  **Department of Computer Science and Engineering**

**Company Profile**

Company Profile Link : http://harha.liveblog365.com/

**Course Code : CSE236**

**Course Tittle : Software Project II**

**Semester : Fall-2023**

Submitted To Submitted By

Md. Golam Rabbany Anukul Chandra

Lecturer, Dept. of CSE ID: 221-15-5585

Daffodil International University. Section: 61\_U

Submission Date: 29 November, 2023

Prof. Md .Lutfur rah

Vice-Chancelor

Daffodil International Universty

Savar ,Ashulia

Dear Sir,

I hope this letter finds you in good health and high spirits. I am writing to formally request your consideration in establishing a mutually beneficial agreement between our organization and Daffodil International University implement a project “Attendence Management system ”. This collaboration aims to enhance time savings, cost savings, easiest way to handling Students data and Students attendance .

Here is my System Requirement Specification for my proposed “Student Attendence System”. I kindly request to read my specification and a meeting at your earliest convenience to discuss the details of this proposed agreement.

We are excited about the potential collaboration between our organizations and the positive impact it could have on both our customers and the community at large.

Thank you for considering our proposal. We eagerly await the opportunity to engage in a productive discussion with you regarding this potential partnership.

Yours sincerely,

Anukul Chandra

Senior. Software Developer

HHHR

Contact us

Email: [anukulchandra4949@gmail.com](mailto:anukulchandra4949@gmail.com)

Cell: 01957541387

**Software Requirements**

**Specification(SRS)**

Of

**Attendance Management System**

Version 4.0

Table of Contents

[Introduction 6](#_Toc152172361)

[Purpose 6](#_Toc152172362)

[Scope 6](#_Toc152172363)

[Definitions, Acronyms, and Abbreviations 6](#_Toc152172364)

[Overview 7](#_Toc152172365)

[Product Perspective 8](#_Toc152172366)

[Product Functions 8](#_Toc152172367)

[Administrator: 8](#_Toc152172368)

[User: 8](#_Toc152172369)

[User Characteristics 8](#_Toc152172370)

[Assumptions and Dependencies 9](#_Toc152172371)

[Use Cases 10](#_Toc152172372)

[Classes / Objects 11](#_Toc152172373)

[Technology Requirements 12](#_Toc152172374)

[External Interface Requirements 12](#_Toc152172375)

[User Interfaces 12](#_Toc152172376)

[Software requirement : 12](#_Toc152172377)

[Hardware Requirement : 12](#_Toc152172378)

[Software Interfaces 13](#_Toc152172379)

[Communications Interfaces 13](#_Toc152172380)

[Functional Requirements : 13](#_Toc152172381)

[Student Registration: 13](#_Toc152172382)

[Student Attendance Management: 14](#_Toc152172383)

[Opinion Management: 14](#_Toc152172384)

[Non-Functional Requirements 15](#_Toc152172385)

[Performance 15](#_Toc152172386)

[Reliability 15](#_Toc152172387)

[Availability 15](#_Toc152172388)

[Security 16](#_Toc152172389)

[Maintainability 16](#_Toc152172390)

[Portability 16](#_Toc152172391)

[Design Constraints 16](#_Toc152172392)

[Other Requirements 16](#_Toc152172393)

[Project Timeline 17](#_Toc152172394)

[Pricing and Payment 17](#_Toc152172395)

[Installation System 17](#_Toc152172396)

[Software Development Life Cycle (SDLC) Model : 18](#_Toc152172397)

[Waterfall Model : 18](#_Toc152172398)

[RAD Model : 18](#_Toc152172399)

[Spiral Model : 18](#_Toc152172400)

[V-Model : 18](#_Toc152172401)

[Incremental Model : 19](#_Toc152172402)

[Agile Model : 19](#_Toc152172403)

[Iterative Model : 19](#_Toc152172404)

[Big Bang Model : 19](#_Toc152172405)

[SDLC Model for this Project: Agile Methodology : 20](#_Toc152172406)

[Analysis Models 21](#_Toc152172407)

[Data Flow Diagram ( DFD ) 22](#_Toc152172408)

[State-Transition Diagrams (STD) 23](#_Toc152172409)

[TESTING 24](#_Toc152172410)

[TESTING LEVELS 24](#_Toc152172411)

[TESTING TECHNIQUES 26](#_Toc152172412)

[TESTING TACTICS 26](#_Toc152172413)

[Specific testing Consideration For Attendance management System 27](#_Toc152172414)

[Proposed testing process 27](#_Toc152172415)

[MEASUREMENT IN SOFTWARE TESTING 27](#_Toc152172416)

[Effectiveness and Efficiency 28](#_Toc152172417)

[Appendices 29](#_Toc152172418)

[Appendix : Glossary 29](#_Toc152172419)

[Contact Us 29](#_Toc152172420)

[Reference: 30](#_Toc152172421)

# Introduction

Student management system has become important factors in modern education field. This system should help the institutional to streamline the administrative task and provide real-time access to the data. Building this system in web based interface will further help the ease of accessibility through any web browser. The study findings enable the definition of the project problem statement, its objectives, scopes and advantages of the student management system.

