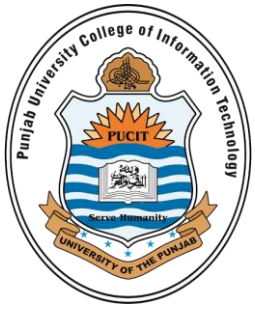
# ICE AGE:Deep Learning Based Framework for classification of Supra and Peri Glaciers using remote sensed Images




**Team ID:BSEF20M - BITF20A**

**SessionID : BSEM- BITA Fall F20**

**Project Advisor: Dr. Muhammad Nadeem Majeed**

**Submitted By**


| | |
| --- | --- |
| Muhammad Usman Khan | BSEF20M540 |
| Qurban Hanif | BSEF20M534 |
| Kainat Bashir | BITF20A503 |
| Alisha Sultan | BITF20A539 |
| Muhammad Usman Zahid | BITF20A545 |

University Of Punjab, Lahore

# STATEMENT OF SUBMISSION

This is to certify that Muhammad Usman Ghias Roll No. BSEF20M540, Muhammad Qurban Hanif Roll No. BSEF20M534, Alisha Sultan Roll No. BITF20A539, Kainat Bashir Roll No. BITF20A503 and Muhammad Usman Zahid Roll No. BITF20A545 have successfully completed their final project named as: Ice Age, at the Punjab University College of Information And Technology Punjab, Lahore to fulfill the partial requirement of the degree of Bachelors in Software Engineering.

_____

Project Office Supervisor

PUCIT, Lahore

_____

Project Primary Advisor

Dr. Muhammad Nadeem Majeed

Associate Professor

PUCIT

_____

Project Examiner

Dr. Muhammad Nadeem Majeed

Associate Professor

PUCIT

# ACKNOWLEDGEMENT

We truly acknowledge the sincere cooperation and help by Dr. Muhammad Nadeem Majeed, Associate Professor, Punjab University College of Information And Technology Punjab, Lahore. He has been a constant source of guidance throughout the course of this project. We would also like to thank Acknowledger from Designation, Address of Organization for his help and guidance throughout this project and the institute PUCIT. We are also thankful to our friends and families whose silent support led us to complete our project. We are also thankful to Allah Almighty.

Date:
May 30, 2024

# ABSTRACT

The project tackles automated identification of debris-covered glaciers, a task traditionally requiring extensive manual effort. By combining deep learning with various data sources like satellite imagery, the system can analyze glacier landscapes and differentiate debris-covered areas from clear ice. This not only saves time and resources but also improves the objectivity and consistency of glacier mapping, providing valuable data for monitoring glacier health in the face of climate change.

# TABLE OF CONTENTS

# 1 Introduction

The role of glaciers in climate and water, especially in the Himalaya-Karakoram (HK) area, is critical given that HK has some of planet's largest and growing glaciers. Offshore glaciers are imperative sources of fresh water, food, and hydropower for millions of inhabitants dwelling downstream from these glaciers. However, conventional glacier mapping techniques are tedious and they involve a lot of manual work, in addition, they give imprecise results because of the confusion that usually comes with the similarity of spectral properties between both supraglacial debris (SGD) and the periglacial debris (PGD).

In order to meet these issues, **Ice Age,** our project, conceptualizes the automated system to detect and map debris containing glaciers. Ice Age uses deep learning and multisource remote sensing data and to delineate accurate glacier boundary, fully connected SGDNet is proposed and used.

Indeed, there is much that can be gained from this automated detection system. From the highly specific and accurate depiction of debris covered glaciers, Ice Age can greatly benefit and improve learning about the movement of glaciers and add value to the overall cause of combating and addressing climate change. Glacier maps will also assist with planning and risk mitigation in regard to other possible natural disasters like glacial flood or landslides and thus safeguard people and facilities within endangered areas. Further enhanced glacier observation will facilitate better water resources management, availability and usage of fresh water and in overall efficiency of hydro power plants in HK region.

## 1.1 Project/Product Feasibility Report

### 1.1.1 Technical Feasibility

Technical Feasibility refers to fundamental decisions that require answering questions related to a project's development possibilities. It is based on evaluating potential theoretical and technological limitations which may detrimental to the project. This stage is of utmost importance since it defines the feasibility of the suggested system or solution. Key considerations include:

**System Development Potential:** This aspect considers whether the proposed system is implementable based on the current technological as well as theoretical standards of possibilities. It evaluates on how viable it is to set up the intended functionalities and

features using the available technologies.

**Theory and Technology Application Limits:** These are some of the fundamental factors that project planners need to consider for them to understand the appropriate theory and technology that should be applied on any development. It involves outlining and studying the processes, methods, algorithms, and ways of computation in order to pursue the goals set in the course of a project.

**Availability of Technical Expertise:** Another vital consideration in the decision-making process is whether the project team has skills in the technical field to implement the development. They should be conversant with the sort of programming languages, frameworks, and tools that would be useful for developing the system and installing it properly.

In the case of the **"Ice Age"** automated debris-covered glacier detection system, the technical feasibility is supported by:

**Use of Advanced Deep Learning Technologies:** The elaboration of deep neural networks including SGDNet in the context of this research proves the capacity to manage and analyse multisource remote sensing data for accurate glacier mapping using TensorFlow and Keras in Python environment.

**Python-Based Backend and JavaScript Framework for Frontend:** For backend operations, Python language will be used, and for the frontend, Owl Framework written in JavaScript (e. g. Functional requirements (Lookup, entry, User registration, User login, Home page, Image upload and Categorization, Result display) provide a strong structure and proper data management & handling along with the interactivity with the users.

**Integration of Multispectral and Multiresolution Remote Sensing Data:** An integration of various data sources from Sentinel-2 MSI, Sentinel 1 SAR, Landsat 8 Thermal Infrared as well as ALOS DEM demonstrates the effectiveness of the system, which is able to organize and process plural geospatial details for exhaustive glacier examination.

**Optimization and Performance Evaluation:** The use of efficient and standard methods like standardization of the data, data augmentation, and adaptive learning rate optimization with the Adam optimizer provide a proper model training it also upheld to proper performance tests using different standard metrics.

**Visualization and Reporting Capabilities:** The usage of Matplotlib and pandas for plotting training metrics and generating reports would help the stakeholders analyze and encourage the usage of the system outputs.

Lastly, analyzing technical feasibility is critical in the assessment of the practicability and success prospects for the Ice Age system implementation, with the initiative conforming to set objectives and utilizing existing toolsets and skills to generate significant

improvements in glacier automated mapping.

### 1.1.2 Operational Feasibility

**Evaluation of Staff Technical Ability**
The technical practicality of the Ice Age project plan focuses on the evaluation of the functionalities of the staff that will be responsible for the project as well as the functionality of the technology that will be infused into the project. This evaluation checks on the ability of the team members to manage and drive the deep learning models, Incorporate the multisource remote sensing data and analyze the outcomes correctly.

**Worthiness of the Problem and Solution**
A part of the operational feasibility is to find out whether the problem tackled by the "Ice Age" project, namely debris-covered glacier monitoring, is really important to be deserving to integrate much of technologies. Furthermore, it assesses whether all the offered solutions including the SGDNet model and application framework, play a positive role in addressing all the envisaged environmental issues in the Himalaya-Karakoram region.

**Stakeholder Perception**
Another aspect of operational feasibility is identifying the perceptions of all the relevant end-users in the organization as well as the manager on the problem statement and its solutions. Development of these envisaged end-user perspectives contributes significantly to fine tune the goals of the project and achieve greater conformity to meet the end-users' requirements thereby helping to optimise the overall improvement and acceptability of the project.

By performing this operational feasibility assessment within the framework of the "Ice Age" project, aspects beyond technical applicability of the methods are covered, such as practicality, relevance, and potential impact as perceived by the stakeholders, this provides the basis for a successful implementation and sustainability of the project.

### 1.1.2 Economic Feasibility

The economic analysis of the Ice Age project cannot be a OPTIONS without considering the feasibility of its benefit-cost narration. This evaluation involves two essential components: The first and second elements of the program are cost estimates and benefit estimates.

**Cost Estimates**
Cost estimates are divided into two main categories:

**Development or Acquisition Costs (One-time):** This entails the financial resources and capital that is needed at the start or at the beginning to set up or purchase basic structures,

equipment, and skills that will be crucial in the execution of the Ice Age project. It is a technique of designing the project in incremental phases and then using models to calculate costs as closely as possible. These estimates are benchmarks with historical data of such similar projects.

**Maintenance and Operation Costs (Ongoing):** These are the costs that are incurred frequently throughout the lifecycle of the Ice Age project, as the project requires funds to ensure that it is maintained and updated, as well as to secure support and operations. Calculating these costs entails projecting the amount of money that will be required to keep the project running incessantly with efficiency in the future.

**Benefit Estimates**
Benefit estimates encompass both tangible and intangible benefits:

**Tangible Benefits:** Some of tangible advantages that can be attributed to the effective management including the following; The cost reduction, the increase in revenues, and the overall efficiency. As for Ice Age, the persons may benefit in terms of improved practices of monitoring glaciers and more effective resource organization.

**Intangible Benefits:** These are all beneficial although they are difficult to assign actual monetary values. Prominent examples include increased precision of data, improved decision-making heuristics, satisfaction among the employees, and strengthening the reputation for monitoring the environment.

Hence, in the context of Ice Age, economic feasibility can be understood in terms of costs and benefits that need to be reckoned with. This way, aspects like costs of production, expected revenue and that of the returns can help stakeholders make wise decisions on the feasibility of the project.

