**NATIONAL INSTITUTE OF TECHNOLOGY**

**TIRUCHIRAPPALLI-15**

****

**Department of Computer Applications**

**Case Tools Lab**

**Software Requirement Specification**

**On**

*A HOSTEL PORTAL*

*SUBMITTED BY****:*** *Under the guidance of*

**AMIT KOLARIYA Dr. B. Janet**

**ROLL NO: 205112077**

***TABLE OF CONTENTS***

1. PROBLEM STATEMENT
2. SOFTWARE REQUIREMENT SPECIFICATION
3. PROCESS MODEL
4. OBJECT MODEL DIAGRAM
5. SEQUENCE DIAGRAM
6. USE CASE DIAGRAM
7. STATECHARTS
8. ENTITY RELATIONSHIP DIAGRAMS
9. DATA FLOW DIAGRAMS
10. TESTING
    1. INTRODUCTION
    2. WHITE BOX/ BLACK BOX THEORY
    3. STRATEGIES FOLLOWED

11. ADVANTAGES & DISADVANTAGES

12. FUTURE SCOPE

13. BIBILOGRAPHY

***PROBLEM STATEMENT***

**NITT : A Hostel Portal**

**Problem :**

Today students has face to do all the thing manually .They have to do paper work for complain related to their respective hostel.

For a particular problems , student have to wait for a long while. **Also they have to wait and want to ask again and again at hostel office for hostel fee receipt.**

Due to manual nature it is difficult to request, grant and manage people availability and also takes a lot of time.

**How to Overcome these problems by proposed system :**

The project is mainly focused on the general problem of this big college. The project is an Intranet based application that can be accessed throughout the college on Delta server.

This system will reduce a lot of paper work and hence load on administrative staff.

**The project is taking care of certain things which are as follows:**

* Hostel related complain.
* Contact of all the wardens and hospital.
* Allotment list of students.
* Feedback Box.

**SOFTWARE REQUIREMENT SPECIFICATION**

1. **INTRODUCTION**
   1. **Purpose**

The origin of most software systems is in the need of a client, who either wants to automate the existing manual system or desires a new software system. The software system is itself created by the developer. Finally, the end user will use the completed system. Thus, there are three major parties interested in a new system: the client, the user, and the developer. Somehow the requirements for the system that will satisfy the needs of the clients and the concerns of the users have to be communicated to the developer. The problem is that the client doesn’t usually design the software or the software development process and the developer does not understand the client’s problem and the application area. This causes a communication gap between the parties involved in the development of the project.

The basic purpose of Software Requirement Specification (SRS) is to bridge this communication gap. SRS is the medium through which the client’s and the user’s needs are accurately specified; indeed SRS forms the basis of software development.

Another important purpose of developing an SRS is helping the clients understanding their own needs. An SRS establishes the basis for agreement between the client and the supplier on what the software product will do.

An SRS provides a reference for validation of the final product.

A high quality SRS is a prerequisite to high quality software and it also reduces the development cost.

The objective is to provide users with software with which they can interact and can get most of their initial work done and also for getting detailed information about judgment made on a case.

* 1. **Definition, acronym, abbreviation**

**Acronyms**

* DFD Data Flow Diagram
* ERD Entity Relationship Diagram
* SRS Software Requirement Specification
* ID no. Identification Number
  1. **Reference**
* Software Engineering, A Practitioners approach Edition 5

by **Roger S. Pressman**

McGraw Hill Publications

1. **Overall Description**
   1. **Product perspective**

* The digitalization of the system will reduce a lot of paperwork and hence the load on the court administrative staff.
* The machine performs all calculations. Hence chances of error are nil.
* The data can easily be retrieved and any required addition can be performed.
* The system provides for user-ID validation, hence unauthorized access is prevented.
  + 1. **User Interface**
* A login Screen (For entering roll number and recording Thumb Impression)
  1. **User Characteristics**
* EDUCATIONAL LEVEL

At least user of the system should be comfortable with English language.

* TECHNICAL EXPERTISE

User should be comfortable using general purpose applications on the computer system.