## Purpose

The purpose this documents is to present a detailed description of the Student Management System. It will explain the purpose and features of the software, the interfaces of the software, what the software will do, the constraints under which it must operates and how the software will react to external stimuli. This document is intended for both the end users and the developers of the software.

## Scope

This document covers the requirements for the Student Management System. This software will provide a graphical environment in which the users of the system will be able to perform various operations that are associated with storing, marinating, updating and retrieving Student information. The purpose of this is to guide developers in selecting a design that will be able to accommodate the full-scale application.The system will capture information about student’s personal details lectures and the courses**.** Storing updating and retrieving in a fast and accurate way.

## Definitions, Acronyms, and Abbreviations

The Student Management System has to handle records for many number of students and maintenance was difficult. Though it has used an information system, it was totally manual. Hence there is a need to upgrade the system with a computer based information.

## Overview

The purpose this documents is to present a detailed description of the Student Management System. It will explain the purpose and features of the software, the interfaces of the software, what the software will do, the constraints under which it must operates and how the software will react to external stimuli. This document is intended for both the end users and the developers of the softwar

## Product Perspective

**General Description**

The product Student Management system, is an independent product and does not depend on any other product or system. The product will automate various tasks associated with handling student details and better organizing the stored information and optimum performance, thus helping the Colleges to ensure smooth working of these processes.

## Product Functions

Our system has two types of accessing modes,

* + 1. Administrator
    2. User
    3. Teacher
    4. Student

### Administrator:

SMS is managed by Administrator. Administrator has to update and monitor the registered student details, add a new student, provide register number for all students, assign each student a course etc., Administrator can update his profile, and also can give help to the teachers and students.

### User:

There are two users:

* 1. **Student**:

User can only view their personal details, course assigned, and edit their assigned course and can view their attendance.

* 1. **Teache**r:

User can add them onto the portal and view their schedules, marks attendance of the students, also can view the students details in graphical order, also of a single student and about the views from the students.

## User Characteristics

This software gives access to two kinds of users.

* + 1. **Administrator**: The personnel and College administrator will have administrator access to add, delete and modify information stored in the database.
    2. **Authorized User**: Teaching staff will have access to only view the data stored in the database and can update the student’s attendance in the form of formatted reports.

## Assumptions and Dependencies

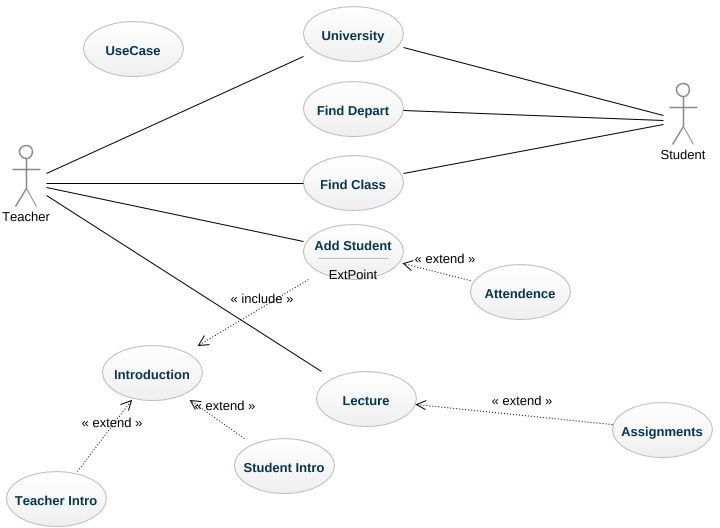
* We assume that the Office personnel do all the data entry based and the correct values obtained from forms and registers.
* We assume that the computers that will use the software will be part of

the college LAN.

