2024-04-25

Optimizing Academic Operations: A Business Process Redesign for iRAS at IUB

Rahman, Md. Mostafizur

IUB, CSE

Optimizing Academic Operations: A Business Process Redesign for iRAS at IUB

By

Md. Mostafizur Rahman

Student ID: **1711445**

Spring, 2024

Supervisor:

Dr. Mahady Hasan

Associate Professor

Department of Computer Science & Engineering

Independent University, Bangladesh

April 25, 2024

Dissertation submitted in partial fulfillment for the degree of Master of Science in Software Engineering

Department of Computer Science & Engineering

Independent University, Bangladesh

Attestation

Md. Mostafizur Rahman, hereby attest that the project titled Optimizing Academic Operations: A Business Process Redesign for iRAS at IUB is my original work. This project represents the culmination of my efforts during 6 months as part of the requirements for my MSc in Software Engineering at Independent University, Bangladesh. I declare that this work is free from any form of plagiarism, and all sources used for information and ideas are appropriately cited. The methodologies, findings, and conclusions presented in this project are the result of my own research and analysis. I further affirm that this project has not been submitted in whole or in part for any other academic qualification at any other institution.

Signature

Date

Md. Mostafizur Rahman

Acknowledgement

I extend my heartfelt gratitude to my esteemed supervisors, Dr. Mahady Hasan, for their invaluable guidance, unwavering support, and expert insights throughout the entire duration of this project. Their mentorship has been pivotal in shaping the direction and enhancing the quality of this work.

I would like to express my sincere appreciation to the faculty and staff at Independent University, Bangladesh, especially the Integrated Registrar's Office, for providing essential resources and creating a conducive environment for research and learning.

A special thank you goes to my classmates and colleagues for their encouragement and constructive feedback, which significantly contributed to the refinement of this project.

I am deeply thankful to my family and friends for their understanding and support during the challenging phases of this endeavor.

Finally, I acknowledge and appreciate the work of individuals and organizations whose contributions have been referenced in this project. Their insights have enriched the content and depth of my research.

I am grateful to everyone who played a role in the completion of this project. Your support has been invaluable.

Md. Mostafizur Rahman

Independent University, Bangladesh

Letter of Transmittal

Dear Sir,

I am pleased to submit the project report titled "Optimizing Academic Operations: A Business Process Redesign for iRAS at IUB" in fulfillment of the requirements for my Software Engineer at Independent University, Bangladesh.

This report encapsulates a comprehensive analysis and redesign of the Integrated Registrar's Office Automation System (iRAS) at IUB, aiming to enhance the quality and efficiency of academic operations. The project explores various aspects, including business processes, system functionalities, and recommendations for improvements.

Throughout this undertaking, I have had the privilege of working under the guidance of my esteemed supervisors, Dr. Mahady Hasan, whose expertise and support have been invaluable in shaping the direction of this project.

The report encompasses detailed insights into the existing system, the proposed redesign, and recommendations for further development. It addresses challenges, outlines advantages, and provides a comprehensive view of the proposed solutions.

I trust that this report will contribute meaningfully to the academic community and serve as a valuable resource for those interested in the optimization of registrar office automation systems.

Thank you for the opportunity to undertake this project, and I look forward to any feedback or insights you may have.

Sincerely,

Md. Mostafizur Rahman
Independent University, Bangladesh

Evaluation Committee

Supervision Panel

.....	.....
Supervisor	Co-supervisor

External Examiners

.....	.....
External Examiner 1	External Examiner 2

Office Use

.....	.....
Program Coordinator	Head of the Department

.....

Director Graduate Studies, Research and Industry Relations

.....

Library

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The project focuses on enhancing the Integrated Registrar Office Automation System (iRAS) at Independent University, Bangladesh (IUB) to address existing challenges and improve its overall efficiency. In response to the dynamic technological landscape, the web-based application has significantly streamlined administrative tasks such as student admissions, course registrations, faculty evaluations, and graduation processes.

However, despite its advantages, iRAS faces challenges, particularly in the manual paperwork involved in certain tasks, dissatisfactions during the course registration period, and inconsistencies in course allocations. This project aims to identify, analyze, and redesign iRAS to provide end-users with an improved user interface and introduce new features, such as online payment options, enhanced admission processes, and comprehensive profiles for students and faculty members.

The scope encompasses the development of a revised version of iRAS, implementing a client-server-based web application, providing guidance on the new system, reducing manual paperwork, and realizing time and cost savings. The objectives include creating a robust system with an enhanced UI, restructuring the backend code, transitioning to REST API services, implementing profile-based admission systems, developing a management dashboard, enhancing registration and payment processes, strengthening security and privacy measures, and improving course and faculty management.

Ultimately, the project seeks to create a more efficient and user-friendly iRAS aligned with the evolving needs of IUB, fostering a seamless transition to a computerized system and minimizing manual paperwork.

1.2 Contribution of the project

I played a pivotal role in the comprehensive development and enhancement of the Integrated Registrar Office Automation System (iRAS) throughout the entire thesis project. Taking charge of various crucial aspects, I undertook responsibilities in coding, architectural design, and continuous refinement of the system to align with the project's objectives.

Specifically, my contributions include:

1. System Architecture and Design:

   I led the architectural design, ensuring a robust and scalable structure for iRAS. This involved conceptualizing and implementing a client-server-based web application, transitioning to REST API services, and restructuring the backend code for improved performance.

2. User Interface (UI) Enhancement:

   Recognizing the importance of a user-friendly interface, I focused on enhancing the UI design. This encompassed creating an improved interface with mobile responsiveness, aiming for an optimal user experience.

3. Profile-Based Admission System:

   A significant addition to the system was the implementation of a profile-based admission system for new students. This included integrating online payment methods, simplifying the admission process, and ensuring a seamless on boarding experience.

4. Security and Privacy Measures:

   Acknowledging the critical nature of sensitive data, I took charge of implementing enhanced security measures and privacy safeguards within iRAS, providing a secure environment for all users.

5. Efficient Registration and Payment Processes:

   Streamlining administrative tasks, I worked on enhancing the registration and online payment processes, striving for increased efficiency and reduced manual efforts.

6. Individual Employee Profiles:

   To cater to the needs of faculty and staff, I conceptualized and implemented individual profiles. These profiles encompass basic information, attendance tracking, grade submission, announcements, and performance evaluations.

7. Course and Faculty Management:

   Recognizing the challenges in course registration and faculty management, I focused on refining these processes. This involved managing course registration, student billing, transcripts, and empowering department heads to streamline faculty assignments.

8. Management Dashboard:

   A comprehensive management dashboard was introduced, providing administrators with a centralized overview of system activities, and fostering informed decision-making.

Throughout the thesis project, my efforts were geared towards not only addressing existing challenges in iRAS but also towards introducing novel features to enhance the overall user experience. My contributions aimed to create a sophisticated and efficient system aligned with the evolving needs of Independent University, Bangladesh.

1.3 Organization of the project

The execution of the project to enhance the Integrated Registrar Office Automation System (iRAS) at Independent University, Bangladesh, adhered to a structured and collaborative organizational framework. The key components of the project organization are outlined as follows:

1. Project Leadership:

   Project Lead: I assumed the role of the project lead, overseeing the entire development lifecycle and steering the team towards the project's objectives.

   Supervisors: Dr. Mahady Hasan provided valuable guidance, support, and expertise throughout the project. Their insights played a crucial role in aligning the project with institutional goals.

2. Development Team:

   Coding and Architecture: The coding and architectural aspects of iRAS were predominantly handled by me. This included system design, backend restructuring, and implementation of new features.

   Collaborators: Ahsanul Haq and Bristi Shaha contributed to the initial phase of the project by collecting requirements. Their inputs were reviewed and incorporated into the evolving system.

3. Collaborative Decision-Making:

   Regular Meetings: Regular team meetings were held to discuss progress, address challenges, and make collaborative decisions. This facilitated a transparent and communicative environment.

   Feedback Loops: Continuous feedback loops were established, allowing team members to share their insights and suggestions, and fostering a culture of iterative improvement.

4. Task Allocation:

   Roles and Responsibilities: Clear roles and responsibilities were defined for each team member. This ensured that tasks were distributed according to individual strengths and expertise.

   Task Monitoring: A systematic approach was adopted for monitoring task progress, identifying bottlenecks, and strategizing solutions to maintain project momentum.

5. Agile Development Methodology:

   Scrum Framework: The project embraced an Agile development methodology, particularly the Scrum framework. This iterative approach allowed for flexibility, adaptability, and continuous improvement.

   Sprints and Milestones: Development cycles were organized into sprints, with clearly defined milestones. This facilitated incremental progress and regular assessments of achieved goals.

6. Quality Assurance:

   Testing and QA: A dedicated focus was given to quality assurance. Thorough testing processes were implemented to identify and rectify bugs, ensuring the reliability and stability of the evolving iRAS.

7. Stakeholder Engagement:

   Supervisor Engagement: Regular updates and consultations with supervisors ensured alignment with academic standards and institutional requirements.

   User Feedback: Continuous engagement with end-users, including faculty and students, played a pivotal role in refining the system based on practical needs and feedback.

8. Documentation and Reporting:

   Documentation: Comprehensive documentation was maintained throughout the project, including requirements, design decisions, and coding practices.

Progress Reports: Regular progress reports were generated, highlighting achievements, challenges, and proposed strategies for moving forward.

The organization of the project

Chapter 2

Literature Review

I am working on various aspects of academic operations optimization at IUB, including the development life cycle, software development performance and social processes, business process reengineering, redesigning the registration integration system, and describing the current system. To support each component, I am conducting a literature review for targeted areas

2.1 Agile Software Development:

The adoption of agile development methodologies, exemplified by Extreme Programming (XP), has become prominent for its focus on time-to-market constraints and adaptability during software development [1]. However, challenges arise in the form of architectural scalability, developer-related issues, and organizational complexities. Studies by Cao et al. (2009) emphasize the need for adaptive structuration and context-awareness to tailor agile practices effectively [2]. In enterprise environments, the coexistence of agile methods with traditional plan-driven development brings about challenges related to landscape complexity and business involvement, prompting the development of mitigation strategies focused on improved communication . Additionally, the necessity of a robust methodology for software development and validation is highlighted, emphasizing formal specifications, dual design, and validation against functional requirements [3]. Overall, the literature stresses the importance of adaptability, effective communication, and reliable methodologies to ensure the success and quality of software development.

2.2 Software Development Performance and Social Processes:

In the landscape of software development, this research delves into the intersection of production methods and social processes and their influence on performance [4]. Draw-

ing insights from 40 software development teams, the study scrutinizes their impact on both software product quality and overall team performance. Strikingly, the findings reveal that conventional production methods, encompassing software methodologies and automated tools, do not emerge as significant contributors to the variations observed in software product quality or team performance [5]. In stark contrast, the study underscores the pivotal role of social processes, particularly informal coordination, conflict resolution, and team supportiveness, in accounting for the observed outcomes.

2.3 Business Process Reengineering (BPR):

Business Process Reengineering (BPR) involves the radical redesign of business processes for improved performance, with the integration of the Internet and Workflow Management Systems (WfMS) playing a key role in enhancing process automation and coordination [6]. In Public Administration (PA), BPR is driven by new legislation, services, and technology, offering the potential for significant improvements in service delivery. AIPA in Italy focuses on leveraging information technology to enhance public services, leading to projects that reengineer processes and migrate software systems. Interestingly, the success factors in BPR are remarkably similar between the private and public sectors, underscoring the importance of efficiency and performance [7].

2.4 Online Course Registration System:

The literature on online registration systems in educational institutions highlights a significant shift from traditional to electronic methods, emphasizing enterprise portals' principles for simplicity, dependability, and systematic management [**?**]. Case studies showcase positive outcomes such as enhanced efficiency and cost-effectiveness. Methodologies like SSADM and OOADM are discussed, emphasizing the need for comprehensive approaches involving technological integration, process automation, and user capacity building[8] [9]. A specific case in the Bangladesh context focuses on the development of an online examination system, showcasing the potential benefits of transparency, efficiency, and risk avoidance associated with traditional paper-based methods.

2.5 Microservice Architecture and Security:

The proliferation of network speed, reliability, and security has driven the shift from local to third-party-managed software and services accessible through the network. This evolution birthed the microservices architecture, a novel software development and architectural style tailored to address the maintenance and scalability demands of online

service providers [10]. As the microservices architecture represents a burgeoning research area, a systematic mapping study becomes essential to consolidate progress, identify gaps, and delineate future study requirements. The study delves into microservices architectures and their implementation, focusing on architectural challenges, diagrams/views, and quality attributes. Simultaneously, another study scrutinizes the Quality Attributes (QAs) associated with Microservices Architecture (MSA), offering a systematic understanding based on a Systematic Literature Review (SLR)[11] [12]. The SLR reveals six prominent QAs in MSA, including scalability, performance, availability, monitorability, security, and testability, with 19 identified tactics addressing these critical QAs [13]. A third study explores the integration of Microservices Architecture (MSA) in DevOps, systematically mapping the literature to identify themes, problems, solutions, visualization methods, quality attribute impacts, and supporting tools. The findings underscore the multifaceted aspects of MSA in DevOps, from development and operations integration to migration experiences and tool support[14]. Together, these studies illuminate the architectural, quality, and operational dimensions of microservices, offering insights into their implementation, challenges, and impact on software development practices.

Chapter 3

System Overview

3.1 General Description of Current Process

In the current business environment, we have identified several distinct subsystems within the existing business system. For each of these subsystems, a rich picture [15] has been created and is accompanied by a detailed description. They are as follows:

Admission System (Rich Picture)

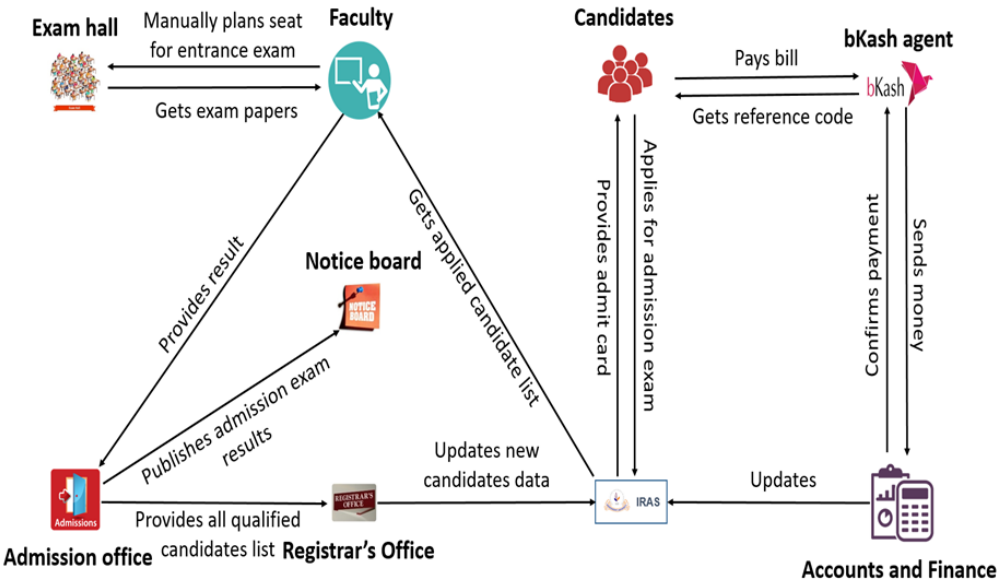

Figure3.1:Admission System

The admission process begins with qualified candidates applying online for admission. Our Integrated Registrar Office Automation System (iRAS) facilitates this process, guiding applicants through the payment system. Payments are made via bKash, and upon completion, the funds are transferred to the accounts department. The Accounts and Finance Office updates the database using iRAS. Once the payment is confirmed, candidates receive a reference code, which they use to download their admit cards from

iRAS. As the admission exam date approaches, iRAS temporarily suspends the payment system, and faculties access the list of applied candidates through iRAS. Faculties then manually create seating plans for the exam. On the day of the admission exam, candidates take their seats and complete the examination. Afterward, faculties review and assess the exam papers, providing the results to the Admission Office. The Admission Office subsequently publishes the results and shares the list of qualified students with the Registrar's Office.

- **Course Allocation and Course Offerings**

- **Scholarship**

After the admission exam, the top ten performing students are awarded a 100 percent scholarship. These students are required to submit all the necessary documents to the Admission Office for verification. Students who do not rank among the top ten may still apply for scholarships through various special quotas. The list of students applying for scholarships is forwarded from the Admission Office to the Registrar's Office by the admission office staff. Subsequently, this list, along with all the required documents, is sent to a decision board or committee responsible for verification. Once the verification process is complete, the decision board provides the Registrar's Office with a list of approved scholarship recipients. In addition to this, regular students have the opportunity to apply for merit-based scholarships if they meet the minimum required CGPA. Eligible students can submit their scholarship applications to the Dean's Office. The list of applicants is then sent from the Dean's Office to the Registrar's Office for processing. Regular students can also request special scholarships from the department head or dean, and the list of such requests is sent to the Dean's Office and, subsequently, to the Registrar's Office for consideration. All the relevant information pertaining to scholarship applications, approvals, and disbursements is meticulously recorded and updated in the Integrated Registrar Office Automation System (IRAS).

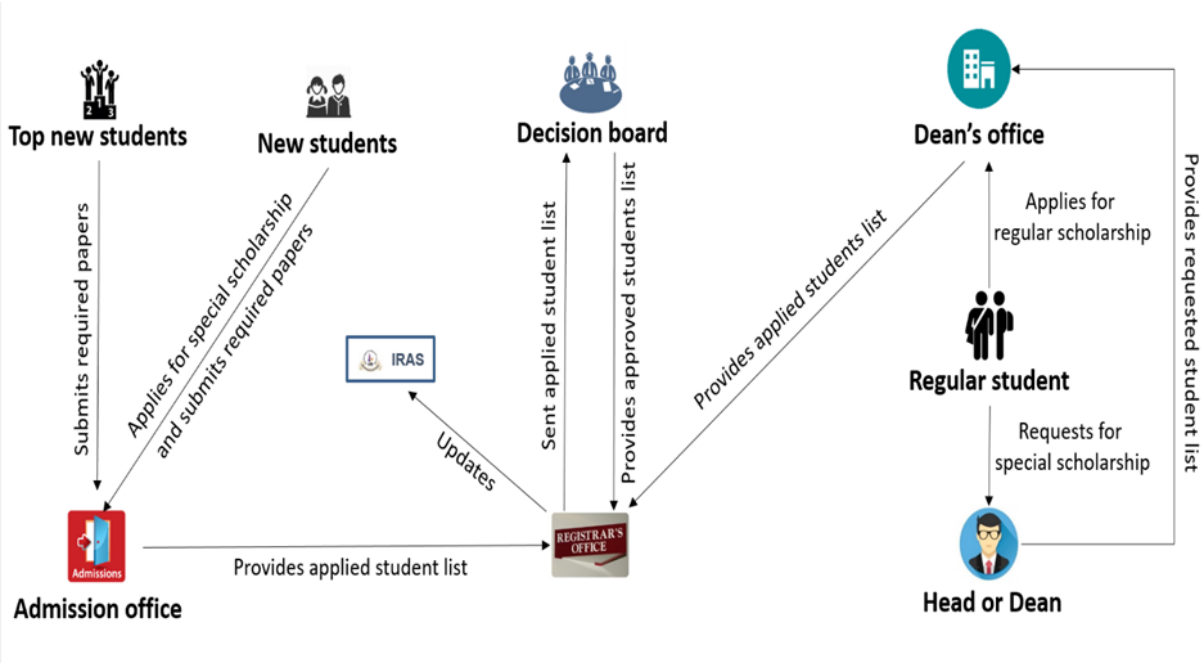

Figure 3.3: Scholarship

- **Registration, payments and Attendance**

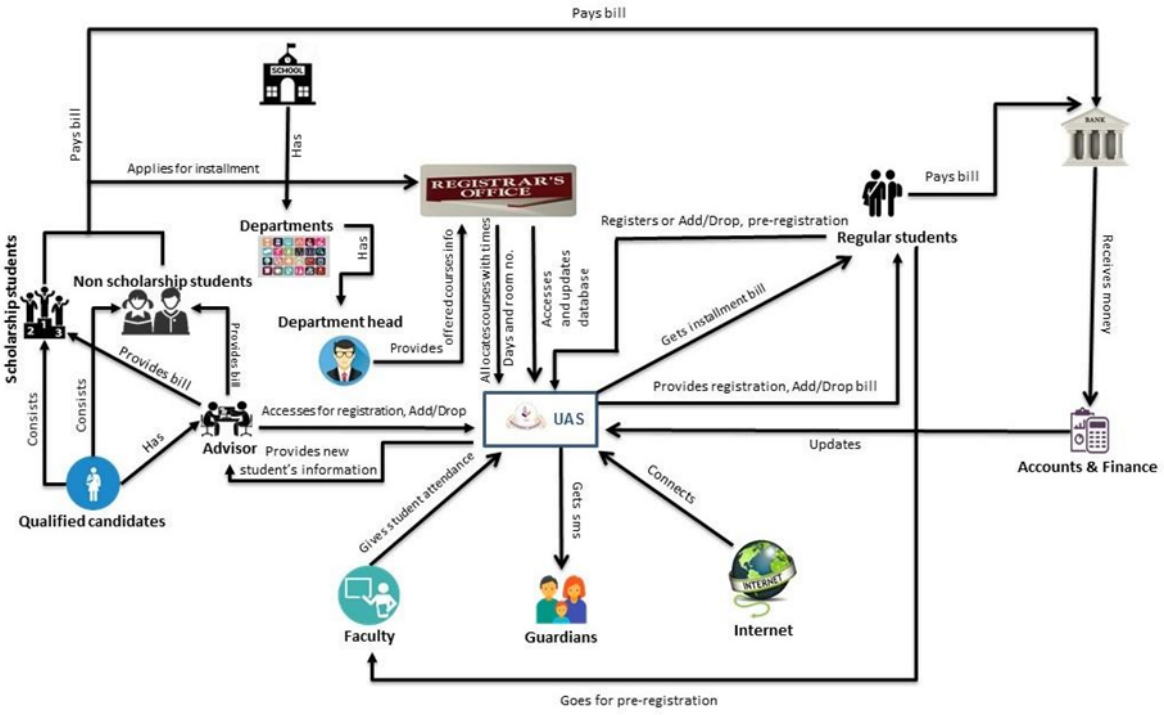

Figure 3.5: Registration, payments, and Attendance

In the current system, prospective university candidates apply online. Payment details are depicted in the rich picture. After applying, students sit for exams, and qualified students are listed for admission, with top students receiving scholarships. The admission office processes admissions and shares student data with the registrar's office. Upon admission, students are assigned advisors who use iRAS to counsel students on course

selection, facilitate registration, and allocate courses. Department heads provide course information to the registrar's office, which handles course allocation through iRAS. Bill generation, payment, and record-keeping occur through the accounts and finance office. Regular students access iRAS for registration, course selection, and bill payment. Students seeking installment payment apply at the registrar's office.An 'Add/Drop' phase exists for course adjustments. Regular students use iRAS, while new students consult advisors.

