TITLE OF THE PROJECT

IUMS Admission Service

Course Code: CIS499

Fall: 2023

Submitted By

Kazi Sakib

ID: 201-16-511

Department of Computing and Information System (CIS)

DAFFODIL INTERNATIONAL UNIVERSITY

Supervised By

Mr. Md. Mehedi Hassan

Lecturer

Department of Computing and Information System (CIS)

DAFFODIL INTERNATIONAL UNIVERSITY

Date of Submission:

PROJECT BASED LEARNING

APPROVAL

This Project titled "IUMS Admission Service", Submitted by Kazi Sakib, ID No: 201-16-511 to the Department of Computing & Information Systems, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computing & Information Systems and approved as to its style and contents. The presentation has been held on 23-01-2024.

BOARD OF EXAMINERS

Md Sarwar Hossain Mollah — **Chairman**
Associate Professor and Head
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Dr. Shekh Abdullah-Al-Musa Ahmed — **Internal Examiner**
Assistant Professor
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Md. Nasimul Kader — **Internal Examiner**
Assistant Professor
Department of Computing & Information Systems
Faculty of Science & Information Technology
Daffodil International University

Prof. Mohammad Shorif Uddin, — **External Examiner**
Professor
Department of Computer Science and Engineering
Jahangirnagar University

Declaration

I hereby declare that; this project has been done by me under supervision of **Md Mehedi Hassan Lecturer** department of Computing and Information System (CIS) of Daffodil International University. I am also declaring that this project or any part of there has never been submitted anywhere else for the award of any educational degree like, B.Sc., M.Sc., Diploma or other qualifications.

Supervised By

23.07.2024

Mr. Md. Mehedi Hassan
Lecturer
Department of CIS
Daffodil International University

Submitted By

Sakib

Name: Kazi Sakib
ID: 201-16-511
Department of CIS
Daffodil International University

Acknowledgment

I want to start by saying how grateful I am to Almighty Allah for giving me this chance. Without the Almighty's help, I would be unable to complete this project. Finally, I want to thank DCL for choosing me as an intern trainee since this internship program has taught me a lot. It is believed that a teacher is a student's second parent since they provide them with the proper guidance, allowing them to progress in life. In my internship program, I had two professors who always advised me in making the best decisions and provided me with advice on how to successfully perform challenging tasks. They never allowed me to struggle alone and were constantly reducing my stress by providing me with mental fortitude. One of them is Md. Mazharul Islam Masum is my intern trainer and Mr. Md. Mehedi Hassan is supervising my academic work.

Additionally, I am appreciative of them for their kindness, readiness to provide a help, and faith in me. I want to thank them for helping me, for their instruction, for teaching me how to work through any challenges, and for teaching me how to handle a crucial project in the future.

Dedication

This project is dedicated to Almighty Allah, secondly my Mom, Dad and my elder sister who have all make it possible for me to complete my BSC Program, I return glory and honor to Almighty God for supporting me, protecting me and also guiding me. I also thank God for the blessing throughout my program.

Abstract

“IUMS Admission Service" is a web-based application. IUMS Admission Service can be used to manage daily operations at their university that will use university admission service purpose. In this Application I used Agile Model because Agile Model is an iterative and incremental process.

IUMS Admission Service will help to improve their university services. This system will able to process, organize, manage and store all their university related process. A well-designed university admission service system will lessen staff workload, enable cost savings, improve data security, and assist students and instructors save time. All of these will eventually boost the productivity and cost-effectiveness of your institution.

Table of Content:

Chapter 1 – Introduction 1
1.1 The system developed 1
1.2 Feasibility analysis 1
1.3 Contribution 2
Chapter 2 – Initial Study 2
2.1 Project Proposal 2
2.2 Background of the IUMS Admission Service System 3
2.3 Problem Area 3
2.4 Possible solution 3
2.5 Objective 4
Chapter 3 – Literature Review 4
3.1 Discussion on problem domain based on published articles 5
3.2 Discussion on problem solutions based on published articles 5
3.3 Comparison of three/four leading solutions 6
3.1.1 Best features 6
3.3.2 Limitations 7
Chapter 4 – Methodology 7
4.1 What to use 7
4.2 Why to use 7
4.3 Sections of methodology 11
4.4 Implementations plans 12
Chapter 5 – Planning 12
5.1 Project Plan 12
5.1.1 Management Plan / Work Breakdown Structure (WBS): 13
5.1.2 Resource Allocation (H/W, S/W. Models, Documentation): 13
5.1.3 Time Duration / Time Boxing: 14
5.1.4 Gantt chart: 15
Chapter 6 – Feasibility 16
6.1 All Possible Types of Feasibility 16
6.1.1 Operational feasibility 16
6.1.2 Technical feasibility 16
6.2 Cost Benefit Analysis 17
6.3 DSDM – good or not for this project -PAQ 17
Chapter 7 – Foundation 18

7.1 The Problem Area Identification 18
7.1.1 Interview 18
7.1.2 Observation 18
7.1.3 Interview 18
7.2 Rich Picture 19
7.3 Specific problem area identification and description 20
7.4 Possible Solution 20
7.5 Overall Requirement List 20
7.5.1 Functional Requirements 20
7.5.2 Functional Requirements 21
7.6 Which Technology to be implemented 21
7.7 Recommendation and Justifications: 22
Chapter 8 – Feasibility 23
8.1 Old System Use Case: 23
8.2 Activity Diagram: 24
8.3 Full System Use Case Diagram: 25
8.4 Full System Activity Diagram: 26
8.5 Requirements Catalogue: 27
8.6 Requirement Prioritized: 28
Chapter 9 – Feasibility 29
9.1 Modules of the new system: 29
9.2 Use Case of IUMS Admission Service System: 30
9.3 Class Diagram of IUMS Admission Service System: 31
9.4 ERD Diagram of IUMS Admission Service System: 31
9.5 Sequence Diagram of IUMS Admission Service System: 32
9.6 Component Diagram of IUMS Admission Service System: 33
9.7 Deployment Diagram of IUMS Admission Service System: 33
Chapter 10 – Deployment / Development 34
10.1 Core Module Coding Samples: 34
10.2 Possible problem breakdown: 37
10.3 Prioritization while developing: 38
Chapter 11 – Testing 39
11.1 Old System Use Case: 39
11.2 Test Case: 41
11.3 Unit Testing: 42

11.4 Module Testing: 43
11.5 Integration Testing: 43
11.6 Acceptance Testing: 44
11.7 Security Testing: 44
Chapter 12 – Implementation 46
12.1 Training 46
12.2 Big Bang 47
12.3 Scaling 47
12.4 Load Balancing 47
Chapter 13 – Critical Appraisal and Evaluation 48
13.1 Objective that could be met 48
13.1.1 Success rate against each objective 48
13.1.2 How much better could have been done 48
13.1.3 Why it could not be done 48
13.1.4 Which objectives have been missed 48
13.1.5 Why these objectives have missed 48
13.2 Objectives totally not met / touched 49
13.2.1 Why it could not be touched 49
13.2.2 What could have been done 49
Chapter 14 – Lessons Learned 49
14.1 Pre project – review – closing 49
14.2 What have I learned 49
14.3 What problem I have faced 50
14.4 What solutions occurred 50
Chapter 15 – Conclusion 51
15.1 Summary of the project 51
15.2 Goal of the project 51
15.3 Success of the project 51
15.4 What I have done in the documentation (stages / activities / plans) 51
15.5 Value of the project 52
15.6 My experience 52
Appendix: 53
Reference 54
Plagiarism report 55

List of figure:

Figure 1: Waterfall Methodology 8
Figure 2: V Methodology 9
Figure 3: DSDM Methodology 10
Figure 4: Gantt chart 15
Figure 5: Rich picture 19
Figure 6: Simple Data Flow 19
Figure 7: Old System Use Case 23
Figure 8: Activity Diagram 24
Figure 9: Full System Use Case Diagram 25
Figure 10: Full System Activity Diagram 26
Figure 11: Use Case 30
Figure 12: Class Diagram 31
Figure 13: ERD Diagram 31
Figure 14: Sequence Diagram 32
Figure 15: Component Diagram 33
Figure 16: Deployment Diagram 33
Figure 17: Admission Center Service 34
Figure 18: Admit Card Signature Service 35
Figure 19: Applicant Assigned to center Service 36
Figure 20: Payment 37
Figure 21: IUMS Admission Service 39

List of Tables:

Table 1: Breakdown Structure of DSMS 13
Table 2: Resource Allocation List 14
Table 3: List of the Time Boxes 15
Table 4: Technical Feasibility 16
Table 5: Cost Benefits 17
Table 6: Requirements Catalogue 27
Table 7: Requirement Prioritized 28
Table 8: Prioritization Requirement 38
Table 9: Test Case 42
Table 10: Unit Test Case 1 42
Table 11: Unit Test Case 2 42
Table 12: Module Test Case 1 43
Table 13: Module Test Case 2 43
Table 14: Integration Test Case 1 43
Table 15: Integration Test Case 2 44
Table 16: Acceptance Test 44
Table 17: Security Test 1 44
Table 18: Security Test 2 45
Table 19: Final Test Report 46
Table 20: Training Based 47

Chapter 1 – Introduction

1.1 The system developed

In this project I have developed IUMS Admission Service. This system is designed to streamline and enhance the entire admission process, from application submission to the final enrollment, incorporating various essential components to ensure efficiency, transparency, and user satisfaction. Admission Service Center System aims to optimize the admission process, providing administrators with powerful tools to manage applications, communicate effectively, and ensure a seamless experience for both applicants and staff. By integrating various modules, the system is poised to enhance transparency, reduce manual effort, and facilitate informed decision-making throughout the admission lifecycle.

1.2 Feasibility analysis

Conducting a feasibility analysis is a crucial step in determining whether the implementation of the Admission Service Center System is viable and practical for the educational institution.

Technical Feasibility:

- **Hardware and Software Requirements:** Assess whether the institution has the necessary hardware and software infrastructure to support the new system.
- **Technology Compatibility:** Ensure that the new system is compatible with existing systems and technologies used by the institution.

Operational Feasibility:

- **User Training:** Evaluate the ease of use and complexity of the system, considering the training required for administrators, staff, and applicants.
- **Integration with Existing Processes:** Examine how well the new system can integrate with current admission processes and other university systems.

Economic Feasibility:

- **Cost-Benefit Analysis:** Conduct a detailed cost-benefit analysis to determine the financial viability of implementing the new system.
- **Return on Investment (ROI):** Assess the expected returns and benefits over time, comparing them to the initial investment in the system.

Environmental Feasibility:

- **Sustainability:** Consider the environmental impact of the new system, such as energy consumption and resource usage.
- **Scalability:** Evaluate whether the system can accommodate future growth and changes in admission processes.

1.3 Contribution

The contribution of the developed Admission Service Center system is multi-faceted, impacting various stakeholders within the educational institution.

- **Operational Efficiency:** The system significantly enhances operational efficiency by automating manual processes, reducing paperwork, and streamlining the entire admission lifecycle. This leads to faster application processing and decision-making.
- **User-Centric Access and Experience:** The Applicant Privilege module ensures a user-centric approach, allowing applicants tailored access to information relevant to their application status and requirements. This contributes to a positive and personalized experience for applicants.
- **Financial Management and Transparency:** The system's financial management components, including Fees Head, Fees Structure, Fees Structure Type, and Payment modules, contribute to transparent and accurate financial transactions related to admission, promoting financial accountability.
- **Security and Compliance:** Robust security measures and compliance with legal and regulatory requirements contribute to the protection of sensitive data, ensuring the privacy and integrity of applicant information.

Chapter 2 – Initial Study

2.1 Project Proposal

Admission Service System is designed to revolutionize the way we handle admissions, providing a centralized and efficient platform for both applicants and administrators. This system will streamline the application process, enhance communication, and provide powerful tools for managing admissions seamlessly. The proposed Admission Service System is crucial for ushering in a new era of efficiency and transparency in our admission processes.

2.2 Background of the IUMS Admission Service System

This is a reputable educational institution committed to providing high-quality education to aspiring students. With a diverse range of academic programs and a growing number of applicants each year, the admission process has become increasingly complex. The manual and decentralized nature of the existing admission procedures led to inefficiencies, delays, and challenges in maintaining transparency. Our system has two roles:

- **Admin:**
  Responsible for system administration, configuration, user management, and overall control of the Admission Service Center system.
- **Teacher:**
  Involved in application review, conducting interviews, and managing admission-related processes.

2.3 Problem Area

An admission service module in a university could revolve around addressing challenges and inefficiencies in the existing admission process. The existing admission process at this educational institution is marred by several inefficiencies and challenges, stemming from manual workflows, disjointed systems, and a lack of modern technology integration. To address these issues, we have developed the Admission Service System. However, the problems it seeks to solve are rooted in the current state of our admission processes.

2.4 Possible solution

Transformation of the Admission Service Center through an Integrated and Automated Module. Automated and Integrated Admission Processes, Centralized Communication Hub, Enhanced Admit Card Security, Comprehensive Applicant Privilege System, Improved Room and Building Allocation, Integrated Email Communication and Continuous Monitoring and Updates. By addressing these aspects, the expected solution aims to transform the Admission Service Center System into a modern, cohesive, and user-centric platform that enhances the overall admission experience for both applicants and administrative staff.

2.5 Objective

The objectives of the Admission Service Center system are likely to be aligned with the goals and improvements desired in the admission process of an educational institution.

- Automation of Admission Processes.
- Enhanced Applicant Experience.
- Efficient Administrative Tools.
- Financial Tracking and Transparency.
- Effective Communication.
- Role-Based Access Control.
- Optimized Room Allocation.
- Exception Handling.
- Comprehensive Reporting and Analytics.
- Security and Data Privacy.
- Scalability.
- User Training and Support.
- Customization.
- Integration with Other System.

Chapter 3 – Literature Review

In Bangladesh, the landscape of higher education institutes is diverse, yet the utilization of modern software solutions for admission services remains limited. This literature review aims to explore the current state of admission service coordination in educational institutes in Bangladesh, emphasizing the prevalence of traditional management systems and the need for the adoption of software solutions.