### 1.1.4 Schedule Feasibility

Project planning and management are some of the key components, and this cannot fail to mention the time factor in relation to Ice Age. Schedule feasibility is about evaluating the possibilities of executing the project plan from the time estimations table within the necessary time with the required endowment and personnel. It therefore requires a scenario per phase in order to enable one to set realistic time horizons to be met during the project life cycle. These performance indicators, therefore, include the optimization of available time, organization of work tasks, resources, and timeframes for monitoring project performance. To that end, Ice Age will ensure that the schedule proposed in the current project plan is as realistic and achievable as possible, which will help it attain its objective of efficiency and timely delivery.

### 1.1.5 Specification Feasibility

Specification feasibility assesses the clarity, completeness, and boundaries of project requirements:

**Clarity of Requirements:**

**System Features:** When defining the features that the system ought to consist of such as traces of glacier boundary detection, classification, and monitoring, then one should do this with clarity.

**Data Requirements:** Describe the types and sources of multispectral imagery required for conclusion-making related to glaciers.

**User Interaction:** Identify interface features working as user attractions (login button, dashboard button, image array input) and features relevant to the system operation.

**Scope Boundaries Assessment:**

**Project Scope:** Demarcate line of viable action Train the highlight on automating glacier surveillance within the region encompassing Himalaya-Karakoram Himal using deep learning.

**Exclusions:** Code redundancy, explicitly marking out other features or functionalities that are not needed presently or in the immediate future.

**Integration Points:** Point out here on what aspects or process should the client focus when connecting with another system or data (e. g. Also included in the package are the required prerequisites for comprehensive glacier evaluation, such as climatic data base and geographic databases.

## 1.1.6 Information Feasibility

In our project information feasibility examines data credibility, relevancy and accuracy as key ways through which we measure the appropriateness of data in meeting certain project objectives. It includes determining the missing and redundant information, evaluating the credibility of sources, and checking if data collected corresponds to the project objectives. Reliability encompassing the feasibility of information also defines consistencies and timeliness which are important factors influencing social decisions and positive outcomes while implementing a project.

## 1.1.7 Motivational Feasibility

Motivational feasibility assesses the readiness and commitment of client staff to effectively and promptly execute necessary project tasks:

**Employee Engagement:** Assess the amount of interested and passionate the staff of the client company is towards the new technology that will enable glacier surveillance automation.

**Skill Acquisition:** Evaluate readiness of the staff for developing relevant competencies regarding the manifestations of the features, such as deep learning interpretation and geospatial analytics.

**Change Management:** Assess the readiness of staff to embrace of new working techniques and process brought about by the automated system.

**Leadership Support:** Assess the level of organizational support and promotion that leaders extend in order to enhance the success of the system adoption and incorporation into organizational processes.

**Training and Development:** Establish the attitudes of the staff towards continued training and development programs for improving their proficiency in deploying the system for monitoring and analyzing glaciers.

### 1.1.8 Legal & Ethical Feasibility

Ice Age project entails careful consideration of legal and ethical aspects to ensure compliance and ethical conduct throughout its operations:

**Data Privacy and Security:** Ice Age complies the policy of privacy keeping the identity of the users, the multispectral imagery and the geospatial data secret. Amidst changing trends and innovative modes of operating, measures such as encryption and access control are put in place to ensure that sensitive information is not prone to malicious attacks or intrusion.

**Intellectual Property Rights:** Copyright considerations are held in high regard due to the position of the movie in the animation business. The use of any proprietary algorithms, models, or data processing techniques is as per the classified level. It is crucial to follow the rules of data sourcing and annotate any original ideas: contracts with data sources and partners help to stipulate all the rights for using the intellectual property.

**Ethical Guidelines:** In this light, Ice Age does provide an ethical framework for the use of the kind of environmental technology the company employs when managing the climate. This touches on issues of accountability where data collection methods must be disclosed while data utilization must be done after seeking consent from the concerned parties where necessary; and where the project results in effects on scientific research or environmental conservation, the impacts must be positive.

**Liability and Risk Management:** The possibility of such a threat to exist is unusual but it is identified and managed in the project for automated data processing and interpretation. Unintended storage and usage liabilities are mitigated through contractual

agreements with all relevant stakeholders, as well as the users of the system in terms of their obligations and the capabilities and limitations of the system.

**Environmental Impact:** Through its mission of presenting updated research information on glacier evaluations, Ice Age is aimed at contributing to environmentalism. Some of the limitations are on ethical grounds such as reduction of the impact that is made on environment during conducting the study and ensuring that the project is sustainable in an aspect that is in harmony with the principles of conservation.

**Compliance:** Ice Age currently respects state and national laws as well as international laws that have been put in place concerning environmental auditing, data privacy, and information technology regulation as a business organization. The implementation of audit and reviews is useful to check for compliance regularly besides adapting to changes in laws.

**Stakeholder Engagement:** Ice Age always involves stakeholders such as various communities, researchers, and governments in matters of moral concern and success in the decision making process. The present AAI approach leads to the development of trust, effective communication and understanding of each other in relation to the project implementation and utilization of the results.

## *1.2 Project/Product Scope*

The automated debris glacier detection system is a powerful new tool to re-invigorate glacier mapping by providing deep learning and multisource remote sensing data. The specification of requirements identifies the elements of the system to be developed, its constraints, and the measures of effectiveness, which offers a schematic plan for work.

**System Functionalities**
The core functionalities of the system encompass:

**Data Acquisition:**

- **Data Sources:** It will also establish a connection with reliable remote-sensing data suppliers such as European Space Agency's Sentinel-2 by integrating multisource data including Landsat-8 and ASF Alaska.

- **Data Types:** The system will establish a set of diverse data types, as pointed out below:

    - Optical data (visible, near-infrared, shortwave infrared) for spectral information
    - Radar data (coherence images) for detailed surface structure analysis (Subject to data availability)
    - Elevation data (ALOS DEM) for accurate terrain representation

- Slope data for understanding topographic variations

**Deep Learning Model (SGDNet):**

- **Pre-trained Model Integration:** Specifically, the system will build on the pre-trained deep neural network (DNN), named Seasonal Glacier – Debris Net (SGDNet), which has been demonstrated effective for debris-covered glacier mapping.

- **Model Functionality:** In this respect, multisource data shall be obtained Some of the multisource data include; SGDNet shall be used to differentiate the acquired multisource data between supraglacial debris (SGD) and periglacial debris (PGD).

**Prediction and Output:**

- **User Interface:** Web based graphical user interface shall be designed using a tool called Owl Framework which is a JavaScript based tool for providing smooth and easily navigable interfaces.

- **Frontend Features:** The basic actions that will be supported by the web interface include the following:

  - One of the most vital features that any website should incorporate is the ability to signup and Login to offer client a secure user access.
  - Application for users: Personal control panel that allows easy managing of tasks and data.
  - Upload an Image for Glacier Analysis for any user to upload any new imagery related to glaciers.
  - In order to present the results in a more easily understandable fashion it was necessary to visualize the shape and boundary of the predicted glacier.
  - Functionality to output the results (the suggested glacier delineation) in raster format for further processing.

**Backend Integration:**

- **Backend Development:** The back-end part will be created with Python and implemented with the use of the Odoo 17 Enterprise Framework to guarantee the efficient and reliable development.

- **Backend Responsibilities:** Some of what the backend will address include:

  - Data storage and management.

- This is more feasible if the developed system will be integrated with remote sensing data sources.
- In this case, the evaluation involves interacting with a pre-trained SGDNet models to get further analysis.
- Processing and visualization of predicted glacier boundaries.
- Account management / User management.

**System Limitations**

While the system offers robust functionalities, it is crucial to acknowledge its limitations:

- **Data Dependency:** The system's performance is inherently dependent on the availability and quality of multisource remote sensing data. Data gaps or inconsistencies may impact prediction accuracy.
- **Current Scope Exclusions:** The current scope excludes certain functionalities, including:
    - Terrain correction of input data for enhanced accuracy
    - Uncertainty quantification in model predictions to assess confidence levels
    - Advanced data visualization tools beyond basic boundary maps for comprehensive analysis

**Future Enhancements**

As the project progresses, potential enhancements include:

- **Data Integration:** Expanding data sources to include additional remote sensing data or in-situ observations for improved accuracy.
- **Model Development:** Investigating the development of an in-house deep learning model tailored to specific user requirements or regional characteristics.
- **Advanced Functionalities:** Implementing terrain correction and uncertainty quantification functionalities to enhance data quality and reliability.
- **Visualization Tools:** Creating advanced data visualization tools, such as 3D models and interactive maps, for comprehensive glacier analysis and communication.

**Exclusions**

To ensure project focus and feasibility, certain aspects are excluded from the current scope:

- **Mobile Application Development:** While the system will prioritize a

responsive web interface built with Owl Framework and Odoo, a mobile application may be considered for future enhancements.

- **On-site Data Collection:** The system relies on pre-existing remote sensing data sources and does not encompass on-site data collection capabilities.

**Success Criteria**

The project's success will be evaluated based on the following criteria:

- **Functional Implementation:** Successful development of a user-friendly web interface that enables seamless data upload, visualization, and export of predicted glacier boundaries.
    - Responsive design across different devices
    - Integration of Signup, Login, Dashboard, and other user-centric features
- **Model Integration:** Effective integration of the pre-trained SGDNet model to accurately detect debris cover and map glacier boundaries.
- **Backend Integration:** Robust backend development using Python and Odoo for data management, model interaction, and result processing.

## *1.3 Project/Product Costing*

Financial planning is crucial for my automated debris glacier detection system. Key costs include developing the web interface and backend using Owl Framework and Odoo, potential fees for remote sensing data access (Sentinel-2, Landsat-8, ASF Alaska), and cloud hosting fees for system operation. To estimate the overall budget, I'll consider expert advice, industry benchmarks, and a breakdown of individual development and infrastructure costs.