Scholarship applications are manual, with the board selecting recipients and informing the registrar's office. Classes begin after registration. Faculty take online attendance via iRAS, with automatic course withdrawal after eight missed classes. iRAS is also used for grade submission

- **Grade submission**

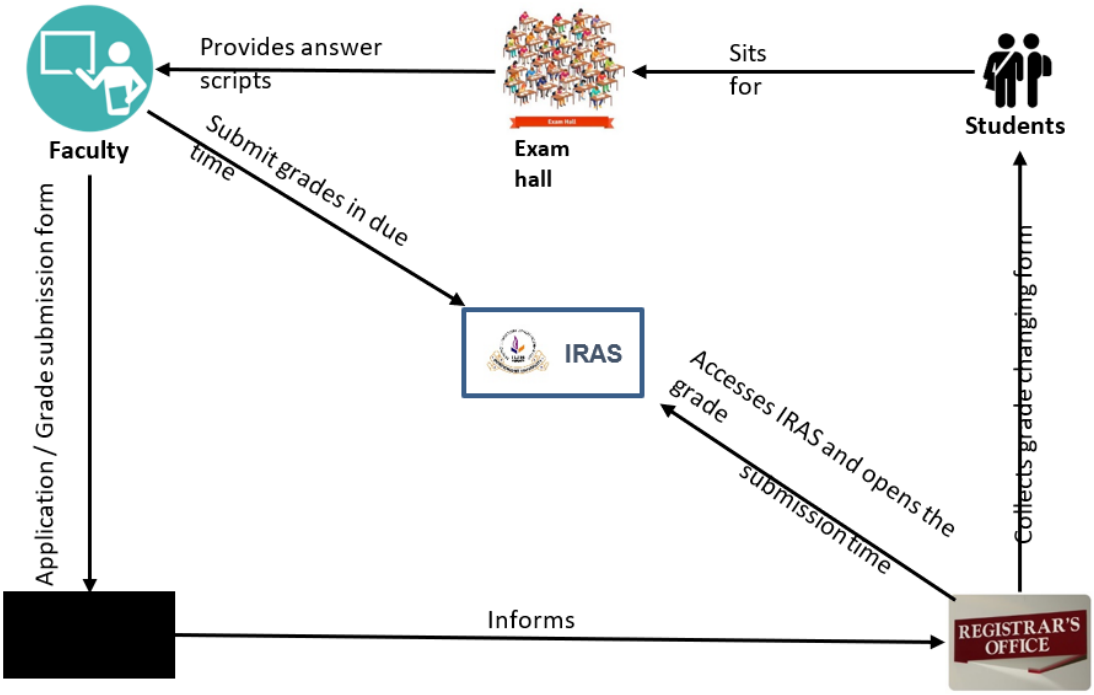

Figure 3.6: Grade submission

After students complete exams in the exam hall, the answer scripts are provided to faculty members for grading. Faculty members are responsible for timely grade submissions to iRAS. In case of delays, faculty members must submit a grade submission form to the Controller of Examination. If grades are not submitted on time, the Controller of Examination notifies the Registrar's Office. The Registrar's Office then accesses iRAS to extend the grade submission deadline for faculty members. Students who believe they received an undeserved grade can challenge it. To initiate a grade change request, students must obtain the required form from the Registrar's Office with a faculty reference

3.2 Detail view of the current working process

The existing business system has been segmented into several distinct subsystems, comprising approximately nine critical operational processes. These processes entail a series of interdependent actions that collaborate to yield specific outcomes. Since the inception of iRAS, both thick and fat clients have been utilized. Certain vital processes, such as registration, continue to be executed through the fat client interface. Notably, mobile applications have not been integrated, and some pages are not optimized for mobile devices. Below, we outline the fundamental steps of the iRAS processes:

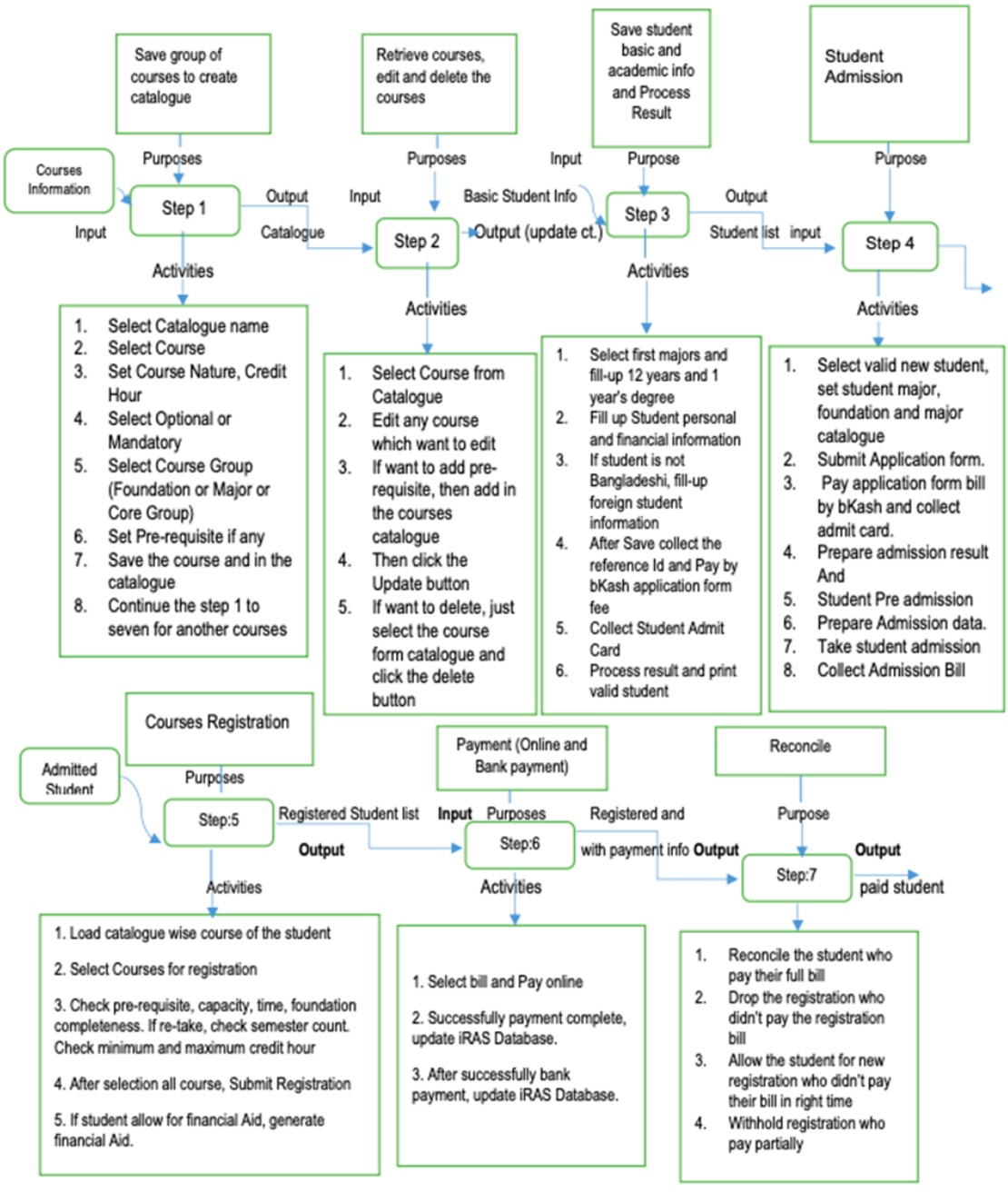

Figure 3.7: Detail view of the current working process

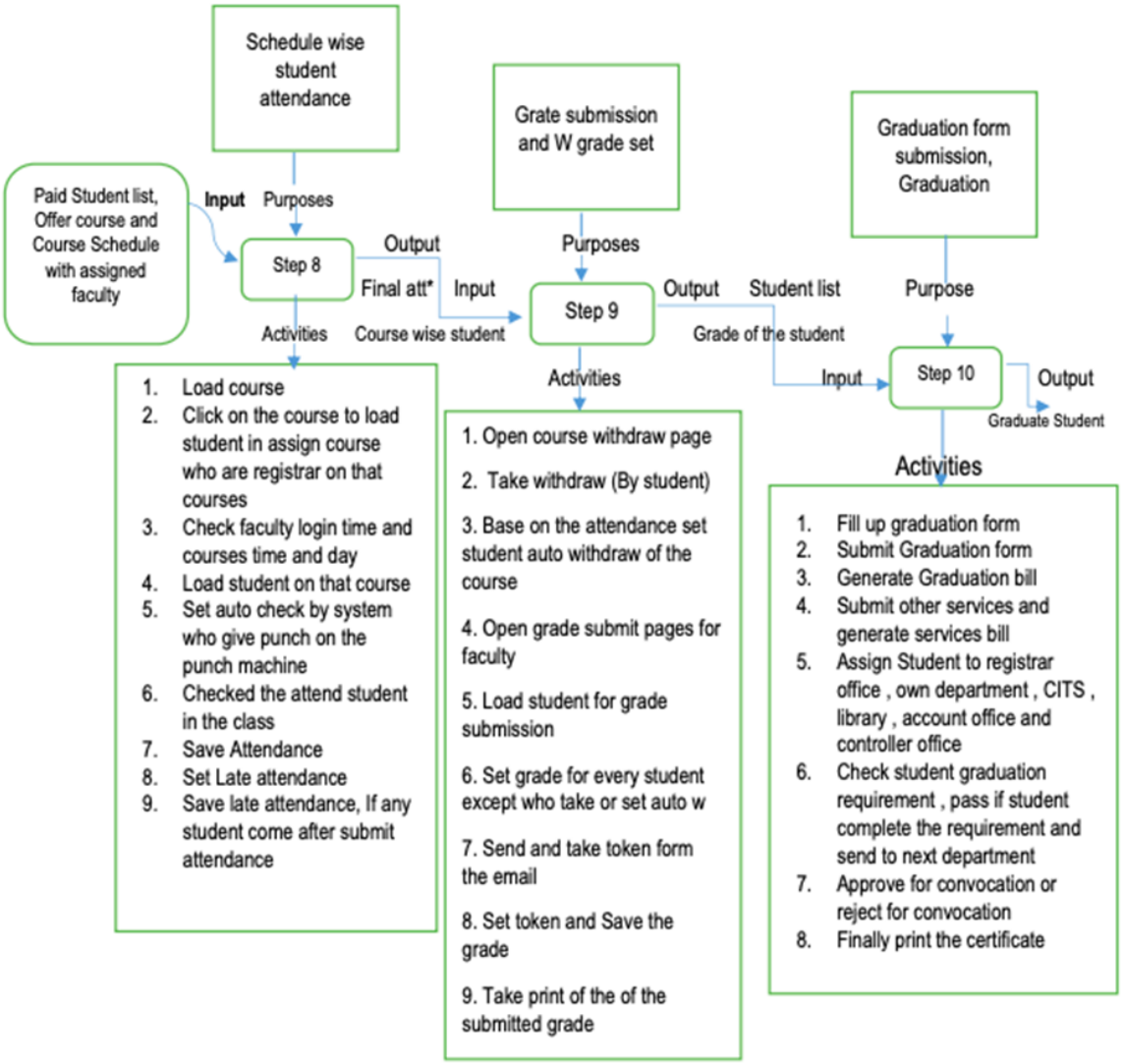

Figure 3.8: Detail view of the current working process

3.3 Detail six element analysis of the proposed business process

I. Human:

- Officer in Charge: Responsible for creating work manifestos, drafting tasks, and overseeing critical processes like late registration, accounts, admission, and add-drop.
- Dean: Plays a vital role in course modeling, registration, and overall student degree progression.
- Faculty and Departmental Head: Involved in various aspects including admission, course registration, grade submission, and course design.

- VC Office: Holds a key role in establishing academic rules.
- Students: Actively participate in admission, course registration, add-drop, and payment processes.
- Bank Officer: Provides information regarding student payments.
- Academic Board and UGC: Contribute to significant decisions regarding education courses.
- Admission Board: Manages admission-related issues.

II. Non-computing Hardware:

- Papers: Used for issuing official decisions, drafts, and official documents.
- Calculator: Utilized for computing balances in some instances.

III. Software:

- iRAS: Manages all university-related activities.
- Banking Software: Facilitates student and university transactions.
- bKash: Simplifies student payment methods.

IV. Hardware:

- Computers: Used by students, officers, faculty, and others to complete tasks within iRAS.
- Mobiles and Tablets: Provide flexibility for students to interact with iRAS.
- Printer and Scanner: Occasionally used for document-related tasks.

V. Database:

- Oracle DB: Serves as the database for iRAS, storing essential university data.

VI. Connectivity:

- Internet and Intranet: Facilitate official work processes.
- Mobile Data: Enables students to access and use iRAS for various tasks.

3.4 Process diagram using the proposed business process

These processes encompass a series of interdependent actions designed to collaboratively achieve desired outcomes. In the proposed system, the entirety of the process is web-based to ensure security and ease of maintenance. It operates as a client-server application with each process functioning as a service, enabling seamless operation between mobile and web applications. Below, you'll find detailed views of the iRAS processes:

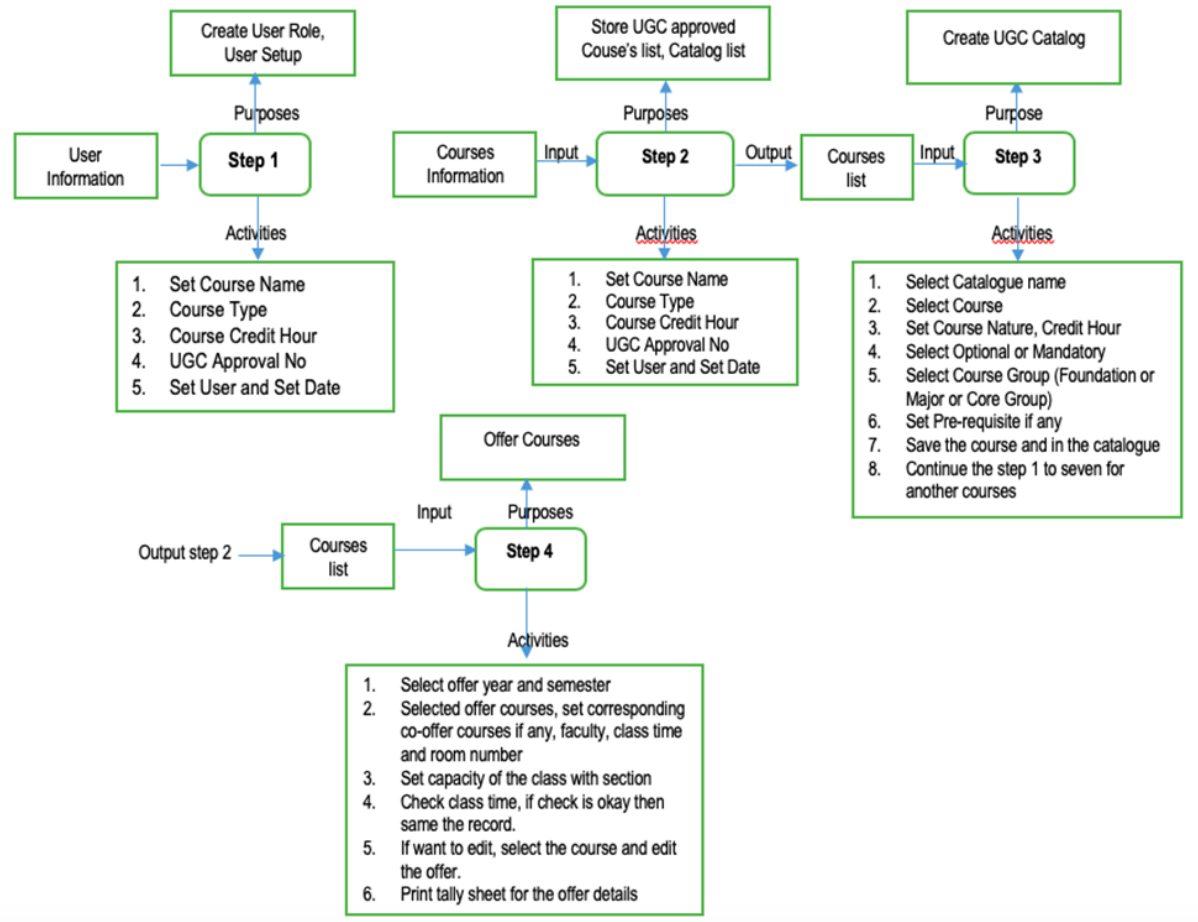

Figure 3.9: Detail view of the configuration module

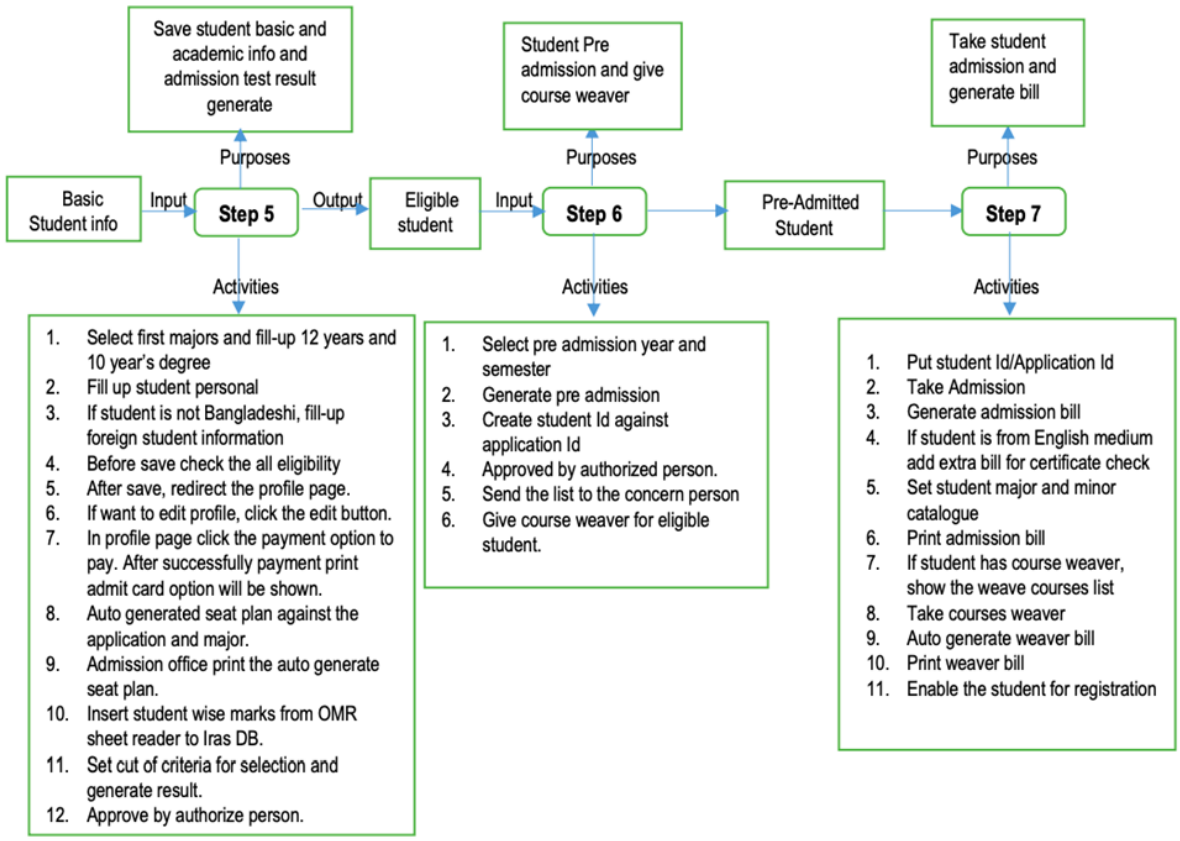

Figure 3.9: Detail view of the admission module

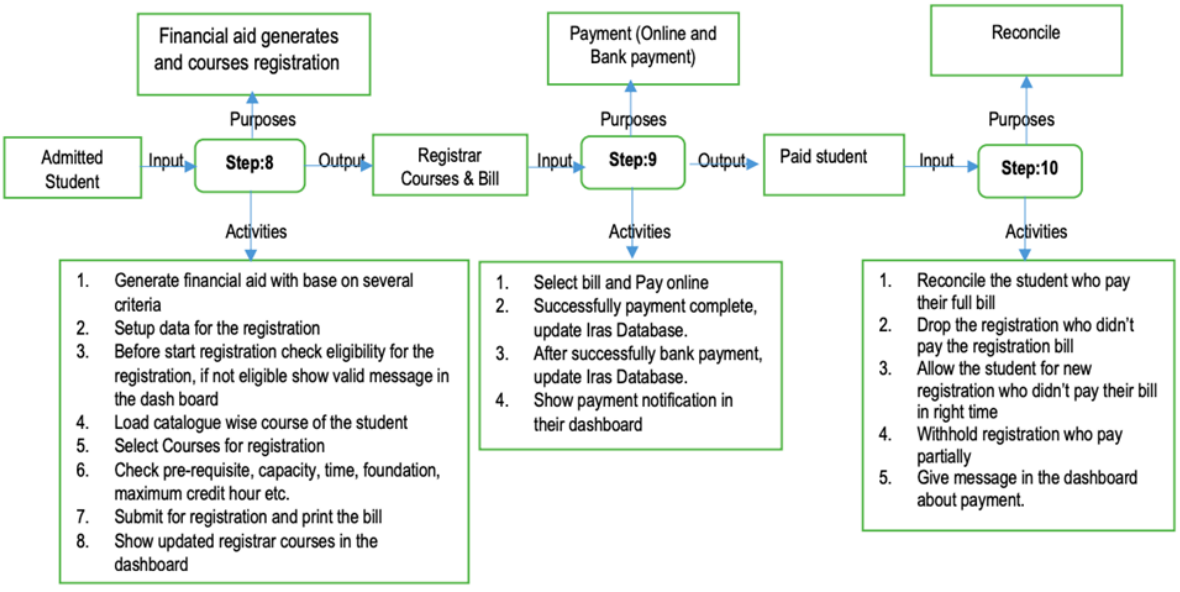

Figure 3.10: Detail view of the registration module

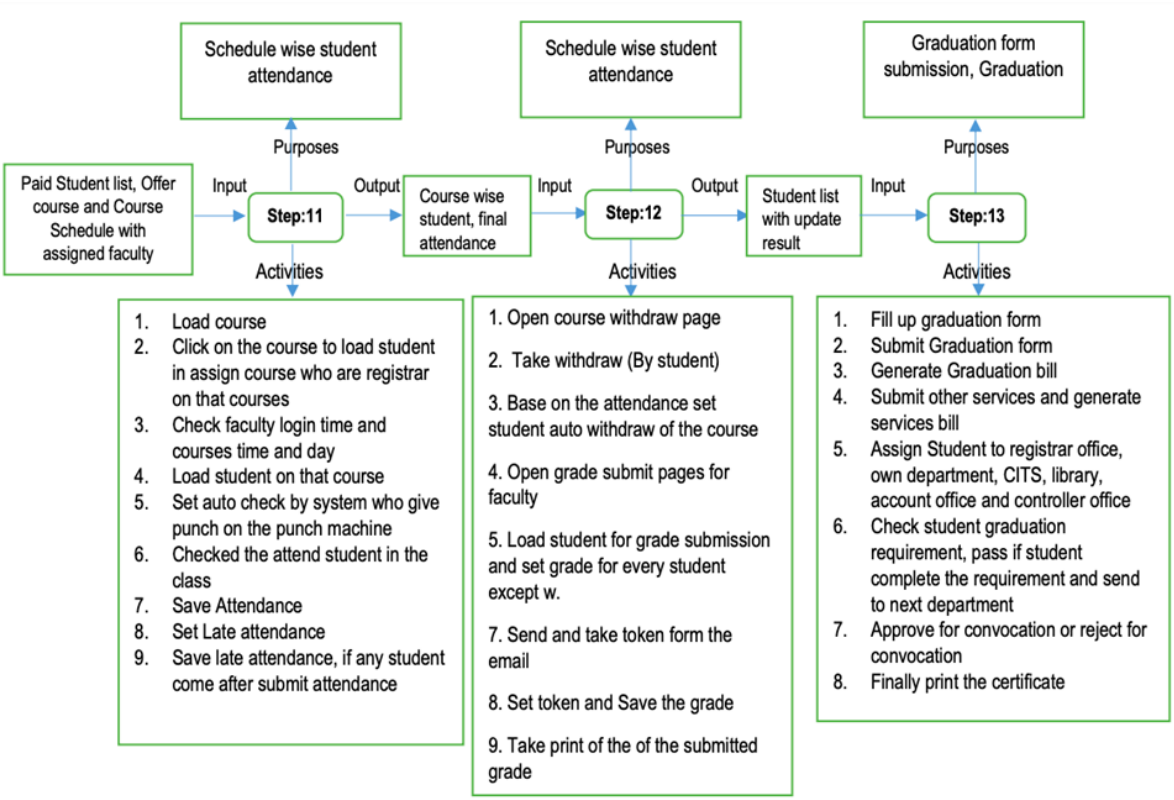

Figure 3.11: Detail view of the attendance, grade submission, and graduation module

Chapter 4

Methodology

For the successful realization of the "Business Process Redesign for Improvement of Quality and Efficiency of Integrated Registrar's Office Automation System (iRAS) for IUB," an Agile development methodology will be employed. Agile offers a dynamic and adaptive framework that aligns well with the project's objective of creating a user-friendly, computerized system while continually addressing evolving needs. This methodology will enable our development team to collaborate closely with stakeholders, including university administrators, faculty, and students, throughout the project's lifecycle. Agile emphasizes iterative development, allowing us to incorporate feedback and make adjustments as we progress. It promotes flexibility, enabling us to respond effectively to changing requirements and unexpected challenges, which is crucial in a project of this scope. Furthermore, Agile's focus on delivering functional components incrementally aligns with our goal to provide immediate value to end-users. As we enhance iRAS with new features and improvements, we will ensure that these elements are tested, refined, and integrated into the system efficiently. Agile will facilitate transparency, ensuring that all stakeholders are engaged and informed, leading to a system that better meets the evolving needs of Independent University, Bangladesh, while saving time and costs.

4.1 Fact Finding Techniques

In the pursuit of redesigning and improving the Integrated Registrar's Office Automation System (iRAS) for Independent University, Bangladesh, various fact-finding techniques [16] have been employed to gather essential information and insights. These techniques have played a pivotal role in identifying existing issues, understanding user needs, and formulating effective solutions.

4.1.1 Sampling

- Purpose: Sampling involves selecting a representative subset of the user population to collect data and feedback. This helps in understanding the general sentiments and preferences of a larger user base.
- Application: In the context of iRAS, sampling was utilized to survey a diverse group of students, faculty, and administrative staff. By analyzing a sample of user experiences and opinions, the project team gained valuable insights into common pain points and areas for improvement.