Historically, educational institutes in Bangladesh have relied on manual and paper-based admission management systems. Studies highlight the challenges posed by these traditional systems, including inefficiencies, data inaccuracies, and prolonged processing times.

Despite the global trend toward digital transformation in education, a significant portion of higher education institutes in Bangladesh continues to lag in adopting software solutions for admission services. Research reveals that only a small fraction of institutions have embraced modern technologies, leaving the majority to grapple with outdated processes.

This literature review emphasizes the urgent need for higher education institutes in Bangladesh to transition from traditional, paper-based admission management systems to modern, software-driven solutions. The findings underscore the potential benefits in terms of efficiency, accuracy, and overall improvement in the quality of admission services.

3.1 Discussion on problem domain based on published articles

Complexity and Usability Challenges: Published articles consistently emphasize the inherent complexity and poor usability of existing admission service systems. The interfaces are often convoluted and non-intuitive, leading to difficulties for both administrators and applicants. Addressing the usability challenges requires a user-centric design approach. Insights from research recommend redesigning interfaces to simplify navigation, reduce cognitive load, and improve the overall user experience.

Document Management and Data Handling Issues: Several articles highlight the inadequacies in document tracking, slow retrieval mechanisms, and data redundancy within admission service systems. These issues result in delays, inaccuracies, and inconsistencies in the admission process. Research underscores the critical role of efficient document management in the admission process. Inefficient systems hinder quick access to relevant documents, leading to bottlenecks in decision-making and potential errors.

The insights derived from published articles collectively underscore the multifaceted challenges within the problem domain of inefficient admission service systems. Usability issues, document management challenges, training deficiencies, and integration hurdles all contribute to a suboptimal admission process. Implementing solutions aligned with the recommendations from research can pave the way for more streamlined, efficient, and user-friendly admission service systems in educational institutions.

3.2 Discussion on problem solutions based on published articles

Usability Improvement through User-Centric Design: Research consistently emphasizes the need for a user-centric design approach to improve the usability of admission service systems. Simplifying interfaces, enhancing navigation, and incorporating user feedback are critical elements of this approach.

Solutions:

- Conduct usability testing to identify pain points and areas of improvement.
- Iteratively redesign interfaces based on user feedback.
- Implement intuitive navigation and provide contextual help features.
- Prioritize accessibility to cater to diverse user needs.

Advanced Document Management Systems: Efficient document management is crucial for the smooth operation of admission service systems. Published articles recommend the implementation of advanced document management systems with features tailored for the admission process.

Solutions:

- Integrate document tracking mechanisms for real-time monitoring.
- Implement quick retrieval mechanisms for efficient access to documents.
- Ensure data accuracy and consistency through version control.
- Explore cloud-based solutions to facilitate seamless document sharing.

3.3 Comparison of three/four leading solutions

Usability:

- Solution A: User-friendly interface, intuitive navigation.
- Solution B: Streamlined user experience, clear workflows.
- Solution C: Intuitive design, accessible features.

Document Management:

- Solution A: Advanced document tracking, quick retrieval mechanisms.
- Solution B: Robust document management, version control.
- Solution C: Efficient document sharing, cloud-based solutions.

Training and Support:

- Solution A: Comprehensive training programs, user manuals.
- Solution B: Hands-on training, continuous learning initiatives.
- Solution C: Ongoing user support, refresher courses.

3.1.1 Best features

- User-Friendly Interface
- Application Management
- Reporting and Analytics
- Customization Options
- Automation of Processes
- Security Measures
- Financial Management

3.3.2 Limitations

- Technical Challenges
- Data Security and Privacy
- Dependency on Internet Connectivity
- Maintenance Challenges
- Incomplete Data

Chapter 4 – Methodology

4.1 What to use

The approach taken to accomplish the project's objective will be described in this chapter. A framework for creating, developing, and sustaining software is the Software Development Life Cycle (SDLC). It provides an overview of the steps, stages, objectives, and advancement of the software development process. The software development process makes use of numerous models. The agile, waterfall, V-, spiral, and dynamic systems development model (DSDM) software development models are all well-known to software engineers.

4.2 Why to use

DSDM is more focused than most other agile approaches since it addresses projects rather than just the creation and delivery of goods, which are usually software. The project context must take into account the overall business demand as well as every element of the solution that emerges to address that need. With a long history of implementing Agile projects successfully in a variety of business contexts, DSDM has proven to be entirely scalable, operating effectively in large, complicated companies, small, simple businesses, and highly regulated environments. Additionally, it has been shown to work just as well for non-IT projects like business change initiatives as for IT ones. I'll outline a few situations in which this project will benefit from this procedure. I'm now going to discuss a few models that have benefits and drawbacks.

Waterfall Model:

One of the software development models, the waterfall model, has a step-by-step or linear process. A waterfall model needs to have well-defined exit conditions, such getting approval from every project member, and each stage needs to be finished in the correct order. A set of tasks, accompanying documentation, and exit requirements for every step of each phase are all part of the waterfall paradigm.

When utilizing this model:

- Requirements are not frequently changing.
- The application is not intricate or drawn out.
- The undertaking is short.
- The prerequisite is unchanging.
- The surroundings remain stable.

Waterfall Model Diagram:

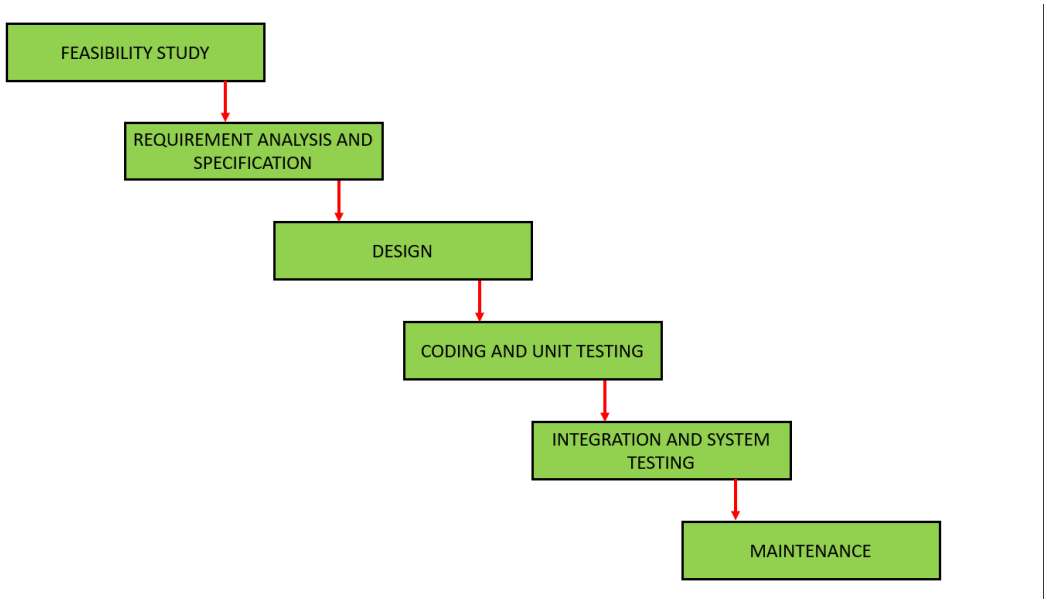


Figure 1: Waterfall Methodology

The waterfall model has the following advantages:

- It is ideal for smaller projects with well-defined objectives
- Each step of development must be completed before going on to the next.
- They should do quality assurance testing (verification and validation) prior to concluding each step.
- Comprehensive documentation is required at every phase of the software development cycle.
- The project team is mostly responsible for its success; client engagement is low.
- Any changes made to software are done so during development.

Drawback of the waterfall model is:

- Errors may only be fixed within the phase.

- It is unacceptable for complex projects where needs are subject to frequent changes.
- The development process's testing stage comes at a comparatively late point.
- It is impossible to incorporate customer feedback into the continuous development process; instead, the majority of the time spent by developers and testers is on documentation.
- Minor adjustments or errors in the final output could cause a lot of problems.

V model:

The software development life cycle's (SDLC) verification and validation are represented by the V model. A V-shaped model's life cycle is carried out in sequential order. Before moving on to the next phase in the V model for software testing, each step must be completed. It is a methodical approach to software development and problem solving compared to the waterfall technique.

When utilizing this model:

- Efficient when the project's needs are definite and consistent.
- Advantageous when documentation is given first priority.
- Preventing early-stage problems.

V Model Diagram:

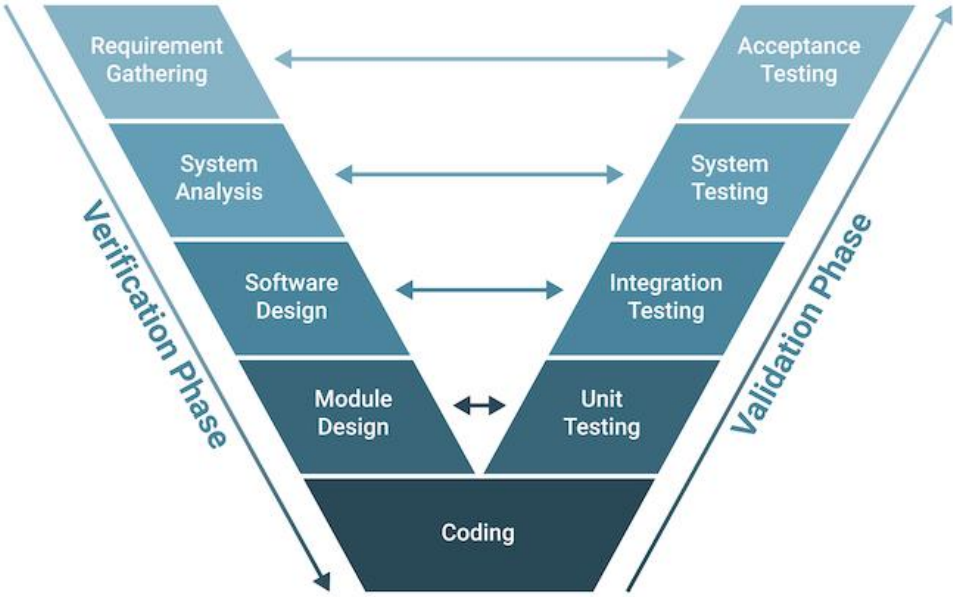


Figure 2: V Methodology

Benefit of the V model:

- It is easy to operate.
- Considerable changes have been made to identify problems early.
- In this level, test tasks like as planning and designing tests are completed before coding.
- It was suitable for small tasks with straightforward requirements.
- It is a very methodical way of developing software.

Drawback of the waterfall model is:

- There is a high and unpredictable danger.
- This method doesn't work well for complicated tasks.
- It is difficult to go back to the first step after completing the test phase.
- This paradigm is inappropriate for long-term initiatives.

Dynamic systems development model (DSDM):

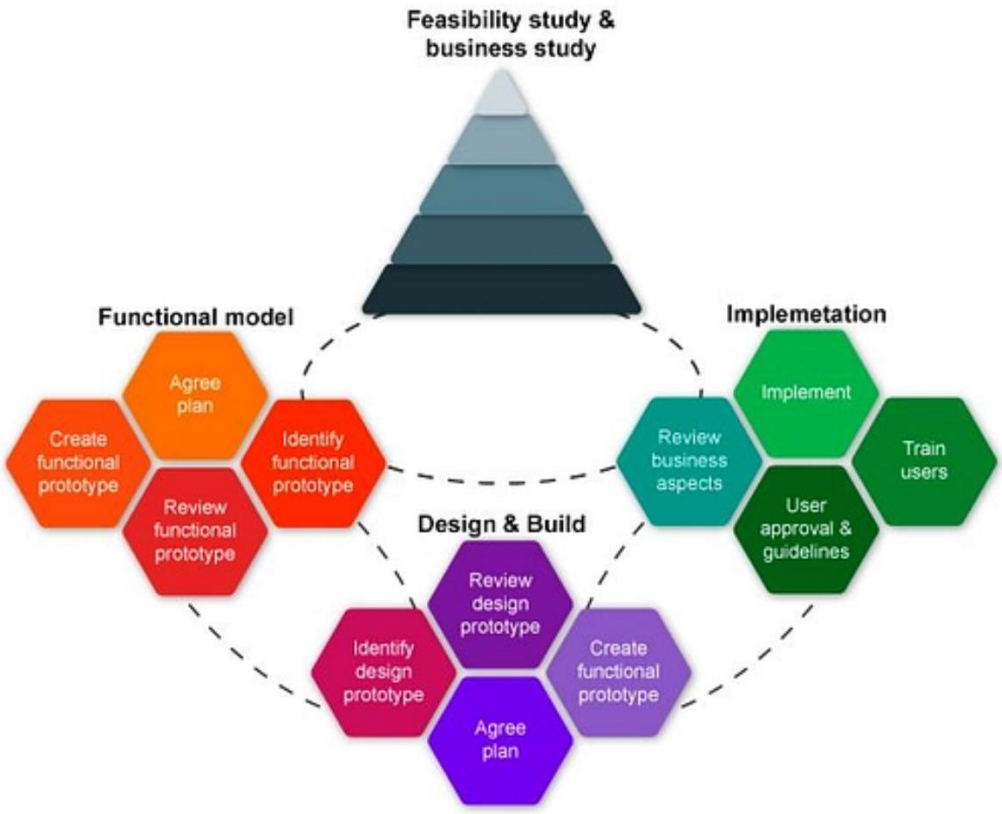


Figure 3: DSDM Methodology

Benefit of the DSDM model:

- Projects are finished on time while remaining flexible, and everyone in the organization can see the results right away.
- Business cases, the fundamental building block of the DSDM paradigm, guarantee that the initiatives offered have true economic value.

Drawback of the DSDM model is:

- Generally speaking, managerial costs can be high.
- Small businesses may find installation costs to be prohibitively high.
- Why DSDM tends to stifle developer innovation, making it unsuitable for small firms due to high management overhead and costly deployment.
- Projects are likely to be completed precisely as asked, even in the case that more sophisticated techniques are available.

Utilize the IUMS Admission Service technique:

I've decided to employ the DSDM approach for the IUMS Admission Service due to the above mentioned reasons. The DSDM approach will be the best option for the IUMS Admission Service since it regularly generates actual outputs and delivers goods on time. The ease of customization of the DSDM approach makes it suitable for implementation by any type of business or organization. As a project progresses, the client or customer may alter their requirements if they believe it is necessary.