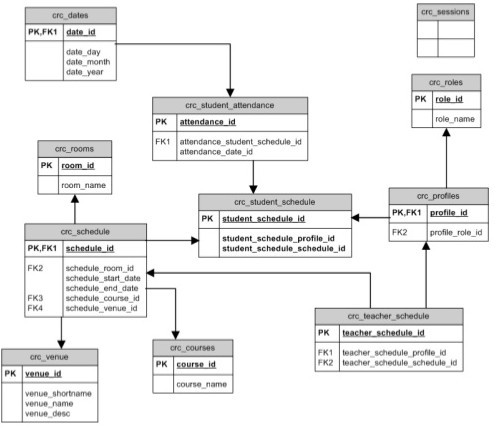
* Users with administrator access should be careful in deleting or modifying any information knowingly or unknowingly which will lead to inconsistency of the database.
* The end users of this software are assumed to have basic level of

computer knowledge i.e. point and click.

# Use Cases



# Classes / Objects



# Technology Requirements

## External Interface Requirements

## User Interfaces

* + - * GUI along with meaningful Frames and buttons
      * Reports are generated as per the requirement
      * Refer Appendices 2.

## Software requirement :

|  |  |
| --- | --- |
| **Name of component** | **specification** |
| Operating System | Win XP/ Vista/ 7,8,10,11 Linux |
| Language | Php |
| Data Base | Mysql Server |
| Web Browser | Mozila firefox ,opera ,chrome |

## Hardware Requirement :

|  |  |
| --- | --- |
| **Name of component** | **specification** |
| Processor | Dual Core 2nd Gen |
| RAM | 512 MB |
| HDD | 80 GB |
|  |  |

## Software Interfaces

|  |  |
| --- | --- |
| Front End | PHP 5.3.0 |
| Back End | MySQL 5.1.36 |

When invalid inputs are given to the modules then the error messages will be popped up in order to inform the user that the input provided is not taken by the database. When incomplete information is provided by the user and the user tries to submit the form in order to store the details in the database the system will pop up a message box asking the user to enter all the details required.

### Communications Interfaces

The machine will have to be part of the college Local area Network to access the central database.

# Functional Requirements :

Student Management System involves the following functions

### Student Registration:

* SMS provides on line registration and status information to the student to view their status.
* SMS provides automatic student register number generation basedon course and year

SMS provides to students to add them in their course they want to study.

### Student Attendance Management:

* Easily track attendance information of students.
* Quickly produce single or multiple day attendance bulletins.

### Opinion Management:

* SMS provides a comprehensive opinion scheduling based on course.
* Students can facilitate to give their opinions by giving the teacher rank

# Non-Functional Requirements

### Performance

Easy tracking of records and updating can be done. All the requirements relating to performance characteristics of the system are specified in the section below. There are two types of requirements.

1. **Static Requirements**

These requirements do not impose any constraints on the execution characteristics of the system. They are:

* 1. **Number of Terminals :**

The software makes use of an underlying database that will reside at the server, while the front end will be available online to the administrative and departmental computers as well as students and teachers.

* 1. **Number of Users :**

The number of users may vary, as this software finds applications in almost all department of the organization.

1. **Dynamic Requirements**

These specify constraints on the execution characteristics of the system. They typically include response time and throughout of the system. Since these factors are not applicable to the proposed software, it will suffice if the response tine is high and the transactions are carried out precisely and quickly.

### Reliability

The software will not be able to connect to the centralized database in the event that the college LAN fails or in the event of the server being down due to a hardware or software failure.

### Availability

The software will be available only to authorized users of the colleges like teachers to mark the students attendance, student to view their enrolled course, admin to add an update students records

### Security

The security requirements deal with the primary security. The software should be handled only by the administrator and authorized users. Only the administrator has right to assign permission like creating new accounts and generating password. Only authorized users can access the system with username and password.

### Maintainability

Backups for database are available.

### Portability

The Software is a web-based application and is built in PHP and MYSQL so it is platform independent and is independent of operating system.

## Design Constraints

This software provides security. The login form prevents the system from being misused by unauthorized users. Only an authorized operator will be granted rights to modify as per requirements. This software is also reliable and fault tolerant. The system developed is designed to handle invalid inputs. Since reliability is major area of concern the system has a backup to avoid data loss. The user should know the programming language very well that is used to develop a software.

# Other Requirements

**Database :** All the data will be stored in a relational database.

# 

# Project Timeline

|  |  |
| --- | --- |
| **Milestone** | **Time** |
| Analysis | 3 days |
| Requirements Collection | 7 days |
| Development | 30 days |
| Testing | 10 days |
| Deployment | 5 days |
| Delivery | 5 days |

# Pricing and Payment

The entire fees will be Seventy Thousand taka(70000) only

## Installation System

25% (25%)

Paid on acceptance of this proposal.