4.1.2 Questionnaire

- Purpose: Questionnaires are structured sets of questions designed to collect standardized information from a large number of participants. They provide quantifiable data on user preferences and issues.
- Application: Interviews were conducted with key stakeholders, including department heads, faculty members, and university administrators. These interviews provided qualitative data, allowing the project team to delve deeper into specific challenges and gather suggestions for system enhancements

4.1.3 Interview

- Purpose: Interviews involve direct, one-on-one or group conversations with users or experts. They are valuable for obtaining in-depth insights, clarifying issues, and uncovering specific pain points.
- Application:Interviews were conducted with key stakeholders, including department heads, faculty members, and university administrators. These interviews provided qualitative data, allowing the project team to delve deeper into specific challenges and gather suggestions for system enhancements.

4.1.4 Brainstorming

- Purpose: : Brainstorming sessions are creative, idea-generating meetings where participants freely share thoughts, suggestions, and innovative solutions. It fosters collaborative problem-solving.
- Application: Brainstorming sessions were organized with cross-functional teams comprising developers, designers, and end-users. These sessions encouraged creative thinking and generated ideas for new features, enhanced UI/UX design, and more efficient processes.

4.1.5 Observation

- Purpose: Ongoing observation of users interacting with iRAS can help identify real-time issues and challenges.
- Application :Continue to observe users as they work with the system. Look for patterns of behavior, usability issues, and areas where users may encounter difficulties. Use these observations to inform incremental improvements

By employing these fact-finding techniques, the project team gained a comprehensive understanding of the challenges and opportunities within the existing system. This information served as a foundation for the project's redesign and improvement efforts, ensuring that the revamped iRAS aligns more closely with the needs and expectations of its users.

4.2 Design System

Data Flow Diagram

iRAS, as the fully integrated system of Independent University, Bangladesh, connects various departments through an intricate network of modules. The interconnected nature of these modules is illustrated in the accompanying diagram. This visual representation provides a clear overview of how data flows and interacts between different departments and modules within the iRAS ecosystem, serving as a foundational blueprint for the system's design and functionality

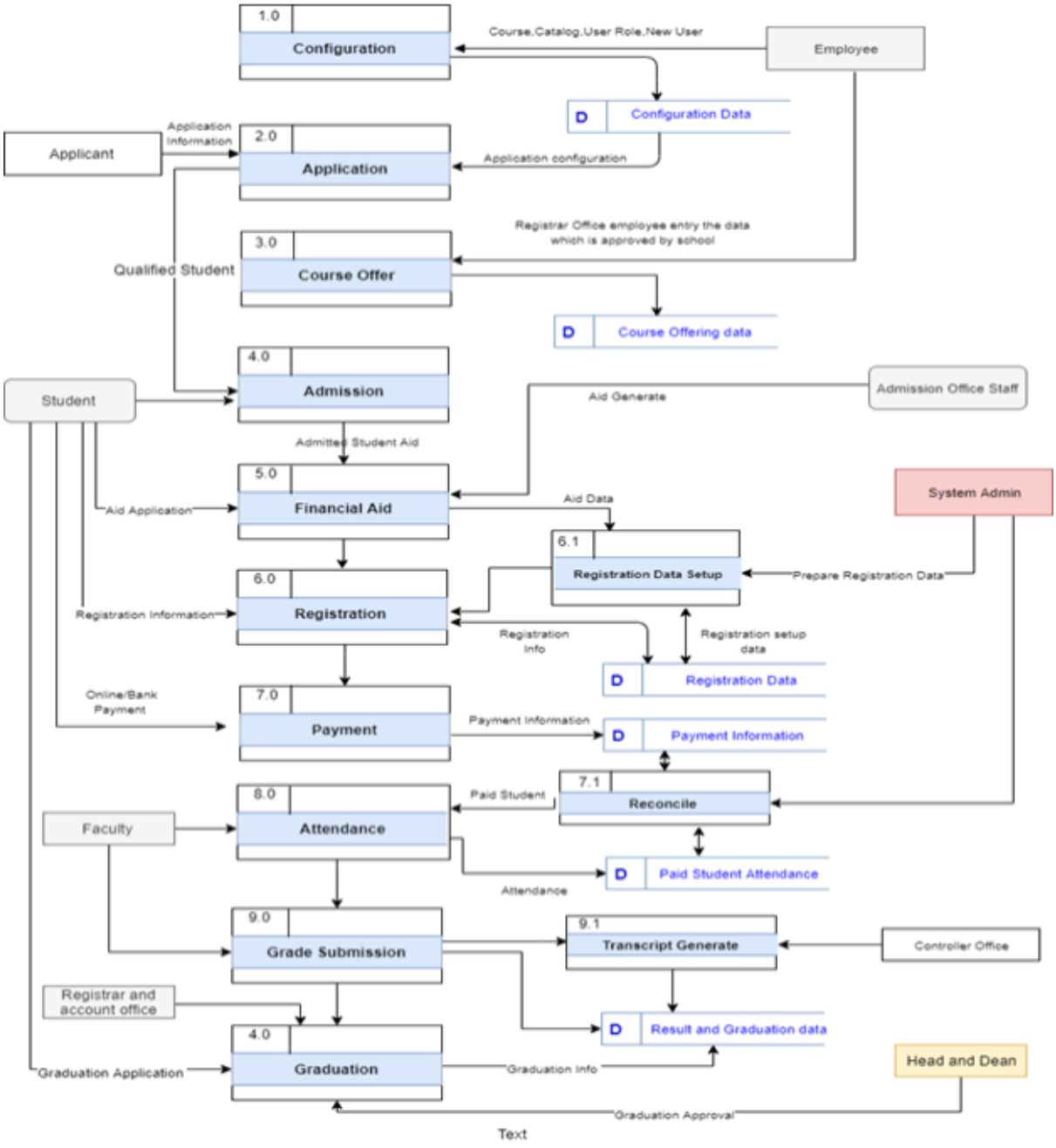

Figure 4.1: Flow diagram

4.2.1 Entity Relationship Diagram

Within the iRAS framework, the Configuration Module plays a critical role in managing the system's settings and parameters. The accompanying Entity Relationship Diagram (ERD) offers a visual representation of the entities and their relationships within this specific module. This ERD serves as a valuable tool for understanding how various elements within the Configuration Module are structured and interconnected, aiding in the design and implementation of this pivotal component of the system. It provides a comprehensive view of how data entities are organized and how they relate to one another, facilitating efficient system configuration and maintenance.

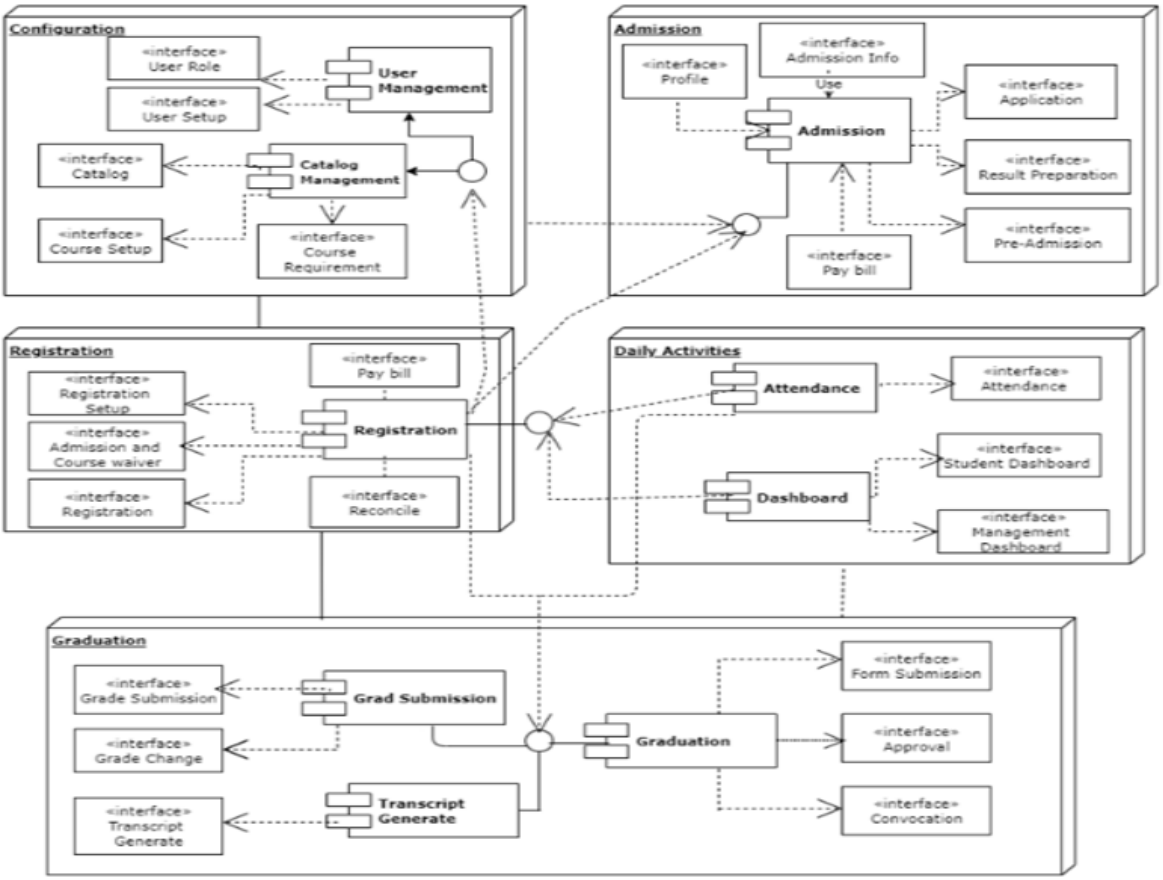

Figure 4.2: Configuration Module

4.2.2 Normalization

The relationships between various modules and the interfaces connecting them are visually represented through the Unified Modeling Language (UML) diagram. This diagram provides a concise and intuitive view of how the different components within iRAS interact and collaborate, aiding in the system's comprehensive understanding and design.

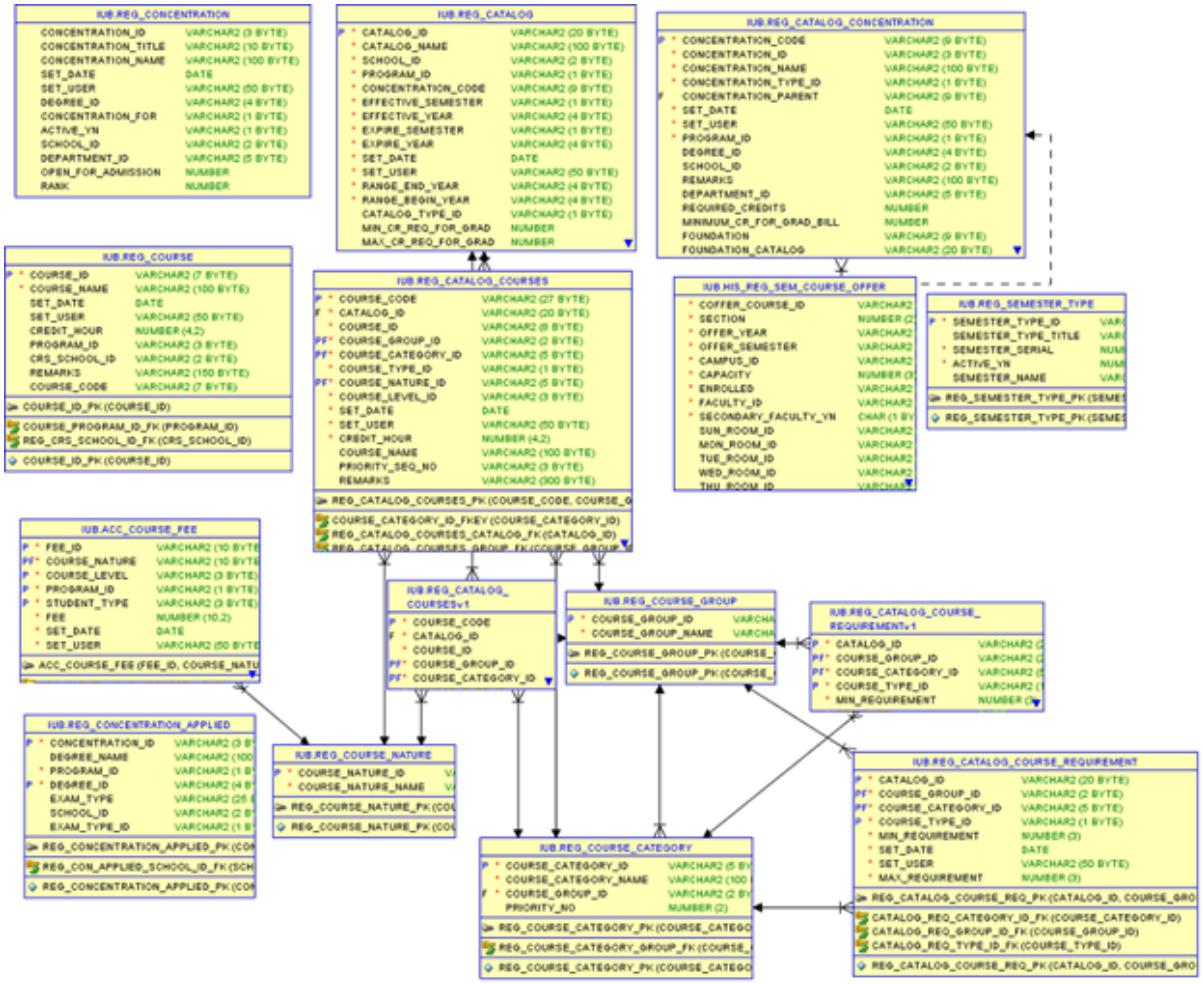

Figure4.3 : Unified Modeling Language (UML)

4.2.3 Programming Methodology

Here's a suggested programming methodology:

1. **Requirements Gathering and Analysis :** Collaborate with stakeholders, including university staff and administrators, to gather and analyze requirements for the IRAS system. Identify user stories, features, and functionality needed.

2. **Agile Development Approach:** Adopt an Agile development approach, such as Scrum or Kanban, to accommodate the dynamic nature of university requirements and ensure flexibility in responding to changes [17].

3. **Design Phase:**

   a. Create a clear and intuitive user interface (UI) using Angular for the front end. Ensure that the UI is user-friendly, responsive, and aligned with the university's branding and usability standards.

   b. Design the architecture of the ASP.NET Web API 2 backend to support the

required functionality. Plan API endpoints and data models for efficient data retrieval and manipulation.

c. Design the database schema using Oracle 19c, considering data integrity, normalization, and performance optimization.

4. **Development Phase:**

a. Develop the front-end components using Angular, implementing the UI design and user interactions.

b. Implement the business logic and API endpoints on the ASP.NET Web API2 backend. Ensure security, 'authentication, and authorization mechanisms are in place.

c. Implement database functionality using Oracle 19c, including data storage, retrieval, and management.

5. **Testing and Quality Assurance:**

a. Conduct thorough testing, including unit testing, integration testing, and end-to-end testing to identify and fix any issues.

b. Perform performance testing to ensure the system can handle the expected load

6. **Deployment and Continuous Integration/Continuous Deployment (CI/CD):**
Set up a CI/CD pipeline to automate the deployment process.Deploy the application to a test environment for further testing and validation.Once validated, deploy the application to the production environment.

7. **Monitoring and Maintenance:**
Implement monitoring and logging tools to track application performance and identify issues.Continuously monitor the system and address any bugs or enhancements promptly.Regularly update and maintain the software to keep it secure and up to date.

8. **User Training and Documentation:**
Provide training to users and administrators on how to use the IRAS system effectively.Create comprehensive documentation for users and developers to reference.

9. **Feedback and Iteration:**
Collect feedback from users and stakeholders to make iterative improvements to the system.Use Agile retrospectives to assess the development process and make necessary adjustments.

10. **Scalability and Future Expansion:**

a. Ensure that the system is designed for scalability to accommodate potential

growth in users and features.

b. Plan for future expansion by keeping the architecture modular and adaptable.

This methodology combines best practices from Agile development, front-end and back-end technologies, and database management to deliver a robust and responsive IRAS system for your university. It emphasizes flexibility, collaboration, and user-centric design throughout the development process.

4.3 Testing Methodology

For the evaluation of iRAS's functionality and reliability, two fundamental testing methodologies are employed: Unit Testing and Black Box Testing.

4.3.1 Unit Testing

Purpose: Unit testing focuses on verifying the correctness of individual components or units of the system. It ensures that each module and function performs as expected in isolation.

Application: Comprehensive unit tests are conducted to assess the behavior of specific code units within iRAS, validating their functionality and identifying any potential issues.

4.3.2 Black Box Testing

Purpose: Black Box Testing evaluates the system's functionality from an external, user-centric perspective, without knowledge of the internal code. It assesses whether the system meets specified requirements.

Application: Rigorous black box testing is performed to validate iRAS's overall behavior and its alignment with user expectations and project objectives.

4.4 Implementation Technology

Backed:

- C-charp: C-charp stands as a modern, object-oriented programming language known for its ability to construct robust and scalable applications on the .NET platform. It enjoys extensive usage in enterprise settings for its performance and reliability.
- .NET Core: .NET Core, an open-source and cross-platform framework, serves as the foundation for developing contemporary applications. This framework offers a rich suite of tools and libraries for creating robust and scalable web APIs.

- Web API: Leveraging .NET Core, Web API furnishes a streamlined approach to crafting HTTP-based services. It facilitates the creation of elegant and RESTful services, consumable by diverse clients
- Entity Framework: Entity Framework, an object-relational mapping (ORM) framework, empowers developers to interact with databases using strongly-typed entities. It offers a robust toolkit for tasks like data modeling, querying, and data access
- SignalR: SignalR, a real-time communication library, enables bidirectional interaction between clients and servers. It is a valuable resource for building real-time applications, such as chat platforms and dynamic dashboards
- gRPC: gRPC serves as a modern, high-performance RPC framework with open-source roots. It empowers bidirectional communication between clients and servers, underpinned by strongly-typed contracts

Frontend:

- Angular: Angular, a widely adopted open-source web application framework developed by Google, equips developers with the tools to build scalable and dynamic web applications.
- Angular Material: Angular Material functions as a comprehensive UI component library for Angular. It streamlines the development process by offering pre-built UI components like buttons, cards, and tables, enabling the creation of visually appealing and responsive user interfaces
- SCSS: SCSS, a potent CSS prepossess, enhances maintainability and modularity by allowing developers to write more structured CSS code
- HTML: HTML, the standard markup language for web pages, provides a rich set of semantic elements for structuring content and a range of attributes for styling and interactivity

Database:

Oracle19C: Oracle19C, a widely employed relational database management system, serves as the cornerstone for data storage and management in enterprise environments. It offers robust scalability and supports diverse data types, encompassing text, numbers, dates, and binary data.

Chapter 5

Requirement analysis

5.1 Software Developments Methodologies

As a private university with multiple stakeholders, frequent changes in decisions and new requirements are expected. In this dynamic environment, an Agile development methodology [1] like Scrum would be an ideal fit. Scrum's iterative approach [18] allows for continuous feedback and adaptation, ensuring that the final product meets the needs of all stakeholders. Additionally, Scrum promotes collaboration and communication among team members and stakeholders, which can help ensure that everyone is aligned and working towards the same goals. With its emphasis on flexibility, adaptability, and continuous improvement, Scrum can help the institution deliver high-quality software that meets changing needs and expectations

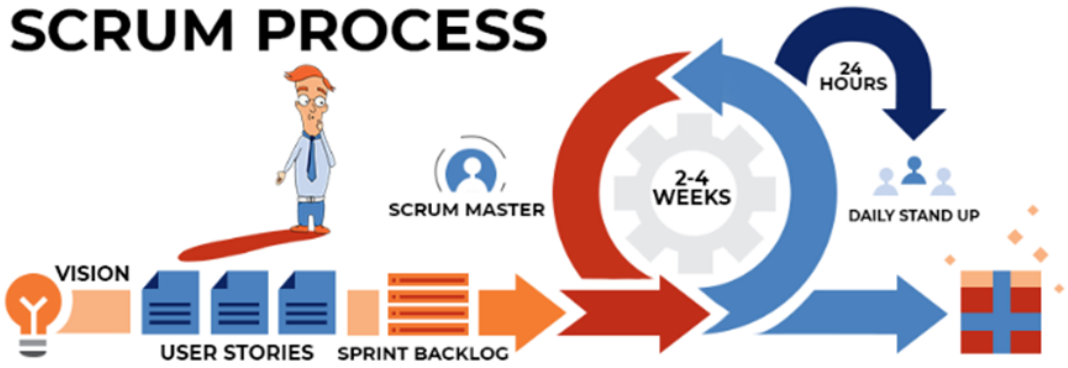

Figure 5.1: Software Development Methodology

5.2 Functional Requirement

5.2.1 Create Role

<table>
<tr><td colspan="3">Name of the function: Create and update user role</td></tr>
<tr><td colspan="3">Description: Insert user role name, access level. Here we have severed user role, like Super Admin, Admin, Dean, Head, Faculty, Executive (School Manager, Registrar Office, Admission, Controller, Library, CITS, and Account).</td></tr>
<tr><td colspan="3">Process description: After submitting the user role, role id will be auto-generated and the user access level should be Super Admin >Admin>Dean >Head>Faculty & Executive. Our highest access level is 10 and the lowest level is 1. Successfully save enable the role for the update.</td></tr>
<tr><td>Input:
1. Role Name
2. Access Level
3. Set user name</td><td>Process:
1. Insert role name and access level
2. Check validation
3. Click Save
4. Create auto role id, get the system date and save in database
5. After save successfully give a success message or give an exception for error.
6. Show role name in the table
7. Now role is ready to edit. If needed, click on the role for edit, save button text would be “Update”, edit name or access level and click the update button.
8. Success update or give an error message.</td><td>Output:
iRAS user roles</td></tr>
<tr><td colspan="3">Alternative option:</td></tr>
<tr><td colspan="3">Post Condition: After success save and show the role in the role list</td></tr>
</table>

5.2.2 Create User

Name of the function: Create User		
Description: Give all basic information, send to the registrar office for approval and validate the email of the user.		
Process description: Account give a user Id for all employee and student Id create on admission time. Insert all basic information against the user id if already not exist in the iRAS (for student all information will load from data). After save successfully, the user end waiting for approval. The registrar office set the user role against designation and give approval. The user will get an email with iRAS approval link. The user will click on the link user will be approved for iRAS. Users must active the user within 24 hours. After 24 hours the link would be invalid.		
Input: 1. User Id 2. User Name 3. Designation 4. Date of Birth 5. Email 6. Mobile 7. Present Address 8. Permanent Address 9. Blood Group	**Process:** 1. Input user Id 2. If already exist the user information, load the information and fill up the field 3. If not exist, give a User name, designation, Date of birth, Blood group, Mobile Number, Home Phone, email. 4. After save successfully give a success message or give an exception for error. 5. Show user in the user list and registrar office approval waiting list 6. Set the user role against the designation and give approval and an auto-generated email sent to the user email. 7. Click on the iRAS user validation link. 8. User redirect to the password setup page and set the password. 9. After successfully saving the user redirects to the user profile page.	**Output:** iRAS user.
Alternative option:		
Post Condition: Successfully set up the user and the user are now able to edit his/her profile. Get access to the IRAS.		

5.2.3 Course Setup

Name of the function: Setup Courses		
Description: Create approved courses list by UGC for the university		
Process description: Give courses basic information and save the information in the database.		
Input: 1. Courses Id 2. Courses Title 3. Credit hour 4. Course code 5. Approved for School	**Process:** 1. Give Course Id, the Course title, credit hour and school ID for that course 2. Click the save button 3. After successfully save give the success message	**Output:** University courses list
Alternative option: If fail to save from web page or causes list is very big, admin can give upload the causes list from the excel sheet.		
Post Condition: Approved courses list		

5.2.4 Course fee setup

Name of the function: Course fee setup		
Description: Setup all fee bill against program id		
Process description: Set per credit hour bill against fee type		
Input: 1. Fee type 2. Per credit hour bill 3. Admission bill 4. Late admission bill 5. Semester fee 6. Graduation fee 7. Late registration percentage 8. Course Waiver fee 9. Remarks	**Process:** 1. Select fee type 2. Select Program 3. Give all necessary information 4. Save 5. Show in the fee list 6. If, want to update select the fee type 7. Load existing data and fill-up all field 8. Which field want to change, update on that field. 9. Update the bill	**Output:** Approved fee list
Post Condition: After save go the fee list table and show the updated fee list		

5.2.5 Create Catalog

Name of the function: Create Catalog		
Pre-Condition: Course list, Course type, Course group		
Description: Create a school/department wise Foundation catalog and major wise core catalog. In the core, the catalog contains student core and focus courses.		
Process description: Set per credit hour bill against fee type		
Input: 1. Catalog name 2. Catalog type 3. School Id 4. Course Id 5. Course type 6. Course group 7. Course prerequisites 8. Requirement for graduation	**Process:** 1. Select catalog name 2. Select course name 3. Select course type, course group 4. If pre-requisite exists to give the prerequisite list 5. Save 6. Give successful message 7. To complete the catalog repeated step 1 to 6. 8. To update any catalog, select the catalog and load existing data and update the field. Click the Update button.	**Output:** Catalog
Alternative option:		
Post Condition: After finish, the catalog goes to the catalog list.		

