## CHAPTER ONE: GENERAL INTRODUCTION

### Introduction

IGICUPURI is a mobile application developed to provide students at Mount Kigali University with easy and consistent access to past examination papers. The app allows lecturers to upload past papers into a central digital archive, where students can search and download them for study and revision. **students** can search, view, and download past exam papers for their respective courses and departments to aid in study and revision. **Lecturers** are responsible for uploading past exam papers, ensuring they are correctly categorized by course, year, and department. **Admins** oversee the system by reviewing uploaded papers, approving or rejecting them to maintain content quality and compliance. This chapter introduces the project by covering the background of the study, the problem it aims to solve, the main objectives, the scope and limitations, justification for the project, potential risks with their solutions, the required budget, and the proposed project schedule.

### Background of the Study

Mount Kigali University (MKU) is a private institution of higher learning located in Kigali, Rwanda. The university was established in 2010 as a campus of Mount Kenya University, a Kenyan-based institution founded by Prof. Simon N. Gicharu.In April 2023, following accreditation by the Rwanda Higher Education Council under the Ministry of Education, the institution was granted full autonomy and rebranded as Mount Kigali University. MKU offers various academic programs such as business, law, computer science, and health sciences. The university is committed to delivering quality education and enhancing student services through technological innovation .Mount Kigali University produces thousands of examination papers every semester, but access to these past papers has been inconsistent

### 1.3 Problem Statement

Currently, students at Mount Kenya University obtain past exam papers informally through **WhatsApp groups**, **photocopied handouts**, or by requesting them directly from lecturers. This method is often inconsistent, time-consuming, and prone to data loss. This addresses the current challenge of limited and informal access to these academic resources, especially affecting new and remote learners. IGICUPURI supports the university’s move toward digital learning by offering a more efficient, paperless, and accessible system. Students often lack consistent access to official past papers. Currently, exam paper sharing is informal and prone to data loss. Lecturers also have no centralized place to submit their exams for student revision. This results in unequal access, especially for new or remote learners.

### 1.4 Objectives

#### General Objective

To develop IGICUPURI mobile app .

#### Specific Objectives

1. To develop a database that will store past exam papers.
2. To design an interface that all user to interact with the application.
3. To implement application that will generate a report of past papers uploaded.

### 1.5 Scope and Limitations

The **IGICUPURI App** is designed specifically for **Mount Kigali University** , this will take time between July , August and September targeting students and lecturers as its primary users. It enables lecturers to upload past exam papers and allows students to search, download, and study them with ease, promoting digital academic access. Geographically, the app is intended for use within **Mount Kigali University campuses across Kigali**, but can be accessed remotely from any location with internet connectivity. Contextually, it addresses the challenge of inconsistent access to past exam papers by providing a centralized, secure, and efficient platform for academic resources. However, it requires a stable internet connection for both uploading and accessing content, which may affect users in low -connectivity areas. The effectiveness of the app also depends heavily on lecturers actively uploading exam papers. Initially, the system will support only a few selected departments, with plans to expand to all faculties in **future academic years**.

### 1.6 Project Justification

The **IGICUPURI App** supports student academic performance by providing centralized and easy access to past exam papers, helping learners prepare effectively for their exams. It also enhances lecturers’ efficiency by simplifying the process of sharing academic resources, thus saving time and reducing repetitive requests from students. Additionally, the app contributes to environmental sustainability by minimizing paper usage and promoting digital resource management.

From the **developer’s perspective**, the project showcases technical skills in mobile and cloud-based application development, database management, and secure user authentication, making it a valuable academic and professional milestone. It also creates an opportunity for future developers to enhance and maintain the app by introducing advanced features such as **offline access**, **AI-powered recommendations**, and integration with the university’s student portal. As technology and user needs evolve, future developers can easily scale and upgrade the system thanks to its flexible architecture. Overall, the app not only benefits students and lecturers but also serves as a foundation for continuous innovation and digital transformation at Mount Kigali University.