4.3 Sections of methodology

Pre-Project Phase: During the pre-project phase, a number of ideas are evaluated, and one is ultimately selected. At this point, both the project's viability and funding requirements are estimated.

Project lifecycle Phase: The project lifecycle for developing an admission service system typically involves several phases, each with its own set of activities, goals, and deliverables. Here's an outline of the project lifecycle phases:

- **Initiation Phase:**
  - Define the purpose and scope of the admission service system.
  - Identify key stakeholders and establish project goals.
- **Planning Phase:**
  - Develop a comprehensive plan for project execution.
  - Establish timelines, resources, and project management processes.

- **Requirements Gathering and Analysis Phase:**
  - Understand and document the specific requirements of the admission service system.
  - Establish a clear understanding of user needs and expectations.
- **Design Phase:**
  - Create the architectural and structural design of the admission service system.
  - Plan the overall system structure based on requirements.
- **Implementation Phase:**
  - Build the admission service system based on the approved design.
  - Write code and develop functionalities according to specifications.
- **Testing Phase:**
  - Validate that the admission service system meets requirements and functions without errors.
  - Identify and address any defects or issues.

4.4 Implementations plans

At this point in the project, the finished application is made available to the general public for use. The new system must to be put into service as soon as a flaw is discovered and fixed. In this part, the methods, parameters, and criteria for release are selected. The new system is then put to the test and put into use if everything goes as planned.

Chapter 5 – Planning

5.1 Project Plan

A collection of official documents known as a project plan describes the stages of project implementation and monitoring. The plan addresses resource management, risk management, and communication in addition to the project's overall scope, cost, and schedule. In order to ensure that risks and problems are minimized and the project moves forward smoothly, we must finish the project planning process before beginning any work on the actual execution of the project. Every aspect of your project is discussed in detail in these project management papers. Project planning will be beneficial to all parties involved, including the project manager and his team. The achievement of essential activities including goal-setting, risk minimization, deadline avoidance, and delivery of the intended product, technical support, or solution will be made possible by planning. In general, project planning—which includes a Gantt chart, time boxing, and WBS—will be covered in this part. Here are some instances of these:

5.1.1 Management Plan / Work Breakdown Structure (WBS):

A project's tasks are arranged hierarchically using the Work Breakdown Structure (WBS). The work breakdown structure (WBS) of a project is "broken down" into smaller, more manageable components. At each delivery, tasks may also be further divided into subtasks, contingent upon the needs of the project. A part of project lifecycle management called work breakdown structure (WBS) divides large, complex projects into smaller, more manageable parts that can be delegated to specific individuals or groups.

No.	Task Name	Duration	Start	End
1	Introduction	5 Days	7/10/2023	7/14/2023
2	Initial Study	4 Days	7/15/2023	7/18/2023
3	Literature Review	6 Days	7/19/2023	7/24/2023
4	Methodology	5 Days	7/25/2023	7/29/2023
5	Planning	7 Days	7/30/2023	8/5/2023
6	Feasibility	5 Days	8/6/2023	8/10/2023
7	Foundation	12 Days	8/11/2023	8/22/2023
8	Exploration	22 Days	8/23/2023	9/13/2023
9	Engineering	13 Days	9/14/2023	9/26/2023
10	Deployment	14 Days	9/27/2023	10/10/2023
11	Testing	7 Days	10/11/2023	10/17/2023
12	Implementation	4 Days	10/18/2023	10/21/2023
13	Critical Appraisal and Evaluation	5 Days	10/22/2023	10/25/2023
14	Lesson Learned	5 Days	10/26/2023	10/30/2023
15	Conclusion	1 Day	11/1/2023	11/1/2023
	Total	115 Days		

Table 1: Breakdown Structure of DSMS

5.1.2 Resource Allocation (H/W, S/W. Models, Documentation):

When resource allocation—the process of allocating the best available resources to projects and activities—is used. It monitors workloads to avoid misuse or underuse. Then, staff members might need to be reassigned based on the project's deadlines and the availability of resources at the moment. The organization's resources will receive assistance in realizing their full potential through this initiative. This is because, when done well, it could result in happier clients and staff. In order to fulfill the prearranged project delivery timeline, the IUMS Admission Service project must allocate its resources as follows:

No.	Task Name	Duration	Resource
1	Introduction	5 Days	Analyst, User
2	Initial Study	4 Days	Analyst
3	Literature Review	6 Days	Analyst, Team Leader
4	Methodology	5 Days	Analyst, Developer
5	Planning	7 Days	Analyst, Developer, Team Leader, Designer
6	Feasibility	5 Days	Analyst, Developer
7	Foundation	12 Days	Analyst, Designer, Developer
8	Exploration	22 Days	Analyst, Designer, Developer
9	Engineering	13 Days	Designer, Developer
10	Deployment	14 Days	Analyst, Developer
11	Testing	7 Days	Developer, Designer, Analyst
12	Implementation	4 Days	Developer, Analyst, User
13	Critical Appraisal and Evaluation	5 Days	Analyst, Developer,
14	Lesson Learned	5 Days	Analyst, User**,** Developer,
15	Conclusion	1 Day	Analyst

Table 2: Resource Allocation List

5.1.3 Time Duration / Time Boxing:

When a clearly defined deliverable needs to be finished in the allotted time and resources, the term "Time Box" is employed. Deliverables have a set time constraint that cannot be altered. One way that a Time Box differs from standard progress control is that it incorporates project management variables such as the deliverable's scope. However, quality is always the same. The scope, quality, and anticipated completion date of the project must be continuously assessed by the project manager. The deadline cannot be fulfilled if it is not possible to further reduce the delivery's scope or quality.

Time Box	Task Name	Duration	Resource
TB1	Introduction	5 Days	Analyst, User
	Initial Study	4 Days	Analyst
	Literature Review	6 Days	Analyst, Team Leader
TB2	Methodology	5 Days	Analyst, Developer
	Planning	7 Days	Analyst, Developer, Team Leader, Designer
	Feasibility	5 Days	Analyst, Developer
TB3	Foundation	12 Days	Analyst, Designer, Developer
TB4	Exploration	22 Days	Analyst, Designer, Developer
	Engineering	13 Days	Designer, Developer
TB5	Deployment	14 Days	Analyst, Developer
	Testing	7 Days	Developer, Designer, Analyst
TB6	Implementation	4 Days	Developer, Analyst, User
TB7	Critical Appraisal and Evaluation	5 Days	Analyst, Developer,
	Lesson Learned	5 Days	Analyst, User, Developer,
TB8	Conclusion	1 Day	Analyst

Table 3: List of the Time Boxes

5.1.4 Gantt chart:

The intricacy of a difficult project will help the Gantt chart in project management to successfully finish it. The Gantt chart will help with planning and scheduling, which will make it easier to complete all projects of various sizes. As a result of different scheduling techniques, a horizontal bar representing the quantity of work that will be finished at each stage as well as deadlines for each system step and the start and completion dates of each assignment will be displayed.

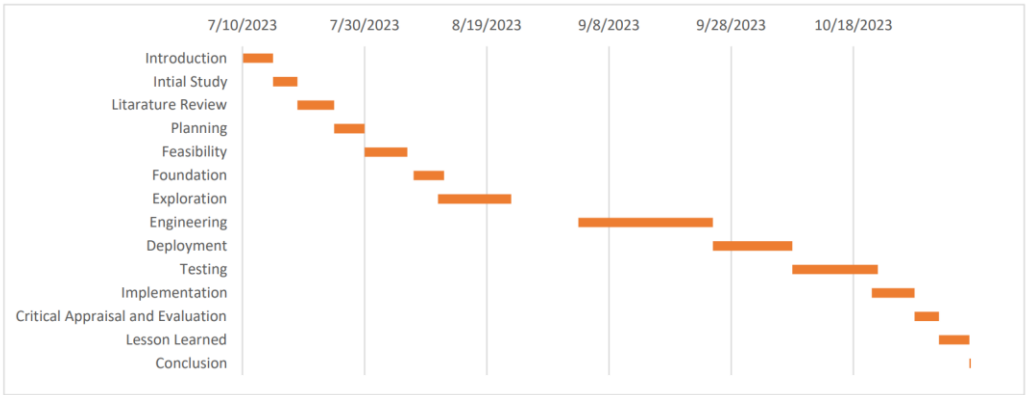


Figure 4: Gantt chart

Chapter 6 – Feasibility

The feasibility analysis offers a thorough grasp of the benefits and potential drawbacks of putting in place an admission service system. It supports stakeholders in making well-informed choices on whether to move forward with the project as is, make changes to the plan, or scrap it entirely. As the project develops and new information becomes available, it could be required to update the feasibility analysis on a regular basis.

6.1 All Possible Types of Feasibility

6.1.1 Operational feasibility

The analysis of operational feasibility offers valuable perspectives on the pragmatic elements involved in integrating the admission service system into the organization. It highlights areas that might need attention both during and after implementation, assisting stakeholders in making well-informed judgments about the organization's readiness to accept the new system. Ongoing evaluations and regular interactions with end users can help ensure that the admission service system is successfully operationally integrated.

6.1.2 Technical feasibility

Hardware	Software	Database
Wi-Fi Router, Laptop	Xampp, MS Word, Google Sheets, Google Chrome Browser, Windows 10	MySQL

Technology:

Server Side
Java

Table 4: Technical Feasibility

6.2 Cost Benefit Analysis

In project management, the idea of cost-benefit analysis is used to weigh the advantages and drawbacks of various project routes, including transactions, activities, investments, and business demands.

Project Name: IUMS Admission Service

Total Cost

Item	Year 1	Year 2	Year 3	Total
Web Based Application	150000	0	0	150000
Domain & Hosting	20000	20000	20000	60000
Software	2500	2500	2500	7500
Internet	3500	3500	3500	10500
Training	7000	0	0	7000
Development	2000	2000	2000	6000
Maintenance	12000	12000	12000	36000
Total	197000	40000	40000	277000

Table 5: Cost Benefits

6.3 DSDM – good or not for this project -PAQ

The vendor-neutral Dynamic System Development Method (DSDM) covers the entire project lifecycle and provides best practices for delivering projects on schedule and under budget. Its scalability has proven to be capable of managing projects of any scale and for any industry. The vendor-independent framework and agile base of DSDM make it a perfect fit for this project.

Chapter 7 – Foundation

7.1 The Problem Area Identification

The majority of software is created in reaction to a specific circumstance. It is necessary to determine the particular scenario in order to develop software that is useful. Without knowing the precise solution, it is impossible to make an accurate solution. Because of this, identifying the problem regions is essential for system development. In this domain, users furnish the software organization with information. Since the user will be using the solution and is aware of the problem for which it will be used, their information will be more helpful in building the solution. There are several approaches to identify the issue. There are two mentioned, and both are used in this solution:

7.1.1 Interview

An interview is a process in which two or more individuals have a direct conversation about a particular problem or possible solutions. It's among the best ways to determine requirements. It may occur face-to-face with staff members, teachers, students, etc.

7.1.2 Observation

Real-world behaviors are observed and rated using observational methods. But a major challenge for the researcher is deciding what to look for and how to describe a particular behavior. Defining the behaviors that draw the researcher's attention is crucial. so that the person observing knows what to look out for and gauge; they can then be totaled to get a score. Numerous types of observations exist, each having pros and cons of their own. These types of observations include participant and non-participant observation, naturalistic and controlled observation, and overt and covert observation.

I'm going to this university to work on this project. After that, I'll try to watch every step of their procedure and figure out what they need.

7.1.3 Interview

An additional tool for gathering data is the questionnaire. Written type interviews are one specific kind of interview. When using this strategy, interviewers need to prepare a list of questions to ask users in order to obtain information.

Note: The process of identifying trouble areas has been completed by the user and DCL analyst. I am familiar with this approach because I did an internship with Daffodil Computers Limited.

7.2 Rich Picture

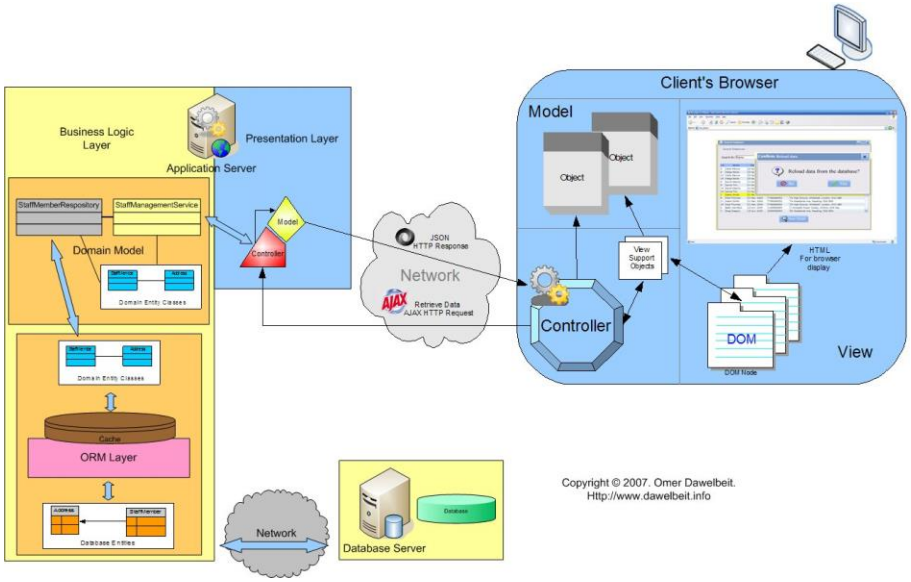


Figure 5: Rich picture

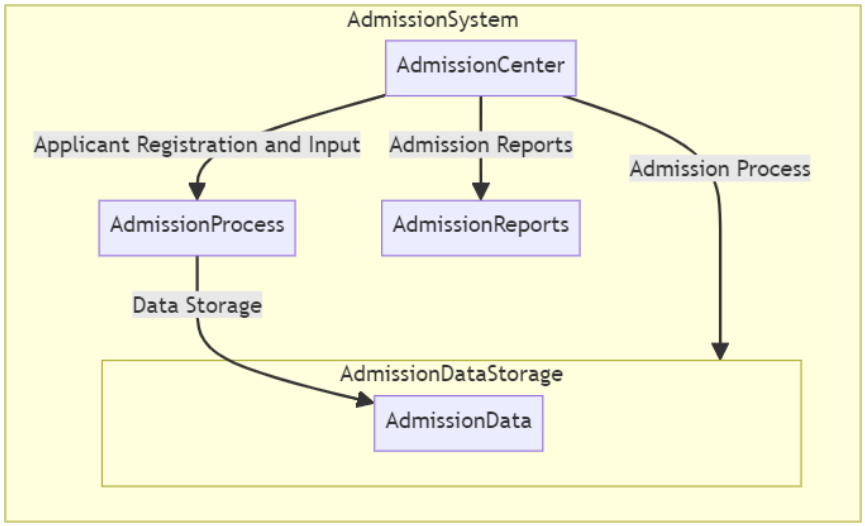


Figure 6: Simple Data Flow

7.3 Specific problem area identification and description

The applicant application processing process in the existing admissions system is inefficient. This inefficiency causes mistakes, delays, and a less than ideal experience for admissions staff as well as applicants. Install an automated admissions service system to expedite the application process, reduce the need for manual intervention, and give admissions personnel and applicants real-time access to application status updates. The objectives of this solution are to increase productivity, shorten processing times, improve communication, and make the application process better for applicants.