* 1. **Constraints**

The system will run under windows XP or higher platforms of operating system.

* 1. **Assumptions**

Administrator will be having a valid account type, roll number to access the software.

1. **Specific Requirements**

**3.1 Performance Requirements**

There are certain performance requirements that the software tries to meet. These are:

* + - * **User Friendliness -** The software terms should be self explanatory and use –friendly. This makes the task of the user much easier.
      * **Ease To Maintain -** Software should not have high maintenance cost. Design should be simple and concise so that problems regarding maintenance are least expected.
      * **Error Free - S**oftware should be developed in such a manner that it is most accurate. Scope for error within the software should be minimized.
  1. **Other Requirements**

Software should satisfy following requirements as well:-

* + - * Security
      * Portability
      * Correctness
      * Efficiency
      * Flexibility
      * Testability
      * Reusability

1. **Design constraints**

There are a number of factors in the client’s environment that may restrict the choices of a designer. Such factors include standards that must be followed, resource limits, operating environment, reliability and security requirements, and policies that may have an impact on the design of the system. An SRS (Software Requirements analysis and Specification) should identify and specify all such constraints.

**Standard Compliance:** This specifies the requirements for the standards the system must follow. The standards may include the report format and accounting procedures.

**Hardware Limitations**: The software may have to operate on some existing or predetermined hardware, thus imposing restrictions on the design. Hardware limitations can include the type of machines to be used, operating system available on the system, languages supported, and limits on primary & secondary storage.

**SYSTEM PROCESS MODEL**

To solve a particular problem a software engineer or a team of software engineers must incorporate a development strategy than encompasses the process, methods and tools. This strategy is often referred to as a process model or a “SOFTWARE ENGINEERING PARADIGM”. The use of a particular process model or software paradigm is based on the nature of the application.

A software life cycle is the series of identifiable stages that a software product undergoes during its lifetime. A software life cycle model is a descriptive and diagrammatic representation of the software life cycle. A life cycle model represents all the activities required to make a software product transit through its life cycle phases. It also captures the order in which these activities are to be undertaken. A life cycle model maps the different activities performed on a software product from its inception to retirement.

Different life cycle models may map the basic development activities to phases in different ways. Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models. During any life cycle phases, more than one activity may also be carried out.

A process covers all the activities beginning from product inception through delivery and retirement. It addresses the broader issue of reuse, documentation, testing, parallel work carried out by team members and coordination with the customer.

A software application may be for an application of real time system, a database system, an operating system or a web based application. Primary advantage of using a life cycle model is that it encourages development of software in a systematic & disciplined manner.

Using a process model gives a good understanding of project to all the team members i.e. what to do & when? It prevents misinterpretations, helps in identifying inconsistencies, redundancies.

Some of them are given below: -

* Waterfall model
* Prototype model
* Iterative enhancement model
* Spiral model
* Rapid action development model (RAD Model)
* Fourth generation technique (4GT)

The following points states the need of a particular software paradigm for development of a software.

* To improve the quality of software.
* To increase the productivity of software development.
* To development of software on time.
* To produce a reliable software.
* To develop the software with in cost estimates.

Thus the above said discussion states that a software paradigm is very necessary to develop software.

So in our project we are following the Iterative model for the development of our project.

**Iterative Enhancement Model: -** This model provides the benefit of both Prototype and Waterfall model. The basic idea is that the software should be developed in increments, each increment adding some functional capability to the system until the full system implemented.

Feasibility study

Requirement analysis and specification

Design

Coding and unit testing

Integrating and system testing

Maintenance

#### ITERATIVE WATERFALL MODEL

Object Model Diagrams:

# HostelPortal

****

***SEQUENCE DIAGRAM***

****

Use Case Diagrams:

# HostelPortal

****

***STATECHARTS***



***STATECHARTS***



***ENTITY RELATIONSHIP DIAGRAM***

**Server**

**Student**

verify

enter

**Administrator**

Drawing Entity Relationship diagram between the various entities of Hostel Portal

**DATA FLOW DIAGRAM**

response

feedback

complain

verify

enter

**Administrator**

**Student**

Drawing Data flow diagram between the both student’s and administrator

**FEASIBILITY STUDY**

There are three major areas in feasibility study:

**1. Operational feasibility**

This feasibility is dependent upon determining human resources for the project.