30% (55%)

Paid on signing of our Application development agreement.

30% (85%)

Paid at 70% Application Demonstration.

15% (100%)

Paid at completion the Application.

# Software Development Life Cycle (SDLC) Model :

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**." Each process model follows a series of phase unique to its type to ensure success in the step of software development. **Here, are some important phases of SDLC life cycle:**

## Waterfall Model :

The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

## RAD Model :

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period. The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.

* Business Modelling
* Data Modelling
* Process Modelling
* Application Generation
* Testing and Turnover

## Spiral Model :

The spiral model is a **risk-driven process model**. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.

Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).

The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.

## V-Model :

In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.

## Incremental Model :

It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.

## Agile Model :

Agile methodology is a practice which promotes continues interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

Any agile software phase is characterized in a manner that addresses several key assumptions about the bulk of software projects:

1. It is difficult to think in advance which software requirements will persist and which will change. It is equally difficult to predict how user priorities will change as the project proceeds.
2. For many types of software, design and development are interleaved. That is, both activities should be performed in tandem so that design models are proven as they are created. It is difficult to think about how much design is necessary before construction is used to test the configuration.
3. Analysis, design, development, and testing are not as predictable (from a planning point of view) as we might like.

## Iterative Model :

It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

## Big Bang Model :

Big bang model is focusing on all types of resources in software development and coding, with no or very little planning. The requirements are understood and implemented when they come.

This model works best for small projects with smaller size development team which are working together. It is also useful for academic software development projects. It is an ideal model where requirements are either unknown or final release date is not given.

# SDLC Model for this Project: Agile Methodology :

We have used **Agile Methodology** for this project, and here are some reasons:

• **Improved quality**: The agile model's sequential approach helps to ensure that each phase of the development process is completed thoroughly and that any defects are identified and fixed early on. This can lead to improved quality and fewer defects in the final product.

• **Reduced risk**: The agile model's emphasis on planning and risk

management can help to reduce the risk of project failure.

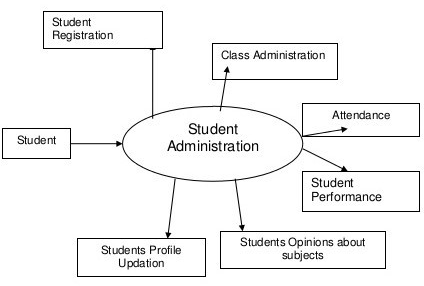
• **Increased predictability:** The agile model's sequential approach and well-defined phases make it easier to predict the project's timeline and budget. This can be important for hospital management systems, which often need to be deployed on a specific schedule and within a specific budget.

* **Easily and Quickly Adapt to Change:** Teams not only adapt to change in Agile, they are encouraged to embrace the practice. Agile acknowledges that customer needs change and that teams must be able to adapt. Working in time-boxed iterations means the team does not need to wait on a lengthy requirement change, review and approval process. Any change or maintenance item is added to the backlog and allotted to an upcoming sprint based on priority and business need.
* **Better Project Control:** Teams work together, along with the product owner, to determine what goes into each sprint. That way, the team is on the same page about what needs to be delivered. Also, there is less of a chance of surprises or unplanned

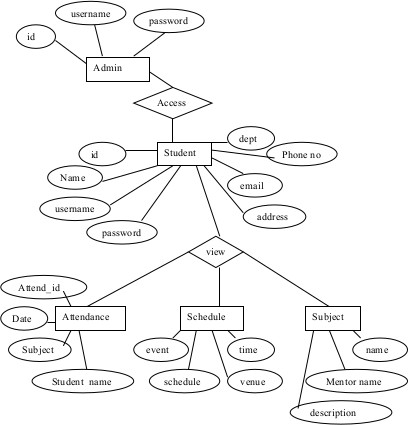
The agile model also has some disadvantages, such as its inflexibility and difficulty to make changes. However, for large and complex projects like Attendance management system projrct , the benefits of the agile model typically outweigh the disadvantages.

Overall, the agile model is a good choice for Attendance management system projects because it provides a structured and disciplined approach to software development that can help to ensure that the project is completed on time, within budget, and to the required quality stc.