5.2.6 Setup Catalog Requirement

Name of the function: Catalog requirement setup		
Pre-Condition: Catalog list, Course group		
Description: The graduation requirement breaks down. Course group-wise requirements summation should be the same as the graduation requirements.		
Process description: Set course group-wise requirement.		
Input: 1. Catalog 2. Course group 3. Minimum requirement 4. Requirement type (optional or mandatory)	**Process:** 1. Select catalog 2. Select course group 3. Select requirement type 4. Give minimum and maximum requirement 5. Save	**Output:** Catalog requirements
Post Condition: After saving go to catalog requirement table show course group-wise requirement		

5.2.7 Course fee setup

Name of the function: Course fee setup		
Description: Setup all fee bill against program id		
Process description: Set per credit hour bill against fee type		
Input: 1. Fee type 2. Per credit hour bill 3. Admission bill 4. Late admission bill 5. Semester fee 6. Graduation fee 7. Late registration percentage 8. Course Waiver fee 9. Remarks	**Process:** 1. Select fee type 2. Select Program 3. Give all necessary information 4. Save 5. Show in the fee list 6. If, want to update select the fee type 7. Load existing data and fill-up all field 8. Which field want to change, update on that field. 9. Update the bill	**Output:** Approved fee list
Post Condition: After save go the fee list table and show the updated fee list		

5.2.8 Course fee setup

Name of the function: Course fee setup		
Description: Setup all fee bill against program id		
Process description: Set per credit hour bill against fee type		
Input: 1. Fee type 2. Per credit hour bill 3. Admission bill 4. Late admission bill 5. Semester fee 6. Graduation fee 7. Late registration percentage 8. Course Waiver fee 9. Remarks	**Process:** 1. Select fee type 2. Select Program 3. Give all necessary information 4. Save 5. Show in the fee list 6. If, want to update select the fee type 7. Load existing data and fill-up all field 8. Which field want to change, update on that field. 9. Update the bill	**Output:** Approved fee list
Post Condition: After save go the fee list table and show the updated fee list		

5.2.9 Semester Course Offer

Name of the function: Semester courses offer		
Description: Semester wise offer courses. The student can registrar only these courses. The school is offering this course.		
Process description: select courses, give all mandatory data and save the offer courses. If you want the Co offer course gives the co-offer list against the offer course id.		
Input: 1. Course Id 2. Co offer courses list 3. Primary Faculty 4. Secondary Faculty 5. Course section 6. Classroom 7. Class start and end time 8. Remarks	**Process:** 1. Select course 2. Select faculty, Classroom, class start and end time. 3. If class time conflict gives a message 4. Set section and class capacity 5. If the course has co-offered course, select the courses list 6. Save 7. For updates select the course, existing information will show and the same as saving we can update.	**Output:** Offer Course list, Tally sheet, Student wise update offer list in the dashboard.
Alternative option: If fail to save give the error message		
Post Condition: After save go the fee list table and show the updated offer list.		

5.2.10 Setup Admission Requirement

Name of the function: Setup admission requirement		
Description: Set the admission requirements. If the student full -fill the requirement then he/she will be allowed to submit the application form.		
Process description: Set all requirements and save. After saving it will appear for updating.		
Input: 1. 10- and 12-years degree minimum result requirements. 2. SAT and TOEFL requirement for exemption from admission test	**Process:** 1. Give requirement name (Like HSC minimum GPA) 2. Give minimum requirements (like GPA: 3.5) 3. Give Code of the requirement 4. Save the requirement	**Output:** Requirements for the application
Alternative option: If fail to save go to the requirement list and give the message		
Post Condition: Store application requirement and show the in the requirement list		

5.2.11 Setup Calendar

Name of the function: Setup Calendar		
Description: Admission calendar setup		
Process description: Set Admission test number, set admission test date, last date of form submission, etc. and save. After save if you want to edit, click the edit button and update the calendar.		
Input: 1. Admission test no 2. Last date of form submission 3. Admission test date 4. Result publishes and registration date 5. Orientation class commence date	**Process:** 1. Set all mandatory data 2. Click Save button 3. If valid the form input saves and give success message	**Output:** Admission Calendar
Post Condition: After save go to the calendar page and show admission calendar		

5.2.12 Create User

Name of the function: Create User		
Description: For application, everyone must create his/her user in iRAS. Give only Name, Email, contact number Date of birth and create a user.		
Pre-Condition: Click Apply button, user create option will show. Give the necessary information and create a user.		
Input: 1. First Name, Last Name 2. Email and Cell Phone 3. Password, Confirm password 4. Date of Birth	**Process:** 1. Input all mandatory data 2. Compare password and confirm password 3. If want to show the password, click show password option 4. Click create user button 5. Give success message or error message	**Output:** iRAS user
Alternative option: If already existing user, click the login link and go to the login page. Give the user name and log in to the admission form.		
Post Condition: Create user and go to user basic information page or admission dashboard page.		

5.2.13 Login to Application Form

Name of the function: Login to The Admission Form		
Description: User login in the admission form,		
Process description: Give Email and password and login to the admission form.		
Input: 1. Email 2. Password	**Process:** 1. Give email and password 2. Click Login button 3. If match email, password combination then logs in. 4. If not match give proper message	**Output:** Login in the Admission form,
Alternative option: If not existing user, click the create user link and go to create a user page. Create users and login to the admission form.		
Post Condition: Login and go to the user basic information page or admission dashboard page.		

5.2.14 Redirect Page after Login

Name of the function: Redirect to Dashboard or Basic Information page		
Pre-Condition: Load admission calendar information		
Description: If the application date is over and still user didn't complete his/her degree plan information, redirect users to Dashboard. If the date is not over and the degree plan is already fill up go to a profile page, otherwise go to the user basic information.		
Process description: After login, Check the application date is over or not and application form fillip. Base on status redirect the page.		
Input: 1. User Id 2. Form fill-up Status 3. Form open - close status	**Process:** 1. Load current form fill up status 2. Check if the form is not fill up check to the last date of form submission 3. If the date is over the re-direct to the dashboard page 4. If the date is not over and form is not fill up redirect to personal information page otherwise send to a profile page.	**Output:** User is now on admission dashboard or personal information page or profile page
Alternative option:		
Post Condition: Page is now a dashboard or personal information page or profile page.		

5.2.15 Admission Dashboard

Name of the function: Admission dashboard		
Description: Show application information and give a message about the form submission date		
Process description: Load Application information and show		
Input:	Process: 1. Load application information 2. Give message	Output: Application Information
Alternative option:		
Post Condition:		

5.2.16 Upload Photo

<table>
<tr><td colspan="3">Name of the function: Upload Photo</td></tr>
<tr><td colspan="3">Description: Upload Student Photo for admission</td></tr>
<tr><td colspan="3">Process description: Click the upload photo button, Photo popup will appear, select and upload the photo.</td></tr>
<tr><td>Input:
1. Student Email
2. User Id
3. Photo Upload Status
4. Student Photo
5. Application Year
6. Application Semester</td><td>Process:
1. Click Upload Photo
2. Photo Popup will appear
3. Select Photo
4. If needed resize, resize the photo
5. Click Save button
6. If want to close the popup click the close button</td><td>Output:
Uploaded Student photo</td></tr>
<tr><td colspan="3">Post Condition: Changes the photo inside and nav bar and student Photo option</td></tr>
</table>

5.2.17 Personal Information

<table>
<tr><td colspan="3">Name of the function: Personal Information (Step 1)</td></tr>
<tr><td colspan="3">Description: Save parent's information, present address, permanent address. If a student is not Bangladeshi save passport information as well. Check the attached document.</td></tr>
<tr><td colspan="3">Process description: Input parent information, present address, permanent address. Check Bangladeshi or not. If not, Bangladesh shows a foreign student field (Passport number, Birth country). Click the "next to academic" button.</td></tr>
<tr><td>Input:
1. Father Name, Mother Name
2. Blood group
3. Guardian phone, email
4. Nationality status
5. Present address (Home/village, District, Police Station, Post office)
6. Permanent address (Home/village, District, Police Station, Post office)
7. Country of the citizen, country of birth, Passport number, country of passport issuance, Dual citizenship status and visa number</td><td>Process:
1. Select blood group
2. Fill up Father and Mother first and last name
3. Give guardian phone number and email
4. Select Nationality (Are You Bangladeshi?)
5. If Bangladesh, visible present and permanent address section
6. Fill up present and permanent village/house, Road, Block, Sector
7. Select district and filter Upazilla/ Police Station and Post office
8. Select Police Station and Post office
9. Click Save and Next to Academic
10. Check validation, if valid go to the next step otherwise give validation error and indicate on that field</td><td>Output:
Candidate Personal information</td></tr>
<tr><td colspan="3">Alternative option: If validation fails, stay on that step and indicate the invalid field</td></tr>
<tr><td colspan="3">Post Condition: Go to the academic information page</td></tr>
</table>

5.2.18 Academic Information

Name of the function: Update Personal and Academic		
Description: Students are allowed to update his/her personal and academic information before a paid admission bill.		
Process description: Show the update profile button for the unpaid student. Click the button and redirect the page to step 1 (personal information). Same as saving personal information to update the information if needed then click "Save and next to the academic" button. Then go step 2 and complete the update.		
Input: 1. User Id 2. Updated Personal Information 3. Update Academic information	**Process:** 1. Click the Update profile button 2. Go to step 1 (Personal information) 3. Load personal information and fill up on those filled 4. Correct which field want to update 5. Click Save and next to the academic link 6. Go to step 2 (academic information) 7. Corrected which field want to update 8. Save and finish	**Output:** Save user wise photo and show photo on the menu
Post Condition: Successfully save got the profile page.		

5.2.19 Update Information

Name of the function: Update Personal and Academic		
Description: Students are allowed to update his/her personal and academic information before a paid admission bill.		
Process description: Show the update profile button for the unpaid student. Click the button and redirect the page to step 1 (personal information). Same as saving personal information to update the information if needed then click "Save and next to the academic" button. Then go step 2 and complete the update.		
Input: 1. User Id 2. Updated Personal Information 3. Update Academic information	**Process:** 1. Click the Update profile button 2. Go to step 1 (Personal information) 3. Load personal information and fill up on those filled 4. Correct which field want to update 5. Click Save and next to the academic link 6. Go to step 2 (academic information) 7. Corrected which field want to update 8. Save and finish	**Output:** Save user wise photo and show photo on the menu
Post Condition: Successfully save got the profile page.		

5.2.20 Pay Admission Bill (Using SSL Commerz)

Name of the function: Pay Admission Bill (using SSL Commerz)		
Description: Click Pay online. Pay admission bill by SSLCommerz		
Process description: Click Pay online. Go to the SSLCommerz site. Select the payment method and pay the admission bill. After successful payment update the payment status. Disable the edit option. Show the print button. Application information taken from the admission system.		
Input: 1. Application No 2. Payment Amount 3. Store Id 4. Payment Method 5. Admission Year 6. Admission Semester	**Process:** 1. Click Pay online 2. Go to the SSLCommerz site 3. Select Payment method 4. Pay the bill 5. Auto back to the application form's profile page. 6. Give payment success message/ receive message 7. Show Print button and hide the update profile button	**Output:** Paid Candidate
Alternative option: Student Can Pay Using bKash USSD		
Post Condition: Enable print button and update payment information of the candidate.		

5.2.21 Pay Admission bill (using bKash USSD)

Name of the function: Pay Admission Bill (using bKash USSD)		
Description: Take Admission Id and Pay by bKash USSD		
Process description: Pay by bKash USSD		
Input: 1. Application No 2. Payment Amount	**Process:** 1. Using Application, No. as reference number in bKash, Pay 1000 TK 2. After pay within 30 s update payment status in student profile 3. Enable print admit card button	**Output:** Paid Candidate
Alternative option: Student Can Pay Using SSL Commerz		
Post Condition: Enable print button and update payment information of the candidate.		

5.2.22 Seat Plan

Name of the function: Create Seat plan		
Pre-Condition: Exam Hall, Exam type, Seat capacity setup. Major wise exam type setup.		
Description: After payment auto-generate admission seat plan. After payment, students major wise find exam type. Load exam type-wise exam hall number. The exam hall will fill up sequentially. When any exam type hall will be fill-up, the student would go to under reserve hall. It is fully system generate. The seat plan would be only for the exam venue Dhaka IUB campus.		
Process description: Full process done by the back end and it is system generate seat plan. All input takes from the system.		
Input: 1. Application No 2. Exam hall no Application Year 3. Application Semester 4. Admission Test No 5. Student major 6. Exam Hall Id 7. Exam Type	**Process:** 1. Load student majoring wise exam type. 2. Load exam type-wise exam hall. 3. Insert application year, semester, application test no, hall and application no.	**Output:** Admission seat plan.
Alternative option: Manually create seat plan.		
Post Condition:		

5.2.23 Print Admit Card

Name of the function: Print Seat plan		
Pre-Condition: Create a system to generate a seat plan.		
Description: Exam type and exam hall wise seat plan create. Where a student photo, major and the signature option would be included.		
Process description: Click the print exam seat plan. Download PDF formatted seat plan.		
Input: 1. Admission Year 2. Admission Semester 3. Admission Teat No	**Process:** 1. Click the download seat plan 2. The download would start and downloaded.	**Output :** Seat plan.
Alternative option: Manually create a seat plan.		
Post Condition: Download Seat Plan		

5.2.24 Print Seat Plan

Name of the function: Print Seat plan		
Pre-Condition: Create a system to generate a seat plan.		
Description: Exam type and exam hall wise seat plan create. Where a student photo, major and the signature option would be included.		
Process description: Click the print exam seat plan. Download PDF formatted seat plan.		
Input: 1. Admission Year 2. Admission Semester 3. Admission Teat No	**Process:** 1. Click the download seat plan 2. The download would start and downloaded.	**Output :** Seat plan.
Alternative option: Manually create a seat plan.		
Post Condition: Download Seat Plan		

5.2.25 Create Admission test result

Name of the function: Create Admission test result		
Pre-Condition: Upload admission test marks		
Description: Give different parameters and create result and save the result		
Process description: Set parameter to create result, Click create result and show the result. After finally select the result save in the database.		
Input: 1. Admission Year 2. Admission Semester 3. Admission Test No 4. Select medium 5. Subject wise minimum marks 6. Board 7. SSC and HSC result range 8. School and Major 9. A-Level GPA 10. O-Level GPA	**Process:** 1. Select and set the parameter 2. Click Create Result 3. Save the result 4. Export the result in PDF and Excel sheet 5. Send for approval	**Output:** Admission Test Result
Alternative option: Manually upload exam test number		
Post Condition: Show number of selected students on administration dashboard		

5.2.26 Admission

Name of the function: Admission		
Pre-Condition: Registration data setup and Check Registration validity		
Description: Student Id wise check the student admission. If the student is still not admitted, then admission year and semester wise complete the student admission. Create Student admission code and set student major, minor, foundation and its corresponding catalog.		
Process description: After login in iRAS go to the registration page. If a student has no admission, give a message to take admission, populate popup and confirm student admission. Without students, the user will give student id then load the student registration information and complete the next process.		
Input: 1. Student Id 2. Admission year 3. Admission Semester 4. School Id 5. Major Id 6. Major Catalog 7. Foundation Catalog 8. Admission code 9. Active Status 10. Set User 11. Set Date	**Process:** 1. Other than the student group user, give the student Id and click the load button. 2. Open the student admission popup. 3. Click the admission button 4. Give a success message	**Output:** Get an admitted student
Alternative option:		
Post Condition: Show the success message and show the print admission bill button.		

5.2.27 Courses Waiver

Name of the function: Courses waiver		
Pre-Condition: Admission		
Description: If the student has a course waiver show the courses list in the popup with course information. Click the causes waiver button. Complete the course waiver.		
Process description: Show the waiver		
Input: 1. Student Id 2. Waive Course 3. Course bill 4. Set User 5. Set Date	**Process:** 1. Click the download admission bill button 2. Check the waiver courses and bill 3. Confirm the waiver button 4. Start to download the admission bill.	**Output:** Waiver done
Alternative option: Avoid to courses waiver, apply for cancel the waiver		
Post Condition: After the download, close the popup.		

5.2.28 Check Course Advising Validity

Name of the function: Course advising validity		
Pre-Condition: Registration data setup		
Description: Check the student is valid for registration or admission for student group go to check the validation and for others group go show student id field and load option. Give student Id and click the load button. Check the validity.		
Process description: Without student group show student id and load button. Give student Id and click load. Check the validity. If not valid give proper message, why invalid.		
Input: 1. Student Id 2. Registration date 3. Registration Permission 4. Add/drop Permission	**Process:** 1. For, without student group show Student Id filed and Load button 2. Give student Id 3. Click load (for student start auto load) 4. Check has student data in registration data setup field 5. If no data in registration data setup table 6. Check student is new 7. Check student is English medium in pre admission table if yes give message for take pre admission from admission office 8. Check all permission base on registration type 9. If all is okay set eligible to true other wise set false and give proper message	**Output:** Validity for registration
Post Condition: Start loading registration data		

5.2.29 Load valid offer course

Name of the function: Load valid courses		
Pre-Condition: Courses advising validity		
Description: Load only valid course form student catalog. Removes unnecessary courses.		
Process description: Selected catalog courses and remove necessary courses.		
Input: 1. Student Id 2. Done courses 3. Done registration year and semester	**Process:** 1. Select catalog courses which has done pre-requisite or has no any pre-requisite 2. Remove courses which complete before 4 semesters ago 3. Remove courses which has grade A 4. Return these valid courses	**Output:** Valid courses for registration
Post Condition: Show the courses on the registration pages.		

5.2.30 Show basic information for registration

Name of the function: Show basic information for registration		
Pre-Condition: Setup registration data and registration validity checkup		
Description: Show student financial aid, student name, major, foundation, major, core and minor done and minimum credit.		
Process description: Load and show student basic registration data.		
Input: 1. Student Id 2. Registration Year 3. Registration Semester	**Process:** 1. Load basic information 2. Show financial aid percentage and minimum credit for financial aid. 3. Show foundation, major, core and minor done and minimum credit 4. Show mandatory fail course 5. Show mandatory courses for this semester	**Output:** Show valid courses
Post Condition: Show basic information		

5.2.31 Select courses for registration

Name of the function: Select courses for registration		
Pre-Condition: Load valid courses		
Description: Check courses for registration		
Process description: Check course, check validity if valid then checked the course otherwise give message for invalidity and keep unchecked.		
Input: 1. Student Id 2. Courses Id 3. Section 4. Capacity 5. Enrolled 6. Class time	**Process:** **9.** Check course **10.** Check capacity is available **11.** Check time conflict 12. Check same courses is not selected 13. Check pair course take same section 14. Check mandatory fail courses is taken 15. When check drop courses for dropping, checking is it pair courses or not 16. Show courses list and total added credit hours top on the courses list	**Output:** Selected valid courses
Post Condition: Show selected courses		

5.2.32 Show confirmation popup

Name of the function: Show confirmation popup		
Pre-Condition: Selected courses for registration		
Description: Show selected courses and credit hours in popup		
Process description: Click registration and show popup		
Input: 1. Course Id 2. Section 3. Class time 4. Credit hours	**Process:** 1. Click register button 2. Show popup 3. Show course summery for add and drop 4. Show total credit hour 5. Show financial aid message for minimum credit 6. Click Confirm	**Output:** Start registration
Post Condition: Close the popup and show spinner		

5.2.33 Save advising courses

Name of the function: Save advising courses		
Pre-Condition: Selected courses for registration and submit confirm for registration		
Description: Save selected courses and create bill for registration		
Process description: After checking save the selected courses information.		
Input: 1. Selected courses list 2. Sections of courses 3. Course code and credit hours 4. Courses advisor id 5. Registration code 6. Initial course grade 7. Registration type 8. Late registration status 9. Set user 10. Set date	**Process:** 1. Save selected courses list with all information 2. Create registration code 3. Call Create bill process	**Output:** Store selected courses
Alter native option: If has special permission can overwrite this checking.		
Post Condition: Update enrolled for the courses (do it section wise)		

5.2.34 Load registered courses

Name of the function: Load registrar course		
Description: load the advising courses after give success message and show on the right-side box		
Process description: Show saving success message, load the courses and show right side table.		
Input: 1. Student id 2. Registration year and semester	**Process:** 1. Load course 2. Show right side table 3. Set disables for selection 4. Show success message	**Output:** Show registrar course
Post Condition: Show courses list		

5.2.35 Pay registration bill (Online)

Name of the function: Pay online registration bill		
Description: Pay registration bill through SSL Commerz		
Process description: Click online bill, go to SSL Commerz and pay		
Input: 1. Registration bill 2. Student Id 3. Bill Id	**Process:** **1.** Click online payment **2.** Go to SSL Commerz site **3.** Select payment method **4.** Pay the bill amount **5.** Go back to the bill page **6.** Update payment status	**Output:** Pay registration bill
Alter native option: Pay the bill in the selected bank (MTB and Bank Asia)		
Post Condition: Update payment status		

5.2.36 Load Course for attendance

Name of the Function: Valid Course Load		
Description: At the beginning of the semester department head search for available courses and allocate them for faculty.		
Pre-condition: 1. Valid faculty ID 2. Course list 3. Valid semester and year		
Input: 1. Faculty ID 2. Semester 3. Year 4. Attendance	**Process:** 1. Click load process 2. Show course list 3. If error, no data show	**Output:**
Alternate Options: 1. Manually check available courses 2. Assign them as per requirements		
Post condition: 1. Offered courses will be shown in iRAS for registration		

5.2.37 Class attendance

Name of the Function: Class Attendance		
Description: To take student attendance select particular course. Check the students who are present. Submit the primary list. If late attendance taken that will be final list, otherwise automatically it will be the final one.		
Pre-condition: 1. Faculty ID 2. Class time 3. Students have to be registered in the course 4. Separate attendance for individual course		
Input: 1. Student ID 2. Faculty ID 3. Course ID 4. Semester and Year 5. Select category	**Process:** 1. Select course 2. Select categories like class, midterm, final. 3. Check the attendant 4. Submit the list 5. It can be taken again within class time for late attendance.	**Output:** 1. Get present student list.
Alternate Options: 1. Have to take attendance manually with paper		
Post condition: 1. Course automatic withdrawal for undergrad students' can be possible 2. Attendance marking		

5.2.38 Course withdraws

Name of the Function: Course Withdraw		
Description: Student can withdraw course(s) from registered course list before midterm. Course bill be adjusted with next semester bill.		
Pre-condition: 1. Within deadline it has to be done		
Input: 1. Student ID 2. Semester and Year 3. Course ID/Name	**Process:** 1. Go to Registration tab 2. Select withdraw 3. Check course 4. Submit	**Output:** 1. Course removed from student profile
Alternate Options: 1. Apply for course withdrawal at registrar office		
Post condition: 1. Student will no more for responsible for that course 2. Account office will adjust the course bill at next semester		

5.2.39 Grade submission

Name of the Function: Grade Submission		
Description: Submit student grade		
Pre-condition: 1.		
Input: 1. Faculty ID 2. Course ID and name 3. Student ID 4. Grade	**Process:** 1. Select grade submission from reports tab 2. Courses list will be shown 3. Select course 4. Assign grade for students 5. Submit	**Output:** 1. Course result for students
Alternate Options: 1. Faculty give grade in a sheet of student list 2. Submit it to registrar office		
Post condition: 1. Student will get result 2. Different action for different result can be taken like 'F' grade students will retake course, 'I' grade holder can continue next time without paying.		

5.3 Non functional Requirements

The primary functions of iRAS are aimed to empower our students, faculty, and administrators to achieve their academic and institutional goals while ensuring the integrity, confidentiality, and security of all academic records.

- Security: Only authorized users can access the system with a username and password.

  i) User Identification: The system requires the user to identify himself/herself using

login credentials
ii)Modification: Super admin can monitor any modification (insert, update, delete) for the database and will be synchronized with timestamp log history
iii) Employee Rights: The employee can see all the information but the super admin allocates the responsibilities and accessibility of them.
iv) Administrator Rights: Super admin will be able to view and modify all the information. Admin will manage different directories like accounts, library, etc. and other employees will be accountable to them.

- Performance: Easy tracking of records and updating can be done.
  i) Response Time: Proposed iRAS will give responses in 1 Second after checking the individual information
  ii)Capacity: New iRAS must support 1000 people at a time.
  iii) Conformity: The system must conform maintaining the integrity of academic policies and the student information system.
  iv) Efficiency: According to the test result, system is working accurately. Efficiency has more increased due to versioning.

- User Friendly: The System is very interactive.
  i) Adaptability: The UX and UI are arranged in such a way to give the user the right direction for certain quarries.
  ii) Readability: The system is designed according to the working criteria of the users of the existing system. Students and faculty get all necessary information like attendance, course info, registration etc. through a dashboard.