### 1.7 Project Risks and Mitigations

#### The **IGICUPURI App** may face several risks, including **data security threats**, where unauthorized access to sensitive exam papers or user data could compromise academic integrity and privacy. To mitigate this, the app will use **Firebase Authentication** with encrypted logins, secure cloud storage policies, and regular system audits to ensure safety. Another risk is **low lecturer participation**, which could limit the availability of past papers; this will be addressed through **training, administrative support**, and a simplified upload process featuring **bulk uploads** and **drag-and-drop options** to make sharing easier. Additionally, the app may encounter **virus attacks** or **system crashes** that can corrupt data or disrupt operations; to prevent this, a **secure cloud infrastructure**, **real-time antivirus protection**, **automated backups**, and **disaster recovery mechanisms** will be implemented. Finally, the app could be targeted by **hackers** through cyberattacks such as **DDoS**, **data breaches**, or unauthorized access; these threats will be mitigated using **advanced encryption**, **firewalls**, **intrusion detection systems**, and **multi-factor authentication** to safeguard the system and user data.

## Budget and resources

|  |  |
| --- | --- |
| **ITEMS** | **Amount (rwf)** |
| Laptop | 500,000rwf |
| Paper package | 10,000rwf |
| Internet | 75,000rwf |
| Flash disk | 16000rwf |
| Other expenses | 20,000rwf |
| **Total Cost** | 621000 |

**Table 1.1: Budget**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Time** | **JULY** | | | | **AUGUST** | | | | **SEPTEMBER** | | | |
| Weeks | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| Chapter One:  Introduction |  |  |  |  |  |  |  |  |  |  |  |  |
| Chapter Two:  Literature Review |  |  |  |  |  |  |  |  |  |  |  |  |
| Chapter Three:  Methodology |  |  |  |  |  |  |  |  |  |  |  |  |
| Chapter Four:  System Analysis |  |  |  |  |  |  |  |  |  |  |  |  |
| Chapter Five:  System Design |  |  |  |  |  |  |  |  |  |  |  |  |
| Chapter Six:  System Implementation |  |  |  |  |  |  |  |  |  |  |  |  |
| Chapter 7:  Conclusion and  Recommandation |  |  |  |  |  |  |  |  |  |  |  |  |

## Project schedule

**Table 1. 2:Schedule**

## ****CHAPTER TWO: LITERATURE REVIEW****

### ****2.0 Introduction****

This chapter reviews scholarly literature, digital system case studies, and academic repositories related to educational resource management, digital archiving, and e-learning platforms. The goal is to establish a foundation upon which the **IGICUPURI** system is built by identifying existing solutions, evaluating their strengths and limitations, and highlighting the gap that this app aims to fill.

Through this review, relevant technologies, methods of deployment, user needs, and security considerations are also examined.

### ****2.1 Definition of Key Terms****

#### ****2.1.1 Digital Repository****

A digital repository is an online archive that stores and manages digital content, allowing users to search, access, and retrieve documents such as journals, eBooks, or academic papers. IGICUPURI functions as a specialized repository for past exam papers.

#### ****2.1.2 Learning Management System (LMS)****

An LMS is a software platform that facilitates the administration, documentation, tracking, and delivery of educational content. Examples include Moodle, Blackboard, and Google Classroom. Unlike a full LMS, IGICUPURI focuses solely on academic archives (past papers), not assignment submissions or quizzes.

#### ****2.1.3 Cloud Storage****

Cloud storage allows users to save data on remote servers accessed via the internet. It ensures high availability, automatic backups, and scalability. Services like Firebase, Google Drive, and AWS S3 are commonly used in academic apps.

#### ****2.1.4 Educational Content Management****

This is the process of collecting, organizing, and publishing learning materials to aid teaching and study. IGICUPURI is built on this principle by enabling structured storage and retrieval of exam content.

### ****2.2 Review of Existing Systems****

#### ****2.2.1 Moodle****

Moodle is an open-source learning management system (LMS) widely used in universities for managing courses, allowing teachers to upload exams, lecture notes, and assessments. However, it often requires institutional licensing and complex setup, making it resource-intensive. Additionally, Moodle combines numerous functions, which can overwhelm users who only need access to past exam papers, limiting its effectiveness as a simple archival tool since it is not specialized for that purpose.