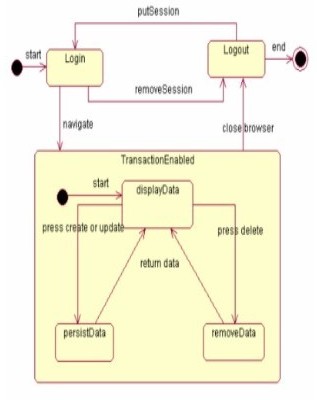
# Analysis Models



# Data Flow Diagram ( DFD )



# State-Transition Diagrams (STD)



# TESTING

PROPOSED TESTING APPROACH COMPARED TO OTHERS

There are various software testing approaches, each with its own strengths and

weaknesses. The most common approaches include:

**Waterfall :** This traditional approach involves a sequential testing process, where

each phase is completed before the next one begins. This approach is well-structured

and easy to understand, but it can be inflexible and time-consuming.

**Agile :** This approach is more flexible and iterative than waterfall, as it allows for

testing to be conducted throughout the development process. This approach is better

suited for dynamic projects where requirements may change frequently.

**DevOps:** This approach emphasizes collaboration between development and

operations teams to ensure that software is delivered quickly and reliably. Testing is

integrated into the development process, and automated tests are used to provide

continuous feedback.

## TESTING LEVELS

A comprehensive testing strategy for an AMS should encompass various testing

levels, each focusing on different aspects of the system's functionality and

performance. These levels can be broadly categorized as follows:

**Unit Testing:** Unit testing is the foundation of software testing, focusing on

individual units or modules of the code. It involves testing the smallest testable

components of the system to ensure they function correctly and adhere to the design

specifications.

**Integration Testing:** Integration testing verifies that different modules or

components of the system interact seamlessly with each other. It involves testing the

interfaces between modules to ensure data exchange and communication are

accurate and consistent.

**System Testing:** System testing evaluates the overall functionality of the HMS from

a user's perspective. It ensures that the system meets all specified requirements and

functions as intended from an end-to-end standpoint.

**User Acceptance Testing (UAT):** UAT is a critical phase in software testing that

involves real users interacting with the system to validate its usability and

acceptance.

**Performance Testing:** Performance testing assesses the system's ability to handle

expected workloads and maintain responsiveness under various load conditions.

**Security Testing:** Security testing evaluates the system's vulnerability to

cyberattacks and data breaches. It aims to identify and address potential security

weaknesses that could compromise patient data or disrupt system operations.

**Compatibility Testing :** Compatibility testing ensures that the HMS functions

correctly across different hardware configurations, operating systems, and web

browsers.

**Regression Testing:** Regression testing is an ongoing process that ensures that new

features or bug fixes do not introduce unintended regressions or break existing

functionality.

By implementing a comprehensive testing strategy that encompasses these various

testing levels, healthcare providers can ensure that their AMS is reliable, secure, and

meets the critical needs of patients and healthcare professionals.TESTING TYPES

There are three main types of software testing:

**Black-box testing:** (also known as behavioral testing) focuses on the software's

output without examining its internal structure or code. This type of testing is

typically done by testers who are not familiar with the software's code. Black-box

testing techniques include:

**•Equivalence partitioning**: Dividing the input data into groups based on

common characteristics and testing one representative from each group.

**•Boundary value testing**: Testing the software at the boundaries of its input

specifications, such as minimum, maximum, and invalid values.

•**Decision table testing**: Creating a table that maps all possible input

combinations to their expected outputs.

**White-box testing:** (also known as structural testing) examines the software's

internal structure and code to ensure that it meets its requirements. This type of

testing is typically done by testers who are familiar with the software's code. White-

box testing techniques include:

**•Unit testing**: Testing individual units of code, such as functions or modules.

•**Code coverage testing**: Measuring how much of the software's code has been

tested by the test suite.

•**Integration testing:** Testing how different units of code interact with each

other.

**Grey-box testing:** is a combination of black-box and white-box testing. Grey-box

testers may have some knowledge of the software's internal structure, but they do

not have access to the code. Grey-box testing techniques include:

•**Exploratory testing**: Ad-hoc testing where the tester tries to break the

software by exploring its functionality.•Scripted testing: Testing where the tester follows a pre-written script of steps

to execute.

•**Fuzz testing**: Testing where the tester feeds random or invalid data to the

software to see how it reacts.

## TESTING TECHNIQUES

There are many different testing techniques that can be used to test software,

depending on the type of testing being performed. Some common testing techniques

include:

•**Manual testing:** Testing the software by hand, without the use of automated

tools.

•**Automated testing:** Testing the software using automated tools, such as

Selenium or TestRail.