* Whether the system will fulfill its department’s requirements or not?
* This test of feasibility asks if the system will work when developed.
* Is there sufficient support from the management?

**2. Technical feasibility**

This feasibility determines whether the project is technically feasible.

* Does the necessary technology exist to do what is proposed?
* Will there be guarantee of accuracy and reliability?
* Will there be easy access to data?
* Will the data be secure?

**3. Economic feasibility**

This feasibility determines the resource requirements and the basic resources that are considered-

* Management time
* Time spent by the system analyst
* Cost of hardware and software

**TESTING**

**INTRODUCTION**

Software Testing is the process of executing a program or system with the intent of finding errors. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible.

It is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation. Once source code has been generated, software must be tested to uncover as many tests as possible before delivery to the customer.

These techniques provide systematic guidance for designing tests that exercise the internal logic of software components and exercise the input and output domains of the program to uncover errors in program function, behavior, and performance. Unlike most physical systems, most of the defects in software are design errors, not manufacturing defects. Software does not suffer from corrosion, wear-and-tear generally it will not change until upgrades, or until obsolescence. So once the software is shipped, the design defects or bugs will be buried in and remain latent until activation.

**TWO WAYS OF TESTING**

**1. BLACK BOX THEORY**

Also called behavioral testing, focuses on the functional requirements to the software. That is, black box testing enables the software engineer to derive sets of inputs conditions that will fully exercise to white-box techniques. Alludes to tests that are conducted at the software interface. Although they are designed to uncover errors, black box tests are used to demonstrate that software functions are operational, that input is properly accepted and output is correctly produced, and the integrity of external information is maintained. A black box test examines some fundamental aspects of a system with little regard for internal logical structure of the software.

**2. WHITE BOX THEORY**

Sometimes called glass-box testing, is a test case design that uses the control structure of the procedural design to derive test cases. It is predicated on close examination of procedural details. Logical paths through the software are tested by providing test cases that exercise specific sets of conditions and/or loops. The “status of the program” may be examined at various points to determine if the expected or asserted status corresponds to the actual status.

**software testing strategy**

there are four types of software testing strategy:-

1. Unit testing: begins at the vortex of the spiral and concentrates on each unit of the software as implemented in source code.
2. Integration testing: focus is on design and the construction of the software architecture.
3. Validation testing: where requirements established as part of software requirements are validated against the software has been constructed.
4. System testing: where the software and other system elements are tested as a whole.

**Now, we are using system testing in our project because**:

These tests fall outside the scope of the software and are not conducted solely by software engineers. However, steps taken during software design and testing can greatly improve the portability of successful software in the larger system.

**Recovery Testing:** Is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic, reinitialization, check pointing mechanisms, data recovery, and restart are evaluated for correctness.

**Security Testing:** Attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration. During security testing, the tester plays the role of the individual who desires to penetrate the system.

**Performance Testing:** Designed to test the run-time performance of software within the context of an integrated system. Performance testing occurs throughout all steps in the testing process. Performance tests are often coupled with stress testing and usually require both hardware and software instrumentations.

**LIMITATIONS OF THE SOFTWARE**

* Extensive help is not provided.
* Not support in the OS lower than windows XP.3

**ADVANTAGES OF PROPOSED SYSTEM**

**There are many advantages by the online hostel portal for student:**

1. Eliminates paper based leave application forms
2. Complains can be submitted online
3. Complain can be approved online
4. Student will get fees receipt on the hostel portal.
5. Time will be reduces for both administrator and student.

**FUTURE SCOPE :**

In present all complains and their solution has been computerized through centralized software. So by these the project is very useful for completing the website of NITT.

**BIBLIOGRAPHY**

**1. INTERNET RESOURCES**

* www.google.com

**2. BOOKS**

* SOFTWARE ENGINEERING - BY PRESSMAN