#### ****2.2.2 Research Gate and Academia.edu****

These platforms host academic publications and enable authors to share their work, serving a global audience primarily of researchers. While they provide powerful access to scholarly content, they are not tailored to the needs of undergraduates or exam preparation. Their focus on peer-reviewed material rather than past examination papers, along with requirements for account creation and potential access restrictions, limits their usefulness for students seeking past exams.

#### ****2.2.3 MKU Student Portal (Current System)****

Mount Kenya University’s student portal offers access to registration, fee statements, and exam results, but it lacks a centralized repository for past exam papers across faculties and years. As a result, students often rely on peer-to-peer sharing or emails from lecturers, which are inconsistent and fragmented, making access to past papers informal and unreliable.

#### ****2.2.4 Google Drive and WhatsApp Groups****

Students often share academic files through WhatsApp groups or Google Drive links, but this approach is prone to data loss, mislabeling, and security risks. The lack of indexing, search filters, or user authentication makes it unsustainable and insecure for reliable institutional academic storage.

**Proposed System as a Solution**

The proposed **IGICUPURI App** offers a centralized, secure, and user-friendly platform where Mount kigali University lecturers can upload past examination papers, and students can easily access and study them. It addresses the challenges of inconsistent access, data loss, and informal sharing methods currently used. By using role-based access and cloud storage, it ensures only authorized users manage content while maintaining data integrity. The app features search and filter functions to enhance usability. Notifications alert students to newly uploaded papers, increasing engagement. Ultimately, IGICUPURI improves academic preparation and supports equitable access to learning resources.

## ****CHAPTER THREE: METHODOLOGY****

### ****3.0 Introduction****

This chapter presents the research methodology employed in the development of the **IGICUPURI App**, a digital platform designed to allow lecturers at Mount Kigali University to upload past examination papers and provide students with seamless access for study and revision. The methodology outlines the data collection procedures, software development lifecycle (SDLC) model, and tools used. Both qualitative and quantitative methods were adopted, including literature reviews, interviews, and surveys, to ensure the resulting application is user-centered, technically sound, and effective.

### ****3.1 Data Collection and Procedures****

To understand the needs of the target users—lecturers and students—this study used a mixed-methods approach. Primary data was collected through structured interviews and questionnaires directed at both students and faculty members. Secondary data was sourced from academic articles, institutional policies, and related education technologies. The combination of these methods provided a solid foundation for identifying functional requirements and system design.

### ****3.2 Primary Data****

#### ****3.2.1 Documentation****

The documentation phase involved reviewing internal Mount Kigali University academic policies, guidelines on exam paper handling, and existing digital learning resources. This helped in understanding the institutional framework and identifying any regulatory limitations or requirements for content management. The documentation also clarified how past exam papers were traditionally distributed and accessed.

#### ****3.2.2 Interviews****

Structured interviews were conducted with lecturers from various departments and a focus group of students. Lecturers provided insight into how they store and share past exams, while students shared challenges they face in accessing reliable study materials. These interviews revealed a significant gap in centralized academic resources and informed several core features of the app, such as department-based filtering and paper categorization by academic year.

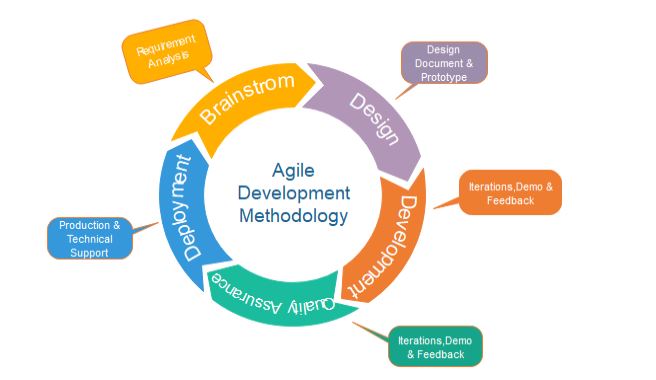
### ****3.3 Secondary Data****

Secondary data was collected from research papers, journal articles, and online reports on the implementation of academic content management systems. Studies on the use of digital libraries and mobile learning applications were also reviewed. This data provided essential information on usability, accessibility, and scalability requirements in academic environments. Furthermore, literature on data privacy, content validation, and educational technology adoption helped shape the system's security and interface design.