- Maintainability: Backups for database are available.
  i) Back Up: The system backs up all necessary data twice in a day.
  ii)Errors: As it is an in-house product, so quick response for fixing bugs, modifications will be attainable.
  iii) Modularity: The whole system is divided into several modules like configuration, admission, for proficient management.

- Re usability: For any type of future development, code and feature all are reusable.
  i)Portability: The system will work in multi station.
  ii) Usability: It can be implemented on windows platform. So, it got the highest usability

5.4 System Requirement (Hardware, Software, Architecture, User)

5.4.1 Hardware Requirements

Azure App Service Plan

- Choose Premium v3 (P2mv3) pricing tier.
- Configure the service plan with 8 cores, 64 GB RAM, and 250 GB storage space.

Azure Web App (App Service)

- Deploy .NET Core 7 backend and Angular client application on an Azure Web App.
- Configure the Web App to use the created Premium v3 Service Plan.

5.4.2 Software Requirements

Server Software

- Operating System for development: Windows Server compatible with ASP.NET Core and Node.js.
- Web Server for development: Nginx or Apache for serving front-end assets.
- ASP.NET Core: The latest stable version for backend API development.
- Node.js: Required for server-side scripting and building APIs.
- Oracle Database: Oracle 19c or later for data storage.
- Database Management System: Oracle Database Management System (DBMS) tools for administration.

Client Software

- Web Browsers: Compatible modern web browsers (Chrome, Firefox, Edge, Safari).
- Front-end Frameworks: Angular, React, SolidJs.
- API Client: Tools like Postman for testing and debugging API endpoints.

5.4.3 Architecture Requirements

Architecture

- Microservices Architecture: Consider a microservices-based architecture for scalability and modularity.
- RESTful API Design: Implement RESTful API design principles for efficiency and interoperability.
- Single Page Application (SPA): Develop a responsive and dynamic SPA architecture using Angular, React, or SolidJs.

5.4.4 Deployment:

Deploy .NET Core Backend and Angular Client:

- Deploy your .NET Core 7 backend and Angular client to the Azure Web App using your preferred deployment method (Azure DevOps, GitHub Actions, Visual Studio Deployment, etc.).

Configure Environment Variables:

- Set environment variables in the Azure Web App for configurations such as connection strings, API keys, etc.

5.4.5 User Requirements:

User Access:

- User Types: Define user roles and access levels, such as administrators, faculty, students, and super admins.
- Authentication: Implement secure authentication mechanisms (e.g., JWT, OAuth) for user login and access control.
- User Interface: Create intuitive and user-friendly interfaces for different user types to access and interact with the system.

5.4.6 Compliance and Standards:

Compliance:

- Ensure compliance with data security and privacy regulations (e.g., GDPR, HIPAA, FERPA) relevant to your institution.

- Follow coding standards and best practices for ASP.NET Core, Node.js, and front-end frameworks.

These system requirements encompass hardware, software, architecture, and user-related considerations to ensure the successful development and deployment of your application using ASP.NET Core, Node.js, Angular, React, or SolidJs, and Oracle as the database.

Chapter 6

Logical Design

6.1 Object Oriented Design Using UML

In this UML design [19], I present comprehensive Use Case and Class Diagrams, offering a holistic depiction of the entire IRAS system. The emphasis lies on effective design abstraction management to refine hardware aspects. The seamless integration of this process into UML is facilitated through its extensibility mechanism, ensuring a cohesive representation of the system's intricacies.

6.1.1 Use Case

Use Case [20] highlights the challenge of ensuring clear communication between domain experts and developers during the analysis phase. The examination of UML Use Case diagrams reveals their limited capability to express variability effectively. To address this limitation, we propose an enhancement to the Use Case meta-model. This enhancement introduces new modeling elements that facilitate the explicit expression of identified variability types, providing a more robust solution.

i) Context Level

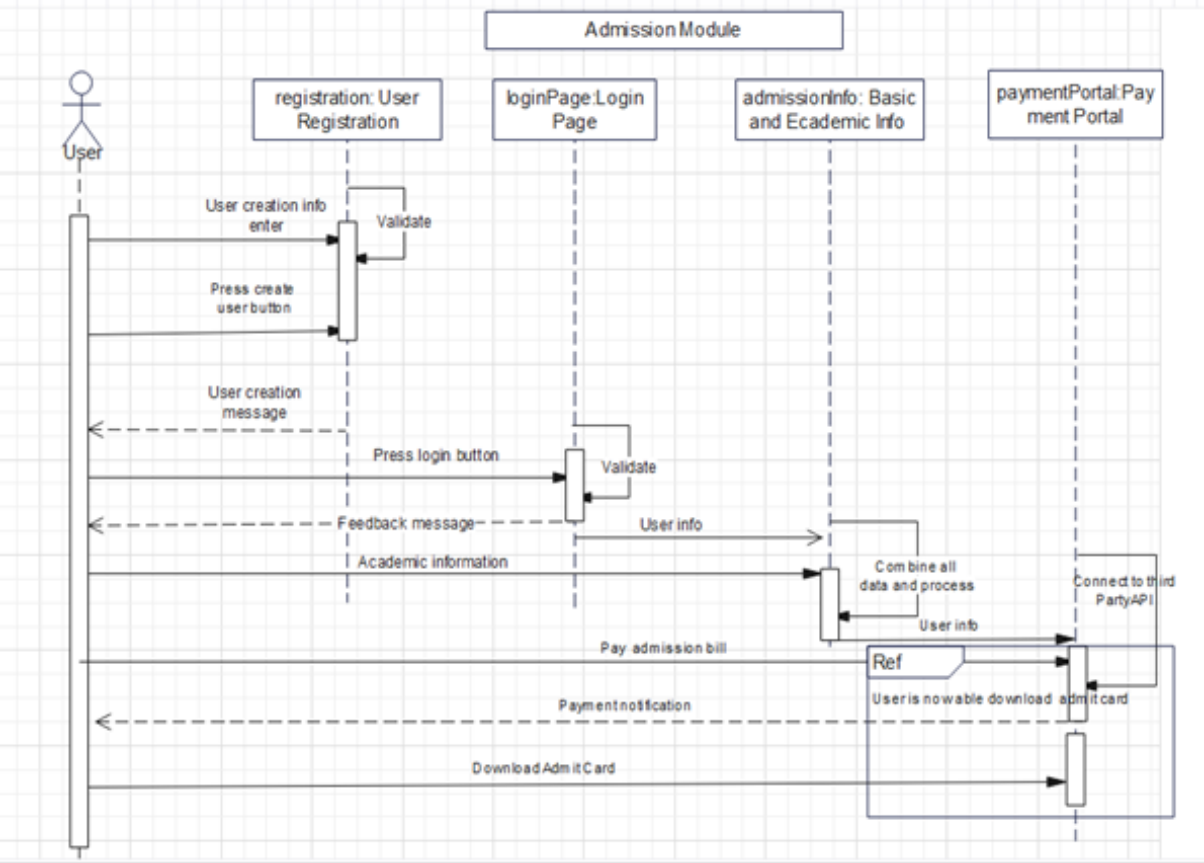

Figure 6.1 : Admission portal

ii) Registration

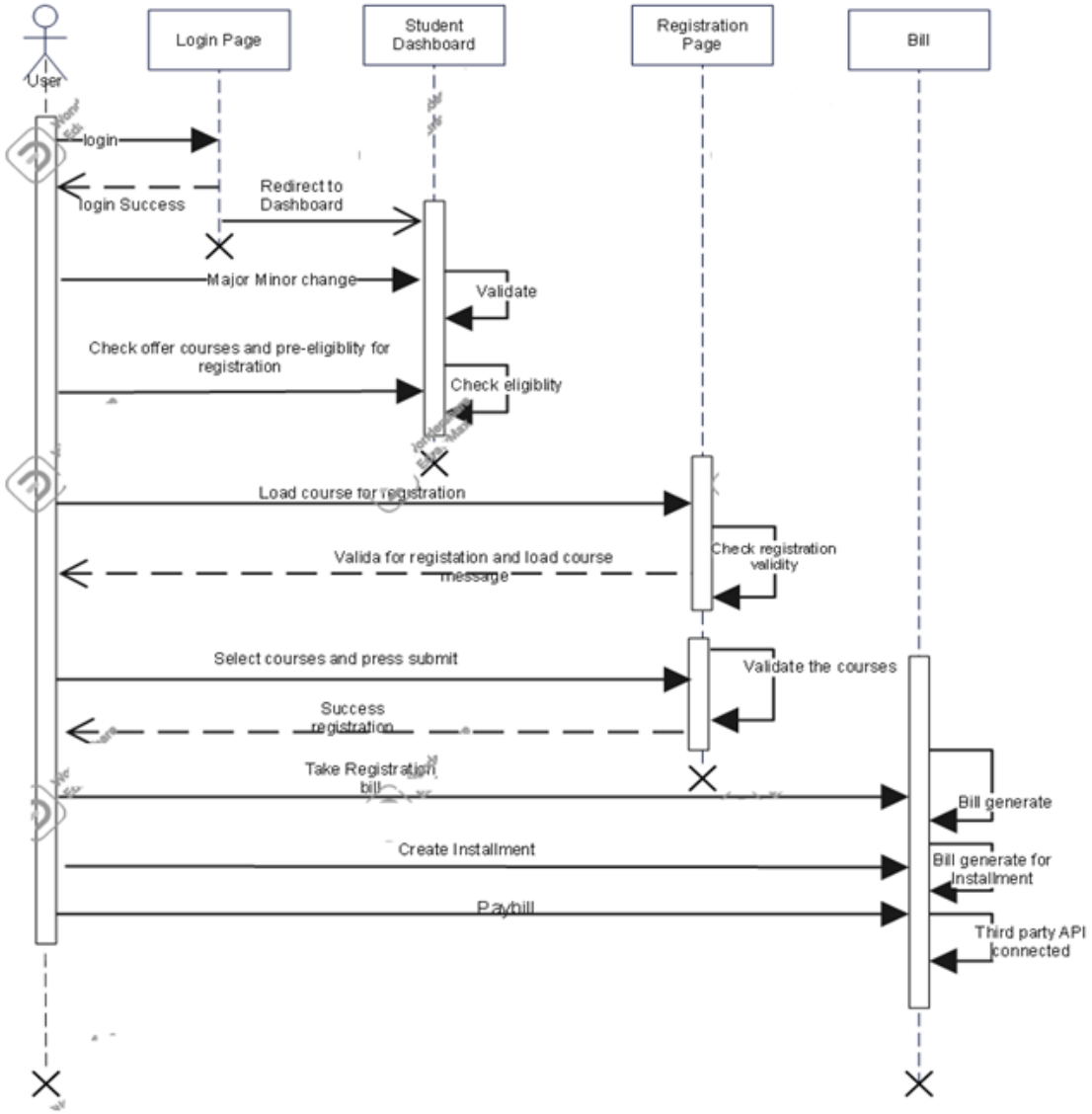

Figure 6.2 : Admission portal

6.2 Class Diagram

Visualizing UML class diagrams [21], emphasizing a balanced mix of aesthetic criteria such as crossing and bend minimization, uniform direction within hierarchies, and more. The proposed approach, implemented in the GoVisual graph drawing library, demonstrates superiority over existing layouts, offering enhanced visualization for class hierarchies, associations, aggregations, and compositions.

i)Admission Portal Class Diagram

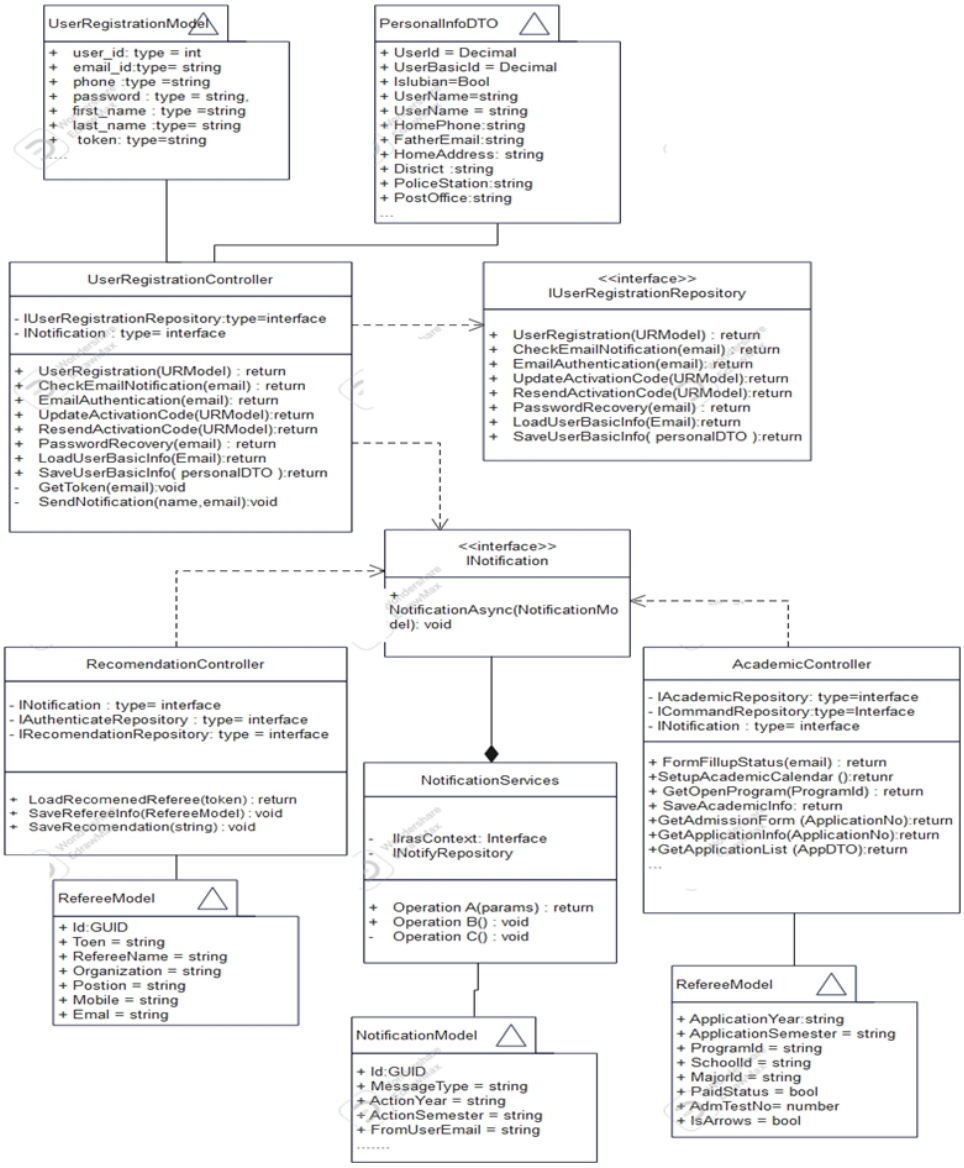

Figure 6.3 : Admission portal

ii)Registration Portal Class Diagram

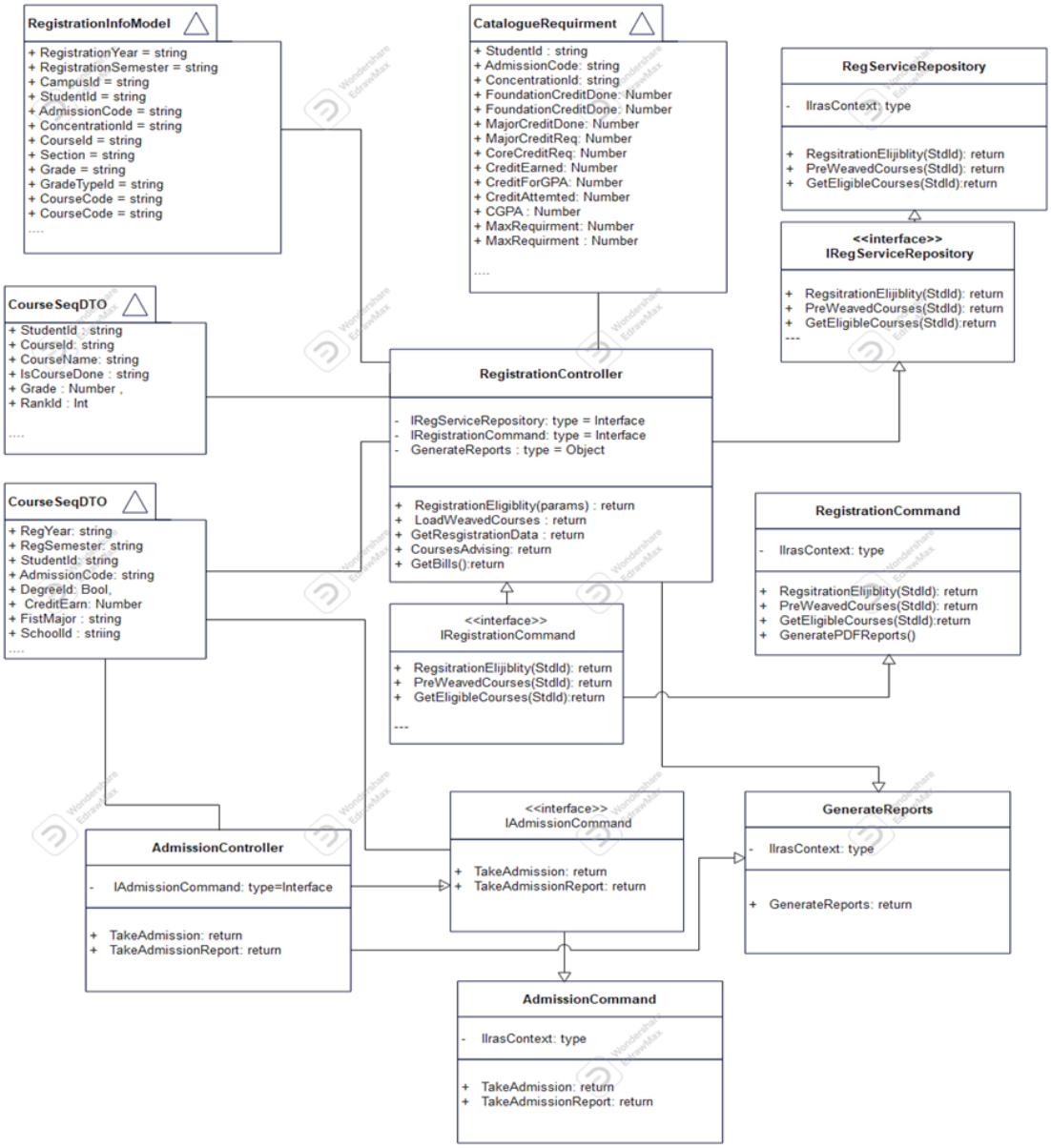

Figure 6.4 : Registration Portal Class Diagram

Chapter 7

Physical Diagram

7.1 Architecture

i)Client Server Architecture

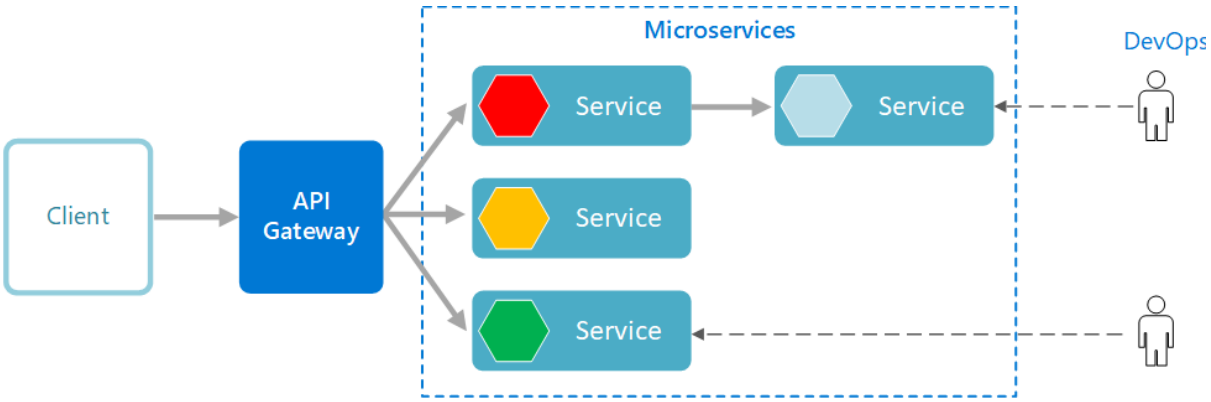

Figure 7.1: Client-server

The architecture of IRAS (Integrated Registration and Admission System) is a client-server model leveraging microservices. At its core, the system follows a distributed approach where clients interact with various microservices deployed on the server-side. These microservices handle specific functionalities such as user authentication, course registration, payment processing, and student data management. Through this modular architecture, IRAS ensures scalability, flexibility, and maintainability, allowing seamless integration of new features and efficient handling of user requests.

7.2 Input Form

7.2.1 Create User

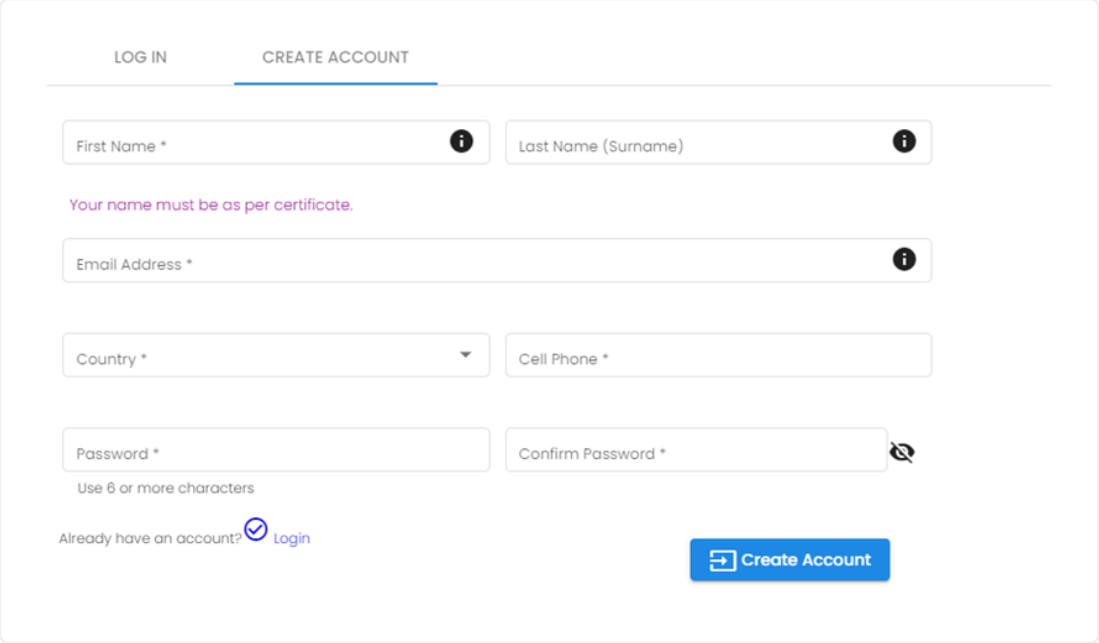

Figure 8.1: Create user

User Information: Input user details such as name, email address, cell phone, country, and password.
Verification Code (Optional): Authentication is enabled, input the verification code received via email or SMS.
Purpose:
User account creation serves the purpose of allowing individuals to register and access the system securely. It establishes user identity and provides necessary credentials for system interaction.

Implementation Method:
User Registration Form: Provide a web-based registration form that prompts users to input their personal information, including name, email, username, and password.
Two-Factor Authentication (Optional): Implement optional two-factor authentication (2FA) to enhance security. Users receive a verification code via email or SMS, which they enter during the registration process to verify their identity.
Control and Flows for Documentation:
User Registration Process: Document the steps involved in user registration, from the user's initial input of information to the system's response.
Two-Factor Authentication (2FA): If 2FA is implemented, document the process of

sending and verifying the verification code.
Error Handling: Describe how the system handles errors, such as duplicate usernames or invalid passwords, and how it informs users about these issues.
Security Measures: Highlight security measures in place to protect user data, including password hashing and encryption.
User Notifications: Document how users are notified of successful registration or any issues encountered during the process

7.2.2 Login

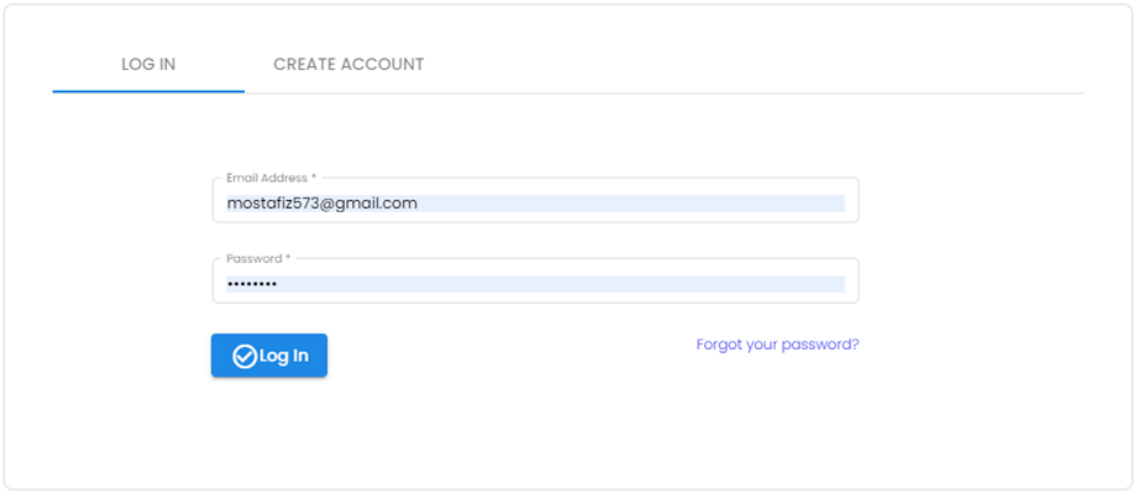

Figure 8.2 : Login

Inputs for Login Page:
Username: User's unique identifier.
Password: User's confidential access code.
Purpose:
The login page serves the purpose of allowing registered users to access the system securely by providing their username and password.