•**Static testing:** Analyzing the software code without executing it, to identify

potential defects.

•**Dynamic testing:** Executing the software to identify defects during runtime.

## TESTING TACTICS

Testing tactics are specific strategies that can be used to apply testing techniques

effectively. Some common testing tactics include:

•**Testing early and often:** The earlier defects are found, the cheaper they are

to fix.

•**Testing in a realistic environment:** Test the software in an environment that

is as close to the production environment as possible.

•**Testing for both positive and negative scenarios:** Test the software to

ensure that it handles both valid and invalid input.

•**Testing for performance and security:** Test the software to ensure that it

# Specific testing Consideration For Attendance management System

In addition to the general testing types, techniques, and tactics discussed above, there

are some specific testing considerations for hospital management systems:

•**Data integrity:** Attendance l management systems store and process sensitive

patient data, so it is critical to ensure that the data is accurate, complete, and

secure.

•**Compliance with regulations:** Attendance management systems must comply

with a variety of regulations, such as HIPAA in the United States.

•**Usability:** Attendance management systems must be easy for users to learn and

use, as they will be used by a variety of people with different levels of

technical expertise.

By following these testing types, techniques, and tactics, you can help ensure that

your hospital management system software is of high quality and meets the needs of

its users.

## Proposed testing process

Outline the step-by-step testing process. This may include:

**Test Planning:** Define the scope, objectives, resources, and schedule for testing.

**Test Design:** Develop detailed test cases and test scripts based on requirements.

**Test Execution:** Implement the tests and record the results.

**Defect Tracking:** Establish a system for identifying, documenting, and resolving

defects.

**Regression Testing:** Ensure that changes do not negatively impact existing

functionality.

**User Acceptance Testing (UAT):** Involve end-users in the testing process to ensure

the system meets their needs.

## MEASUREMENT IN SOFTWARE TESTING

Measurement is an essential part of software testing. It allows testers to assess the

effectiveness and efficiency of their testing efforts, and to identify areas for

improvement. There are a number of different metrics that can be used to measure

software testing, but some of the most common include:

**• Defect density:** This metric measures the number of defects per unit of code

or functionality.

**• Defect detection rate:** This metric measures the percentage of defects that

are detected by testing.

**• Test coverage:** This metric measures the percentage of code or functionality

that is covered by tests.

**• Test execution time:** This metric measures the time it takes to execute a test

suite.

In addition to these metrics, the Hierarchy of Testing Difficulty (HTD) can be used

to measure the difficulty of testing different aspects of software. The HTD is a

classification scheme that categorizes the difficulty of testing into five classes:

**• Class I:** There exists a finite complete test suite.

**• Class II:** Any partial distinguishing rate (i.e., any incomplete capability to

distinguish correct systems from incorrect systems) can be reached with a

finite test suite.

**• Class III:** There exists a countable complete test suite.

**• Class IV:** There exists a complete test suite.

**• Class V:** All cases.

The HTD can be used to identify areas of software that may be more difficult to test

and to focus testing efforts on these areas.

## Effectiveness and Efficiency

Effectiveness and efficiency are crucial aspects of software testing. While

effectiveness measures the ability to identify defects, efficiency focuses on finding

them quickly and resourcefully. Testers aim to achieve both effectiveness and

efficiency, but often face a trade-off. Extensive testing may be more effective but

also more costly and time-consuming. Careful consideration of this trade-off is

essential for designing testing approaches. Testers should utilize metrics to track

progress and identify areas for improvement.

# Appendices

## Appendix : Glossary

Following abbreviations have been used throughout this document:

**DFD** : Data Flow Diagram

**ERD** : Entity Relationship Diagram

**SRS** : Software Requirement Specification

**SQL** : Structured Query Language

**SMS** : Student Management System

**STD :** State Transition Diagram

# Contact Us

You can get in touch with us in any of the below ways:

By Phone:

01957541387

Anukul Chandra

By Email

Chandra15-5585@diu.edu.bd

On our website

By post

Savar , Ashulia

We look forward to hearing from you soon!

**Agreement Signed By:**

|  |  |  |
| --- | --- | --- |
| ……………………………    Client Signature  S M MASUM  Vice chancellor  DIU | …………………………..    Order Provider Signature  Officer  FnF IT | ……………………………    Authority Signature  Golam Rabbany  Managing Director (MD)  FnF IT |

# Reference:

<https://www.wikipedia.org/>

<https://www.geeksforgeeks.org/>