7.4 Possible Solution

We must utilize a technique to determine the user requirements for this system. Like observations, questionnaires, and interviews. Conversely, we can split this user need into two parts. Both useful and useless.

- Interview: It can happen face to face with student, teacher, staff etc.
- Observation: In order to gather user needs, we can visit the field and watch every step of their job process.

7.5 Overall Requirement List

Two type of requirements:

- Functional Requirements
- Non-Functional Requirements

7.5.1 Functional Requirements

- The system should allow administrators and staff to register with unique usernames and passwords.
- Users should be able to log in securely to access their respective roles and functionalities within the system.
- The system must provide a user-friendly interface for applicants to submit their applications, including details such as preferred courses, educational background, and personal statements.
- Applicants should be able to upload and submit supporting documents required for the admission process, such as transcripts, recommendation letters, and identification documents.
- Admission staff should have the ability to review and evaluate applications, including access to applicant profiles, submitted documents, and academic records.

- The system should facilitate admission staff in making admission decisions based on predefined criteria, and notify applicants of the decision.
- The system must automatically send notifications and updates to applicants regarding their application status, additional document requests, and admission decisions.
- The system should generate reports and analytics for administrators, providing insights into application trends, admission rates, and other relevant metrics.
- Allow admission staff to schedule and manage applicant interviews through the system.

7.5.2 Functional Requirements

- The system should have an intuitive and user-friendly interface, ensuring ease of use for both applicants and admission staff.
- The system should handle a scalable number of concurrent users and maintain optimal performance during peak admission periods.
- Implement robust security measures, including data encryption, secure authentication, and access controls, to protect sensitive applicant information.
- The system should be highly reliable, with minimal downtime, ensuring continuous availability during critical admission periods.
- Ensure the system's compatibility with existing institutional systems, allowing seamless data exchange and integration.
- The system should be accessible to users with disabilities, following accessibility standards to promote inclusivity.
- Implement performance monitoring tools to track system metrics, identify bottlenecks, and optimize performance continuously.

7.6 Which Technology to be implemented

The specific requirements of the system, the development team's experience, the necessity for scalability, and compatibility with the current infrastructure all play a role in the technology selection for an admission service system.

Java Spring Boot: A framework built on Java that makes it easier to create apps that are suitable for production. It works great for creating reliable and scalable web apps.

- **Convention over Configuration:** Spring Boot reduces the requirement for developers to define specific configurations by adhering to the convention over configuration paradigm. Because of its sensible defaults, developers may concentrate more on business logic.

- **Embedded Web Server:** With the help of Spring Boot's embedded web server—often called Tomcat, Jetty, or Undertow—deploying apps is made simple and eliminates the need for external servers.
- **Auto-Configuration:** Based on the dependencies included in the project, Spring Boot uses auto-configuration to automatically set up application components. Developers will need to spend less time manually configuring systems as a result.
- **Micro services Support:** With built-in capability for generating standalone, independently deployable services, it is ideal for designing micro services architectures.
- **Spring Ecosystem Integration:** Because of Spring Boot's smooth integration with the larger spring ecosystem, developers can use a variety of spring projects for messaging, security, data access, and other purposes.
- **Spring Boot Dev Tools:** A collection of tools called Dev Tools, which include live reloading, automated program restarts, and improved debugging assistance, are available to make development easier.

7.7 Recommendation and Justifications:

Justification:

In order to develop my admission service system, I provided security, usability, comfort, and efficiency a lot of consideration. Because of this, selecting an approach is necessary. A well-liked Java-based framework called Spring Boot makes it easier to create apps that are ready for production. It offers tools like embedded servers, auto-configuration, and a range of modules to help developers create applications that are effective and scalable.

Spring Boot is a good option for Java development because of its large ecosystem and community support. I used MySQL to manage the database and the Java Spring Boot Framework to run the backend. Due to the fact that large libraries don't need to be loaded repeatedly, it becomes extremely safe as well as quick.

Chapter 8 – Feasibility

8.1 Old System Use Case:

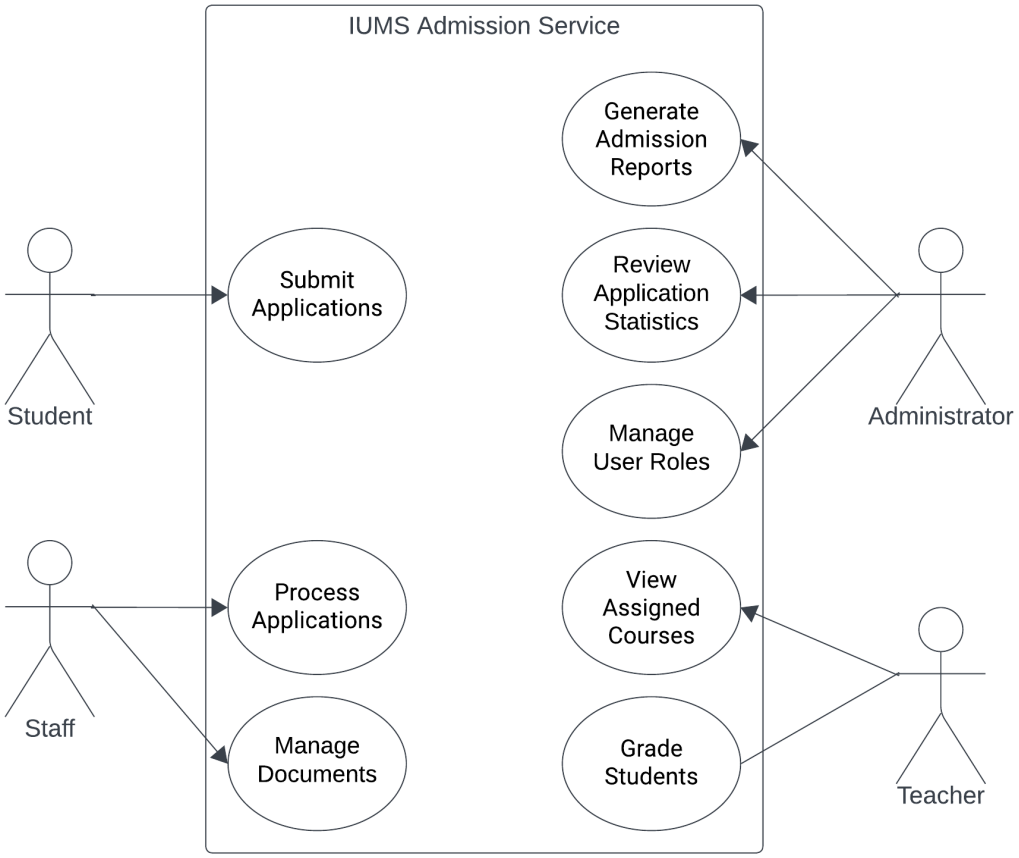


Figure 7: Old System Use Case

In this old IUMS Admission Service have missing many modules. Such as: Check Application Status, Manage Admission Center etc.

