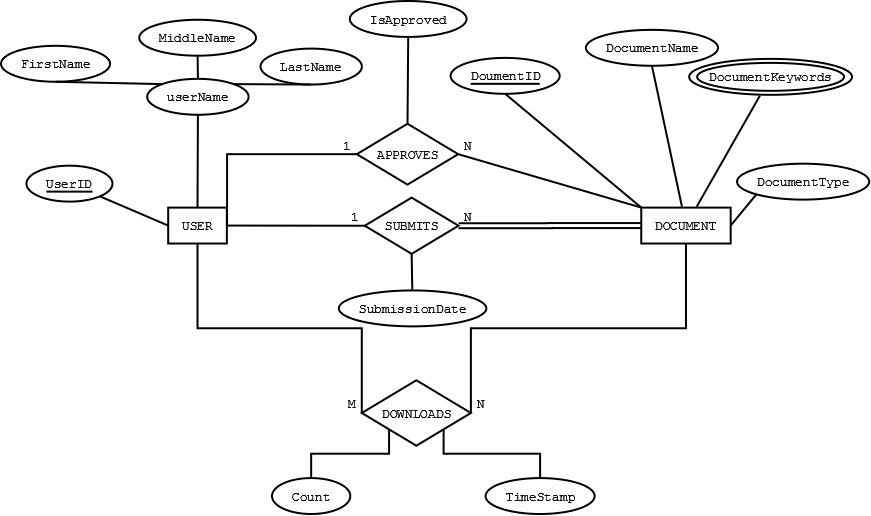
Often, a customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements. In other cases, the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human/machine interaction should take. In these, and many other situations, a prototyping paradigm may offer the best approach.

The prototyping paradigm begins with **requirements gathering**. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory. A **"quick design"** then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats). The quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

Ideally, the prototype serves as a mechanism for identifying software requirements. If a working prototype is built, the developer attempts to use existing program fragments or applies tools (e.g., report generators, window managers) that enable working programs to be generated quickly.

**7. Data Model**

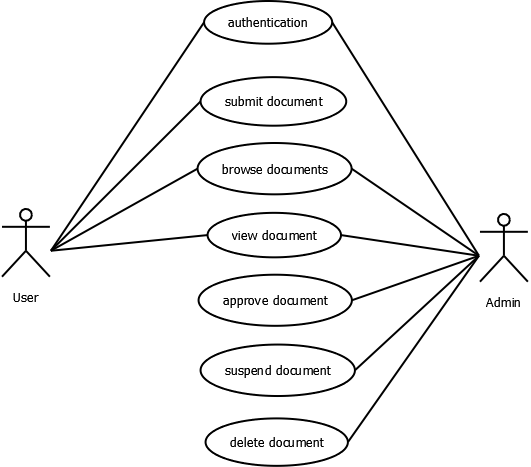
**ER Diagram**

****

**8. Functional Requirements**

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

**Use Case Diagram**

****

**Use Case Descriptions**

|  |  |
| --- | --- |
| **Use Case Name:** | Authentication |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | User is connected to the Internet and on the KMS home page |
| **Basic Path** | 1. User enters username and password. 2. The username and password is matched with the record in the database. 3. If the authentication parameters are correct the user is directed to the user’s main page, otherwise an error message is displayed. |
| **Alternate Path** | NA |
| **Post Condition** | The user is on the User Home Page |
| **Exception Path** | If there is a connection failure the server returns to the wait state |
|  |  |
| **Use Case Name:** | Submit document |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | User is connected to the Internet and on the user’s main page |
| **Basic Path** | 1. User selects a document. 2. User clicks on the upload document button. 3. The server side program receives the document and saves it in the server system’s file system. |
| **Alternate Path** | NA |
| **Post Condition** | The user uploads the document. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Browse document |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | User is connected to the Internet and on the user’s main page |
| **Basic Path** | 1. User selects the browse document link. 2. The server side program returns the list of documents. |
| **Alternate Path** | NA |
| **Post Condition** | The user has the list of all documents. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Approve document |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | Admin is connected to the Internet and on the admin’s main page |
| **Basic Path** | 1. The server program returns list of submitted non approved documents. 2. Admin selects the documents he/she wishes to approve. 3. The approval acknowledgement goes to the user whose document is approved. |
| **Alternate Path** | NA |
| **Post Condition** | The admin approves a document. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | View document |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | User is connected to the Internet and on the user’s main page |
| **Basic Path** | 1. User selects a document. 2. User clicks on the link. 3. The server side program receives the request and sends the document. |
| **Alternate Path** | NA |
| **Post Condition** | The user views a document. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Delete document |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | Admin is connected to the Internet and on the admin’s main page |
| **Basic Path** | 1. The server program returns list of all documents. 2. Admin selects the documents he/she wishes to delete. 3. The deletion acknowledgement goes to the user whose document is deleted. |
| **Alternate Path** | NA |
| **Post Condition** | The admin deletes a document. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