### 1.3.1 Project Cost Estimation by using COCOMO’81 (Constructive Cost Model)

Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since ‘functionality’ cannot be measured directly, it must be derived indirectly using other direct measures. Function-oriented metrics were first proposed by Albrecht, who suggested a measure called the function point. Function points are derived using an empirical relationship based on countable (direct) measures of software’s information domain and assessments of software complexity.

Function Point Analysis can provide a mechanism to track and monitor scope creep. Function Point counts at the end of requirements; analysis, design, code, testing, and implementation can be compared. The function point count at the end of requirements

and/or designs can be compared to function points actually delivered. If the project has grown, there has been scope creep. The amount of growth is an indication of how well requirements were gathered by and/or communicated to the project team. If the amount of growth of projects declines over time it is a natural assumption that communication with the user has improved.

To compute function points (FP), the following relationship is used:

**FP est.=Count Total×[0.65+0.01×(ΣF i)]**

Where count total is the sum of all FP entries obtained from the table below is the value adjustment factor (VAF) based on 14 general system characteristics (GSCs) that rate the general functionality of the application being counted. Each characteristic has associated descriptions that help determine the degrees of influence of the characteristics. The degrees of influence range on a scale of zero to five, from no influence to strong influence.

**Calculations**

**Type of Components and Complexity:**

| **Type of Component** | Complexity of Components | Total |
| --- | --- | --- |
| | Low | Average |
| External Inputs | | 5 x 4 = 20 |
| External Outputs | | 1 x 5 = 5 |
| External Inquiries | | |
| Internal Logical Files | | 2 x 7 = 14 |
| External Interface Files | | 2 x 7 = 14 |
| Total Number of Unadjusted Function Points | 53 | |
| Multiplied Value Adjustment Factor | [0.65 + 0.01 * Σ(Fi)] | |
| Total Adjusted Function Points | Calculated Below | |

**Total Number of Unadjusted Function Points: 53**

**Value Adjustment Factor (VAF):**

Given 14 General System Characteristics (GSCs), assume moderate influence for each characteristic:
If we assume each characteristic has an influence degree of 3 (mid-scale for 0-5 range), then Σ(Fi) = 3 * 14 = 42

**Function Point Adjustment Formula:**

**FP est.=53×[0.65+0.01×42]=53×[0.65+0.42]=53×1.07=56.71**

**Thus, the Total Adjusted Function Points are 56.71.**

**Additional Calculations**

Using the formulae provided:

1. **Cost / FP:**

$$\text{Cost per FP} = \frac{\text{Productivity Parameter}}{\textit{Labour Rate}}$$

2. **Total Project Cost:**

$$\text{Total Project Cost} = 56.71 \times (\text{Cost per FP})$$

3. **Total Estimated Efforts:**

$$\text{Total Estimated Effort} = \frac{56.71}{\text{Productivity Parameter}}$$

## *1.4 CPM - Critical Path Method*

The Critical Path Method (CPM) is a project management tool that represents the sequence of activities and events in a project network. The CPM helps in:

- Providing a visual representation of the project timeline.
- Estimating the time required for project completion.
- Identifying the tasks that are critical to maintaining the project schedule.

For our project, the CPM analysis reveals the following schedule, with a total duration of approximately 15.5 months:

| Activity | Task Name | Immediate Predecessor | Duration (Months) |
| --- | --- | --- | --- |
| **T-1** | Group Meetings | None | 0.5 |
| **T-2** | Meeting with Supervisor | T-1 | 0.5 |
| **T-3** | Project Proposal | T-2 | 1 |
| **T-4** | Requirements Gathering | T-3 | 1.5 |
| **T-5** | Feasibility Analysis | T-4 | 1 |
| **T-6** | Define Project Scope | T-5 | 0.5 |
| **T-7** | Market Research | T-6 | 1 |
| **T-8** | Cost & Benefit Analysis | T-7 | 0.5 |
| **T-9** | Develop Signup Interface | T-8 | 1 |
| **T-10** | Develop Login Interface | T-8 | 1 |
| **T-11** | Develop Image Upload | T-8 | 0.5 |
| **T-12** | Develop Results Show | T-11 | 0.5 |
| **T-13** | Import Libraries | T-9, T-10 | 0.5 |
| **T-14** | Import Dataset | T-13 | 0.5 |
| **T-15** | Reshape Data | T-14,T-13 | 0.5 |
| **T-16** | Split Dataset | T-15 | 0.5 |
| **T-17** | Scale Data | T-16 | 0.5 |

| | | | |
| --- | --- | --- | --- |
| **T-18** | Define Model | T-17 | 1 |
| **T-19** | Train Model | T-18 | 1 |
| **T-20** | Evaluate Model | T-19,T-18 | 1 |
| **T-21** | Predict on New Data | T-20 | 0.5 |
| **T-22** | Export Results | T-21 | 0.5 |
| **T-23** | Analyze Results | T-22 | 0.5 |
| **T-24** | Write Documentation | T-23 | 0.5 |
| **T-25** | Save Model | T-24 | 0.5 |
| **T-26** | Mobile App Development | T-20 | 3 |

| Activity | Task Name | Duration (Months) | ES | EF | LS | LF | TS | FS |
|---|---|---|---|---|---|---|---|---|
| **T-1** | Group Meetings | 0.5 | 0 | 0.5 | 0 | 0.5 | 0 | 0 |
| **T-2** | Meeting with Supervisor | 0.5 | 0 | 0.5 | 0 | 0.5 | 0 | 0 |
| **T-3** | Project Proposal | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| **T-4** | Requirements Gathering | 1.5 | 1 | 2.5 | 1 | 2.5 | 0 | 0 |
| **T-5** | Feasibility Analysis | 1 | 2.5 | 3.5 | 2.5 | 3.5 | 0 | 0 |
| **T-6** | Define Project Scope | 0.5 | 3.5 | 4 | 3.5 | 4 | 0 | 0 |
| **T-7** | Market Research | 1 | 4 | 5 | 4 | 5 | 0 | 0 |
| **T-8** | Cost & Benefit Analysis | 0.5 | 5 | 5.5 | 5 | 5.5 | 0 | 0 |
| **T-9** | Develop Signup Interface | 1 | 5.5 | 6.5 | 5.5 | 6.5 | 0 | 0 |
| **T-10** | Develop Login Interface | 1 | 5.5 | 6.5 | 5.5 | 6.5 | 0 | 0 |
| **T-11** | Develop Image Upload | 0.5 | 5.5 | 6 | 5.5 | 6 | 0 | 0 |
| **T-12** | Develop Results Show | 0.5 | 5.5 | 6 | 5.5 | 6 | 0 | 0 |
| **T-13** | Import Libraries | 0.5 | 6.5 | 7 | 6.5 | 7 | 0 | 0 |
| **T-14** | Import Dataset | 0.5 | 7 | 7.5 | 7 | 7.5 | 0 | 0 |
| **T-15** | Reshape Data | 0.5 | 7.5 | 8 | 7.5 | 8 | 0 | 0 |
| **T-16** | Split Dataset | 0.5 | 8 | 8.5 | 8 | 8.5 | 0 | 0 |
| **T-17** | Scale Data | 0.5 | 8.5 | 9 | 8.5 | 9 | 0 | 0 |
| **T-18** | Define Model | 1 | 9 | 10 | 9 | 10 | 0 | 0 |
| **T-19** | Train Model | 1 | 10 | 11 | 10 | 11 | 0 | 0 |
| **T-20** | Evaluate Model | 1 | 11 | 12 | 11 | 12 | 0 | 0 |
| **T-21** | Predict on New Data | 0.5 | 12 | 12.5 | 12 | 12.5 | 0 | 0 |
| **T-22** | Export Results | 0.5 | 12.5 | 13 | 12.5 | 13 | 0 | 0 |
| **T-23** | Analyze Results | 0.5 | 13 | 13.5 | 13 | 13.5 | 0 | 0 |
| **T-24** | Write Documentation | 0.5 | 13.5 | 14 | 13.5 | 14 | 0 | 0 |
| **T-25** | Save Model | 0.5 | 14 | 14.5 | 14 | 14.5 | 0 | 0 |
| **T-26** | Mobile Application Development | 1 | 14.5 | 14 | 15.5 | 15 | 0 | 0 |