8.2 Activity Diagram:

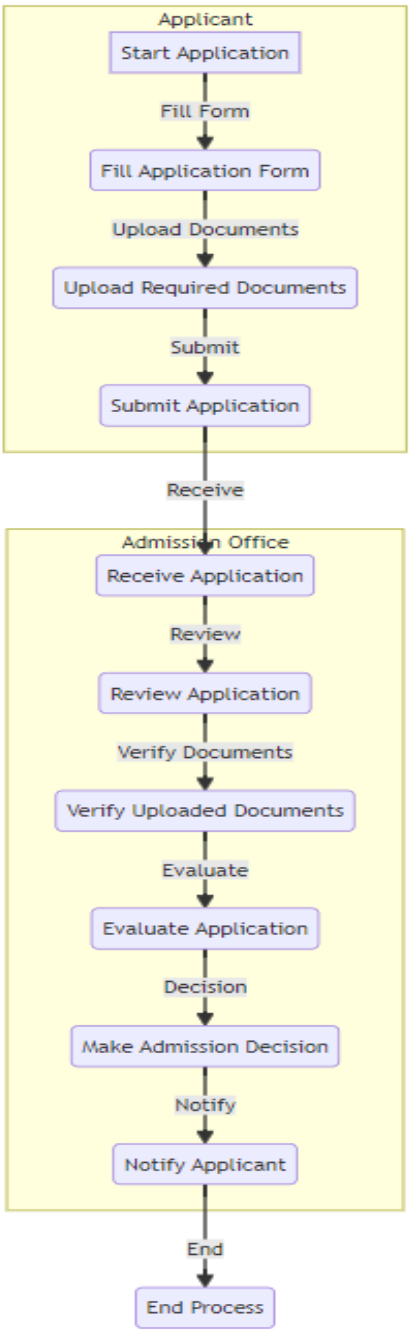


Figure 8: Activity Diagram

8.3 Full System Use Case Diagram:

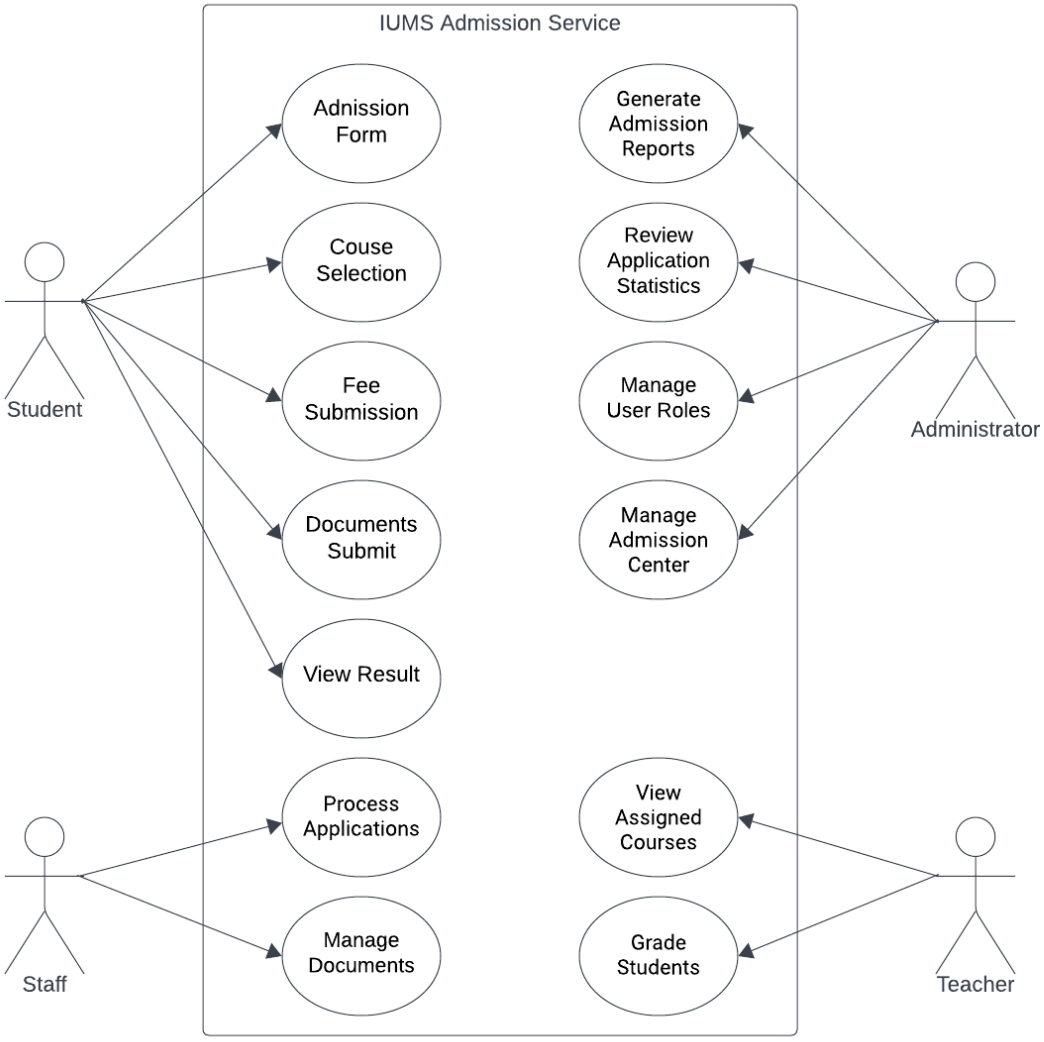


Figure 9: Full System Use Case Diagram

8.4 Full System Activity Diagram:

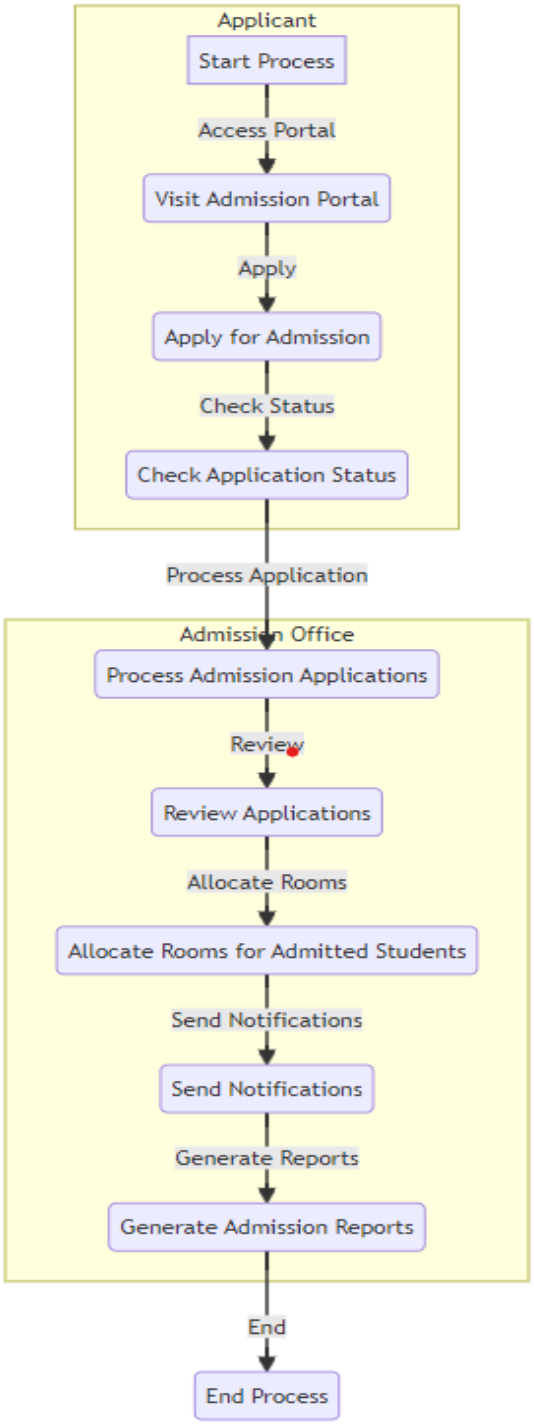


Figure 10: Full System Activity Diagram

8.5 Requirements Catalogue:

Requirement Type	Functional Requirement	
Requirement Name	Admission Circular Management	
Description	Administrators should be able to create, edit, and publish admission circulars.	
Requirement ID	**Source**	**Priority**
F.R.001	Stakeholder interviews	High
Requirement Type	Functional Requirement	
Requirement Name	Room Allocation	
Description	Staff should be able to allocate and manage rooms for admission-related activities.	
Requirement ID	**Source**	**Priority**
F.R.002	Stakeholder interviews	Medium
Requirement Type	Non-Functional Requirement	
Requirement Name	Security	
Description	The system must implement secure user authentication and authorization mechanisms to protect sensitive data.	
Requirement ID	**Source**	**Priority**
N.F.R.001	Security assessments	High

Table 6: Requirements Catalogue

8.6 Requirement Prioritized:

The best way to create a project's priority list is to use the MoSCoW priority technique. I'm going to use this technique to prioritize all of the criteria that have been found for the IUMS Admission Service. Here is a list of each of them:

Requirement category	Requirement Name	Explanation
Must have	Admission Circular Management	Administrators must be able to create, edit, and publish admission circulars.
Should have	Room Allocation	Staff should be able to allocate and manage rooms for admission-related activities.
Could have	Scalability	The system could be scalable to accommodate a growing number of admission applications each year.
Won't have	Performance	The system won't be optimized to handle a specific number of concurrent users at this time.

Table 7: Requirement Prioritized

Chapter 9 – Feasibility

9.1 Modules of the new system:

The IUMS Admission Service has a ton of new modules available. As a result, there will be different modules to clarify. Here, I'll go over some of the core modules:

User Management Module deals with the registration, authentication, and authorization of users, including administrators, teachers, staff, and students.

Admission Circular Module, Enables administrators to create, manage, and publish admission circulars providing information about admission processes and requirements.

Room Allocation Module manages the allocation and utilization of rooms for admission-related activities such as exams, interviews, and counseling.

Audit Trail Module overarching module for managing system configurations, security settings, and overall system health.