|  |  |
| --- | --- |
| **Use Case Name:** | Suspend document |
| **Priority** | Essential |
| **Trigger** | Menu selection |
| **Precondition** | Admin is connected to the Internet and on the admin’s main page |
| **Basic Path** | 1. The server program returns list of all documents. 2. Admin selects the documents he/she wishes to suspend. 3. The suspension acknowledgement goes to the user whose document is suspended. |
| **Alternate Path** | NA |
| **Post Condition** | The admin suspends a document. |
| **Exception Path** | If there is a connection failure the server returns to the wait state |

**9. Non Functional Requirements**

In addition to the obvious features and functions that you will provide in your system, there are other requirements that don't actually DO anything, but are important characteristics nevertheless. These are called "non-functional requirements" or sometimes "Quality Attributes." For example, attributes such as performance, security, usability, compatibility. aren't a "feature" of the system, but are a required characteristic. You can't write a specific line of code to implement them; rather they are "emergent" properties that arise from the entire solution. The specification needs to describe any such attributes the customer requires. You must decide the kind of requirements that apply to your project and include those that are appropriate.

Each requirement is simply stated in english. Each requirement must be objective and quantifiable; there must be some measurable way to assess whether the requirement has been met.

Often deciding on quality attributes requires making tradeoffs, e.g., between performance and maintainability. In the APPENDIX you must include an engineering analysis of any significant decisions regarding tradeoffs between competing attributes.

Here are some examples of non-functional requirements:

**Performance requirements**

Requirements about resources required, response time, transaction rates, throughput, benchmark specifications or anything else having to do with performance.

For better performance the application will restrict the document size to 5 MB.

**Operating constraints**

List any run-time constraints. This could include system resources, people, needed software,

The application must run without any manual intervention.

**Platform constraints**

Discuss the target platform. Be as specific or general as the user requires. If the user doesn't care, there are still platform constraints.

Since the application will be developed in PHP it is platform independent.

**Accuracy and Precision**

Requirements about the accuracy and precision of the data. (Do you know the difference?) Beware of 100% requirements; they often cost too much.

**Modifiability**

Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person- months).

Minimal

**Portability**

The effort required to move the software to a different target platform. The measurement is most commonly person-months or % of modules that need changing.

Minimal

**Reliability**

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

**Security**

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time, ...) to break into the system. Do not discuss solutions (e.g. passwords) in a requirements document.

Only secured users can access the application.

No one can go to any independent page without logging in.

**Usability**

Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

**Legal**

There may be legal issues involving privacy of information, intellectual property rights, export of restricted technologies, etc.

**10. Feasibility Study**

You should provide a feasibility report in the following format:

* **Product:**A general statement of the product; give a brief description of what the proposed system will do, highlighting where the proposed system meets the specified business requirements of the organization.
* **Technical Feasibility:**Will the proposed system perform to the required specification? Outline technical systems options you propose to use, which will give a technical solution satisfying the requirements and constraints of the system, as outlined in the terms of reference.
* **Social Feasibility:**Consideration of whether the proposed system would prove acceptable to the people who would be affected by its introduction. Describe the effect on users from the introduction of the new system; consider whether there will be a need for retraining the workforce. Will there be a need for relocation of some of the workforce? Will some jobs become deskilled? Will the current workforce be able to perform effectively any new tasks introduced by the proposed system? Describe how you propose to ensure user co-operation before changes are introduced.
* **Economic Feasibility:**Consider the cost/benefits of the proposed system. Detail the costs that will be incurred by the organisation adopting the new system; consider development costs and running costs. Detail benefits that the new system will bring, direct economic benefits such as reduced costs, and indirect benefits, such as improved management information and better customer service. Illustrate the cost/benefit of the new system by applying a suitable cost/benefit analysis method such as the payback method.
* **Market Research:**A comprehensive market research identifying a need for the product. Detail all market research you carried out, listing sources of information. Justify any conclusions you have drawn from your research. Identify the potential customer base for your product, together with evidence of customer need for the product. Describe how you propose to compete with similar products on the market.
* **Alternative Solution:**Consideration of alternative solutions should be documented. At least two alternative business or technical systems options should be considered. Detail the differences between these options and the proposed system. Justify your choice of the proposed system and the reasons for rejecting the alternative options.