Implementation Method:

- Username and Password Fields: Provide fields for users to enter their username and password.
- Authentication: Authenticate user credentials against stored data to grant or deny access.
- Error Handling: Handle scenarios like incorrect credentials or locked accounts.
- Password Reset (Optional): Optionally, offer a password reset mechanism for users who forget their passwords.

Control and Flows for Documentation:

- Authentication Process: Document the steps involved in user authentication, including checking credentials and granting access.
- Error Handling: Describe how the system handles login errors, such as incorrect username or password, and how it communicates errors to users.
- Password Reset (If Applicable): Document the process for resetting passwords, including user verification.
- Security Measures: Highlight security measures like password hashing and account lockout policies.
- Session Management: Explain how user sessions are managed, including session timeouts and automatic logouts.

7.2.3 Forget Password

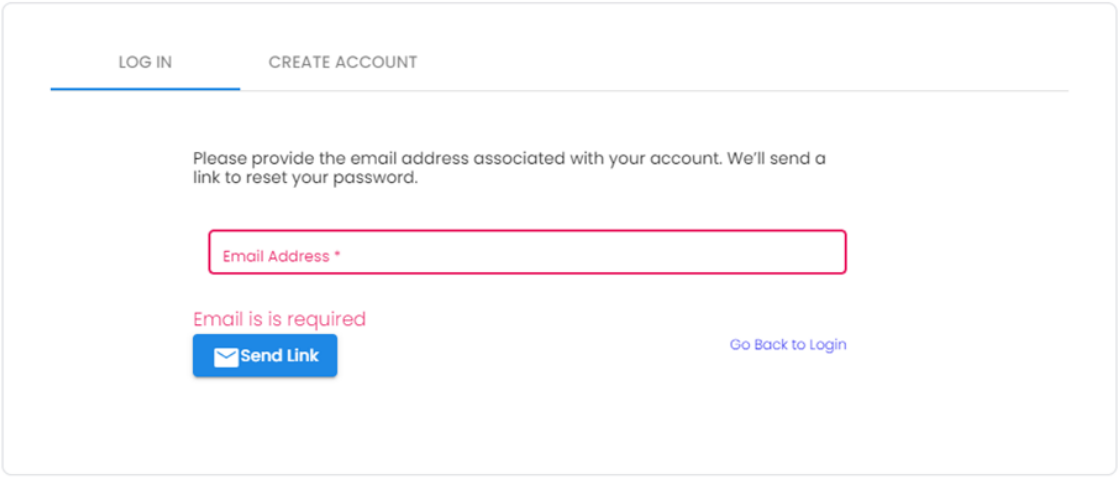

Figure 8.3 : Forget Password

Inputs for Forgot Password:
User ID or Email: User provides either their user ID or the email associated with their account.
Purpose:
The "Forgot Password" feature allows users who have forgotten their password to regain access to their account by receiving a password recovery link via email.

Implementation Method:

- User Input: Provide a field where users can input their user ID or email.
- Submit Request: Include a button or link for users to submit their request.
- Email Confirmation: Upon request submission, send a password recovery link to the user's provided email address.
- Token Generation: Generate a unique, time-limited token or link that allows users to reset their password securely.
- Password Reset Page: Create a web page where users can securely set a new password.

Control and Flows for Documentation:

- Request Submission: Document the process of users submitting their user ID or email for password recovery.
- Token Generation: Describe how the system generates and associates a secure token with the user's request.
- Email Confirmation: Explain how the system sends an email to the user's registered email address, containing the password recovery link.
- Password Reset Page: Detail the steps involved in users accessing the password reset page through the link.
- Security Measures: Highlight security measures to protect the password recovery process, including token expiration and user verification.

7.2.4 User Basic Information

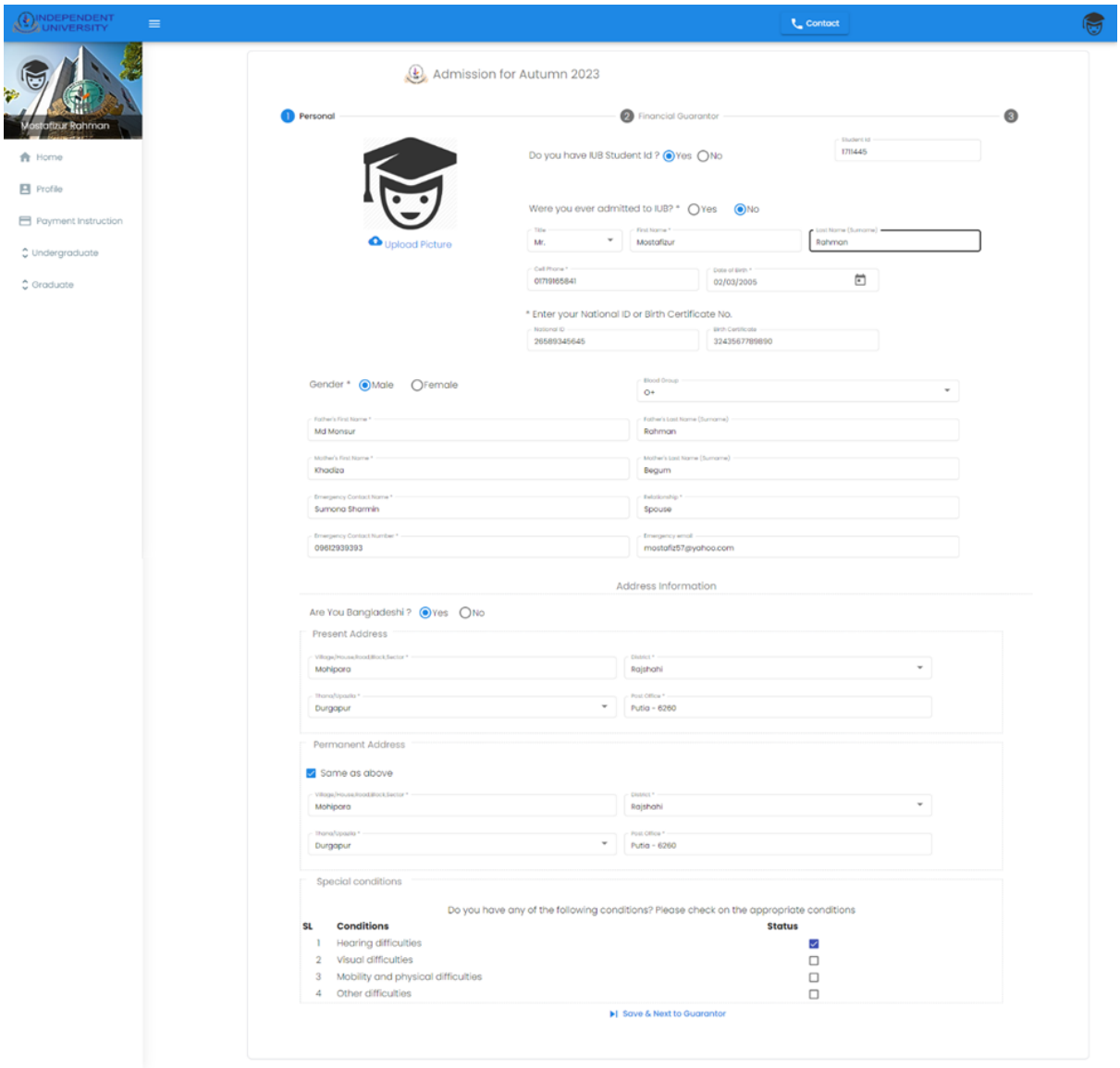

Figure 8.4 : User Basic Information

Inputs for User Basic Information Page:

Father's Name: User provides their father's name.

Mother's Name: User provides their mother's name.

Present Address: User provides their current address.

Permanent Address: User provides their permanent address.

Emergency Phone: User provides an emergency contact phone number.

Blood Group: User specifies their blood group.

Photo: User uploads a profile photo.

Foreigner Status: User indicates whether they are a foreigner or not.

Passport Information (If Foreigner): If the user is a foreigner, they provide passport details.

Visa Information (If Foreigner): If the user is a foreigner, they provide visa information.

Country of Birth (If Foreigner): If the user is a foreigner, they specify their country of birth.

Purpose:
The User Basic Information page aims to collect and store essential personal information, including parental details and emergency contact information, to assist the institution in maintaining accurate records and facilitating communication.

Features and Functionality:

- Form Fields: Include fields in the user profile for entering father's name, mother's name, present address, permanent address, and an emergency contact phone number.
- Data Validation: Implement data validation to ensure the accuracy and completeness of the information entered.
- Data Storage: Store the provided information securely in the user's profile within the database.
  Control and Flows for Documentation:
- Data Entry: Document the process of users entering their personal information into the respective form fields.
- Validation: Describe the validation checks performed to ensure the correctness of the provided data.
- Storage: Explain how the collected data is securely stored in the user's profile.
- Privacy and Security: Highlight measures taken to protect the confidentiality and security of personal information.

7.2.5 Academic form:

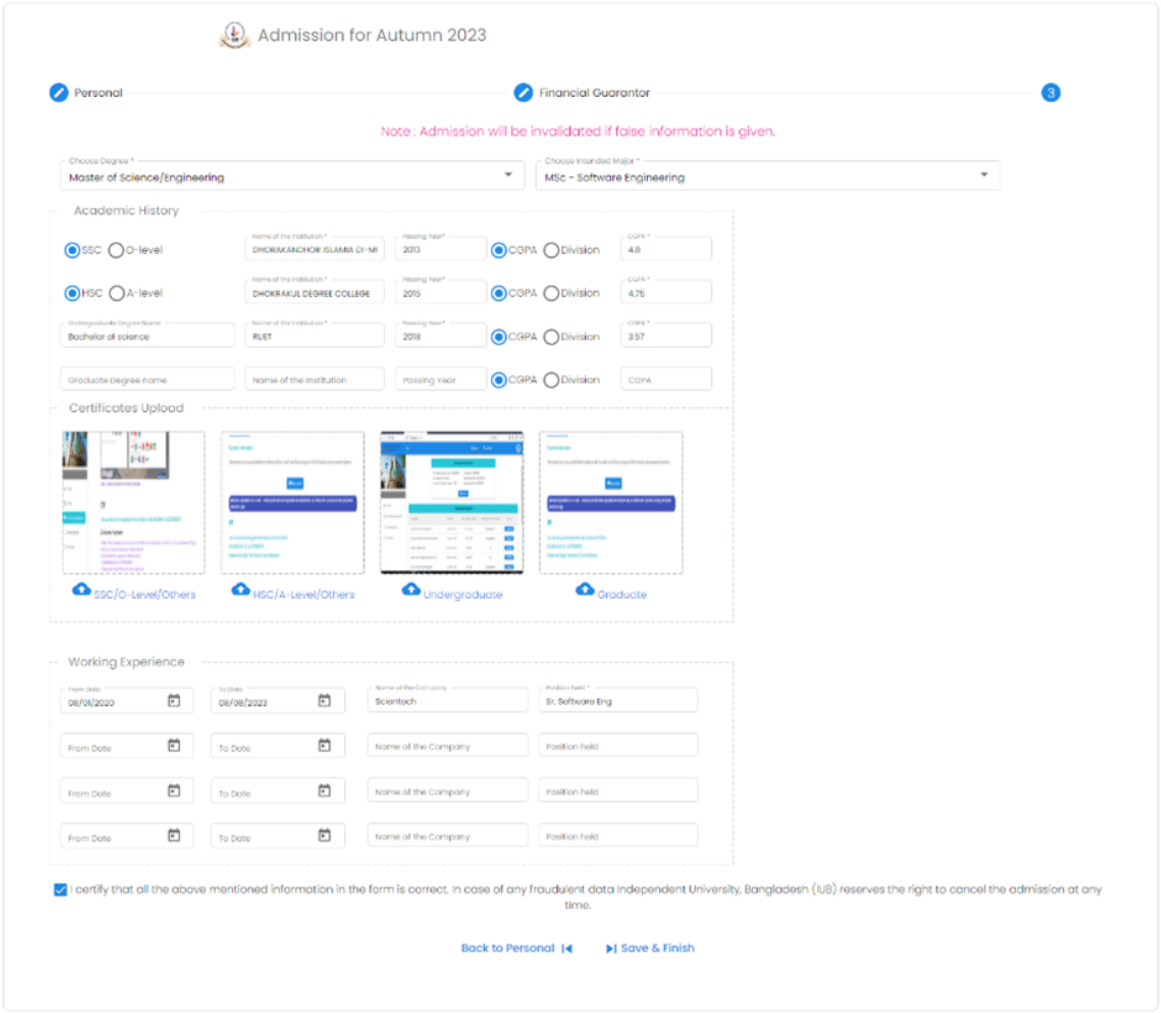

Figure 8.5: Academic form

Inputs for Academic Information Page:

- First Major: User specifies their first major field of study.
- Second Major: If applicable, the user specifies their second major field of study.
- 10 Years Degree Information: Details about the user's academic achievements and courses taken during their 10-year degree.
- 12 Years Degree Information: Details about the user's academic achievements and courses taken during their 12-year degree.
- Upload Result and Transcript Sheet: Users are allowed to securely upload their academic results and transcripts.
- IELTS Information: If applicable, users provide information about their IELTS (International English Language Testing System) scores.

- GRE Information: If applicable, users provide information about their GRE (Graduate Record Examination) scores.

Purpose:

- The Academic Information page aims to collect and document the user's educational background, major fields of study, academic achievements, and relevant standardized test scores to assist in the admission process. **Implementation Method:**

- Form Fields: Include fields in the user profile or application form for entering academic information, majors, and standardized test scores.

- Data Validation: Implement data validation to ensure the accuracy and completeness of the information entered.

- Document Upload: Provide a secure mechanism for users to upload their academic result sheets and transcripts.

- IELTS/GRE Section: Include specific sections for entering IELTS and GRE scores if required.

Features and Functionality:

- Data Entry: Document the process of users entering their academic information, including majors, courses, and scores.

- Validation: Describe the validation checks performed to ensure the correctness of the provided data.

- Document Upload: Explain how users can securely upload their academic result sheets and transcripts.

- IELTS/GRE Section: Document the process for entering IELTS and GRE scores, if applicable.

- Storage: Detail how the collected data and uploaded documents are securely stored and associated with the user's application or profile.

7.2.6 Student Dashboard :

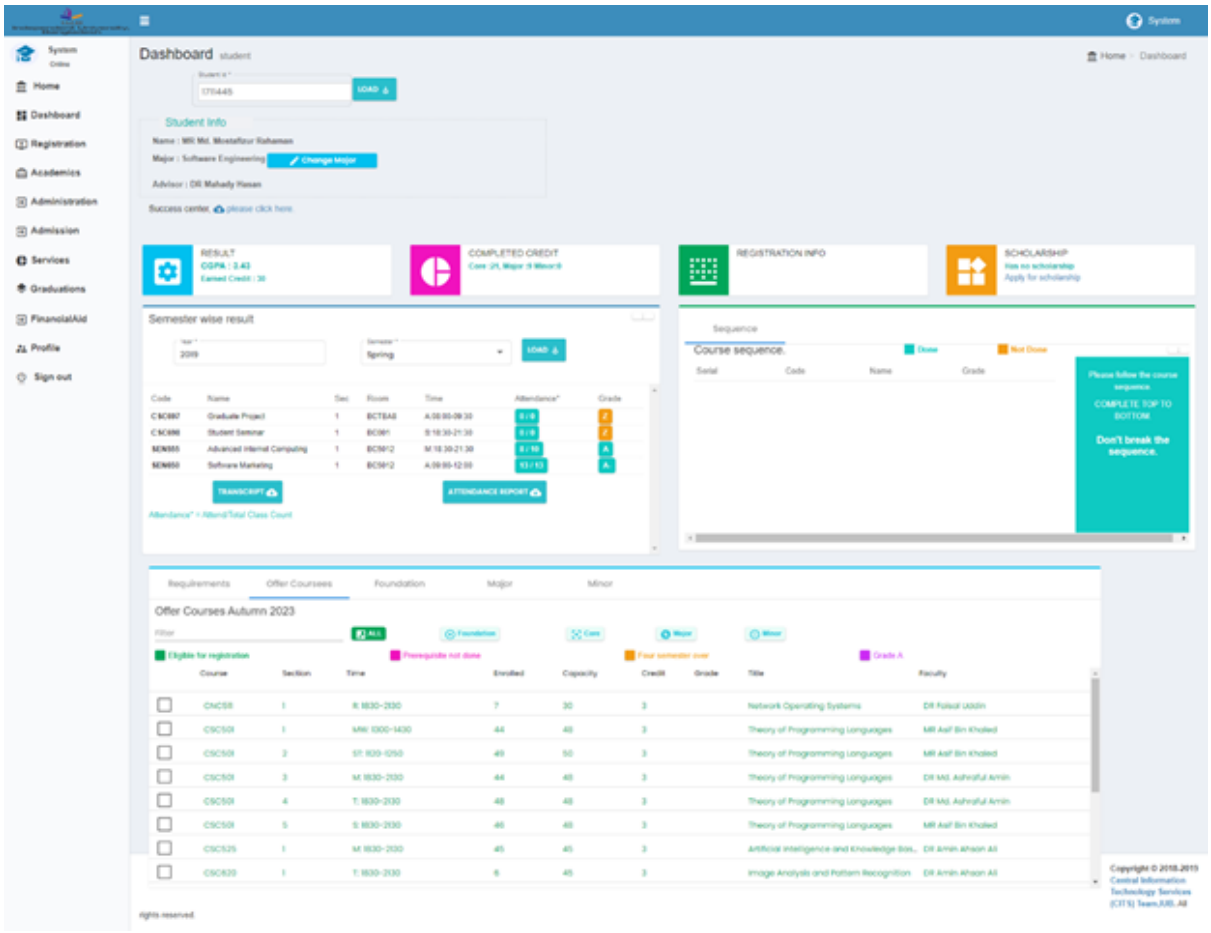

Figure 8.6 : Student Dashboard

Student Dashboard Features (Enhanced):

Transcript Summary: The dashboard continues to provide a comprehensive summary of a student's academic transcripts, including completed courses, grades, and cumulative GPA.

Student Dashboard Features (Enhanced): Student Financial Information: Students can access their financial information, including billing and payment details, outstanding balances, and payment history.

Semester-Wise Student Attendance Info: The dashboard displays attendance records across various semesters, offering insights into attendance percentages and dates of attended classes.

Current Semester Grade: Students can instantly view their current semester's grades for enrolled courses.

Course Sequence: Information about the recommended course sequence for a student's

program of study remains available.

Catalog Information: The course catalog section provides students with detailed information about available courses, prerequisites, course descriptions, and credit hours.
Current Major: The dashboard includes information about the student's current major field of study.
Change Major Application: Students can apply for a major change directly through the dashboard. The application process includes selecting the desired major and providing any necessary documentation.
Declare Minor: Students have the option to declare a minor field of study. The declaration process involves selecting the minor and confirming the intention to pursue it.

Implementation Method (Enhanced):
Major Change Workflow: Implement a workflow for major change applications, including user inputs, document uploads (if required), and approval processes.
Minor Declaration: Set up a user-friendly interface for students to declare their chosen minor and confirm their selection.
Data Integration: Ensure that major change requests and minor declarations are integrated into the IRAS system for tracking and processing.
User Notifications: Implement notifications to inform students of the status of their major change and minor declaration requests. **Documentation Updates:** Document the new features related to major changes and minor declarations, including workflow diagrams and user instructions.

7.2.7 Course Registration :

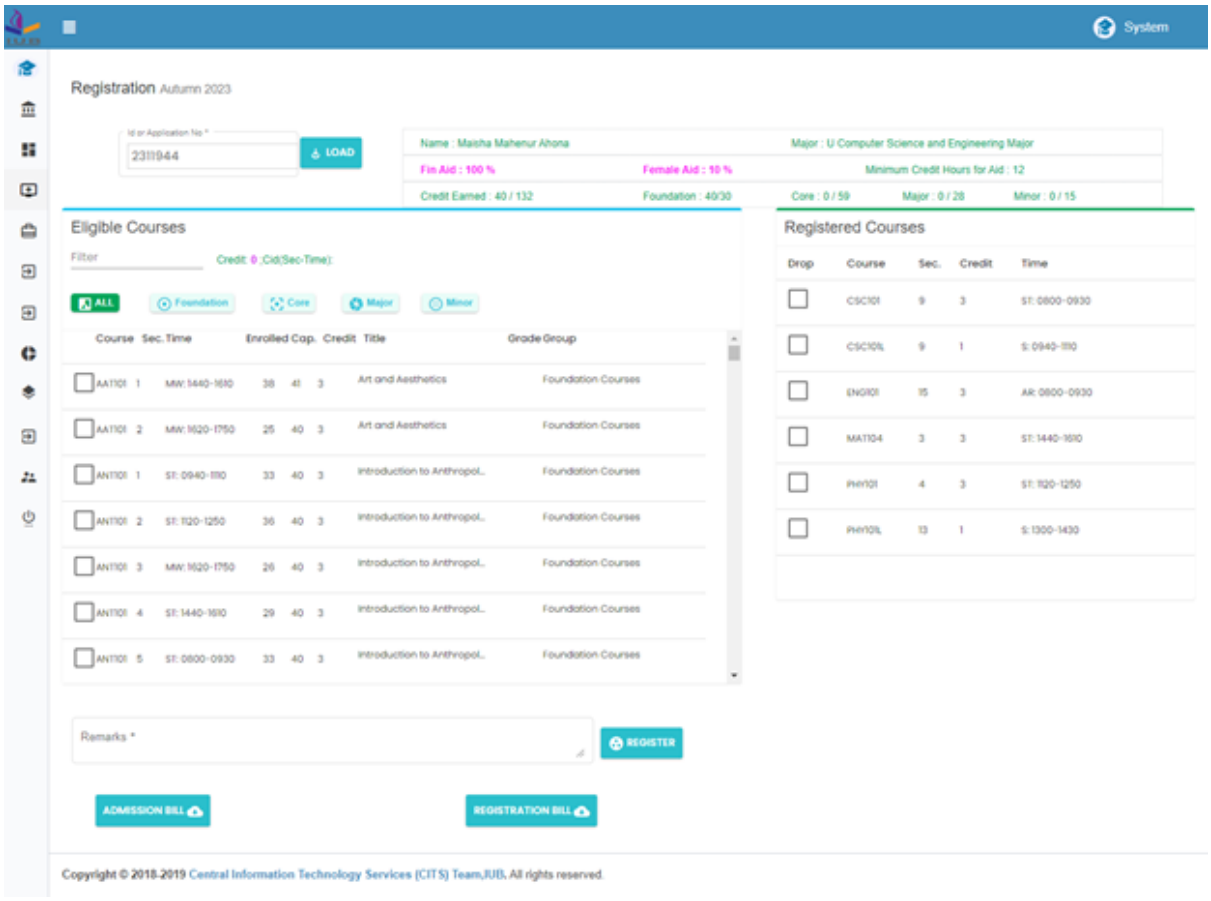

Figure 8.7: Course Registration

Purpose:

The Registration Page within IRAS is designed to facilitate the course registration process for students. It serves as an interactive platform where students can select and enroll in courses for the upcoming semester while ensuring that their selections comply with various validation checks and financial considerations.

Features and Functionality:

Course Selection: Upon accessing the Registration Page, students are presented with a list of available courses for the upcoming semester. They can browse through the course offerings and select the courses they wish to enroll in.

Validation Checks: The Registration Page incorporates multiple validation checks to ensure that students' course selections meet specific criteria. Same course will not be allow to select, For pair course, will be selected same section, couple course will be must be select with lab theory.

Time Conflicts: The system checks for time conflicts between selected courses to prevent scheduling conflicts.

Credit Limit: Some course like LFE201, BUS485, MGT480, student can select when

student credit earned more than 80. Checks if students have exceeded or not met the required credit limits for the semester

Prerequisites: Ensures that students meet the prerequisites for advanced courses.

Registration Button: After making their course selections, students click the "Registration" button to confirm their choices.