9.2 Use Case of IUMS Admission Service System:

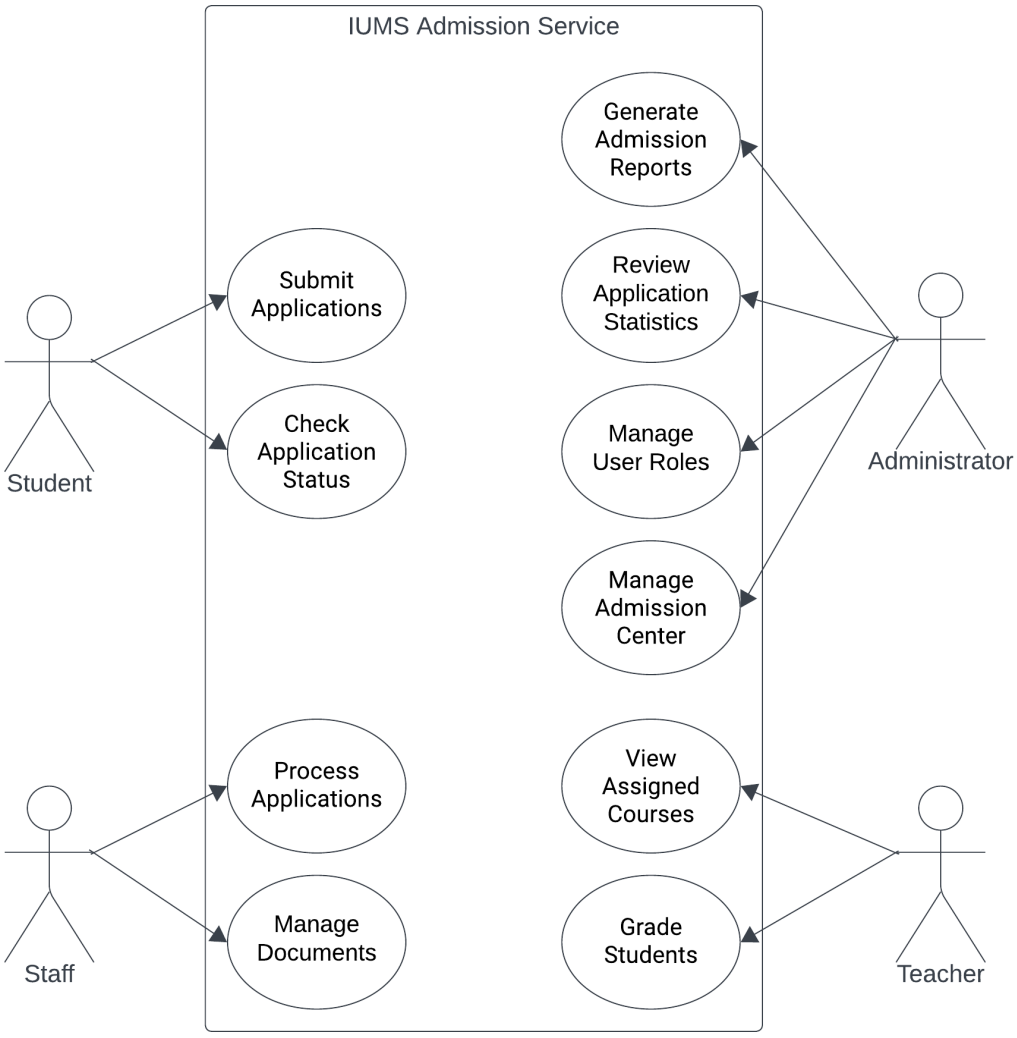


Figure 11: Use Case

9.3 Class Diagram of IUMS Admission Service System:

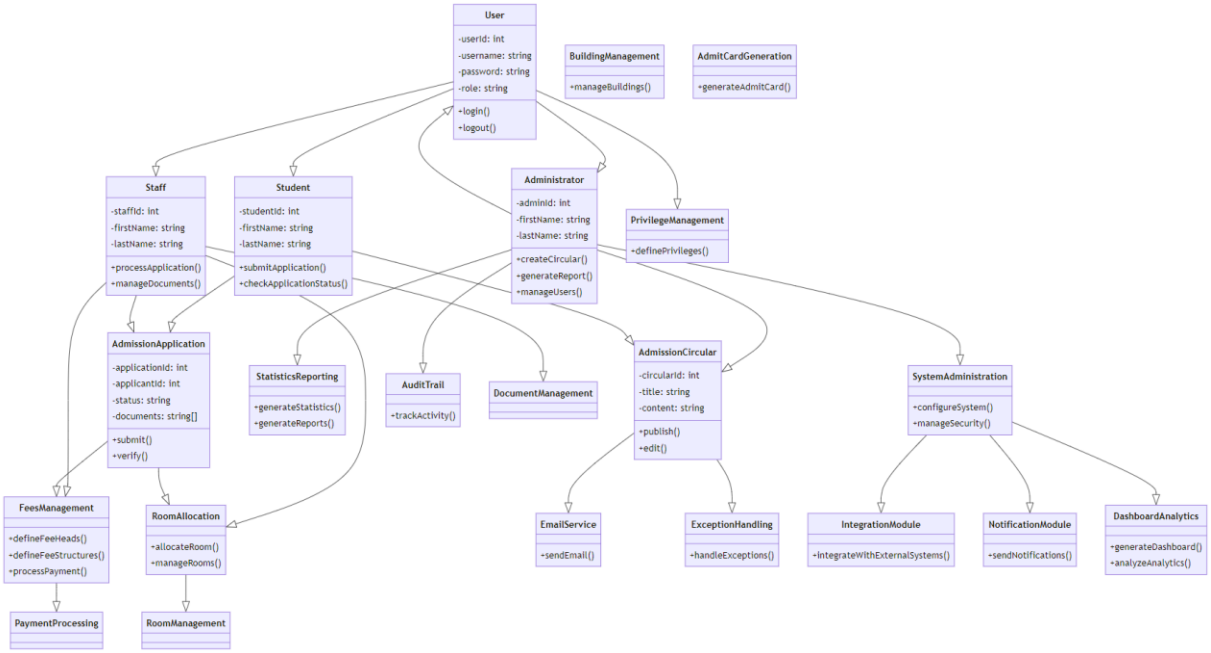


Figure 12: Class Diagram

9.4 ERD Diagram of IUMS Admission Service System:

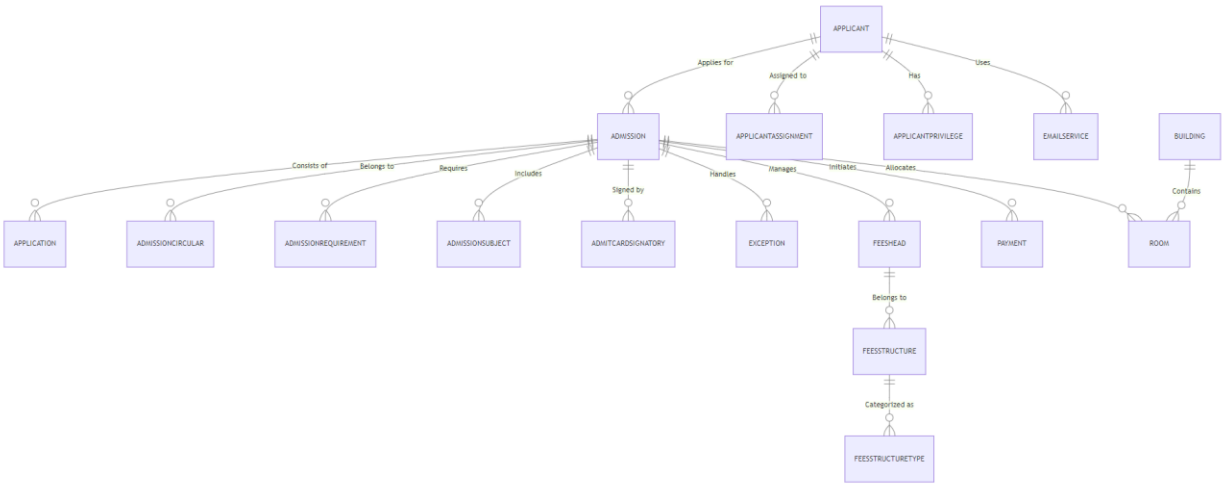


Figure 13: ERD Diagram

9.5 Sequence Diagram of IUMS Admission Service System:

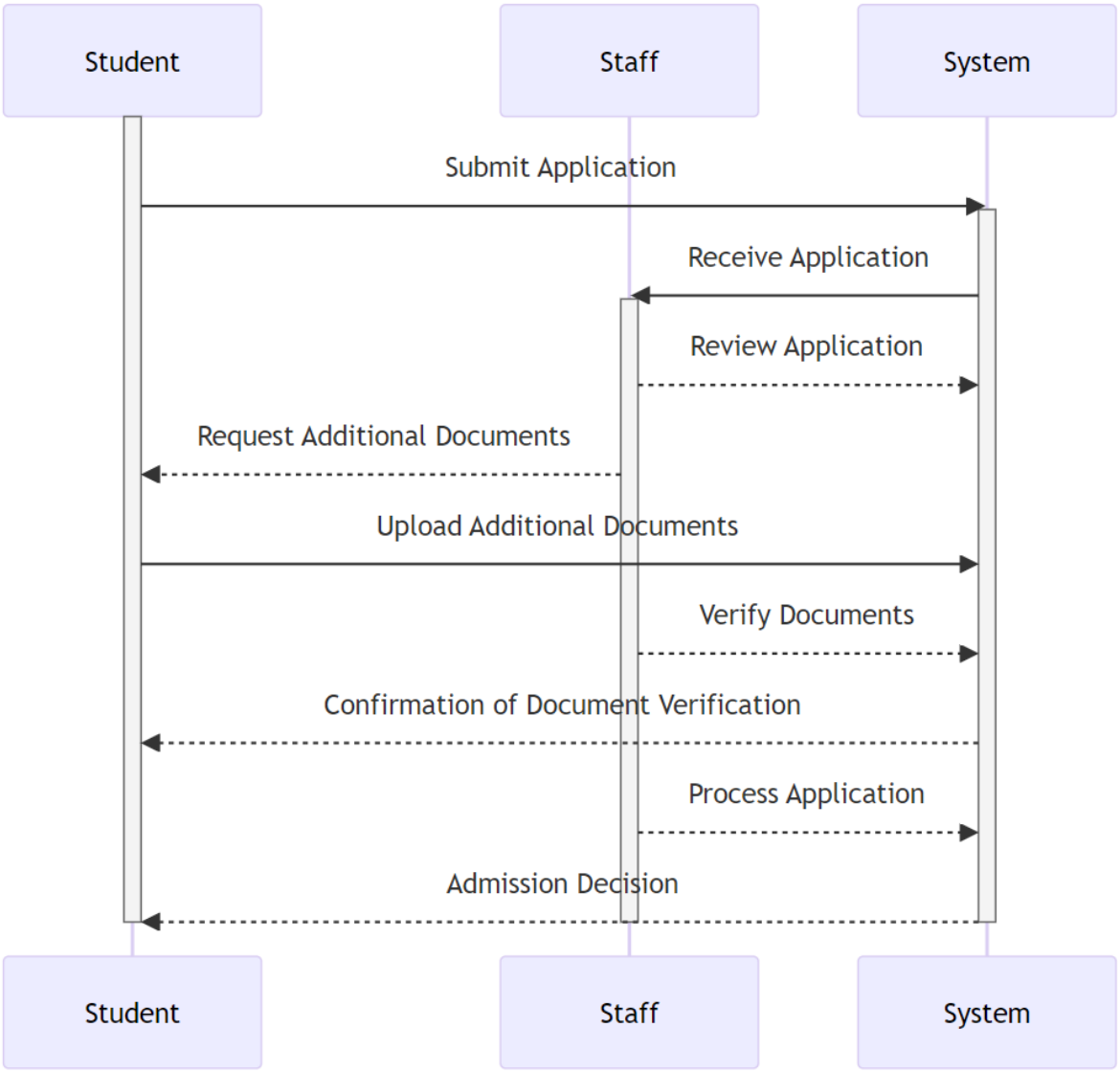


Figure 14: Sequence Diagram

9.6 Component Diagram of IUMS Admission Service System:

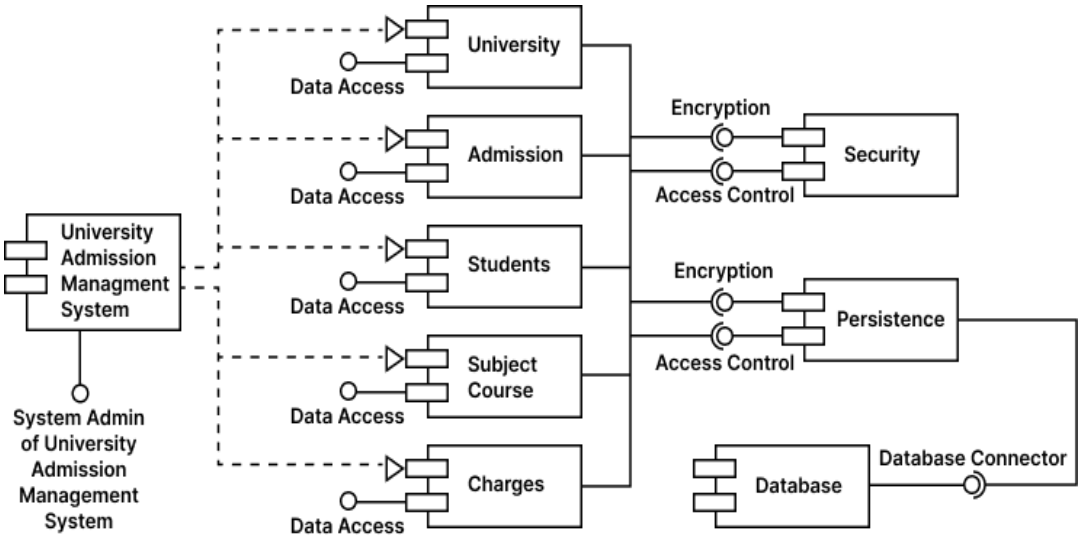


Figure 15: Component Diagram

9.7 Deployment Diagram of IUMS Admission Service System:

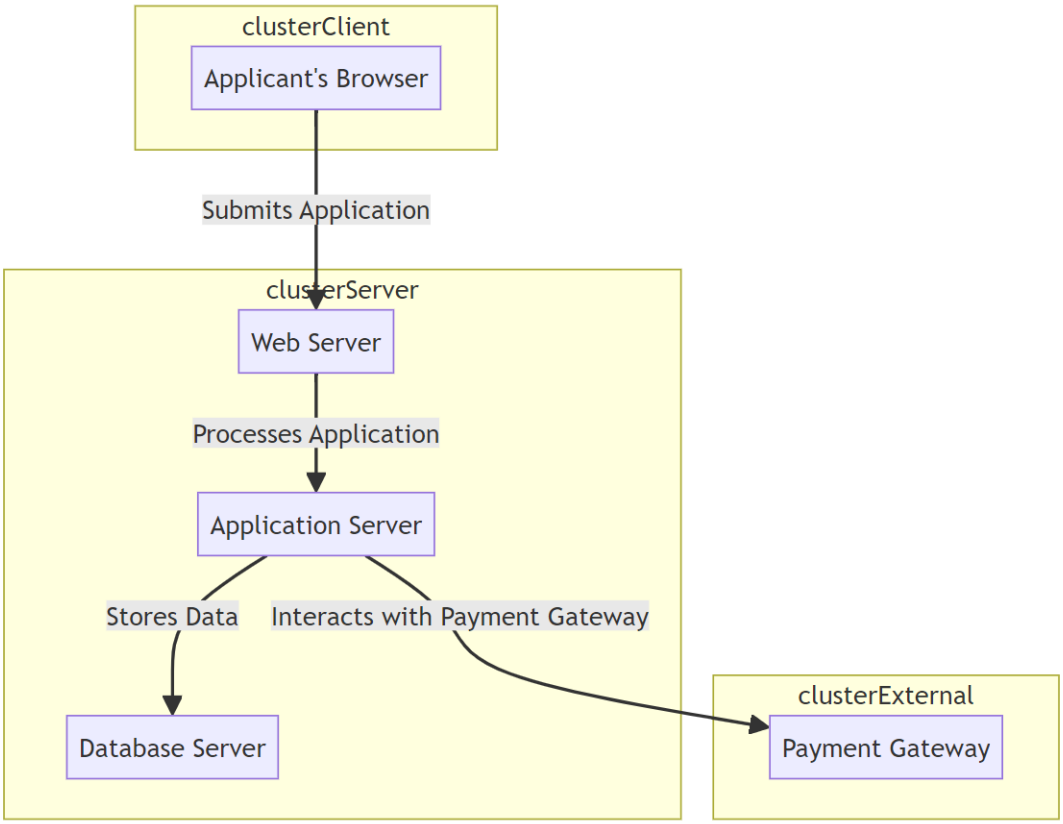


Figure 16: Deployment Diagram

Chapter 10 – Deployment / Development

10.1 Core Module Coding Samples:

Admission Center Service

```java
package com.ums.admissionservice.service.admissioncenter;

import com.ums.admissionservice.domain.model.AdmissionCenter;
import com.ums.admissionservice.domain.repository.AdmissionCenterRepository;
import com.ums.admissionservice.dto.model.AdmissionCenterDTO;
import com.ums.admissionservice.mapper.AdmissionCenterMapper;
import com.ums.admissionservice.service.exception.AdmissionCenterNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

no usages
@Service
public class AdmissionCenterServiceImpl implements AdmissionCenterService {

    7 usages
    @Autowired
    private AdmissionCenterRepository admissionCenterRepository;

    @Override
    public List<AdmissionCenterDTO> index() {
        return AdmissionCenterMapper.map(this.admissionCenterRepository.findAll());
    }

    @Override
    public AdmissionCenterDTO show(Long id) throws AdmissionCenterNotFoundException {
        AdmissionCenter admissionCenter = this.admissionCenterRepository.findById(id)
                .orElseThrow(() -> new AdmissionCenterNotFoundException(String.format("Admission Center with id: %s could not found",id)));
        return AdmissionCenterMapper.map(admissionCenter);
    }

    @Override
    public String store(AdmissionCenterDTO admissionCenterDTO) {
        this.admissionCenterRepository.save(
                new AdmissionCenter()
                        .setName(admissionCenterDTO.getName())
                        .setCapacity(admissionCenterDTO.getCapacity())
                        .setLocation(admissionCenterDTO.getLocation())

        );
        return "Admission Center has been created";

    }
```

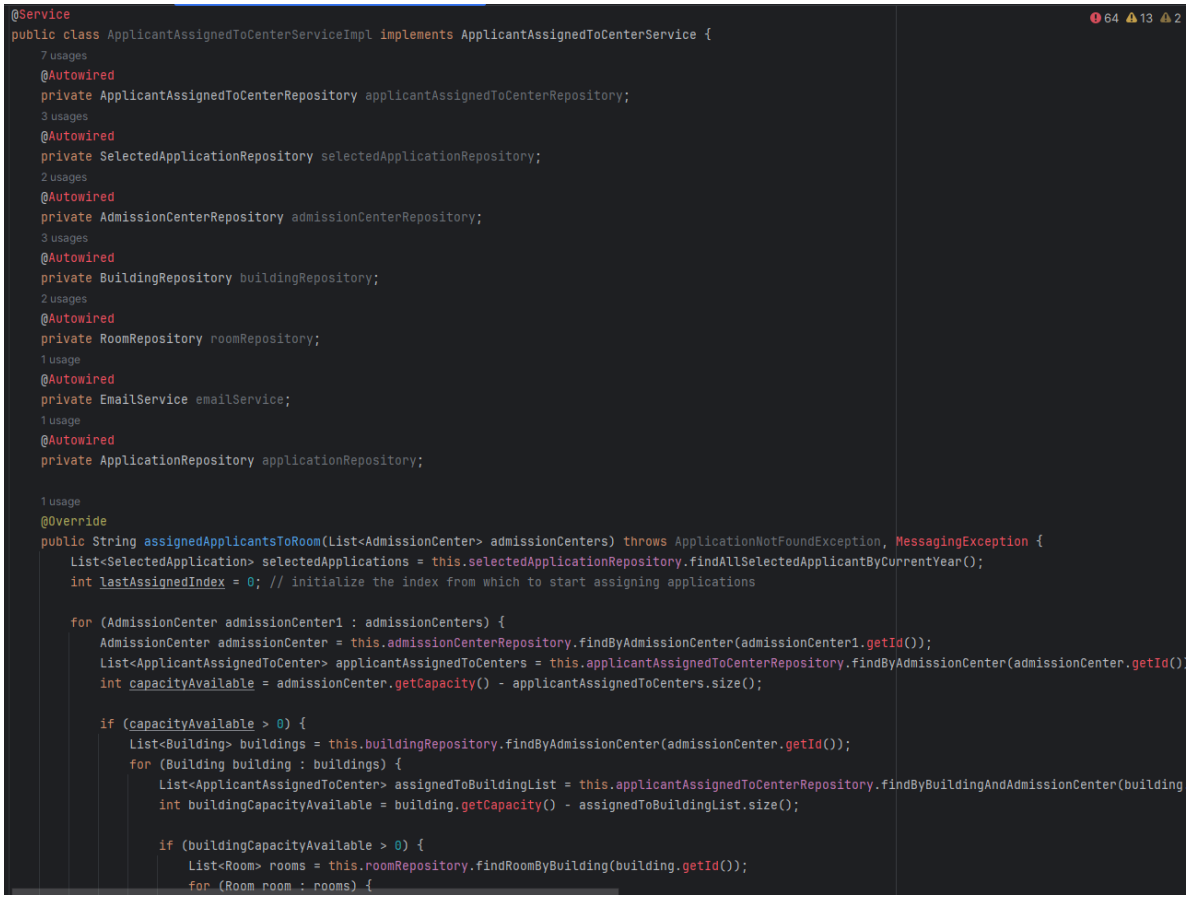
Figure 17: Admission Center Service

Admit Card Signature Service

```java
package com.ums.admissionservice.service.admitcardsignatory;

import com.ums.admissionservice.domain.model.AdmitCardSignatory;
import com.ums.admissionservice.domain.repository.AdmitCardSignatoryRepository;
import com.ums.admissionservice.dto.model.AdmitCardSignatoryDTO;
import com.ums.admissionservice.mapper.AdmitCardSignatoryMapper;
import com.ums.admissionservice.service.exception.AdmitCardSignatoryNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

no usages
@Service
public class AdmitCardSignatoryServiceImpl implements AdmitCardSignatoryService{
    7 usages
    @Autowired
    private AdmitCardSignatoryRepository admitCardSignatoryRepository;
    @Override
    public List<AdmitCardSignatoryDTO> index() {
        return AdmitCardSignatoryMapper.map(this.admitCardSignatoryRepository.findAll());
    }

    @Override
    public AdmitCardSignatoryDTO show(Long id) throws AdmitCardSignatoryNotFoundException {
        AdmitCardSignatory admitCardSignatory = this.admitCardSignatoryRepository.findById(id)
                .orElseThrow(()->new AdmitCardSignatoryNotFoundException(String.format("Admit card signatory with id: %s could not found", id)));
        return AdmitCardSignatoryMapper.map(admitCardSignatory);
    }

    @Override
    public String store(AdmitCardSignatoryDTO admitCardSignatoryDTO) {
        this.admitCardSignatoryRepository.save(
                new AdmitCardSignatory()
                        .setName(admitCardSignatoryDTO.getName())
                        .setDesignation(admitCardSignatoryDTO.getDesignation())
                        .setSignature(admitCardSignatoryDTO.getSignature())
        );
        return "Admit card signatory's information has been stored";
    }

    @Override
    public String update(Long id, AdmitCardSignatoryDTO admitCardSignatoryDTO) throws AdmitCardSignatoryNotFoundException {
        AdmitCardSignatory admitCardSignatory = this.admitCardSignatoryRepository.findById(id)
```