## Network Diagram

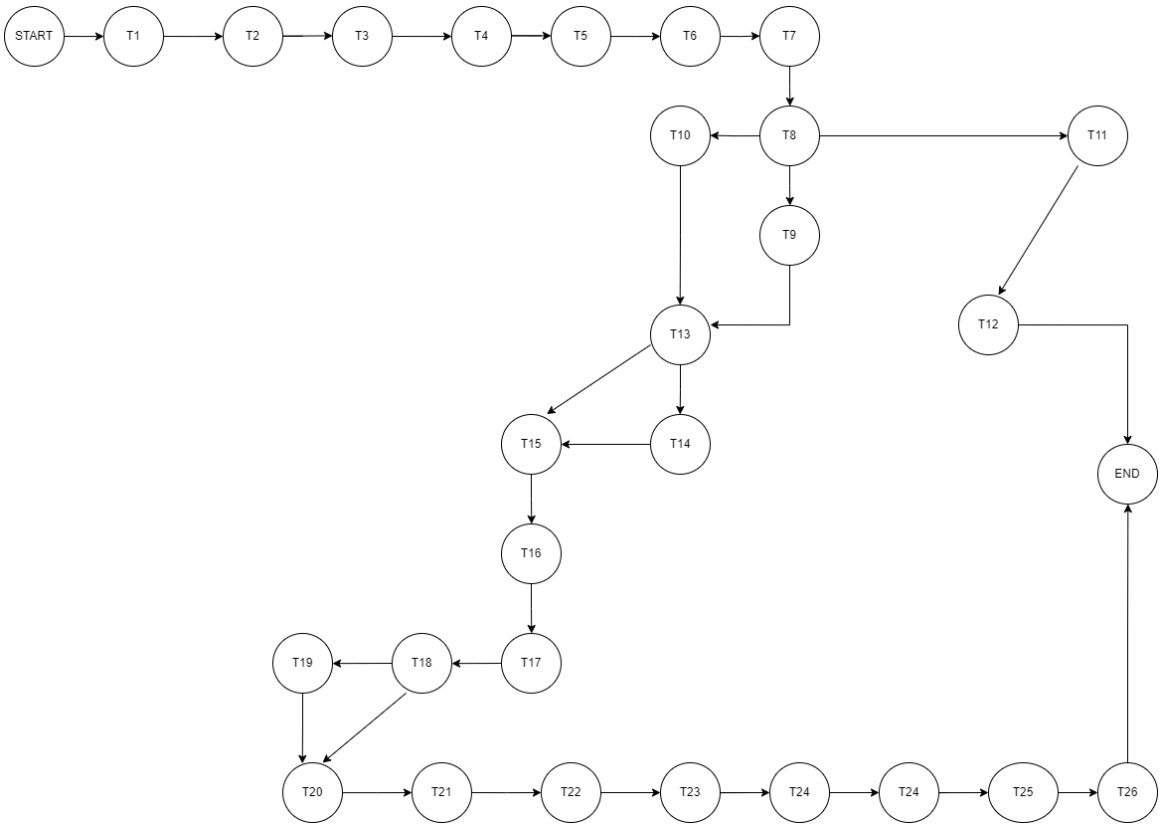



**Figure 1: Network Diagram**

## NETWORK DIAGRAM (CPM)

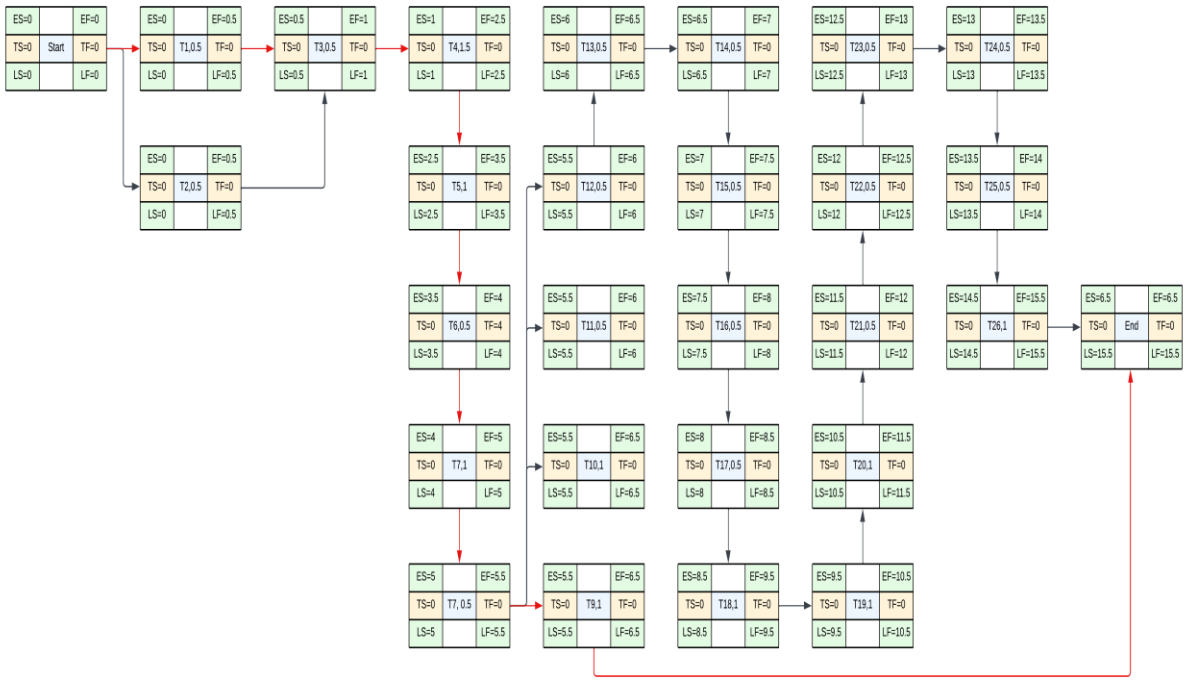



**Figure 2: Network Diagram (CPM)**

### Critical Path Analysis for Project Timeline of Approximately 15.5 Months:

### Possible Critical Paths:

1. **T-1 → T-2 → T-3 → T-4 → T-5 → T-6 → T-7 → T-8 → T-9 → T-13 → T-14 → T-15 → T-16 → T-17 → T-18 → T-19 → T-20 → T-21 → T-22 → T-23 → T-25 → T-26**
   - Total Duration: 0.5 + 0.5 + 1 + 1.5 + 1 + 0.5 + 1 + 0.5 + 1 + 0.5 + 0.5 + 0.5 + 0.5 + 1 + 1 + 1 + 0.5 + 0.5 + 0.5 + 0.5 + 3 = 15.5 months
2. **T-1 → T-2 → T-3 → T-4 → T-5 → T-6 → T-7 → T-8 → T-9 → T-13 → T-14 → T-15 → T-18 → T-19 → T-20 → T-21 → T-22 → T-23 → T-25 → T-26**
   - Total Duration: 0.5 + 0.5 + 1 + 1.5 + 1 + 0.5 + 1 + 0.5 + 1 + 0.5 + 0.5 + 0.5 + 1 + 1 + 1 + 0.5 + 0.5 + 0.5 + 0.5 + 3 = 15.5 months
3. **T-1 → T-2 → T-3 → T-4 → T-5 → T-6 → T-7 → T-8 → T-9 → T-13 → T-15 → T-18 → T-19 → T-20 → T-21 → T-22 → T-23 → T-25 → T-26**
   - Total Duration: 0.5 + 0.5 + 1 + 1.5 + 1 + 0.5 + 1 + 0.5 + 1 + 0.5 + 0.5 + 1 + 1 + 1 + 0.5 + 0.5 + 0.5 + 0.5 + 3 = 15.5 months

## *1.5 Gantt chart*

Use MS Project to create a Gantt chart visualizing our glacier detection system's development timeline, including task durations and dependencies. The Gantt chart, built in MS Project, will map project tasks, their durations, and crucial milestones for our automated debris glacier detection system.

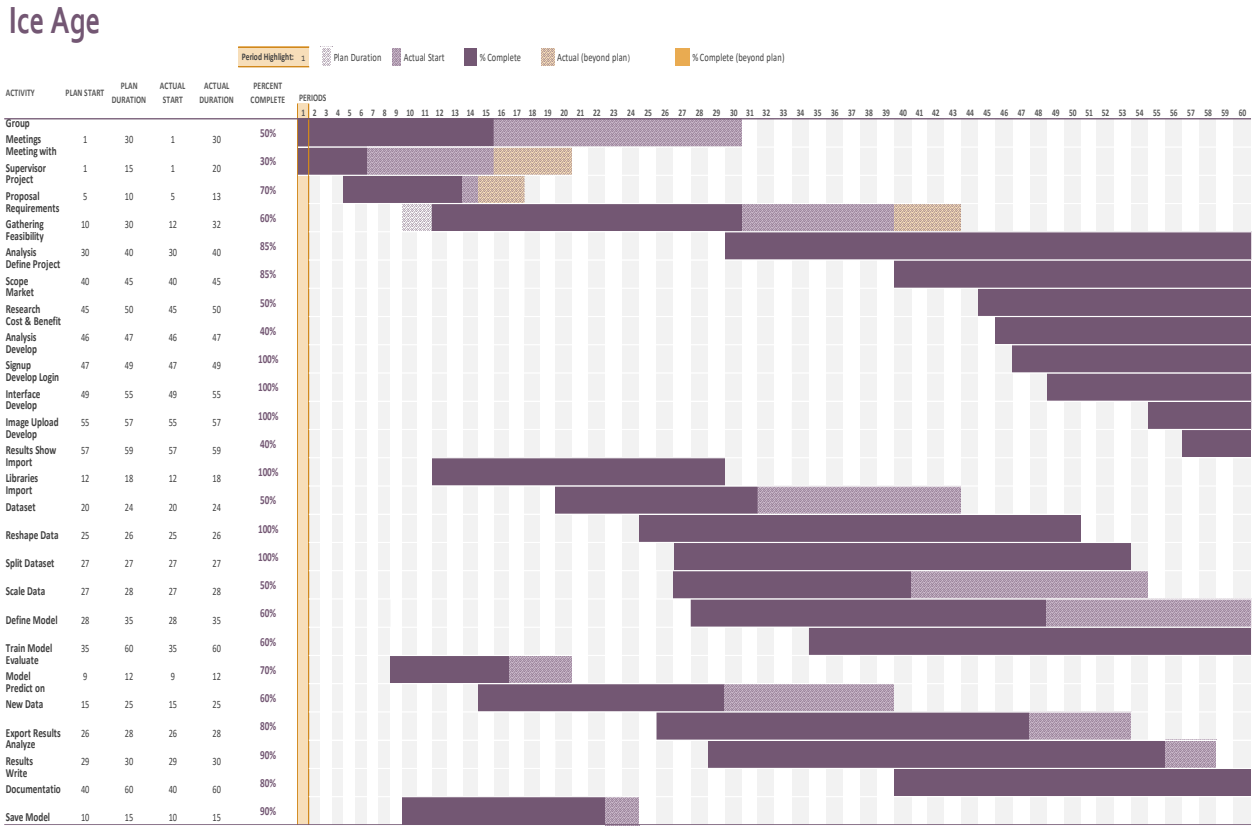



**Figure 3: Gantt Chart**

## *1.6 Introduction to Team member and their skill set*

The automated debris glacier detection system takes flight on the wings of a dedicated and skilled team. Each member brings a unique melody to the development symphony, ensuring a harmonious and successful outcome.