At this point, all of the planning for the project has been done and if the feasibility study has shown that the project is likely to succeed within its constraints, then it only remains for us to start the requirements analysis and thus proceed with the project.

|  |  |
| --- | --- |
| **Feasibility Study** | |
| System: KMS | Date: |
| Author: Saumitra Das | Page: 1 |
| **Product** | |
| The project requires a web application to be developed that will allow online knowledge/document/paper sharing. | |
| **Technical Feasibility** | |
| The web application will be developed using PHP and MySQL. The team is competent in that. | |
| **Social Feasibility** | |
| Some training for the users/admin are required but all users are IT literate. | |
| **Market Research** | |
| Market research says that this application would be useful for the users as it could seamlessly help them to share documents. | |
| **Economic Feasibility** | |
| The application can be developed within budget. | |
| **Alternate Solution** | |
| Could be a desktop system but that would not allow documents to be shared online. | |

**11. Project Planning**

Project planning is concerned with identifying the following for every project:

* Activities
* Milestones
* Deliverables.

A plan must be drawn up to guide the development towards the project goal. A plan is drawn up at the start of a project. This plan should be used as the driver for the project. The initial plan is not static, and must be modified as the project progresses.

Planning is required for development activities from specification through to delivery of the system.

**12. Project Scheduling**

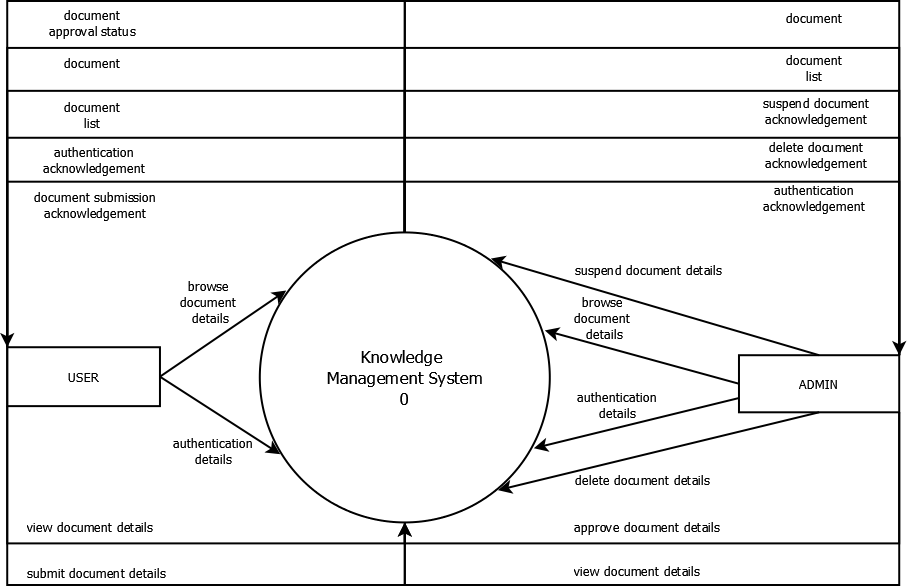
**GANTT chart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **Person(s) Responsible** | **Week 1** | **Week 2** | **Week 3** | **Week 4** |
| **Communication** |  |  |  |  |  |
| **Quick Plan** |  |  |  |  |  |
| **Modeling Quick Design** |  |  |  |  |  |
| **Construction of Prototype** |  |  |  |  |  |
| **Deployment, Delivery and Feedback** |  |  |  |  |  |