Course Enrollment: Upon successful validation and confirmation, the selected courses are officially enrolled for the student. This information is updated in the system, reflecting the student's course schedule.

Registration Bill: The system calculates the registration fees based on the enrolled courses and any applicable financial aid or scholarships. A registration bill is generated for the student, reflecting the total amount due.

Financial Aid Consideration: The Registration Page takes into account any financial aid or scholarships awarded to the student. It calculates the net amount due after deducting financial assistance.

Billing Information: The registration bill includes detailed information about the fees, payment due date, and payment methods accepted by the university.

Course Sequence : Every catalog has some course sequence where student must follow the course sequence on registration.

7.2.8 Student Class Attendance :

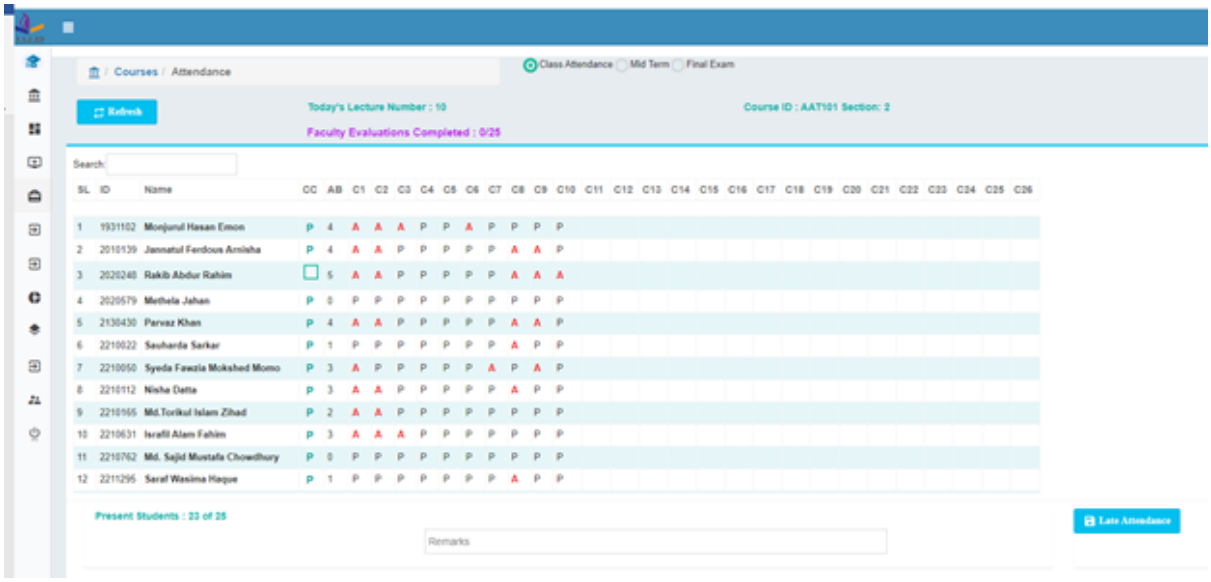

Figure 8.8: Class attendance

Purpose:

The Attendance Page in the IRAS serves as a central platform to manage and monitor student attendance for classes. Its primary function is to facilitate the recording and monitoring of student attendance through integrated biometric devices.

Features and Functionality:

Biometric Integration: The system is integrated with biometric devices, enabling students to register their attendance by providing their fingerprints upon entering the classroom. This integration ensures accurate and reliable attendance records.
Attendance Page: Faculty members have access to an Attendance Page where they can view and manage attendance records. They can mark the attended students for each session and have an overview of the total class attendance.

Informative Display: The Attendance Page provides a detailed overview of each class session, displaying the status of student attendance. This information allows faculty members to quickly assess who is present and absent during the class.
Absenteeism Monitoring: The system tracks absenteeism, and if a student's absences reach a predefined threshold (e.g., 25 percent of the total class sessions), the system marks a warning. If the absenteeism threshold is met, it automatically triggers a process to withdraw the student from the course.

Attendance Reports: The Attendance Page generates comprehensive attendance reports for individual students and overall class attendance. These reports offer insights into attendance trends, patterns, and individual student attendance records.
Faculty Class Reports: Faculty members have the capability to generate class reports to monitor and assess attendance across their courses. These reports provide a comprehensive overview of attendance statistics for each course handled by the faculty.

Implementation Method:
Biometric Integration: Implement a secure and efficient biometric integration system to accurately capture and record student attendance.

Automated Absenteeism Monitoring: Develop an automated monitoring system to track and notify when a student's absences reach the defined threshold for course withdrawal.

Report Generation: Design a reporting mechanism for generating attendance reports and faculty class reports based on the recorded attendance data.
Control and Flows for Documentation:

Biometric Device Integration:
Document the process of integrating the biometric devices with the attendance system.
Attendance Recording: Detail how attendance is recorded and updated in the system, and how it's displayed for faculty members.
Absenteeism Threshold Process: Explain the automated process triggered when

absenteeism reaches the withdrawal threshold.

7.2.9 Faculty Evolution :

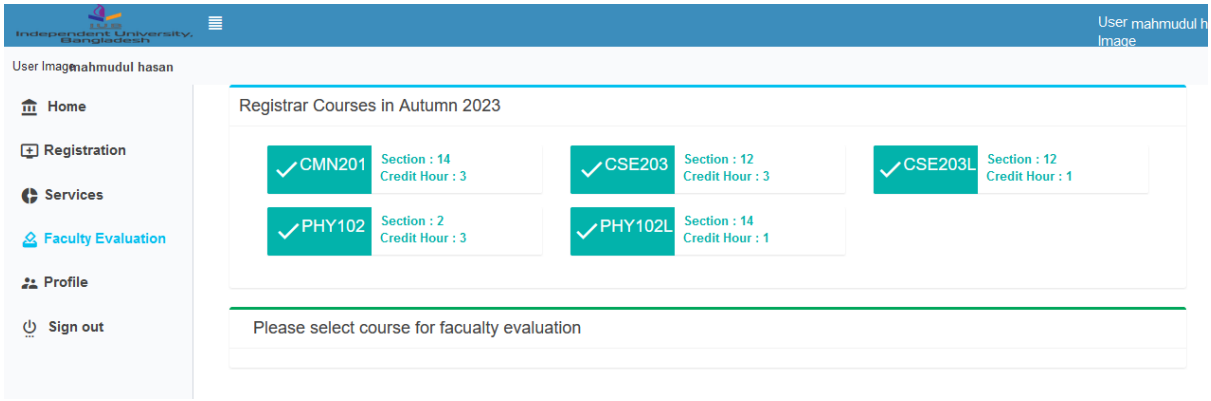

Figure 8.9: Faculty Evolution

Purpose:

The Faculty Evaluation Page is accessible to students after the mid-term of the semester. Its primary objective is to provide students with the opportunity to assess and provide feedback on their courses and faculty.

Features and Functionality:

Questionnaire Access: Once enabled, students gain access to a comprehensive questionnaire designed to evaluate each course and the respective faculty members. **Student Feedback:**

Students complete the questionnaire by providing feedback and answering specific questions related to their courses and faculty members. The questions cover various aspects such as teaching methodologies, course content, faculty interaction, and overall satisfaction. **Evaluation Report Creation:**

Based on the aggregated responses from students, the system generates a Faculty Evaluation Report. This report consolidates and analyzes the feedback received from students for each faculty member and course.

Information Visibility:

The Evaluation Page provides visibility of the questionnaires to students, ensuring they can assess all aspects of their courses and faculty members effectively.

7.3 Interface Design

7.3.1 GUI Style and Consideration

Angular Material for UI Components:

Design Consistency: Utilize Angular Material's pre-built components for consistent

design elements across the application.

Responsive Layout: Implement responsive design principles to ensure seamless functionality across various devices and screen sizes.

SCSS for Styling:

Modular Styles: Organize SCSS stylesheets in a modular fashion to enhance maintainability and readability.

Reusable Styles: Create reusable classes and components within SCSS to ensure consistency in styling and promote code reusability.

Chapter 8

Testing

8.1 Unit Testing

For automated backend testing [22], our chosen framework is XUnit [23], a powerful and versatile testing tool that integrates seamlessly with .NET Core APIs. Our service-based application undergoes comprehensive testing, where each service is rigorously examined, with a primary focus on validating both expected success states and handling potential errors. The decision to leverage XUnit stems from its robust capabilities for unit testing in .NET Core environments. XUnit provides a rich set of features that align with our testing requirements, allowing us to efficiently write and execute tests for our API services. This testing approach ensures that the backend services of our application are thoroughly examined for functionality, validation, and successful execution. By employing XUnit testing, we aim to maintain the reliability and correctness of our services, promoting a resilient and error-free backend for our application.

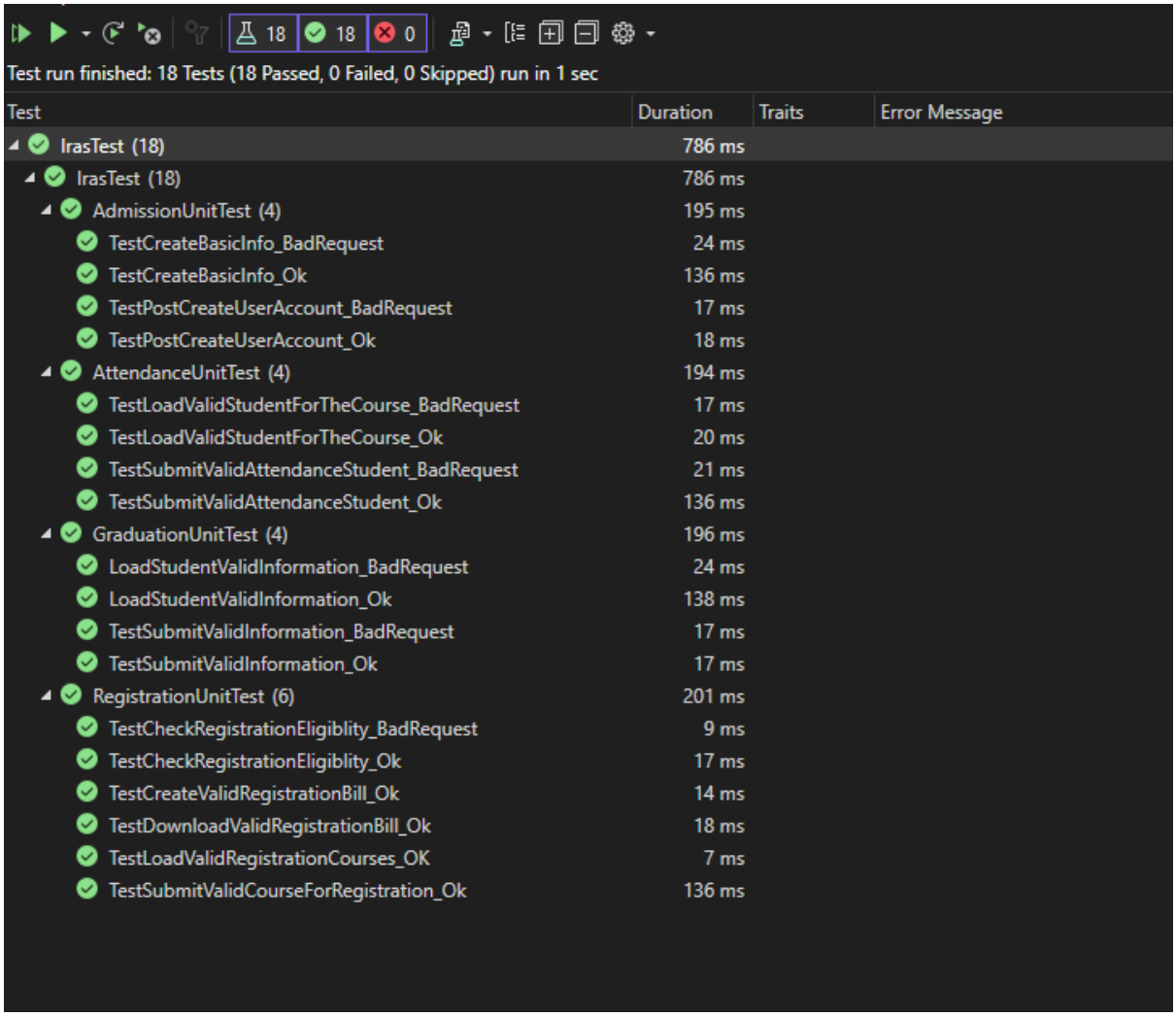

Figure 8.1: Unit Testing

8.1.1 Admission Module

1. **TestPostCreateUserAccount_BadRequest:**
   - *Objective:* Validate the system's response when attempting to create a user account with invalid or missing data.
   - *Scenario:* The test involves intentionally providing incomplete or invalid data for user account creation.
   - *Expected Result:* The system should return a Bad Request response, indicating that the user account creation failed due to data issues.
2. **TestPostCreateUserAccount_Ok:**
   - *Objective:* Verify the successful creation of a user account with valid data.
   - *Scenario:* The test involves providing valid data for creating a user account.

- *Expected Result:* The system should return an Ok response, indicating that the user account has been successfully created.

3. **TestCreateBasicInfo_BadRequest:**

   - *Objective:* Validate the system's response when attempting to create basic information with invalid or missing data.
   - *Scenario:* The test involves intentionally providing incomplete or invalid data for creating basic information.
   - *Expected Result:* The system should return a Bad Request response, indicating that the creation of basic information failed due to data issues.

4. **TestCreateBasicInfo_Ok:**

   - *Objective:* Confirm the successful creation of basic information with valid data.
   - *Scenario:* The test involves providing valid data for creating basic information.
   - *Expected Result:* The system should return an Ok response, indicating that the basic information has been successfully created.

8.1.2 Registration Module

1. **TestCheckRegistrationEligibility_BadRequest:**

   - *Objective:* Validate the system's response when checking registration eligibility for a student who does not have a valid schedule.
   - *Scenario:* The test involves intentionally providing invalid scheduling data or triggering a validation failure.
   - *Expected Result:* The system should return a Bad Request response, indicating that the student is not eligible for registration due to scheduling issues.

2. **TestCheckRegistrationEligibility_Ok:**

   - *Objective:* Verify the system's ability to check registration eligibility for a student with a valid schedule.
   - *Scenario:* The test involves providing a student with a valid schedule.
   - *Expected Result:* The system should return an Ok response, confirming that the student is eligible for registration.

3. **TestLoadValidRegistrationCourses_Ok:**

   - *Objective:* Ensure the successful loading of valid offer courses eligible for registration.

- *Scenario:* The test involves loading courses that are valid and open for registration.
- *Expected Result:* The system should return an Ok response, indicating that the valid offer courses have been successfully loaded.

4. **TestSubmitValidCourseForRegistration_Ok:**

   - *Objective:* Confirm that the system processes the submission of valid selected courses for registration successfully.
   - *Scenario:* The test involves submitting a set of valid courses selected by the student for registration.
   - *Expected Result:* The system should return an Ok response, signifying that the selected courses have been successfully submitted for registration.

5. **TestCreateValidRegistrationBill_Ok:**

   - *Objective:* Verify the creation of a registration bill after a successful registration process.
   - *Scenario:* The test involves completing the registration process successfully.
   - *Expected Result:* The system should return an Ok response, indicating that the registration bill has been successfully created.

6. **TestDownloadValidRegistrationBill_Ok:**

   - *Objective:* Confirm that the registration bill can be downloaded successfully after a successful registration.
   - *Scenario:* The test involves attempting to download a registration bill after a successful registration process.
   - *Expected Result:* The system should return an Ok response, allowing the user to download the registration bill.

8.1.3 Attendance Module

1. **TestLoadRegistrarStudentForCourse_BadRequest:**

   - *Objective:* Validate the system's response when attempting to load registrar students for a course, triggering a validation failure.
   - *Scenario:* The test involves intentionally providing invalid data or triggering a validation failure during student loading.
   - *Expected Result:* The system should return a Bad Request response, indicating that the student loading process failed due to validation issues.

2. **TestLoadRegistrarStudentForCourse_Ok:**

   - *Objective:* Verify the successful loading of registrar students for a course.
   - *Scenario:* The test involves loading registrar students for a course with valid data.
   - *Expected Result:* The system should return an Ok response, signifying that the registrar students have been successfully loaded.

3. **TestSubmitValidStudentForAttendance_Ok:**

   - *Objective:* Confirm that the system processes the submission of valid student attendance successfully.
   - *Scenario:* The test involves submitting attendance for a student with valid data.
   - *Expected Result:* The system should return an Ok response, indicating that the student attendance has been successfully submitted.

8.1.4 Graduation Module

1. **TestLoadStudentValidInfoForGraduation_BadRequest:**

   - *Objective:* Validate the system's response when attempting to load valid student information for graduation but encountering a validation failure or missing requirements.
   - *Scenario:* The test involves intentionally providing incomplete or invalid data for loading student information for graduation.
   - *Expected Result:* The system should return a Bad Request response, indicating that the loading process failed due to validation issues or missing requirements.

2. **TestLoadStudentValidInfoForGraduation_Ok:**

   - *Objective:* Verify the successful loading of valid student information for graduation.
   - *Scenario:* The test involves loading student information for graduation with all requirements completed.
   - *Expected Result:* The system should return an Ok response, indicating that the student information for graduation has been successfully loaded.

3. **TestPostValidInfoForGraduation_BadRequest:**

- *Objective:* Validate the system's response when attempting to post valid information for graduation but encountering a validation failure.
- *Scenario:* The test involves intentionally providing incomplete or invalid data for posting information for graduation.
- *Expected Result:* The system should return a Bad Request response, indicating that the posting process failed due to validation issues.

4. **TestPostValidInfoForGraduation_Ok:**

- *Objective:* Confirm that the system processes the posting of valid information for graduation successfully.
- *Scenario:* The test involves posting valid information for graduation.
- *Expected Result:* The system should return an Ok response, indicating that the information for graduation has been successfully posted.

8.2 Integration Testing

Integration testing [24] ensures that the different components of the system work together as expected. In our case, we are focusing on testing the interactions between modules, such as Common Setup, Admission, Registration, Attendance, Graduation, and Account.

8.2.1 Testing Environment Setup

Before running integration tests, set up a dedicated test environment that closely resembles the production environment. Use containerization tools like Docker for easy deployment and teardown.

8.2.2 Database Setup

Create a separate test database to avoid any impact on production data. Populate the test database with relevant sample data for testing scenarios.

8.2.3 Testing Tools

For integration testing, we'll use testing tools compatible with our technologies:

- **.NET Core:** Utilize xUnit for .NET Core integration tests.
- **gRPC:** Leverage Grpc.Net.Client for gRPC integration testing.

- **Web API:** Use HttpClient for interacting with Web API endpoints.
- **Azure Service Bus:** Implement tests for sending and receiving messages through Azure Service Bus.

8.2.4 Integration Testing Scenarios

Define various integration testing [24] scenarios to cover different interactions between modules. Include both success and failure scenarios to ensure robust error handling.

8.2.5 Repository Pattern Testing

Verify interactions with the data storage layer using the test database. Ensure that the repository pattern functions correctly for fetching and persisting data.

8.2.6 Factory and Strategy Pattern Testing

Test that factories create instances correctly, and strategies are applied as expected. Include scenarios where different strategies or factory implementations are used.

8.2.7 Azure Service Bus Integration

For systems using Azure Service Bus, create tests for sending and receiving messages. Ensure proper processing of messages by the relevant modules.

8.2.8 Concurrency and Parallelism Testing

Test scenarios where multiple modules process requests concurrently. Verify that the system behaves correctly under load and is scalable.

8.2.9 Logging and Monitoring

Integrate logging and monitoring tools into the test environment. Capture and analyze logs during integration tests for debugging and analysis.

8.2.10 Clean-Up

Implement a clean-up mechanism to reset the test environment between test runs.

8.3 Test Cases

8.3.1 Black Box Test Cases

Before each version release of the iRAS system, thorough testing is conducted, including major black box testing [25]. This encompasses functional testing to ensure proper system behavior, usability testing for an optimal user experience, compatibility testing across platforms, performance testing for responsiveness, security testing to address vulnerabilities, and regression testing to prevent new issues. This systematic testing approach aims to identify and rectify any potential issues, contributing to a reliable and high-quality iRAS system upon deployment.

Id:	BBTC004
Name:	Create and Update User Role
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Blocker
Step:	1. Enter valid Role Name. 2. Enter a valid Access Level (e.g., 1 to 10). 3. Set a valid username. 4. Click the "Save" button.
Expected Result:	• A new user role is created successfully. • Role ID is auto-generated. • The system date is captured and saved in the database. • Success message is displayed. • The created role is listed in the user role table.
Status:	Not Pass

Table 8.1: Create and Update User Role

Id:	BBTC005
Name:	Update Existing User Role
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Locate an existing user role in the role list. 2. Click on the role for editing. 3. Change the Role Name or Access Level. 4. Click the "Update" button.
Expected Result:	• The selected user role is updated successfully. • The system captures the update timestamp. • Success message is displayed.
Status:	Not Run

Table 8.2: Update Existing User Role

Id:	BBTC006
Name:	Create User
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Input User Id. 2. If the user information already exists, load the information and fill up the fields. 3. After saving successfully, display a success message or provide an exception for errors. 4. Show the user in the user list and registrar office approval waiting list. 5. Set the user role against the designation and give approval. 6. An auto-generated email is sent to the user's email. 7. Click on the iRAS user validation link. 8. User is redirected to the password setup page and sets the password. 9. After successfully saving, the user redirects to the user profile page.
Expected Result:	• iRAS user is created successfully. • User role is set against the designation, and approval is given. • Auto-generated email is sent to the user's email. • User validates through the iRAS user validation link. • User sets the password and is redirected to the user profile page.
Status:	Not Run

Table 8.3: Create User

Id:	BBTC007
Name:	Setup Courses
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Give Course Id, Course Title, Credit Hour, and School ID for the course. 2. Click the "Save" button. 3. After successfully saving, display the success message.
Expected Result:	• University courses list is updated with the new course. • Success message is displayed.
Status:	Not Run

Table 8.4: Setup Courses

Id:	BBTC008
Name:	Create Catalog
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Select catalog name. 2. Select course name. 3. Select course type and course group. 4. If prerequisites exist, provide the prerequisite list. 5. Click the "Save" button. 6. Display successful message.
Expected Result:	• Catalog is created successfully. • Success message is displayed.
Status:	Not Run

Table 8.5: Create Catalog

Id:	BBTC009
Name:	Semester Courses Offer
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Select course. 2. Select primary and secondary faculty, classroom, class start and end time. 3. If there is a class time conflict, display a message. 4. Set section and class capacity. 5. If the course has co-offered courses, select the co-offer courses list. 6. Save. 7. For updates, select the course, existing information will be shown, and the same steps as saving can be used to update.
Expected Result:	• Offered course is saved successfully. • Tally sheet is updated. • Student-wise updated offer list is displayed in the dashboard.
Status:	Not Run

Table 8.6: Semester Courses Offer

Id:	BBTC010
Name:	Create User
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Input first name, last name, email, cell phone, password, confirm password, and date of birth. 2. Compare password and confirm password. 3. If needed, show the password by clicking the show password option. 4. Click the "Create User" button. 5. Receive a success message or error message.
Expected Result:	• iRAS user is created successfully. • Success message is displayed.
Status:	Not Run

Table 8.7: Create User

Id:	BBTC011
Name:	Personal Information (Step 1)
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Select blood group. 2. Fill up father and mother first and last name. 3. Provide guardian phone number and email. 4. Select nationality (Are You Bangladeshi?). 5. If Bangladeshi, fill up present and permanent address details. 6. Click "Save and Next to Academic" button. 7. Check validation; if valid, proceed to the next step, otherwise show validation errors.
Expected Result:	• Candidate's personal information is saved successfully. • If valid, proceed to the academic information page. • If validation fails, stay on the step and indicate the invalid field.
Status:	Not Run

Table 8.8: Personal Information (Step 1)

Id:	BBTC012
Name:	Academic Information (Step 2)
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Select first and second major. 2. Set major-wise requirements. 3. Select 12 years of degree and populate the input form. 4. Select 10 years of degree and populate the input form. 5. Check the minimum CGPA requirement. 6. Select exam venue. 7. If exempt from admission tests, provide SAT/TOFEL or IELTS score. 8. Accept the certification. 9. Click "Save and Finish" button. 10. If you want to go back to personal information, click the back button.
Expected Result:	• Admission form is filled up successfully. • If valid, go to the profile page. • If invalid, stay on the step and indicate the invalid field.
Status:	Not Run

Table 8.9: Academic Information (Step 2)

Id:	BBTC013
Name:	Print Admit Card
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Fill up the admission form and set up a calendar. 2. Click the "Download Admit Card" button.
Expected Result:	• Admit card is downloaded in PDF format.
Alternative Option:	If unable to download, display a proper error message.
Post Condition:	Admit Card is downloaded.
Status:	Not Run