- **Muhammad Usman Ghias: The Deep Learning Maestro:** A seasoned developer, Muhammad Usman wields the power of deep learning in Python. His mastery extends to Odoo, a robust open-source framework for business applications. We can expect him to orchestrate the backend development, particularly the seamless integration of the pre-trained SGDNet model for accurate debris cover detection.
- **Alisha Sultan: The User Interface (UI) & Deep Learning Alchemist:** Alisha is a multi-talented developer who excels in both frontend development and deep learning using Python. Her expertise in crafting user interfaces (UI) and her keen understanding of UX principles, honed through her experience with Figma, will be invaluable in composing an intuitive and user-friendly web interface. Additionally, her deep learning skills with Python allow her to contribute to potential improvements within the model itself. This ensures a captivating and user-centered experience while also exploring optimizations for the deep learning aspects of the system.
- **Kainat Bashir: The Versatile Developer:** Kainat possesses a strong foundation in deep learning using Python alongside the ability to develop captivating frontends. This versatility allows her to contribute to both the backend model integration and the user interface development. Her involvement can create a cohesive experience, where the system's functionality and user interaction flow seamlessly together. We can also highlight her expertise in Python by mentioning she can leverage it for various tasks within the system, not just deep learning.
- **Qurban Hanif: The Full-Stack Alchemist:** While Qurban's primary expertise lies in ReactJS, his proficiency in Python suggests the potential for a full-stack development role. He can leverage his Python skills to contribute to both frontend and backend aspects, utilizing Python's versatility for various tasks within the system. Imagine him as the alchemist, transforming raw code into a functional and robust system.
- **Muhammad Usman Zahid: The Infrastructure Architect:** A crucial member for building a strong foundation, Muhammad Usman boasts expertise in computer networks, Python, and AWS (Amazon Web Services). His knowledge will be instrumental in designing and managing the cloud resources for system operation. This ensures scalability, reliability, and accessibility for users across different locations. Think of him as the architect, ensuring the system has a solid and scalable infrastructure to operate upon.

## *1.7 Tools and Technology with reasoning*

The automated debris glacier detection system leverages a strategic combination of tools and technologies to ensure efficient development and functionality across various platforms. Let's delve into the rationale behind our choices:

**Understanding Needs and Constraints:**

- **Development Process (Iterative):** We plan to adopt an iterative development process, fostering continuous improvement through frequent user feedback. This necessitates tools that support:
    - **Version Control System (Git):** Git will be our lifeline, meticulously tracking code changes, managing different versions, and facilitating seamless collaboration throughout the development process. GitHub, a popular Git hosting platform, will provide a central repository for code, issue tracking, and collaboration features.
    - **Automated Testing Framework (Pytest):** Pytest, our automated testing champion, will streamline unit and integration testing. It will catch bugs early and prevent regressions as we iterate on the system.
- **Target Platforms:**
    - **Web Application (Owl):** A web application, built with the lightweight Owl framework (JavaScript), is our primary platform. Owl's focus on minimal dependencies and responsive design guarantees a smooth and user-friendly experience across various devices.
    - **Optional Mobile Application (Flutter):** Depending on project scope and resources, we might explore developing a mobile application using Flutter. This cross-platform framework from Google allows us to create native-looking mobile experiences for both Android and iOS with a single codebase, enhancing accessibility for users on the go.
- **Programming Languages:**
    - **Python:** Python serves as the foundation of our system due to its:
        - **Readability:** Clear and concise syntax fosters collaboration and simplifies maintenance.
        - **Extensive Libraries:** Python boasts a rich ecosystem of libraries that empower us:
            - **Deep Learning (Scikit-learn, TensorFlow, PyTorch):** To leverage pre-trained models like SGDNet or potentially develop custom models for debris cover detection in the

future.

- **Data Manipulation (NumPy, Pandas):** To efficiently handle and analyze multisource remote sensing data.
- **Web Development (Django, Flask - Backend; Owl - Frontend):** To build robust backend functionalities and a user-friendly web interface.

❖ **Existing Tools:**

- **Pre-trained SGDNet Model:** We'll capitalize on this pre-trained deep learning model for debris cover detection. This eliminates the need to develop a model from scratch, saving valuable time and resources.

❖ **Development Team Distribution:**

- **Potential Geographic Dispersion:** If our team is geographically scattered, we'll bridge the distance with:
  - **Collaboration Tools (Slack, Microsoft Teams):** These tools will keep communication flowing, streamline project management, and enable real-time collaboration despite physical separation.

❖ **Project Size:**

- **Open-Source (Git, Pytest, Owl):** Open-source tools optimize budget allocation for core functionalities.
- **Cost-effective Cloud Solutions (AWS, depending on usage):** We'll leverage cloud services efficiently to manage infrastructure costs.

**Additional Tools:**

❖ **QGIS:** QGIS, a free and open-source Geographic Information System (GIS) software, might be instrumental in:

- **Data Preprocessing:** Preparing and analyzing geospatial data (e.g., satellite imagery) relevant to glacier cover and debris detection.
- **Visualization:** Visualizing and exploring geospatial data to aid in understanding glacier dynamics and model development.

❖ **PyCharm:** PyCharm, a powerful Integrated Development Environment (IDE) specifically designed for Python, can significantly enhance our development experience by:

- **Code Completion and Refactoring:** Streamlining code writing and improving code quality.

- **Debugging Tools:** Facilitating efficient debugging and troubleshooting.
- **Version Control Integration:** Simplifying version control workflows within the IDE.

**Justification for Tool Selection:**

The chosen tools strike a delicate balance between:

- **Effectiveness:** Addressing development needs and delivering desired functionalities.
- **Efficiency:** Streamlining workflows, minimizing development time, and optimizing resource allocation.
- **Cost-effectiveness:** Utilizing open-source tools and exploring cost-efficient cloud solutions whenever possible.

This combination of tools ensures a robust development process, a user-friendly web application and for a mobile app, and efficient data handling capabilities, ultimately contributing to a successful and impactful automated debris glacier detection system.

## *1.8 Vision Document*

**Project Overview**

Our project aims to revolutionize glacier mapping by developing an automated debris glacier detection system. Climate change poses a significant threat to glaciers worldwide, and understanding their dynamics is crucial for informed environmental conservation strategies. This system will address the increasing need for efficient and accurate glacier debris cover detection.

**Stakeholders**

❖ **Primary Stakeholders:**

- Glaciologists and researchers
- Environmental policymakers
- Conservation organizations

❖ **Secondary Stakeholders:**

- The general public interested in climate change and environmental issues
- Educational institutions

## Problem Statement

Manual glacier debris cover mapping is a time-consuming and resource-intensive process, often relying on visual analysis of satellite imagery. This approach is prone to human error and inconsistencies, hindering large-scale and timely monitoring efforts.

## System Vision

We envision a user-friendly **web application (primary platform)** that automates debris glacier detection. This system will leverage deep learning models and geospatial data analysis to:

- **Reduce reliance on manual mapping:** The system will automate debris cover detection, significantly saving time and resources for researchers and environmental organizations.
- **Improve accuracy and consistency:** By leveraging deep learning algorithms, the system will provide objective and consistent results, minimizing human error in debris cover assessment.
- **Enhance accessibility:** The web-based platform will offer easy access for researchers and stakeholders worldwide, facilitating collaboration and data sharing.
- **Support informed decision-making:** By providing accurate and timely information about debris cover, the system will empower policymakers and conservation efforts to develop effective strategies for glacier preservation.

**Additionally, depending on project scope and resources, we will explore developing a complementary mobile application using Flutter.** This cross-platform framework from Google allows us to create native-looking mobile experiences for both Android and iOS with a single codebase. A mobile application would:

- **Enhance accessibility:** Provide on-the-go access to the system's functionalities for researchers and stakeholders in the field.
- **Increase user engagement:** Offer a convenient way to interact with the system and its data.

## Core Features

- Automated debris cover detection using pre-trained deep learning models (e.g., SGDNet)
- Integration with geospatial data sources (e.g., satellite imagery)
- User-friendly interface for uploading and analyzing data (optimized for both web and mobile if applicable)
- Visualization tools for exploring glacier cover and debris distribution
- Data export functionalities for further analysis and reporting

### 1.8.6 Success Criteria

The success of this project will be measured by:

- **Accuracy of debris cover detection:** The system should achieve high accuracy in identifying debris cover compared to manual mapping techniques.
- **User adoption:** The system should be user-friendly and adopted by researchers, environmental organizations, and policymakers.
- **Contribution to research and conservation:** The system should provide valuable data and insights to support glacier research and conservation efforts.

## *1.9 Risk List*

This list outlines potential risks associated with developing the automated debris glacier detection system, categorized by their severity (High, Medium, Low).

**High:**

- **Data Availability and Quality:** Insufficient or unreliable geospatial data or satellite imagery could impact the accuracy of debris cover detection.
- **Model Performance:** The deep learning model might not achieve the desired level of accuracy in identifying debris cover.
- **Technical Expertise:** The development team might encounter challenges due to a lack of expertise in specific technologies (deep learning, geospatial analysis).
- **Project Scope Creep:** Unforeseen changes in project requirements or functionalities could lead to delays and budget overruns.