Figure 18: Admit Card Signature Service

Applicant Assigned to center Service



```java
@Service
public class ApplicantAssignedToCenterServiceImpl implements ApplicantAssignedToCenterService {
    7 usages
    @Autowired
    private ApplicantAssignedToCenterRepository applicantAssignedToCenterRepository;
    3 usages
    @Autowired
    private SelectedApplicationRepository selectedApplicationRepository;
    2 usages
    @Autowired
    private AdmissionCenterRepository admissionCenterRepository;
    3 usages
    @Autowired
    private BuildingRepository buildingRepository;
    2 usages
    @Autowired
    private RoomRepository roomRepository;
    1 usage
    @Autowired
    private EmailService emailService;
    1 usage
    @Autowired
    private ApplicationRepository applicationRepository;

    1 usage
    @Override
    public String assignedApplicantsToRoom(List<AdmissionCenter> admissionCenters) throws ApplicationNotFoundException, MessagingException {
        List<SelectedApplication> selectedApplications = this.selectedApplicationRepository.findAllSelectedApplicantByCurrentYear();
        int lastAssignedIndex = 0; // initialize the index from which to start assigning applications

        for (AdmissionCenter admissionCenter1 : admissionCenters) {
            AdmissionCenter admissionCenter = this.admissionCenterRepository.findByAdmissionCenter(admissionCenter1.getId());
            List<ApplicantAssignedToCenter> applicantAssignedToCenters = this.applicantAssignedToCenterRepository.findByAdmissionCenter(admissionCenter.getId())
            int capacityAvailable = admissionCenter.getCapacity() - applicantAssignedToCenters.size();

            if (capacityAvailable > 0) {
                List<Building> buildings = this.buildingRepository.findByAdmissionCenter(admissionCenter.getId());
                for (Building building : buildings) {
                    List<ApplicantAssignedToCenter> assignedToBuildingList = this.applicantAssignedToCenterRepository.findByBuildingAndAdmissionCenter(building
                    int buildingCapacityAvailable = building.getCapacity() - assignedToBuildingList.size();

                    if (buildingCapacityAvailable > 0) {
                        List<Room> rooms = this.roomRepository.findRoomByBuilding(building.getId());
                        for (Room room : rooms) {
```

Figure 19: Applicant Assigned to center Service

Payment

```
private RestTemplate restTemplate;
1 usage
@Override
public String ApplicationPaymentStore(String applicantUniqueId) throws FeesStructureNotFoundException, FeesHeadNotFoundException {
    Application application = this.applicationRepository.findByApplicantUniqueId(applicantUniqueId);
    FeesStructure feesStructure = this.feesStructureRepository.findByFeesStructureTypeApplication();
    FeesStructureFeesHeads feesStructureFeesHeads = this.feesStructureFeesHeadsRepository.getFeesStructureFeesHeadsByFeesStructuresId(feesStruct
    FeesHead feesHead = this.feesHeadRepository.findById(feesStructureFeesHeads.getFeesHeads().getId()).orElseThrow(()->new FeesHeadNotFoundExce
    Random rnd = new Random();
    int reff_number = rnd.nextInt( bound: 99999999);
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    Map<String, String> requestBody = new HashMap<>();
    requestBody.put("user_id", application.getApplicantUniqueId());
    requestBody.put("amount", String.valueOf(feesHead.getAmount()));
    requestBody.put("currency_code", "BDT");
    requestBody.put("cus_name", application.getName());
    requestBody.put("cus_email", application.getEmail());
    requestBody.put("cus_address", "Dhaka");
    requestBody.put("cus_city", "Dhaka");
    requestBody.put("cus_state", "Dhaka");
    requestBody.put("cus_postcode", "1208");
    requestBody.put("cus_country", "Bangladesh");
    requestBody.put("cus_phone", application.getNumber());
    requestBody.put("response_type", "json");
    requestBody.put("service_type", "");
    requestBody.put("success", "http://google.com");
    requestBody.put("redirect", "http://facebook.com");
    requestBody.put("reff_id", String.valueOf(reff_number));
    HttpEntity<Map<String, String>> request = new HttpEntity<>(requestBody, headers);
    ResponseEntity<String> response = restTemplate.postForEntity("https://api.1card.com.bd/odootest/pay", request, String.class);
    String responseString = response.getBody();
    String cleanString = responseString.replace( target: "\\",  replacement: "");
    return cleanString;
}

1 usage
@Override
public String AdmissionTestPaymentStore(String applicantUniqueId) throws FeesStructureNotFoundException, FeesHeadNotFoundException {
    Application application = this.applicationRepository.findByApplicantUniqueId(applicantUniqueId);
    FeesStructure feesStructure = this.feesStructureRepository.findByFeesStructureTypeAdmissionTest();
    FeesStructureFeesHeads feesStructureFeesHeads = this.feesStructureFeesHeadsRepository.getFeesStructureFeesHeadsByFeesStructuresId(feesStruct
    FeesHead feesHead = this.feesHeadRepository.findById(feesStructureFeesHeads.getFeesHeads().getId()).orElseThrow(()->new FeesHeadNotFoundExce
```

Figure 20: Payment

10.2 Possible problem breakdown:

It's challenging to do a big assignment correctly at once. It'll take a long time as well. If the work can be separated into reasonable parts, it will be easier to accomplish and take less time overall. The work will be completed efficiently and competently. As a result, the following inventory of possible issues is given:

- There might be redundant data.
- Standard error management; insufficient requirement analysis.
- Inadequate user support.
- Insufficient testing.
- Insufficient support from users.

I've done the following to help prevent these potential issues:

- The data structure was created without utilizing redundant data.
- Contains all error handling types.
- Created a requirement analysis and had real users evaluate it.
- A number of methods were tried.
- User approval was assessed among actual users.

Every potential issue has been resolved.

10.3 Prioritization while developing:

The best way to create a project's priority list is to use the MoSCoW priority technique. I'm going to use this technique to prioritize the requirements that IUMS Admission Service has found. Below is a summary of each

Prioritization	Requirement Name	Explanation
Must-haves	User registration and authentication Admission application submission Application processing and status tracking	Essential requirements that are critical for the project's success. These are non-negotiable and must be delivered in the initial release for the system to be considered functional.
Should-haves	Document verification and management Room allocation functionality Fee management and payment processing	Important requirements that are significant but not critical for the project's success. These can be deferred to a later release if necessary but are highly desirable for the project.
Could-haves	Advanced analytics and reporting Integration with external systems Enhanced notification features	Desirable but not critical requirements. These are often seen as nice-to-have features that can be implemented if there is enough time and resources available.
Won't-haves	Non-essential features that might be considered for future releases Features that are beyond the current project's scope	Requirements explicitly agreed upon not to include in the current project. These are deferred to future releases or not considered for implementation.

Table 8: Prioritization Requirement

Chapter 11 – Testing

This admissions service system is web-based. It adheres to the agile SDLC concept. I utilize these tools and programming languages (Java Spring Boot, My SQL) for development. Spring framework is mostly utilized.

Project Name	IUMS Admission Service	
Name of Product	IUMS Admission Service	
Product Description	Comprehensive software solution designed to streamline and optimize the admission processes within educational institutions.	
Project Description	Java Spring Boot and MySQL	
Project Duration	Start date	10/07/23
	End date	30/11/23

Figure 21: IUMS Admission Service

11.1 Old System Use Case:

Allow me to now explain this Admission Service System exam. As everyone knows, software testing plays a crucial role in producing a high-quality end result. By definition, software testing is the process of confirming and validating the behavior and artifacts of the software under test. In order to meet user needs or product quality objectives, the testing process must be effectively completed.

The purpose of the Admission Service System test is to:

- Verify that a certain piece of software meets all necessary specifications.
- Use the least amount of money and time possible to confirm the effectiveness of software testing.
- Make the best possible test cases, test software that is effective, and generate useful and accurate problem reports.

This project's testing phase is already complete. Because of the flexible Model I used. The agile software development life cycle (SDLC) approach, which prioritizes process flexibility and customer satisfaction through quick delivery of functional software, uses incremental and iterative process models. The product is divided into smaller, incremental builds using agile methodologies. Every iteration includes cross-functional teams working on many projects at once in the following areas:

- Planning
- Requirements analysis
- Design
- Coding
- Unit testing
- Acceptance testing

The customer and important stakeholders are shown a functioning product following iteration and testing.

I'll now go over the primary portion of the testing using the Admission Service system.

The Acceptance of the Test Plan The test plan includes information about the project's resources, objectives, schedule, estimates, and deadlines in addition to the test methodology. They assist others outside the QA department by providing a detailed description of the testing process. They assist quality assurance engineers in carrying out their testing duties by serving as a guide. They cover a lot of ground, including the exam's cost, scope, and general approach. To make evaluating and using this information simpler for management staff, we've compiled it all into a single document.

The purpose of testing admission service software:

- Identify errors in software
- Build trust in it
- Assess its attributes
- Assess performance
- Assess security
- Assess usability

Now, explain the testing stage:

Functional Testing: The procedure used by quality control specialists to determine whether a piece of software satisfies its stated requirements is called functional testing. Black-box testing techniques are used in functional testing because the tester is not aware of the reasoning behind the system at all. Finding out if a system operates as intended is the only goal of functional testing.

The three types of functional testing that are listed below are as follows:

Unit Testing:

- Verification of the input forms.
- Selected kind filtering.

Module Testing:

- Submitting the login form without a password or username.
- Incorrect registration or login information.
- Send in a form that contains accurate or proper data.

Integration Testing:

- Enter the necessary login credentials to log in.
- Addition and updating of personal data with success.

Non-Functional Testing: This type of testing examines the software's functionality and performance. On the other hand, functional testing seeks to verify the overall functionality of a piece of software. When developing software, testing—both functional and non-functional—is essential. Both contribute to the proper operation of your product.

Various criteria are used to classify non-functional testing:

- Security Testing
- Usability Testing
- Acceptance Testing
- Accessibility Testing

11.2 Test Case:

In software engineering, a system test is a single test that is run to achieve a particular software testing goal, like assessing a particular program path or ensuring that a certain requirement is met. Programs, intended results, testing procedures, and input parameters are a few examples of test

cases. Unlike random testing, intentional testing starts with a test case. A collection of test cases that sufficiently cover the software under test can be assembled.

Test Case Name	Unit/Module/Integration Test		
Test Case			
Test Description			
Source of Data	Test Steps	Expected Result	Actual Result

Table 9: Test Case

11.3 Unit Testing:

Test Case Name	Unit Test		
Test Case	Calculate Fees For Application		
Test Description	Test the calculation of fees for an admission application.		
Source of Data	Test Steps	Expected Result	Actual Result
Admin	1. Select an admission application. 2. Calculate fees.	Fees are calculated based on the defined structure.	Fees are calculated based on the defined structure.

Table 10: Unit Test Case 1

Test Case Name	Unit Test		
Test Case	Allocate Room For Application		
Test Description	Test the allocation of a room for an admitted student.		
Source of Data	Test Steps	Expected Result	Actual Result
Admin	1. Select an admitted student. 2. Allocate a room.	Room is successfully allocated to the student.	Room is successfully allocated to the student.

Table 11: Unit Test Case 2

11.4 Module Testing:

Test Case Name	Module Test		
Test Case	Process Payment Test		
Test Description	Test the processing of payment for an admission application.		
Source of Data	Test Steps	Expected Result	Actual Result
Admin	1. Select an admission application. 2. Initiate the payment process.	Payment is successfully processed.	Payment is successfully processed.

Table 12: Module Test Case 1

Test Case Name	Module Test		
Test Case	Receive Notification Test		
Test Description	Verify that notifications are received.		
Source of Data	Test Steps	Expected Result	Actual Result
Admin	1. Trigger a notification. 2. Check for the received notification.	Notification is received successfully.	Notification is received successfully.

Table 13: Module Test Case 2

11.5 Integration Testing:

Test Case Name	Integration Test		
Test Case	Application Submission And Fees Calculation		
Test Description	Verify the integration between application submission and fees calculation.		
Source of Data	Test Steps	Expected Result	Actual Result
Admin	1. Submit an admission application. 2. Calculate fees for the submitted application.	Application submission and fees calculation integrate seamlessly.	Application submission and fees calculation integrate seamlessly.

Table 14: Integration Test Case 1

Test Case Name	Integration Test		
Test Case	Document Upload And Room Allocation		
Test Description	Verify the integration between document upload and room allocation.		
Source of Data	Test Steps	Expected Result	Actual Result
Admin	1. Upload a document for an admission application. 2. Allocate a room for the admitted student.	Document upload and room allocation integrate seamlessly.	Document upload and room allocation integrate seamlessly.

Table 15: Integration Test Case 2

11.6 Acceptance Testing:

Test Case Name	Acceptance Test		
Test Case	Admission Application Submission		
Test Description	Applicants should be able to fill out the application form, upload necessary documents, and submit the application.		
Source of Data	Probability	Impact	Mitigation
Admin	Medium	Delays in the admission process, frustration among applicants, and potential loss of qualified candidates.	1. Conduct thorough testing of the application submission process. 2. Provide a user-friendly interface with clear instructions.

Table 16: Acceptance Test

11.7 Security Testing:

Test Case Name	Security Test
Test Case	Validate data encryption during transmission.
Test Description	Ensure that data is transmitted over secure channels (HTTPS) to prevent eavesdropping.
Source of Data	Mitigation
Admin	Implement SSL/TLS protocols for secure data transmission.

Table 17: Security Test 1

Test Case Name	Security Test
Test Case	Verify security of file upload functionality.
Test Description	Ensure that file uploads are restricted to specific file types, and implement proper validation to prevent malicious uploads.
Source of Data	Mitigation
Admin	Validate file types, set size limits, and use a secure file storage mechanism.

Table 18: Security Test 2

This section contains a summary of all the testing efforts. The information displayed here comprises

Test cases completed: 75

Test cases that pass: 75

Test cases are unsuccessful: 0

Pass: 100%

Failure: 0%

Note: Developed based on user requirements and successful testing phase.

Defect

One of the most important items in the test report is the defect. The report ought to contain the following data.

Total number of bugs: 10 (open, closed, and resolved).