Table 8.10: Print Admit Card

Id:	BBTC014
Name:	Registration Data Setup
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Load students based on credit earned or select only new students. Consider loading based on the degree category. 2. Set registration date, registration type, and permission status. 3. Save the data.
Expected Result:	• Get a qualified list of students for registration.
Alternative Option:	If any issues occur during data setup, display a proper error message.
Post Condition:	Show registration date and permission status on the dashboard.
Status:	Not Run

Table 8.11: Registration Data Setup

Id:	BBTC015
Name:	Admission
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. If the user is not part of the student group, provide the student ID and click the load button. 2. Open the student admission popup. 3. Click the admission button.
Expected Result:	• Get an admitted student.
Alternative Option:	If any issues occur during the admission process, display a proper error message.
Post Condition:	Show the success message and display the "Print Admission Bill" button.
Status:	Run successfully

Table 8.12: Admission

Id:	BBTC016
Name:	Select Courses for Registration
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide the student ID. 2. Check the desired courses for registration. 3. Validate the selected courses for capacity, time conflicts, and other constraints. 4. If valid, mark the courses as selected; otherwise, display a message for invalidity.
Expected Result:	• Selected valid courses.
Alternative Option:	If any issues occur during the course selection process, display a proper error message.
Post Condition:	Show the list of selected courses and the total added credit hours.
Status:	Not Run

Table 8.13: Select Courses for Registration

Id:	BBTC017
Name:	Save Advising Courses
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide the selected courses list with associated information (sections, course code, credit hours, advisor ID, registration code, initial course grade, registration type, late registration status). 2. Save the selected courses information. 3. Create a unique registration code. 4. Call the Create Bill process.
Expected Result:	• Store the selected courses and associated information.
Alternative Option:	If there is a special permission, allow overwriting of the checking process.
Post Condition:	Update the enrollment for the courses (section-wise).
Status:	Not Run

Table 8.14: Save Advising Courses

Id:	BBTC018
Name:	Generate Registration Bill
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide the selected courses with credit hours. 2. Provide the per credit hour bill. 3. Provide the financial aid percentage and minimum credit hours for financial aid (if any). 4. Calculate the course bill. 5. Calculate the financial aid. 6. Create a unique bill ID. 7. Store the bill information.
Expected Result:	• Generate the total registration bill.
Alternative Option:	
Post Condition:	Generate the bill.
Status:	Not Run

Table 8.15: Generate Registration Bill

Id:	BBTC019
Name:	Valid Course Load for attendance
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide a valid faculty ID. 2. Provide the semester. 3. Provide the year. 4. Click the load process. 5. Verify if the course list is displayed.
Expected Result:	• Successfully load and display the course list.
Alternative Option:	• Manually check available courses. • Assign courses as per requirements.
Post Condition:	Offered courses will be shown in iRAS for registration.
Status:	Not Run

Table 8.16: Valid Course Load

Id:	BBTC020
Name:	Class Attendance
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide a valid faculty ID. 2. Select a course. 3. Choose a category (class, midterm, final). 4. Check the attendance for each student. 5. Submit the attendance list. 6. Verify if the present student list is generated. 7. Optionally, retake attendance within class time for late attendance.
Expected Result:	• Successfully submit and generate the present student list.
Alternative Option:	• Take attendance manually with paper.
Post Condition:	• Automatic withdrawal for undergrad students can be possible. • Attendance is marked.
Status:	Not Run

Table 8.17: Class Attendance

Id:	BBTC021
Name:	Course Withdraw
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide a valid student ID. 2. Go to the Registration tab. 3. Select the withdraw option. 4. Check the course to be withdrawn. 5. Submit the withdrawal request.
Expected Result:	• The course is removed from the student's profile.
Alternative Option:	• Apply for course withdrawal at the registrar's office.
Post Condition:	• The student is no longer responsible for that course. • The accounting office will adjust the course bill in the next semester.
Status:	Not Run

Table 8.18: Course Withdraw

Id:	BBTC022
Name:	Grade Submission
Milestone:	Iras-spring-24
Priority:	High
Bug Type:	Functional
Step:	1. Provide a valid faculty ID. 2. Go to the Reports tab and select Grade Submission. 3. The list of courses is shown. 4. Select a course. 5. Assign grades for students. 6. Submit the grades.
Expected Result:	• Course results for students are submitted.
Alternative Options:	• Faculty gives grades on a sheet of student list and submits it to the registrar's office.
Post Condition:	• Students receive their results. • Different actions are taken for different results, e.g., 'F' grade students retake the course, 'I' grade holders can continue to the next term without payment.
Status:	Grade submit successful

Table 8.19: Grade Submission

Chapter 9

Sustainability of the project

9.1 Sustainability Analysis

The sustainability [26] of the iRAS project is critical for its long-term success and continued effectiveness. The analysis focuses on various aspects, including financial sustainability, environmental impact, and adaptability to future needs.

9.1.1 Financial Sustainability

Assignment and Allocation of Resources

SL No	Name of Resources	Availability Percentage (%)	Hourly Rate (BDT)
1	Project Manager	50	5000
2	Project Architecture	25	5000
3	System Analyst	25	5000
4	Team leader (Database Focus)	100	3000
5	Team Leader (Backend Side Focus)	100	3000
6	Sr. Software Developer -1 (Database Focus)	100	2000
7	Database Developer-2(Database Focus)	100	2000
8	Sr. Software Developer1 (Full-stack)	100	2000
9	Software Developer2 (Backend Focus)	100	2000
10	Software Developer3 (Backend Focus)	100	2000
12	Sr. Front end developer 1 (JavaScript expert +Style sheet develop Focus)	100	1500
13	JavaScript +Style sheet developer 2 (Front end focus)	100	1500
15	Sr. Quality assurance engineer	100	1800
16	Quality assurance engineer1	100	1500
17	Quality assurance engineer3	100	1500

Total Project Cost

9.1.2 Costs and Budgeting

TASK ID	TASK LIST	Days	Hours	Cost (BDT)
1.1	Project Charter			
1.1.1	Define the project scope	3	24	168000.00
1.1.2	Identify the project objectives	3	24	168000.00
1.1.3	Define the project deliverables	3	24	168000.00
1.2	Develop project schedule			
1.2.1	Identify task dependencies	2	16	112000.00
1.2.2	Define project milestones	1	8	56000.00
1.3	Projections			
1.4	Guidelines	1	8	56000.00
1.6	Project Initiation	3	24	168000.00
2.1	Gather requirements			
2.1.1	Meet with stakeholders	10	80	560000.00
2.1.2	Gather and document	8	64	448000.00
2.1.3	Get approval	1	8	56000.00
2.2	Identify gaps and inconsistencies			
2.2.1	Identify Your Current State	5	40	280000.00
2.2.2	Identify Your Future State	5	40	280000.00
2.2.3	Identify the Gaps	2	16	112000.00
2.2.4	Evaluate Solutions	4	32	224000.00
2.3	Prioritize requirements	3	24	52000.00
3.1	Design and develop the system architecture			
3.1.1	Define system components and modules	8	64	448000.00
3.1.2	Create architectural diagrams	4	32	69333.33
3.2	Develop the code			
3.2.1	UI/UX design	41	328	2132000.00
3.2.2	Front end Coing	51	408	2652000.00
3.2.3	Backend Code	51	408	3345600.00
3.3	Implement database design and coding			
3.3.1	Database design	21	168	952000.00
3.3.2	Code in database	41	328	1858666.67
4	Testing			
4.1	End-to-End (E2E) Testing	51	408	1958400.00
4.2	API Integration Testing	29	232	1113600.00
4.3	Execute test plans to verify software quality	15	120	576000.00
5.1	Deploy in production environment	6	48	8200.00
5.2	Release the software to users	5	40	280000.00
			Total :	18301800.00

Monthly Operational Cost (After Deployment)

After deployment, in Azure every month, the deployment server cost will be 1,00,000 TK, and though it is a university application where different types of changes are needed, so every semester needs changes and updates. Therefore, a team of about 4 members is required to update the application, with a monthly salary of 4,00,000 TK.

Total Monthly Operational Cost: 5,00,000 TK

Revenue Generation

The implementation of the iRAS system is expected to significantly impact the revenue and operational efficiency of the university, primarily through streamlined processes and enhanced services.

1. **Time and Cost Savings:** The iRAS system expedites student registration, admission, graduation, and attendance processes. With the new system, the university can complete registrations within a day, compared to the previous four days, resulting in a time saving of seven days per Year. This translates to a cost saving of 30,00,000 TK per year.

2. **Operational Efficiency and Reputation:** The seamless and error-free operation of iRAS significantly contributes to enhancing the university's reputation. The efficient admission portal and prompt financial aid generation not only streamline administrative processes but also serve as attractive features for prospective students. With the accurate enrollment of students in course sections, it is anticipated that at least 200 additional students will enroll, resulting in a monthly gain of 1000,000 TK (Taka). The enhanced capabilities of management, facilitated by iRAS, empower them to make well-informed decisions, potentially saving an additional 100,000 TK per month. This multifaceted improvement in operational efficiency and student enrollment brings both financial and managerial benefits to the university.

3. **Manpower Savings:** iRAS reduces the workload of the financial aid and registrar offices, saving an estimated 2-3 personnel. Additionally, the accounting department benefits from the service-oriented system, enabling students to deposit various bills in banks with live updates. This results in an approximate monthly manpower cost saving of 2,00,000 TK.

4. **Total Revenue Generation:** The cumulative impact of cost and time savings, increased student enrollment, and manpower efficiencies lead to a total revenue generation of a year is 300000+72,00,000 + 2,00,000 = 77,00,000 TK.

5. **Future Development and Cost Reduction:** iRAS, being a service-based architecture, paves the way for future development. The modular structure of the

system reduces the need for purchasing new software, contributing to long-term cost reduction and operational efficiency. As a top-rated university in Bangladesh, IUB's reputation is poised to benefit significantly from these advancements, aligning with its future plans. The system's adaptability also opens opportunities for developing mobile applications, further enhancing accessibility and user experience.

Return on Investment (ROI)

Periodic evaluations of the project's ROI will help measure the effectiveness of financial investments. This includes assessing the benefits derived from improved efficiency, reduced administrative workload, and enhanced user experiences.

	Head	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9
Cost	Maintance	0	6000000	12000000	18000000	24000000	30000000	36000000	42000000	48000000	54000000
	Sum	**18301800**	**24301800**	**30301800**	**36301800**	**42301800**	**48301800**	**54301800**	**60301800**	**66301800**	**72301800**
Profit	Time and Cost Savings		3000000	6000000	9000000	12000000	15000000	18000000	21000000	24000000	27000000
	Operational Efficiency and Reputation	0	7200000	14400000	21600000	28800000	36000000	43200000	50400000	57600000	64800000
	Man Power Saving	0	2400000	4800000	7200000	9600000	12000000	14400000	16800000	19200000	21600000
	Sum	**0**	**12600000**	**25200000**	**37800000**	**50400000**	**63000000**	**75600000**	**88200000**	**100800000**	**113400000**

Figure 9.3: ROI Data

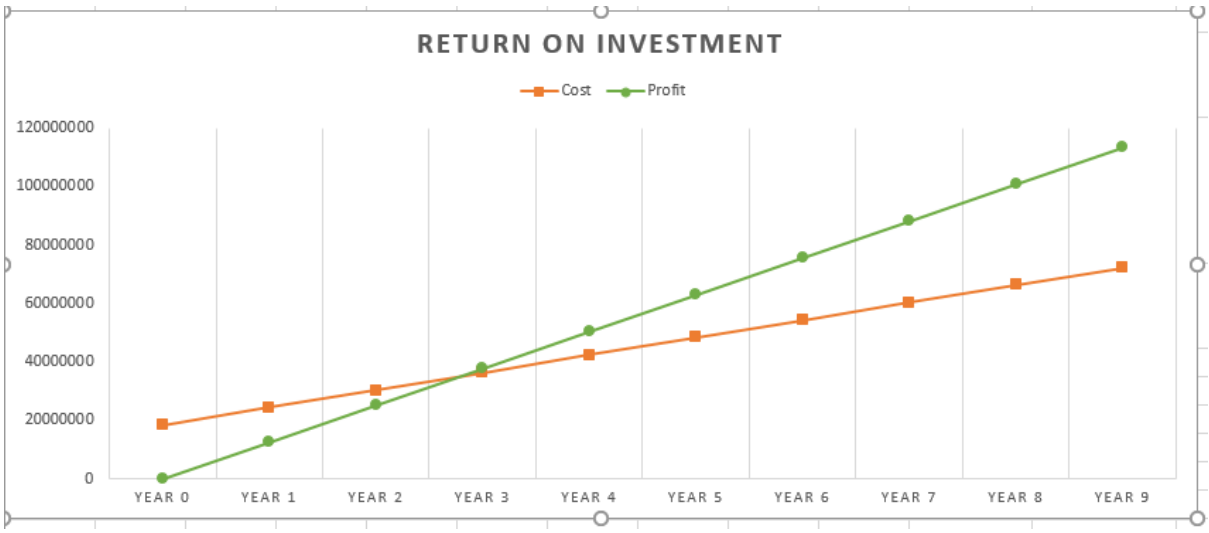

Figure 9.4: ROI

9.1.3 Environmental Impact

Green Computing Practices

Efforts to reduce the environmental impact of the iRAS project can include adopting green computing practices. This may involve optimizing server resource usage, implementing energy-efficient data centers, and minimizing electronic waste through responsible hardware disposal. Additionally, the deployment strategy on Azure aligns with sustainability goals, as Azure emphasizes energy efficiency and eco-friendly data management practices. By choosing Azure, the iRAS project not only benefits from a robust

cloud infrastructure but also contributes to environmentally conscious computing solutions.

Paperless Operations

Promoting paperless operations within the educational institution through iRAS can contribute to environmental sustainability. Reduction in physical paperwork aligns with eco-friendly practices.

9.1.4 Adaptability and Future-Proofing

Scalability

Design considerations for scalability ensure that iRAS can accommodate the growth of users, data, and functionalities over time. This adaptability is crucial for handling increased demand without compromising performance.

Technology Upgrades

Anticipating and planning for technology upgrades, including advancements in programming languages, frameworks, and database systems, will keep iRAS current and compatible with evolving industry standards.

User Feedback and Iterative Development

Regularly gathering user feedback and incorporating iterative development cycles ensure that iRAS remains aligned with user needs. This adaptive approach enhances the system's sustainability by addressing evolving requirements.

9.1.5 Social Impact

User Training and Support

Investing in user training programs and providing ongoing support fosters positive user experiences. This social aspect contributes to the sustainability of iRAS by ensuring user satisfaction and acceptance.

Inclusivity and Accessibility

Ensuring that iRAS remains inclusive and accessible to diverse user groups supports social sustainability. This includes features for users with disabilities and considerations for different demographics within the educational community.

9.1.6 Ethical Considerations

The iRAS project is committed to upholding ethical standards throughout its development, deployment, and operation. Privacy and data security are paramount concerns, and the system is designed to ensure the confidentiality and integrity of academic records. Strict access controls and authentication mechanisms are implemented to safeguard sensitive information. The project adheres to relevant data protection regulations, respecting the rights of individuals and ensuring responsible handling of personal data. Transparency is maintained in communication with stakeholders, and the system aims to provide accurate and unbiased information. Additionally, user training programs will be conducted to promote responsible and ethical usage of the iRAS system. The ethical foundation of the project reflects a commitment to integrity, accountability, and the well-being of the academic community it serves.

9.1.7 Technical Feasibility

Hardware Resources

The technical feasibility of the iRAS project requires robust hardware resources to support its development, deployment, and ongoing operations. The following hardware components are deemed necessary for the successful execution of the project:

1. **Azure App Service Plan:**
    - Pricing Tier: Premium v3 (P2mv3)
    - Configuration: 8 cores, 64 GB RAM, 250 GB storage space
2. **Azure Web App (App Service):**
    - Deployment of .NET Core 7 backend and Angular client application
    - Configuration tied to the Premium v3 Service Plan
3. **Development Server:**
    - Windows Server compatible with ASP.NET Core and Node.js
    - Adequate resources for development tasks
    - Nginx or Apache for serving front-end assets
4. **Database Server:**
    - Oracle Database 19c or later for data storage
    - Oracle Database Management System (DBMS) tools for administration

Software Resources

1. **Server Software:**
    - Operating System: Windows Server for development
    - Web Server: Nginx or Apache for serving front-end assets
    - ASP.NET Core: Latest stable version for backend API development
    - Node.js: Required for server-side scripting and building APIs
    - Oracle Database: Oracle 19c or later for data storage
    - Database Management System: Oracle DBMS tools for administration
2. **Client Software:**
    - Web Browsers: Compatible modern web browsers (Chrome, Firefox, Edge, Safari)
    - Front-end Frameworks: Angular, React, SolidJs
    - API Client: Tools like Postman for testing and debugging API endpoints

Other Resources

1. **Development Team:**
    - Project Manager
    - Project Architecture
    - System Analyst
    - Team Leader (Database Focus)
    - Team Leader (Backend Side Focus)
    - Sr. Software Developers
    - Database Developers
    - Front-end Developers
    - Quality Assurance Engineers
2. **Development Tools:**
    - Integrated Development Environment (IDE) for .NET Core and Angular development
    - Version control system (e.g., Git) for code collaboration
    - Deployment tools for releasing updates

9.1.8 Conclusion

The sustainability of the iRAS project requires a holistic approach encompassing financial stability, environmental responsibility, adaptability to future needs, and positive social impact. By addressing these aspects, iRAS can continue to provide valuable services to its users while contributing to the overall sustainability goals of the educational institution.

Chapter 10

Conclusion

10.0.1 Sustainability

The sustainability of the IRAS BPR project is anchored in its operational cost structure, revenue generation potential, and environmental impact. The system's monthly operational cost is carefully managed to ensure that it remains within budget constraints. The projected revenue generation through increased student enrollment, operational efficiency, and cost savings is expected to exceed the yearly operational cost, ensuring financial sustainability. Additionally, the platform's design allows for scalability, accommodating future growth without compromising performance or user experience. By leveraging digital tools and reducing manual processes, the IRAS system also contributes to resource conservation, aligning with sustainability goals.

10.0.2 Feasibility

The feasibility of the IRAS BPR project has been thoroughly evaluated across multiple dimensions. From a technical perspective, the project utilizes proven technologies, including UML for logical design and robust testing frameworks like XUnit for backend validation. The integration with existing systems is designed to be seamless, minimizing disruptions during implementation. Financially, the project's cost structure and projected revenue generation indicate a favorable return on investment within a reasonable timeframe, supporting its feasibility. The project's manpower requirements have also been carefully considered, with operational efficiencies leading to reduced staffing needs, further enhancing feasibility.

10.0.3 Social and Environmental Impact

The social and environmental impact of the IRAS BPR project is primarily positive. Socially, the system streamlines academic processes, providing students with a more user-friendly platform for course registration and other academic activities. This contributes

to improved student satisfaction and engagement. Environmentally, the project's focus on digitalization and reduced paperwork supports sustainability efforts by decreasing paper waste and energy consumption. The IRAS system's cloud-based infrastructure also allows for efficient resource allocation and minimizes the carbon footprint compared to traditional on-premise systems.

10.0.4 Ethics

Ethics play a crucial role in the IRAS BPR project. The system is designed to protect user privacy and ensure data security, adhering to ethical guidelines and legal requirements for data protection. The use of encryption and secure communication protocols safeguards sensitive information, fostering trust among users. The project also emphasizes fairness and equity, ensuring that all students have equal access to course registration and other academic resources. Additionally, the IRAS team is committed to transparency and accountability, regularly reviewing processes to ensure they meet ethical standards.

10.0.5 Project Summary

The IRAS Business Process Reengineering project represents a significant step toward modernizing academic processes at the institution. Through a comprehensive approach to logical design, rigorous testing, and careful consideration of sustainability, feasibility, social impact, and ethics, the project has successfully created a system that meets the needs of both students and administrative staff. The system's operational efficiencies and financial benefits demonstrate its value, while its design ensures scalability and adaptability for future growth. Overall, the IRAS BPR project is poised to contribute to the institution's success and pave the way for continued innovation in academic administration.

10.0.6 Future Work

Looking ahead, the IRAS BPR project has identified several areas for future work. The system's scalability provides opportunities for expanding its functionality, such as integrating additional modules or enhancing existing ones. Further improvements in user experience and accessibility are also planned, with a focus on making the platform more inclusive for all users. Additionally, ongoing monitoring and feedback mechanisms will be implemented to ensure continuous improvement and adaptation to changing needs. The project team is committed to exploring new technologies and best practices to maintain the system's relevance and effectiveness in the long term.

Bibliography

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," *arXiv preprint arXiv:1709.08439*, 2017.

[2] L. Cao, K. Mohan, P. Xu, and B. Ramesh, "A framework for adapting agile development methodologies," *European Journal of Information Systems*, vol. 18, no. 4, pp. 332–343, 2009.

[3] L. Williams, "Agile software development methodologies and practices," in *Advances in computers*, vol. 80, pp. 1–44, Elsevier, 2010.

[4] M. Ozer and D. Vogel, "Contextualized relationship between knowledge sharing and performance in software development," *Journal of Management Information Systems*, vol. 32, no. 2, pp. 134–161, 2015.

[5] D. F. Galletta and Y. Zhang, "Asynchronous virtual teams: Can software tools and structuring of social processes enhance performance?," *Human-Computer Interaction and Management Information Systems: Applications. Advances in Management Information Systems*, pp. 135–158, 2014.

[6] L. Aversano, G. Canfora, A. De Lucia, and P. Gallucci, "Business process reengineering and workflow automation: a technology transfer experience," *Journal of Systems and Software*, vol. 63, no. 1, pp. 29–44, 2002.

[7] M. C. Jurisch, C. Ikas, W. Palka, P. Wolf, and H. Krcmar, "A review of success factors and challenges of public sector bpr implementations," in *2012 45th Hawaii International Conference on System Sciences*, pp. 2603–2612, IEEE, 2012.

[8] M. Rahman and S. Sagorika, "Web-based resource management system for promoting teaching and learning: Bangladesh perspectives," 2016.

[9] M. Z. Islam, M. M. Rahman, and M. K. Islam, "Online examination system in bangladesh context," *International Journal of Science, Environment and Technology (IJSET)*, vol. 2, no. 3, pp. 351–359, 2013.

[10] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 44–51, IEEE, 2016.

[11] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen, and M. A. Babar, "Understanding and addressing quality attributes of microservices architecture: A systematic literature review," *Information and software technology*, vol. 131, p. 106449, 2021.

[12] T. Cerny, A. S. Abdelfattah, V. Bushong, A. Al Maruf, and D. Taibi, "Microservice architecture reconstruction and visualization techniques: A review," in *2022 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pp. 39–48, IEEE, 2022.

[13] M. Waseem, P. Liang, and M. Shahin, "A systematic mapping study on microservices architecture in devops," *Journal of Systems and Software*, vol. 170, p. 110798, 2020.

[14] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022.

[15] S. Bell, T. Berg, and S. Morse, "Towards an understanding of rich picture interpretation," *Systemic Practice and Action Research*, vol. 32, pp. 601–614, 2019.

[16] A. A. Alzahrani, "Analyzing the factors impacting the choice of the fact-finding technique for requirements elicitation.," *International Journal on Information Technologies & Security*, vol. 12, no. 1, 2020.

[17] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," 2012.

[18] K. Schwaber, "Scrum development process," in *Business Object Design and Implementation: OOPSLA'95 Workshop Proceedings 16 October 1995, Austin, Texas*, pp. 117–134, Springer, 1997.

[19] J. J. Odell, *Advanced object-oriented analysis and design using UML*, vol. 12. Cambridge University Press, 1998.

[20] T. von der Maßen and H. Lichter, "Modeling variability by uml use case diagrams," in *Proceedings of the International Workshop on Requirements Engineering for product lines*, pp. 19–25, 2002.

[21] C. Gutwenger, M. Jünger, K. Klein, J. Kupke, S. Leipert, and P. Mutzel, "A new approach for visualizing uml class diagrams," in *Proceedings of the 2003 ACM symposium on Software visualization*, pp. 179–188, 2003.

[22] W. Pugh and N. Ayewah, "Unit testing concurrent software," in *Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, pp. 513–516, 2007.

[23] Z. Chaczko, R. Braun, L. Carrion, and J. Dagher, "Design of unit testing using xunit. net," in *2014 Information Technology Based Higher Education and Training (ITHET)*, pp. 1–9, IEEE, 2014.

[24] H. K. Leung and L. White, "A study of integration testing and software regression at the integration level," in *Proceedings. Conference on Software Maintenance 1990*, pp. 290–301, IEEE, 1990.

[25] E. Steegmans, P. Bekaert, F. Devos, G. Delanote, N. Smeets, M. van Dooren, and J. Boydens, "Black & white testing: Bridging black box testing and white box testing," *Software Testing: Beheers Optimaal de Risico's van IT in uw Business*, pp. 1–12, 2004.

[26] F. Albertao, J. Xiao, C. Tian, Y. Lu, K. Q. Zhang, and C. Liu, "Measuring the sustainability performance of software projects," in *2010 IEEE 7th International Conference on E-Business Engineering*, pp. 369–373, IEEE, 2010.