**Medium:**

- **Hardware and Infrastructure Limitations:** Computing power or storage capacity limitations could hinder development and processing.
- **Software Development Delays:** Unexpected issues during development could cause delays in completing the system.
- **External Dependency Delays:** Reliance on third-party tools or APIs could introduce delays if they are not readily available.
- **Stakeholder Misalignment:** Differing expectations or priorities among stakeholders could lead to project conflicts.

**Low:**

- **Power Outages and Network Disruptions:** Temporary disruptions could cause minor delays but have workarounds (backups, offline functionality).
- **Resource Availability:** Unexpected team member absences could cause minor delays, mitigated by flexible scheduling or workload adjustments.
- **Open-Source Software Updates:** Updates to open-source libraries used in the

project might require minor adjustments to maintain compatibility.

- **Documentation Gaps:** Incomplete or unclear documentation could lead to minor inefficiencies during development or user adoption.

**Mitigation Strategies:**

For each risk, a mitigation strategy will be developed to minimize its impact. These strategies might include:

- **Data source diversification and quality checks**
- **Rigorous model testing and refinement**
- **Team training and knowledge sharing**
- **Clear project scope definition and change management process**
- **Investment in adequate hardware and infrastructure**
- **Agile development methodologies and regular testing**
- **Contingency plans for external dependencies**
- **Open communication and stakeholder management**
- **Redundancy and backup systems**
- **Cross-training team members**
- **Regular software updates and compatibility checks**
- **Comprehensive and up-to-date documentation**

This risk list will be a living document, continuously updated as the project progresses. By proactively addressing potential risks, we can increase the likelihood of successful development and deployment of the automated debris glacier detection system.

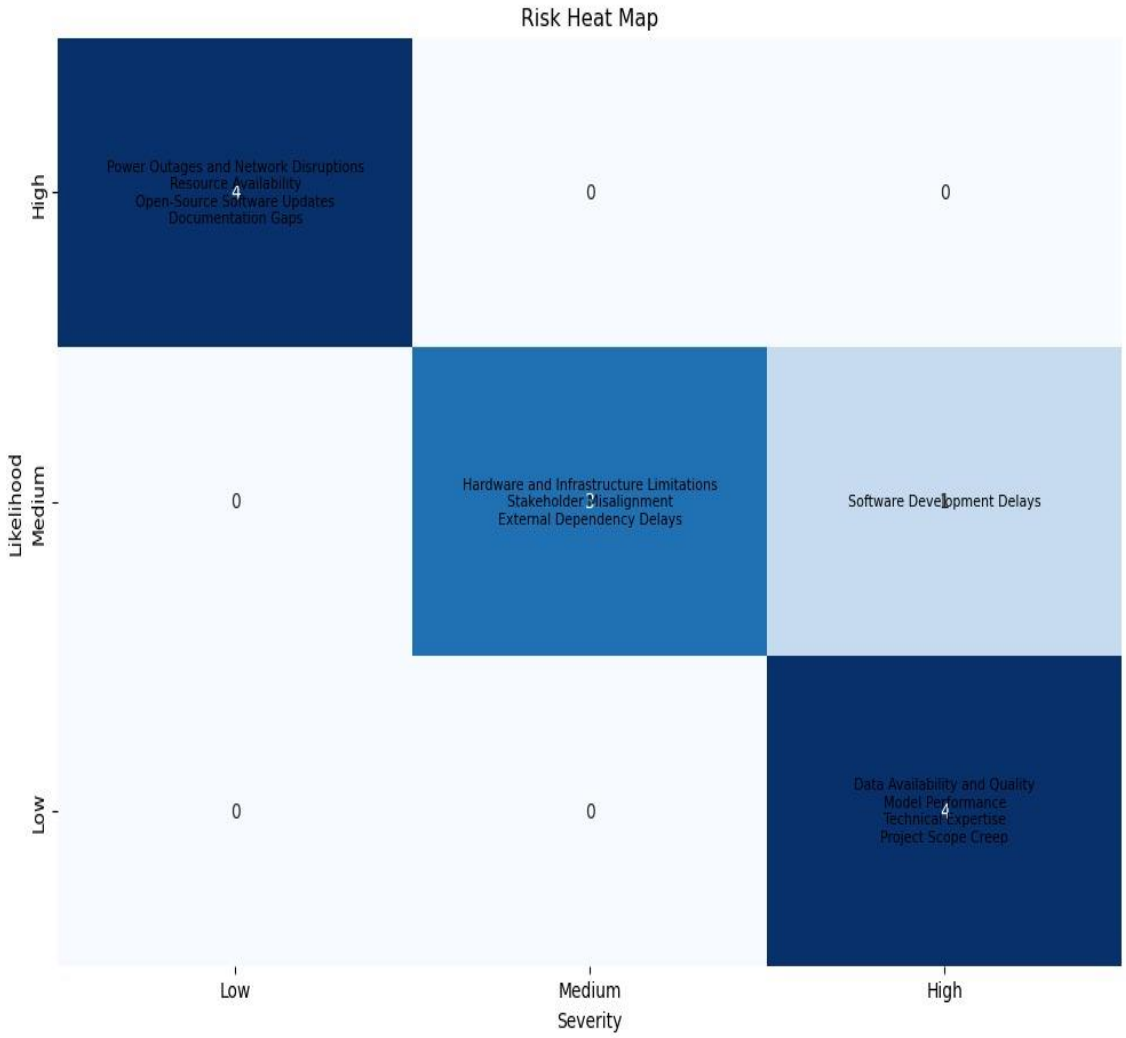



**Figure 4: Risk Matrix**

**********REQUIREMENTS ENGINEERING**********

# *1 Introduction*

Developing an automated debris glacier detection system requires a deep understanding of user needs and translating them into a well-defined set of functionalities. **Requirements Engineering** provides a structured approach to achieve this. Through this process, we'll gather vital information from glaciologists, researchers, and environmental stakeholders to precisely capture their vision for the system's capabilities. By meticulously analyzing and documenting these requirements, we establish a clear roadmap for development, ensuring the final system effectively addresses critical tasks like debris cover mapping and data analysis.

## 1.1 Systems Specifications

The following are the clauses that must be included while describing the system specifications.

### Introduction

This document outlines the system specifications for an automated debris glacier detection system. Glaciers are critical indicators of climate change, with debris cover significantly impacting melt rates and glacial health. This system will be a valuable tool for glaciologists, researchers, and environmental organizations, enabling them to monitor debris cover extent and dynamics more efficiently and at a larger scale.

### Existing System

Currently, debris cover mapping relies heavily on manual analysis of satellite imagery. This process is time-consuming, resource-intensive, and prone to human error, making it challenging for large-scale monitoring efforts. Additionally, inconsistencies can arise due to subjective interpretations by different analysts.

## Organizational Chart

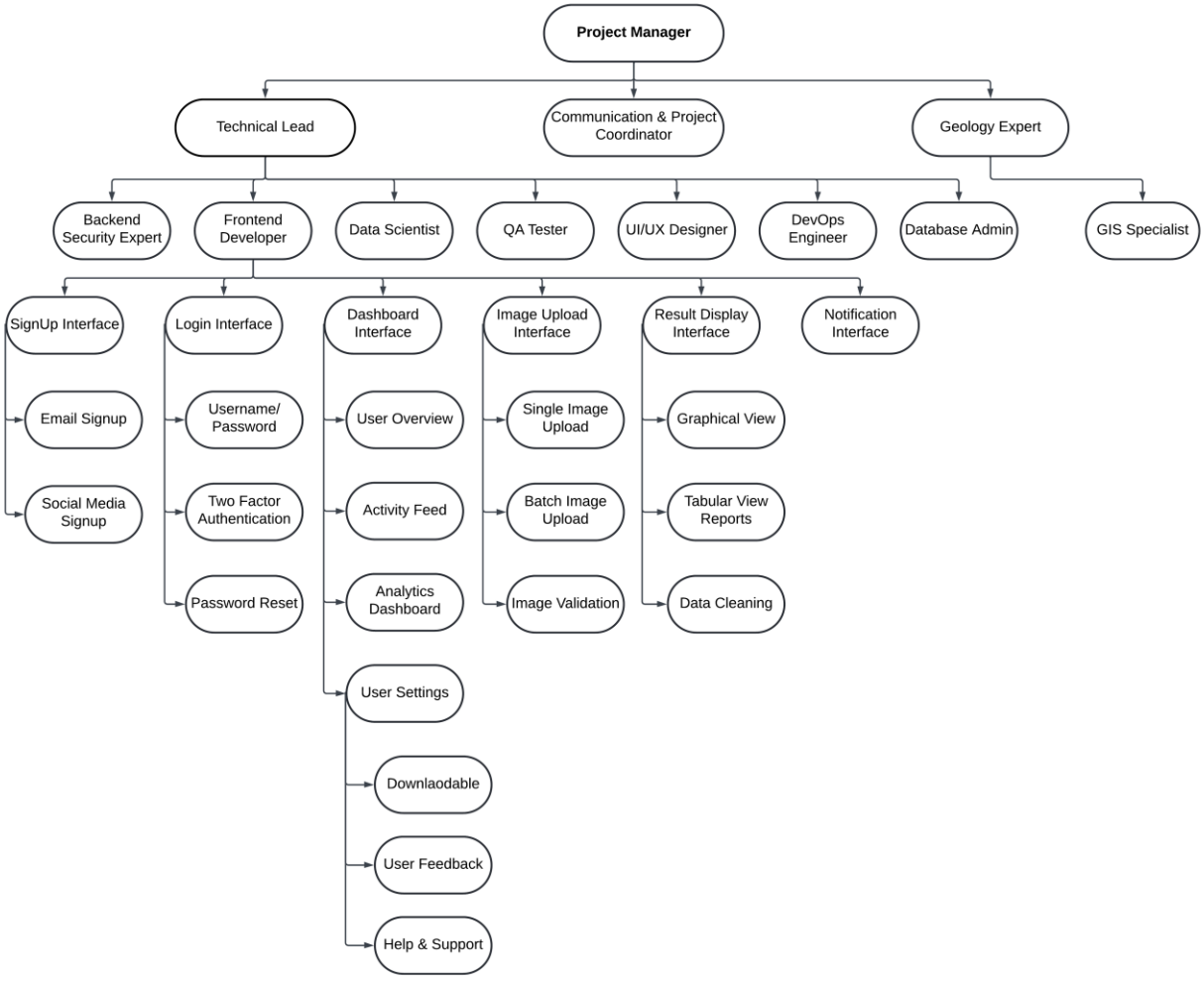



**Figure 5: Organizational Chart**

## Scope of the System

This project focuses on developing a comprehensive solution with two potential deployment options:

- **Web-based Application (Primary Platform):** This user-friendly web application will be accessible through any standard web browser on desktops, laptops, or tablets with an internet connection. The web application will cater to researchers, glaciologists, and environmental organizations, providing them with in-depth data analysis and manipulation capabilities.
- **Mobile Application (Optional):** A secondary mobile application can be

developed as a complimentary tool for field personnel or researchers requiring on-site data access and basic functionalities. The mobile application would offer a streamlined interface for uploading images, viewing basic analysis results, and potentially capturing geospatial data in the field (using GPS functionality).

Both platforms (web and potential mobile app) will leverage deep learning models for automated debris cover detection on glaciers. The core functionalities will include:

- **Data Acquisition:**

  - The web application will allow users to upload geospatial data (primarily satellite imagery) in various formats commonly used in the scientific community.
  - The system will explore integration with established geospatial data sources (e.g., NASA Earth Observing System, USGS Earth Resources Observation and Science Center) to facilitate seamless data acquisition for specific regions or glaciers of interest.

- **Data Preprocessing:**

  - The system will perform automated preprocessing steps on the uploaded imagery to ensure data quality and consistency, such as format standardization, normalization, and potential cloud cover removal techniques.

- **Debris Cover Detection:**

  - The system will employ deep learning models trained on large datasets of labeled satellite imagery to automatically identify and delineate the extent of debris cover on glaciers.
  - The system may offer users the ability to choose between pre-trained models optimized for specific debris types or geographical regions (if data allows).

- **Data Visualization:**

  - The web application will provide interactive visualizations of the analyzed data, including high-resolution maps with clear overlays highlighting the extent and characteristics of debris cover.
  - Users will be able to explore the data through various visualization tools like charts and graphs depicting the percentage of debris cover over time or across different glacier regions.

- **Data Export and Analysis:**

  - The system will allow users to export the analyzed data (debris cover

maps, statistics) in various formats (e.g., GeoTIFF, CSV) for further analysis in external GIS software or generation of reports.
- The web application might offer basic data analysis functionalities like time series analysis of debris cover changes or comparison with other environmental data (e.g., temperature records).

### 1.1.4 Summary of Requirements (Initial)

- **Accuracy:** The system should achieve high accuracy in debris cover detection compared to manual methods, with clear metrics and validation procedures established.
- **User-friendliness:** Both web and mobile interfaces (if applicable) should be intuitive and user-friendly for researchers and stakeholders with varying technical backgrounds. The system should offer clear documentation, tutorials, and user support mechanisms.
- **Data Security:** The system should ensure the security and privacy of user-uploaded data, adhering to relevant data protection regulations.
- **Scalability:** The system should be scalable to handle large datasets from various geographical regions and accommodate future functionalities like advanced data analysis tools or integration with other environmental monitoring systems.
- **Performance:** The system should deliver results within reasonable timeframes, especially for the web application catering to potentially complex analyses.

These initial requirements will be further refined and elaborated upon throughout the requirements engineering process. Stakeholder input and iterative development will be crucial in ensuring the system meets the specific needs of the glaciology and environmental research communities.

## 1.2 Identifying External Entities

Following the two-phase approach for identifying external entities, we can delve deeper into our project's abstract:

**Phase a: Over-Specifying Entities from Abstract**

Based on the abstract, we can brainstorm a comprehensive list of potential external entities:

- **Glaciers (Primary Entity):** Geographical formations of interest for debris cover detection. The system will primarily focus on data acquisition and analysis related to glaciers.
- **Satellite Imagery Providers:** Organizations or platforms that provide satellite imagery data crucial for training the debris cover detection model and analyzing debris cover over time. (e.g., NASA Earth Observing System, USGS Earth Resources Observation and Science Center)

- **Ground Observation Teams (Optional):** Field research teams potentially collecting on-site data that could be integrated with the system in the future (e.g., high-resolution photos for model validation or additional data points).
- **Glaciologists:** Researchers specializing in the study of glaciers and their dynamics, who represent a core user group for the system. They will heavily rely on the system's functionalities for debris cover mapping and analysis.
- **Researchers (Broader Category):** A broader category encompassing scientists studying various aspects of glaciers and their environment, who might also find the system valuable for their research endeavors (e.g., climate change impact studies).
- **Environmental Organizations (Diverse Range):** Non-profit or governmental entities concerned with environmental monitoring and protection. These organizations can leverage the system for large-scale glacier health assessments and informing policy decisions.

**Phase b: Refinement Based on Business Logic**

Considering the system's functionalities and target users, we can refine the initial list:

- **External Data Sources:** This encompasses various sources of geospatial data, primarily satellite imagery providers. However, depending on future developments, it could potentially include other relevant environmental datasets (e.g., digital elevation models, temperature data).
- **Research Institutions:** This broader category includes universities, research institutes, and government agencies employing glaciologists and researchers who would utilize the system. These institutions might also contribute data or collaborate on model development.
- **End-Users (Primary Focus):** This refined category focuses on the individuals directly interacting with the system:

    - Glaciologists: A core user group for in-depth analysis of debris cover and glacier health.
    - Researchers affiliated with research institutions: These researchers might utilize the system for broader studies related to glaciers and their environment.
    - Environmental organization personnel: These users would benefit from the system for large-scale monitoring and data-driven decision making.

**Additional Considerations:**

- **Data Repositories (Optional):** Depending on the implementation, external data repositories for storing and managing user-uploaded data might be considered, especially if the system caters to user-generated datasets beyond satellite imagery.
- **Cloud Service Providers (Potential):** If the system leverages cloud computing for scalability or data storage (e.g., large satellite image archives), cloud service providers become external entities. Security considerations would be paramount

when involving such entities.

## 1.3 Context Level Data Flow Diagram

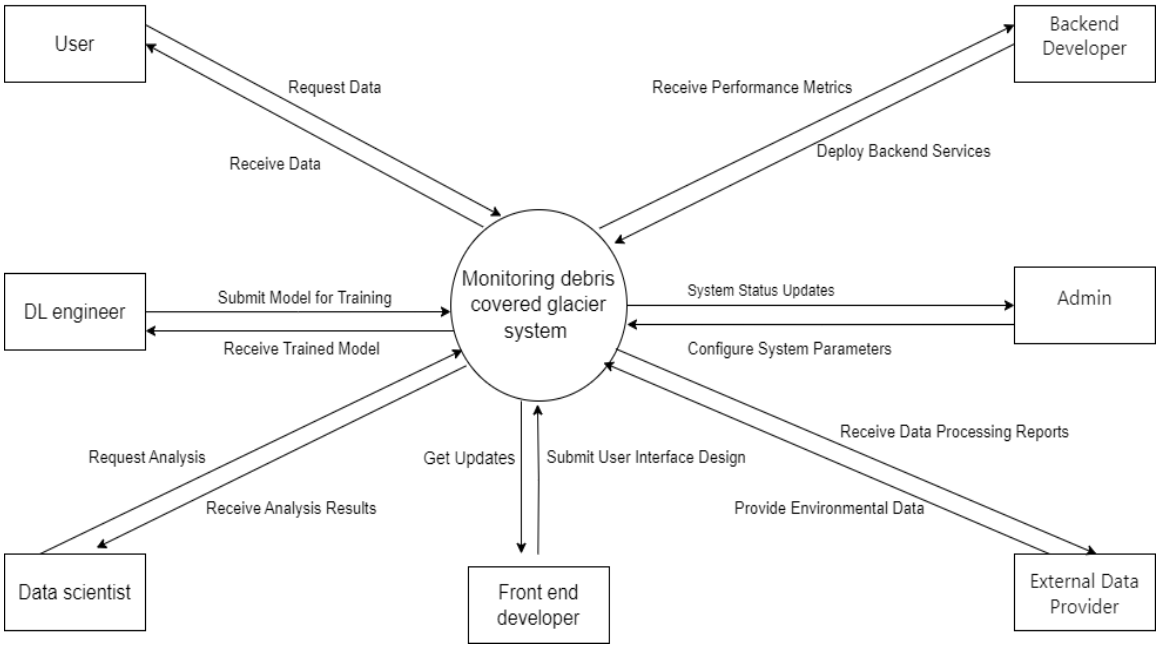



**Figure 6: Context Level Data Flow Diagram**

## 1.4 Capture "shall" Statements

| Section | External Entity | "Shall" Statement | Description |
| --- | --- | --- | --- |
| **User Management** | End-User | Register with the system to create an account. | Establishes user account creation process. |
| **User Management** | System Administrator | Review and approve/reject registration requests. | Defines administrator role in user account management. |
| **User Management** | End-User | Log in to the system using their credentials. | Enables user authentication. |
| **User Management** | System | Allow End-Users to reset their passwords if forgotten. | Provides password recovery functionality. |
| **User Management** | End-User | View and update their profile information. | Allows user profile management. |
| **Data Management** | End-User | Upload geospatial data (satellite imagery) in supported formats. | Enables data submission by users. |
| **Data Management** | System | Validate the uploaded data format and notify the user of any errors. | Ensures data quality and compatibility. |
| **Data Management** | System | Store uploaded data securely. | Ensures data security. |
| **Debris Cover Detection & Analysis** | End-User (Optional) | Select pre-trained debris cover detection models (if applicable). | Allows user control over model selection (if available). |
| **Debris Cover Detection & Analysis** | System | Provide a user interface for selecting debris cover detection models. | Facilitates user interaction with model selection. |
| **Debris Cover Detection & Analysis** | System | Automatically process the uploaded data using the chosen model(s). | Defines the core functionality of debris cover detection. |
| **Debris Cover Detection & Analysis** | System | Display the analysis results within a reasonable timeframe. | Ensures system responsiveness and user experience. |
| **Debris Cover Detection & Analysis** | System | Present the results in a user-friendly format, including maps with debris cover extent highlighted. | Defines data visualization for clear communication of results. |
| **Data Export & Reporting** | End-User | Download the analysis results (maps, data) in various formats. | Enables users to leverage results for further work. |