### ****3.4 Software Development Life Cycle (SDLC)****

The development of the IGICUPURI App followed the **Agile methodology**, known for its iterative, flexible, and user-feedback-driven nature. Agile was chosen because it allows rapid prototyping, continuous user involvement, and adaptability to changes during development. With each sprint, user feedback from both students and lecturers guided feature refinement.

### ****3.4.1 Agile Model****

The Agile model divides the development process into small, manageable cycles called sprints. At the end of each sprint, a functional module is delivered, tested, and refined based on user input. This methodology ensured that the final product was not only technically viable but also aligned with the expectations of its end-users—lecturers and students.

**Figure 3.1: Agile Model (**Agile Manifesto)

#### ****3.4.1.1 Feasibility Study****

This phase evaluated whether the IGICUPURI App could be developed and deployed within available technical, institutional, and budgetary constraints. A SWOT analysis of existing solutions (like student WhatsApp groups and departmental paper-sharing) highlighted inefficiencies and gaps. The feasibility study confirmed that a lightweight, mobile-compatible platform would be ideal for MKU's diverse student population.

#### ****3.4.1.2 Requirements and Analysis****

Requirements were gathered through user stories, stakeholder interviews, and institutional needs assessments. Functional requirements included secure upload and download of past papers, search functionality, paper categorization by course and year, and user authentication. Non-functional requirements included system scalability, platform responsiveness, data privacy, and offline access features.

#### ****3.4.1.3 Design****

The design phase involved developing the app's system architecture, user interfaces, and database schema. User flows were mapped to ensure intuitive navigation for both lecturers and students. Wireframes and UI mockups were created using tools like Figma, focusing on minimalistic design and easy accessibility. The backend was planned to support cloud storage, role-based access control, and efficient data retrieval.

#### ****3.4.1.4 Implementation****

During this phase, actual coding of the application began. The front-end was developed using **Flutter**, allowing cross-platform compatibility (Android, iOS, and Web). The back-end employed **Firebase** for real-time data management, user authentication, and secure cloud file storage. Key modules included: user registration, paper upload form, searchable archive, and admin approval interface.

#### ****3.4.1.5 Testing****

Comprehensive testing was conducted to ensure system reliability. This included:

1. **Unit Testing** for individual features (e.g., login, upload).
2. **Integration Testing** to verify how modules interact (e.g., upload-to-database-to-student-download).
3. **User Acceptance Testing (UAT)** involving a pilot group of lecturers and students to evaluate functionality, ease of use, and performance.

Feedback from this testing phase led to refinements in paper categorization and UI clarity.

#### ****3.4.1.6 Deployment****

Upon completion of testing, the app was deployed to a staging environment for institutional review. Once approved, it was prepared for official release on the **Google Play Store** and accessible via web browser for broader accessibility. Deployment involved setting up domain hosting, security certificates, and database indexing for efficient search performance.

#### ****3.4.1.7 Maintenance****

After deployment, regular maintenance ensures continuous improvement. This includes:

1. Fixing bugs reported by users
2. Enhancing performance based on analytics
3. Adding new features (e.g., marking schemes or faculty-specific dashboards)

Updating security measures to comply with MKU policy and data protection regulations

### ****3.5 Software Technologies****

The IGICUPURI App was developed using the following tools and technologies:

|  |  |  |
| --- | --- | --- |
| **Component** | **Technology** | **Purpose** |
| **Frontend** | Flutter | Cross-platform mobile/web development |
| **Backend** | Firebase Functions & Firestore | Server less backend for scalability |
| **Authentication** | Firebase Auth | Secure user login for students and lecturers |
| **Storage** | Firebase Storage | Secure, cloud-based file upload/download |
| **Database** | Firestore | NoSQL database to manage metadata (course, year, file name) |
| **APIs** | REST APIs | Enable communication between modules (if needed) |
| **Notifications** | Firebase Cloud Messaging (FCM) | To alert students when new papers are uploaded |

By integrating these technologies, the IGICUPURI App achieves a balance between performance, security, and user experience—ensuring students and lecturers have a reliable, efficient academic resource at their fingertips.