Number of bugs:

- Open: 10
- Resolved: 10
- Closed: 10

Sorting by importance and severity: resolve

Test Report of Admission Service System

	Test	

Test Cycle System test

EXECUTED	**PASSED**	**50**
	FAILED	**0**
Pending		**0**
In Progress		**0**
Blocked		**0**
(Sub-Total) Test Passed		**0**
Pending + In Progress+ Blocked + Test Executed		**50**

Table 19: Final Test Report

Chapter 12 – Implementation

12.1 Training

Software is created to minimize discomfort while working and to automate tasks as much as feasible. A well-designed system should be easy for users to use and navigate. Even Nevertheless, some customers may still find it challenging to use the system because they lack IT skills, even if it is user-friendly. For this reason, any large system ought to include a training phase. My method is really simple to use. But I will also be conducting a training phase in the university where I will be selling it. I have to train a group of university representatives so that they can train others in the university. I'll demonstrate to them how the entire system works and is used in a real demo server area. I will test their ability to perform that flawlessly following my presentation. The training step can be effectively completed when you have taught them.

User	Training	Time	Comment
Administrator	• Generate admission reports • Review application statistics • Manage user roles	4 hours	The administrator, teachers and staffs are knowledgeable about every feature and operator.
Teacher	• View assigned courses • Grade assigned courses	3 Hours	
Staff	• Process applications • Manage documents	4 hours	

Table 20: Training Based

12.2 Big Bang

A system can use the big bang approach to automatically adjust previous system data. It's like updating versions when the system is modified but the data is kept intact. These features need a great deal of work to develop. In my system, this capability is absent. Outdated system data requires human input. However, I want to include this feature into my system in the future.

12.3 Scaling

The DCL team leader has been in charge of scaling this project. I was not informed about this scaling acceptance during my internship.

12.4 Load Balancing

Load balancing is the optimization of a system against the impact of its users. The number of users hit indicates how many users are using the system at once and how long it has been running. We call this load balance and load equalization. By spreading the load among several servers, it enables the system to keep running regularly.

Chapter 13 – Critical Appraisal and Evaluation

13.1 Objective that could be met

There is no such thing as a flawless system. Updates are always possible. I've tried to complete all necessary tasks so that a university may operate this system and use it efficiently. Nonetheless, some conditions might be met. These are the following:

- The frontend user interface of the admission service system from a worldwide standpoint.
- Room Allocation
- Management of fees payments

13.1.1 Success rate against each objective

The system as a whole and all of its goals have excellent success rates. All pertinent data is gathered and displayed more effectively for the user's benefit. Teachers and staffs portals accurately handle any issues and provide all pertinent information about them.

13.1.2 How much better could have been done

Because it was created with Java spring boot, the system is more updated and responsive than other systems of a similar nature. It would be preferable, though, if I could integrate library management functions into the system.

13.1.3 Why it could not be done

This system contains all of the necessary core features. Every function is quickly, easily navigable, safe, and well-managed. Using the system is easy and uncomplicated. However, the system could not manage massaging or different statistical data views.

13.1.4 Which objectives have been missed

This system has all the necessary important features. Every function is handled quickly, safely, easily, and efficiently. Using the system is easy and uncomplicated. Nevertheless, the system would not permit multiple views or manipulations of statistical data.

13.1.5 Why these objectives have missed

I could not build those features because client budget issue, time and this features was not client main requirements.

13.2 Objectives totally not met / touched

The system has every feature needed to function as an admission service. It features a payment recode store online. However, this technique has no effect on online bank or other payment transactions.

13.2.1 Why it could not be touched

For this segment, I did some field research. If we wish to develop online transactions, we need to do work linked to APIs. This cannot be developed in the timeframe and money provided by our client.

13.2.2 What could have been done

It's possible to add a mobile banking system to the system. In order for parents to use their particular mobile banking system to pay for their children's education.

Chapter 14 – Lessons Learned

14.1 Pre project – review – closing

The internet is the main Admission Service System platform. My company followed a predetermined framework to build this system, which included obtaining the client's needs, creating a plan for the system's creation, selecting the system's architecture, and deciding on a project name. After that, I built this system with PHP and wrote project-specific documentation. This admission service system handles tracking for administrators, teachers, and staff, among other things.

14.2 What have I learned

This university uses the Admission Service System, a web-based program. As a trainee DCL intern, I provided backend development support for this project. I learned a number of things from working on this project that I will find very helpful going forward. My initial experience with software architecture was with third-party designs. I learned a few new, undiscovered design techniques while I was working on the project. I've researched them and included them in the process I created. Such as how to use templates to generate designs and how to improve their usability. I am ready for the system to be developed after the design is complete. I have studied them and incorporated them into the process I created. Such as how to use templates to generate designs and how to improve their usability. I am ready for the system to be developed after the

design is complete. Any system must first have its database designed before it can be constructed. If a suitable database architecture is not put into practice, a system cannot be completed. I've worked with small databases before, which were necessary for a small project. Nonetheless, this system has given me more experience with a sizable database and multiple data tables.

I helped install the system on the working server when the project was completed. I had never installed a system on a shared server before, until now. I can honestly say that working on this project has been a fantastic experience because I have learned far too much.

14.3 What problem I have faced

I worked as a developer as well as a designer on this admission service project. Over the project's duration, I've encountered several difficulties and issues. I had the most trouble working with the project architecture that the team leader created. Because it might be challenging to work with architecture that has been constructed by someone else. Understanding the architecture takes a lot of time and complexity. I have to confront and resolve issues iteratively because I employed the Agile DSDM strategy, which guarantees iterative development. Stated differently, issues that crop up in one time box get fixed in this other time box and task. I've already begun creating the system, but I was told to stick to the templates. However, the reason it took so long to finish was that I had no knowledge how to make the templates for the design. I also needed to learn new things about user-friendly design. It became obvious that I would need to establish a large database with multiple data tables as soon as I started working on the system's database. But I had never dealt with a database this huge before.

14.4 What solutions occurred

There is a fix for every problem that can be applied to fix it. I have completed the system and come up with a solution for every issue I mentioned earlier. First and foremost, I developed a way to use software architecture that was already produced by others. During the design process, I discovered some new and surprising basic ideas. I obtained them and incorporated them into this system. For example, how to increase usability and create a design using templates. I've got ready for when the design is complete and the system is developed. A database needs to be constructed before any systems can be developed.

Chapter 15 – Conclusion

15.1 Summary of the project

This project was created for an academic setting. Most universities in our country are still analog. The university still maintains its historical handwritten data-keeping system. I tried to figure out what needed to be done in order to build an Admission Service that would benefit all associated users of any university. I used the internet to study a few things that had to be included in the project in addition to doing fieldwork. All users of this system will benefit from its many features, which are intended to make university digital and technologically advanced.

15.2 Goal of the project

Since I took part in a DCL internship program, my organization has determined the goal of my project. In the software development industry, a system's goal is established by the needs of the customer. The goals of the admission service system were likewise set in line with the needs of the customer. The main objective of this project is to automate the manual procedure and lower labor costs.

- Convert every system to a computer.
- Cut down on the time invested and the chance of error.
- Every system has automatic management.
- Administration of databases centralized.
- There is no need for paperwork; the system operator has simple controls.

15.3 Success of the project

I have created a system with every feature that is necessary for a university. It can be used successfully in various universities. My biggest accomplishment to far have been providing a management system to universities in need. I developed the university's admissions service system with success.

15.4 What I have done in the documentation (stages / activities / plans)

I had to first finish it in accordance with the format for academic projects because this documentation is for an academic project. I've included a thorough account of all I accomplished

for the project in this documentation. I began by writing the system's introduction in the documentation, after which I started this project's basic investigation and progressively finished all the procedures required for this system. This documentation was created using Microsoft Word for all tables, and a third-party tool was used to create any figures or diagrams.

15.5 Value of the project

Work that is done entirely by hand is more difficult. If all of the job can be entirely automated, time and hassle might be saved. The same amount of labor can be done more quickly and with less effort when technology is used. It takes a lot of work and time to manually maintain all of an organization's client information.

15.6 My experience

Working on this project taught me a lot, even though I was not familiar with all the technology involved. I had to relearn these concepts as well as how to put them into practice in order to complete the job. I may now work with others on software architecture that has been designed by others. Developing software and writing code can be rewarding and difficult at the same time. There are times when things come together and I feel accomplished, but I also run across unforeseen difficulties that call for problem-solving abilities. Creating software for an admissions service is a complex process that calls for innovative thinking, technological difficulties, and a never-ending learning curve. It's a journey that blends the necessity for adaptability in the face of changing requirements and technologies with the gratification of making a positive contribution to a crucial process.

Appendix:

Applicant: A person who applies for admission to a university or educational institution.

Admission Office: Administrative unit responsible for processing and managing admission applications.

IUMS: Integrated University Management System, the overarching system that includes the Admission Service module.

Use Case: A description of how a system will be used by its actors (users).

Actor: A person or system that interacts with the admission service system.

Java Spring Boot: https://spring.io/projects/spring-boot/

My SQL: https://www.mysql.com/

Reference

Integrated University Management System(IUMS). (2000). Retrieved from ValuePLUS Computer Systems Ltd.: https://www.vpcsbd.com/products/ius/

Jain, S. (n.d.). *Phases of Waterfall Model*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/waterfall-model/

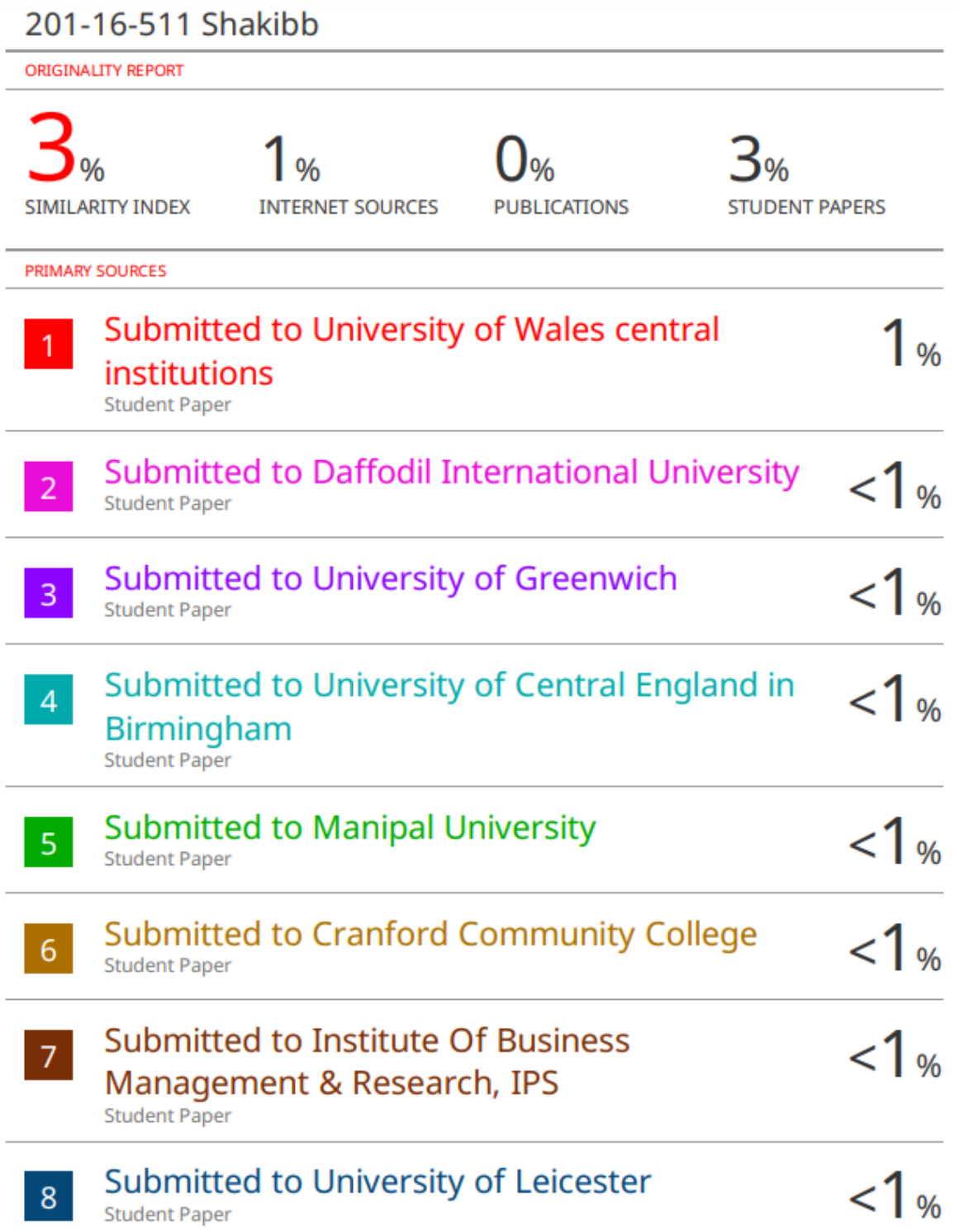
Mazepa, A. (2009). *Dynamic Systems Development Method*. Retrieved from New Line Technologies: https://newline.tech/dynamic-systems-development-method/

Risk Management. (2007). Retrieved from Pacs: https://www.pacs.org.uk/risk-management#:~:text=%EF%BB%BFManagement&text=PACS%20IWMS%20provides%20a%20fully,manage%20performance%20and%20measure%20compliance.

Rungta, K. (2008). *Test Report*. Retrieved from Guru99: https://www.guru99.com/how-test-reports-predict-the-success-of-your-testing-project.html

Schroer, A. (2011). *What Is the V-Model in Software Development?* Retrieved from Built In: https://builtin.com/software-engineering-perspectives/v-model

Plagiarism report



201-16-511 Shakibb

ORIGINALITY REPORT

3%	1%	0%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Wales central institutions Student Paper	1%
2	Submitted to Daffodil International University Student Paper	<1%
3	Submitted to University of Greenwich Student Paper	<1%
4	Submitted to University of Central England in Birmingham Student Paper	<1%
5	Submitted to Manipal University Student Paper	<1%
6	Submitted to Cranford Community College Student Paper	<1%
7	Submitted to Institute Of Business Management & Research, IPS Student Paper	<1%
8	Submitted to University of Leicester Student Paper	<1%