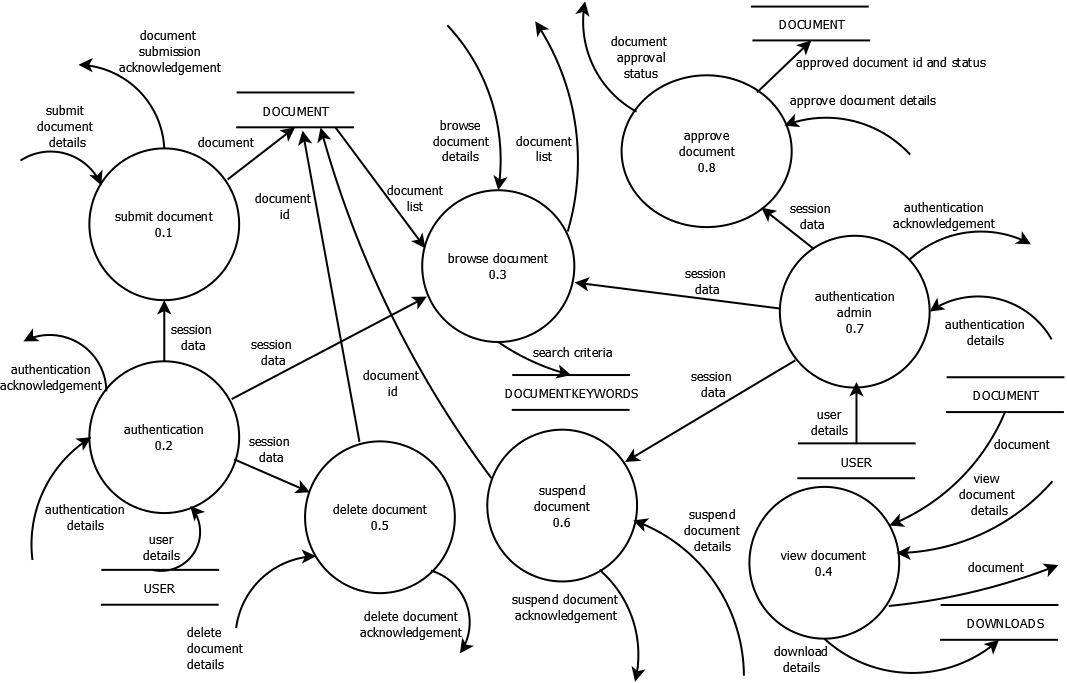
**13. Software Engineering Paradigm Applied**

**Data Flow Diagrams**

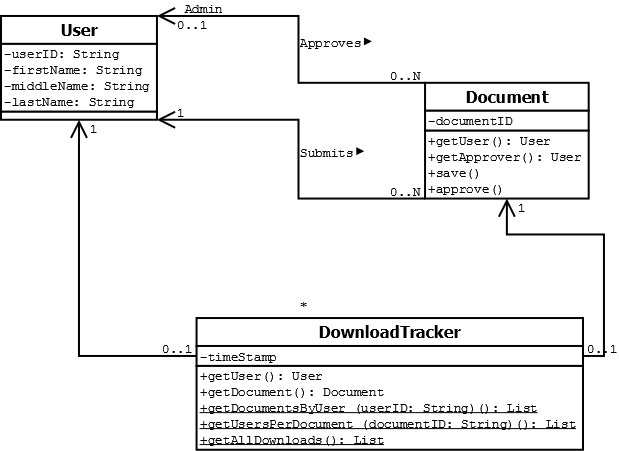
**Level 0**

****

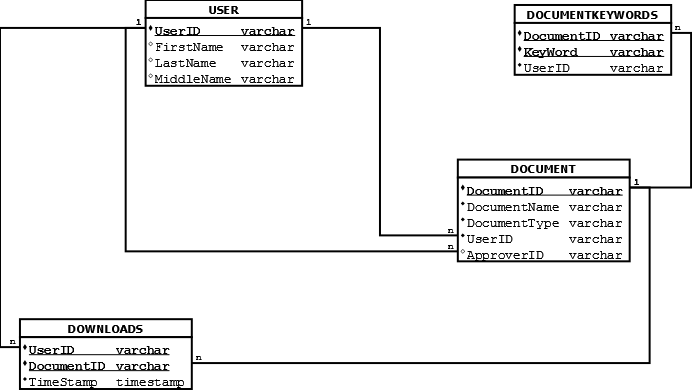
**Level 1**

****

**Class Diagram**

****

**14. Schema/Database Design**

****

**15. User Interface Design**

**16. Coding**

**17. Testing**

## Team Interaction

The following describes the level of team interaction necessary to have a successful product.

* The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.
* The Test Team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.
* Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

# Test Objective

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

# Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.
3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.
4. Identify the expected results for each test.
5. Document the test case configuration, test data, and expected results.
6. Perform the test(s).
7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).
8. Successful unit testing is required before the unit is eligible for component integration/system testing.
9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

# Testing Process

**b.** Design System Test

**e.** Design/Build Test Proc.

**f.** Signoff

**c.** Design/Build Test Proc.

**a.** Organize Project

**d.** Organize Project

The diagram above outlines the Test Process approach that will be followed.

1. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.

**b.** **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

**c.** **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

**d.** **Build Test Environment** includes requesting/building hardware, software and data set-ups.

**e. Execute System Tests –** The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

**f.** **Signoff** - Signoff happens when all pre-defined exit criteria have been achieved.

# Testing Strategy

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The test cases that will be used for designing use cases is shown in Figure 2.1 and onwards.

**Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Student1 name | |
| **Test Type** | | Unit Testing | |
| **Test Case Number** | | 1 | |
| **Test Case Name** | | User Identification | |
| **Test Case Description** | | The user should enter his/ her accurate userid and password so that he/she can able to go for the further options. The test case will check the application for the same since a user can only login with the correct userid , password. | |
| **Item(s) to be tested** | | | |
| 1 | Verification of the userid and password with the record in the database. | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| 1. Correct User id and password      1. Incorrect Id or Password | | | 1. Successful login 2. Failure Message |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tested By:** | | Student2 name | |
| **Test Type** | | Unit Testing | |
| **Test Case Number** | | 2 | |
| **Test Case Name** | | Submit Document | |
| **Test Case Description** | | The User submit document. For document submission the must login. | |
| **Item(s) to be tested** | | | |
| 1 | Check whether the user id logged in. | | |
| 2 | Check if the user has selected a document and the size of that is < 5 MB. | | |
| **Specifications** | | | |
| **Input** | | | **Expected**  **Output/Result** |
| 1. Trying to submit document without logging in. 2. Check whether document is selected. 3. Check whether the size of the document is < 5 MB. | | | 1. The user is redirected to the login page. 2. A message asks the user to select a document. 3. Return an error message. |

## Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

### White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

### Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

## System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case well focus only on function validation and performance. And in both cases we will use the black-box method of testing.

**18. System Security measures (Implementation of security for the project developed)**

* Only authorized users are allowed.
* Without signing in users are not allowed to go an intermediate page by typing an URL. For all such efforts, users will be redirected to the home page.

**19. Database/Data security**

* Database is present in remote machine.
* Oracle’s default securities are applied.

**20. Creation of User profiles and access rights**

* The admin must create users manually
* The admin can create more admins

**21. Cost Estimation of the Project along with Cost Estimation Model**

# Analogous estimate of effort or cost

Used for Early Estimate or Individual Activity Estimate

Sample example shown below is for two major deliverables of a software project. You use a previous project as a benchmark for analogous estimation. Using your experience you will estimate a multiplier.

Multipliers:

1. Prototyping: 0.75.
2. Testing: 0.5
3. Deployment: 0.5

Finally, if you want to convert to cost, you would use current rates for the resource.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **WBS ID** | **Previous**  **Similar Project**  **Activity** | **Previous**  **Effort** | **Current**  **Project**  **Estimate** | **Multiplier** | **Effort**  **(Previous Effort \* 0.75)** | **Cost**  **(Rs. 500/hr.)** |
| 1 | Prototyping | 40 Work-Hours | Prototyping | 0.75 | 30 Work-hours | Rs. 15000/- |
| 2 | Testing | 20 Work-Hours | Testing | 0.50 | 10 Work-Hours | Rs. 5000/- |
| **Total** |  |  |  |  | **40 Work- Hours** | **Rs. 20000/-** |

Note: Effort is also called Size and unit of estimation is called either Work-Hour, person-hours.

**22. Reports (sample layouts should be placed)**

**23. Future scope and further enhancement of the Project**

KMS has lot of enhancement options. In future documents may be classified category-wise. It may try to analyze the user behavior and preferences and accordingly suggest document categories. AI concepts can be applied to make KMS intelligent.

**24. Bibliography**

1. Roger S. Pressman. Software Engineering: A Practioner's Approach (Sixth Edition, International Edition). McGraw-Hill, 2005.
2. Ian Sommerville. Software Engineering (Seventh Edition). Addison-Wesley, 2004.
3. Frederick P. Brooks. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition. Addison-Wesley Pub Co; 1st edition (August 2, 1995).
4. 1st Edition: 1975. Tells the story of the IBM 360 Operating System, and what failed why.
5. Add Some HTML, CSS, Javascript, Jquery and DBMS Book(Minimum 3 book each).