## ****CHAPTER FOUR: SYSTEM ANALYSIS AND REQUIREMENTS MODELING****

### ****4.0 Introduction****

This chapter presents the analysis of the IGICUPURI App, a system designed to facilitate the uploading and accessing of past exam papers by lecturers and students at Mount Kigali University. The focus is on identifying and modeling the system's functional and non-functional requirements, analyzing system interactions, and representing them using appropriate software engineering models. The use of **Unified Modeling Language (UML)** is essential in capturing both structural and behavioral aspects of the IGICUPURI system for effective communication between stakeholders and developers.

### ****4.1 Unified Modeling Language (UML)****

#### ****4.1.1 Introduction to UML****

The Unified Modeling Language (UML) is a standardized language used in software engineering to visualize the design of a system. It helps define the system architecture and relationships between different entities, aiding both in planning and documentation. UML is especially useful in object-oriented systems but can be adapted for various types of applications, including educational platforms like IGICUPURI. The visual diagrams created using UML serve as blueprints for the development process, ensuring that the system meets user needs and is aligned with project goals.

#### ****4.1.2 Purpose of UML****

The primary purpose of UML in the IGICUPURI project is to provide a shared understanding of the system’s architecture among developers, stakeholders, and end-users. UML diagrams offer clear insights into the structure, processes, user roles, and interactions within the app. This enhances communication, reduces development errors, and ensures the final product aligns with user expectations. It also serves as a reference during system maintenance and feature expansion phases.

#### ****4.1.3 UML Building Blocks****

UML consists of various building blocks categorized into **elements**, **relationships**, and **diagrams**.

* **Elements**: The core components representing system functions and structure.
* **Relationships**: Connect elements and define interactions, dependencies, and hierarchies.
* **Diagrams**: Visual tools to map and document how the system operates.

#### ****4.1.3.1 Elements****

* **Structural Elements**: These are the static components of the system, including user roles such as Student, Lecturer, and Admin, as well as data structures like Exam Paper, Course, and Department.
* **Behavioral Elements**: These represent dynamic actions such as Upload Paper, Search Papers, Download File, and Approve Upload.
* **Grouping Elements**: These allow the organization of related elements, such as categorizing user roles under a User Module.
* **Notational Elements**: Used to annotate or explain different parts of the system diagram for clarity.

#### ****4.1.3.2 Relationships****

* **Association**: Links between students and downloaded papers, or lecturers and uploaded content.
* **Dependency**: The student's ability to access papers depends on the lecturer uploading them.
* **Generalization**: The Admin, Lecturer, and Student may inherit from a general User class.
* **Realization**: Demonstrates how system features such as file management are implemented via specific functions.

#### ****4.1.3.3 Diagrams****

UML diagrams for the IGICUPURI system are categorized into:

* **Structural Diagrams**: Include class diagrams and component diagrams that define the system's backbone.
* **Behavioral Diagrams**: Include use case diagrams and sequence diagrams to show how users interact with the system.

### ****4.1.3.3.1 Context Diagram****

The **Context Diagram** defines the boundaries of the IGICUPURI system and illustrates how it interacts with external entities. It represents the system as a single, central process and outlines the flow of information between the system and users.

* **External Entities**:
  + Lecturers: Upload past papers.
  + Students: Search and download papers.
  + Admins: Moderate and manage content.
* **System Center**: IGICUPURI receives paper uploads, stores them, and makes them available to students.

According to Korsakoff (2011), a system context diagram should not reveal internal components but focus on the interaction between the system and its environment. For IGICUPURI, this includes academic departments, user devices (mobile/web), and cloud storage services.

ADIMIN

USER

DATA BASE

**Figure 4.1: context diagram**

**4.1.3.3.2 Dataflow Diagram (DFD)**

A Data Flow Diagram (DFD) is a graphical tool used to model the flow of data within a system, focusing on the inputs, processes, and outputs. It helps in understanding how information moves through a system and the interactions between data and processes. DFDs are commonly used in system analysis and design to visualize the data processing flow and the relationships between system components (Kendall & Kendall, 2011).

**Elements of a Data Flow Diagram:**

**Process**: Represents a function or activity that transforms inputs into outputs.

Depicted as a circle or rounded rectangle.

Process Name

**Data Store**: Represents where data is stored within the system.

Depicted as an open rectangle or two parallel lines.

Labeled with the name of the data store.

**D1**

**Data Store**

**Data Flow**: Represents the movement of data between processes, data stores, or external entities.

Depicted as an arrow.

Labeled with the data being transferred.

Moving Data

**External Entity**: Represents an outside system or person that interacts with the system.

Depicted as an oval.

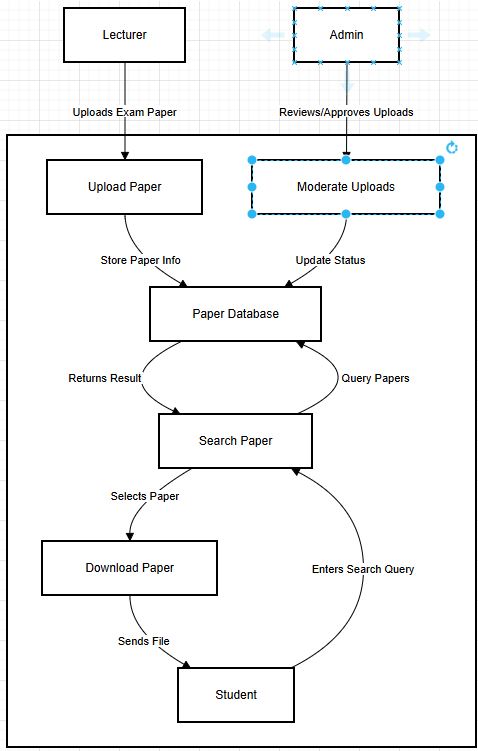
Labeled with the entity’s name.

**Types of Relationships:**

**Data Flow**: Links processes, data stores, and external entities by showing the movement of data.

**Process**: Interacts with both data flows and data stores, transforming data between them

**Data Flow Diagram for IGICUPURI APP**



**Figure 4‑1 DFD**

##### **4.1.3.3.3 Use Case Diagram**

A **Use Case Diagram** is a methodology used in system analysis to identify, clarify, and organize system requirements. It represents a set of possible sequences of interactions between systems and users in a specific environment related to a particular goal. Each use case should encompass all significant system activities related to users and can be thought of as a collection of possible scenarios associated with a specific goal (Dennis, 2012).

**Elements of a Use Case Diagram:**

**Actor:** Represents a person or system that interacts with the subject.

Depicted as a stick figure or a rectangle with <<actor>> for non-human actors.

Labeled with its role and can be associated with other actors using a specialization/superclass association.

**Use Case:** Represents a major piece of system functionality.

Can extend or include another use case.

Labeled with a descriptive verb-noun phrase and placed inside the system boundary.

**System Boundary:** Defines the scope of the subject, such as a system or an individual business or process business.

System

**Types of Relationships:**

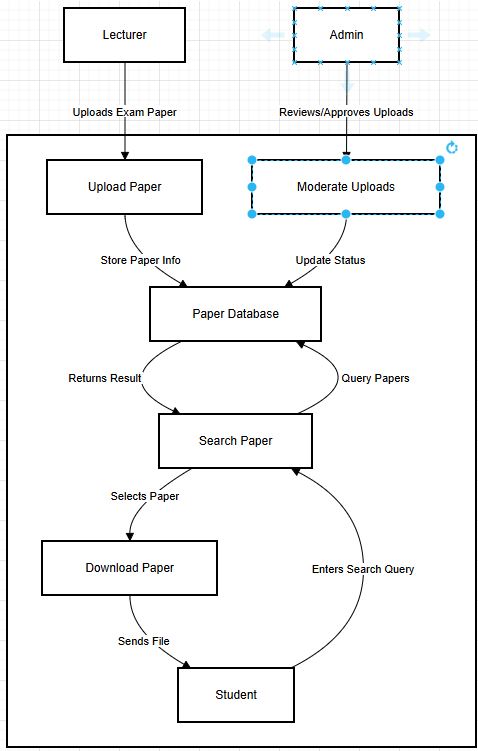
**Include Relationship:** Indicates that one use case includes the functionality of another, with an arrow from the base use case to the included use case.

**Extend Relationship:** Represents optional behavior that extends a use case, with an arrow from the extending use case to the base use case.

**Generalization Relationship:** Shows a specialized use case as a more generalized one, with an arrow from the specialized use case to the base use case.

**Association Relationship:** Links an actor with the use case(s) it interacts with.

**Use Case Diagram for IGICUPURI APP**



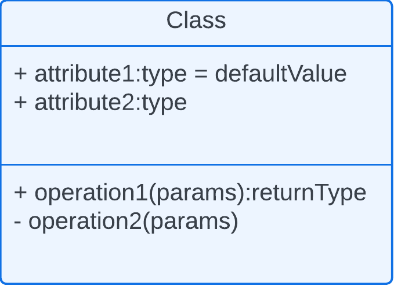
**Figure 4‑2: Use Case Diagram**

##### **4.1.3.3.4 Class Diagram**

A **Class Diagram** is a static diagram that visualizes the structure of a system by depicting classes, their attributes, operations, and relationships among objects. It provides a foundation for understanding the system's architecture and serves as a blueprint for the code implementation phase.

**Elements of a Class Diagram:**

**Class:** Represents a type of object in the system, typically with attributes (properties) and operations (methods).



Name is typed in bold and centered at the top.

Attributes are listed in the middle compartment.

Operations (methods) are listed in the bottom compartment.

**Attribute:** Describes the properties or data fields of a class.

**Operation:** Represents actions or functions that a class can perform.

Classified as constructors, queries (read operations), or updates (write operations).

**Relationships Between Classes:**

**Generalization:** Indicates that a specialized class inherits from a more general superclass.

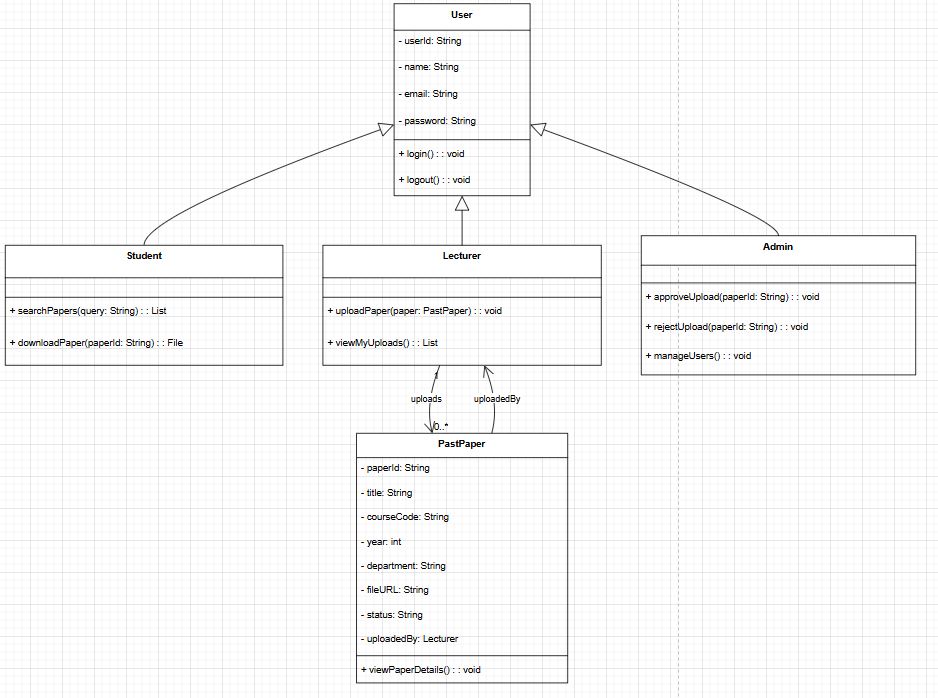
**Association:** Represents a relationship between classes, showing how objects interact with each other.

**Aggregation:** A special form of association where one class (whole) contains another class (part).

**Composition:** A stronger form of aggregation where the part is dependent on the whole and cannot exist independently.

**Dependency:** Shows that a change in one class may affect another class.

**Class Diagram for IGICUPURI APP**



**Figure 4‑3 Class Diagram**