| | | | |
| --- | --- | --- | --- |
| **User Management** | System Administrator | Manage user accounts (e.g., add/remove users). | Defines administrator control over user accounts. |
| **User Management** | End-User | Update their profile information (e.g., contact details, affiliation). | Allows user profile management. |
| **Data Management** | End-User | Upload additional data relevant to debris cover analysis (if the system allows). | Accommodates potential future functionalities for additional data types. |
| **Debris Cover Detection & Analysis** | System | Provide a user interface for selecting and applying appropriate debris cover detection models. | Facilitates user interaction with model selection. |
| **Data Export & Reporting** | End-User | Download the analysis results (maps, data) for further analysis or reporting. | Enables users to leverage results for further work. |

## 1.5 Allocate Requirements

| Para # | Initial Requirements | Use Case Name | Description |
| --- | --- | --- | --- |
| **1.0** | End-User "shall" register with the system to create an account. | UC_Register_End_User | User creates a new account (standard or privileged based on approval). |
| **1.1** | System Administrator "shall" review and approve/reject registration requests based on predefined criteria. | UC_Process_Registration | Administrator reviews and approves/rejects user registration requests. |
| **1.2** | End-User "shall" log in to the system using their credentials. | UC_Login | User authenticates by logging in. |
| **1.3** | System "shall" allow End-Users to reset their passwords if forgotten. | UC_Reset_Password | User retrieves a new password if forgotten. |
| **1.4** | End-User "shall" be able to view and update their profile information (e.g., contact details, affiliation). | UC_Manage_User_Profile | User accesses and updates their profile information. |
| **2.1** | End-User "shall" be able to upload geospatial data (satellite imagery) in supported formats. | UC_Upload_Data | User uploads satellite imagery for debris cover analysis. |

| | | | |
| --- | --- | --- | --- |
| **2.2** | System "shall" validate the uploaded data format and notify the user of any errors. | UC_Validate_Data | System checks the uploaded data format and informs the user of any issues. |
| **2.3 (Optional)** | End-User (Optional) "shall" be able to upload additional data relevant to debris cover analysis (if the system allows). | UC_Upload_Additional_Data (Optional) | User uploads additional data types relevant to debris cover analysis (if supported). |
| **2.4** | System "shall" securely store uploaded data following data privacy regulations. | UC_Secure_Data_Storage | System stores uploaded data securely following data privacy regulations. |
| **3.1** | End-User (Optional) "shall" select pre-trained debris cover detection models (if applicable). | UC_Select_Model (Optional) | User chooses a pre-trained model for debris cover detection (if available). |
| **3.2** | System "shall" provide a user interface for selecting and applying appropriate debris cover detection models. | UC_Provide_Model_Selection | System offers an interface for users to select appropriate debris cover detection models. |
| **3.3** | System "shall" automatically process the uploaded data using the chosen model(s). | UC_Process_Data | System automatically analyzes the uploaded data using the selected model(s). |

| | | | |
| --- | --- | --- | --- |
| **3.4** | System "shall" display the analysis results within a reasonable timeframe. | UC_Display_Results | System presents the analysis results to the user in a timely manner. |
| **3.5** | System "shall" present the results in a user-friendly format, including maps with debris cover extent highlighted and relevant statistics. | UC_Format_Results | System displays the results in a user-friendly format (maps, statistics) for clear communication. |
| **3.6** | System "shall" offer data visualization options (e.g., zoom, layer control) for further exploration of results. | UC_Explore_Results | System provides functionalities for users to explore the analysis results visually (e.g., zoom, layer control). |
| **4.1** | End-User "shall" be able to download the analysis results (maps, data) in various formats (e.g., GeoTIFF, CSV) for further analysis or reporting. | UC_Download_Results | User downloads the analysis results (maps, data) in various formats for further work. |
| **4.2** | End-User "shall" be able to generate custom reports incorporating analysis results and user annotations. | UC_Generate_Report | User creates reports combining analysis results and their own observations. |

| | | | |
| --- | --- | --- | --- |
| **5.1** | System "shall" provide a communication platform for users to share findings and collaborate on research. | UC_Enable_Collaboration (Optional) | System offers functionalities for users to share findings and collaborate (optional). |
| **5.2** | System "shall" implement access control mechanisms to ensure data privacy and security based on user roles. | UC_Implement_Access_Control | System enforces access control based on user roles to protect data privacy and security. |
| **5.3** | System Administrator "shall" monitor system performance and resource usage. | UC_Monitor_System_Performance | Administrator monitors system performance and resource usage. |

## 1.6 Prioritize Requirements

| Rank | Para # | Initial Requirements | Use Case ID | Use Case Name |
| --- | --- | --- | --- | --- |
| **Highest** | 1.0 | End-User "shall" upload geospatial data (satellite imagery) for debris cover analysis. | UC_1 | UC_Upload_Imagery |
| **Highest** | 1.0 | End-User "shall" register with the system to create an account (standard or privileged based on approval). | UC_2 | UC_Register_End_User |
| **Highest** | 2.0 | End-User "shall" select a payment method (cash or credit card) for the analysis. | UC_3 | UC_Select_Payment_Method |
| **Highest** | 2.0 | System "will" generate an invoice and confirmation receipt upon successful order submission. | UC_4 | UC_Generate_Invoice_Receipt |
| **Medium** | 2.0 | Both registered and privileged users "shall" be able to submit debris cover analysis orders. | UC_5 | UC_Submit_Analysis_Order (privileged) |
| **Medium** | 1.1 | System Administrator "shall" review and approve/reject user registration requests based on predefined criteria. | UC_6 | UC_Process_Registration |

| | | | | |
| --- | --- | --- | --- | --- |
| **Medium** | 3.1 (Optional) | End-User (Optional) "shall" select pre-trained debris cover detection models (if applicable). | UC_7 | UC_Select_Model (Optional) |
| **Medium** | 3.2 (Optional) | System "shall" provide a user interface for selecting and applying appropriate debris cover detection models (if applicable). | UC_8 | UC_Provide_Model_Selection (Optional) |
| **Medium** | 3.3 | System "shall" automatically process the uploaded imagery using the chosen model(s). | UC_9 | UC_Process_Imagery |
| **Medium** | 3.4 | System "shall" display the analysis results within a reasonable timeframe. | UC_10 | UC_Display_Results |
| **Medium** | 3.5 | System "shall" present the results in a user-friendly format, including maps with debris cover extent highlighted and relevant statistics. | UC_11 | UC_Format_Results |
| **Medium** | 3.6 | System "shall" offer data visualization options (e.g., zoom, layer control) for further exploration of results. | UC_12 | UC_Explore_Results |

| | | | | |
| --- | --- | --- | --- | --- |
| **Medium** | 4.1 | End-User "shall" be able to download the analysis results (maps, data) in various formats (e.g., GeoTIFF, CSV) for further analysis or reporting. | UC_13 | UC_Download_Results |
| **Medium** | 4.2 | End-User "shall" be able to generate custom reports incorporating analysis results and user annotations. | UC_14 | UC_Generate_Report |
| **Medium** | 2.4 | System "shall" securely store uploaded data following data privacy regulations. | UC_15 | UC_Secure_Data_Storage |
| **Medium** | 1.0 | System "shall" allow End-Users to reset their passwords if forgotten. | UC_16 | UC_Reset_Password |
| **Medium** | 1.2 | End-User "shall" log in to the system using their credentials. | UC_17 | UC_Login |
| **Medium** | 1.4 | End-User "shall" be able to view and update their profile information (e.g., contact details, affiliation). | UC_18 | UC_Manage_User_Profile |

| | | | | |
| --- | --- | --- | --- | --- |
| **Medium** | 2.3 (Optional) | End-User (Optional) "shall" be able to upload additional data relevant to debris cover analysis (if the system allows). | UC_19 | UC_Upload_Additional_Data (Optional) |
| **Lowest** | 5.1 (Optional) | System "shall" provide a communication platform for users to share findings and collaborate on research (optional). | UC_20 | UC_Enable_Collaboration (Optional) |
| **Lowest** | 5.2 | System "shall" implement access control mechanisms to ensure data privacy and security based on user roles. | UC_21 | UC_Implement_Access_Control |
| **Lowest** | 5.3 | System Administrator "shall" monitor system performance and resource usage. | UC_22 | UC_Monitor_System_Performance |
| **Lowest** | 3.0 (Optional) | Corresponding administrator "shall" view his Action List containing different actions, and correspondingly process these pending actions (optional). | UC_23 | UC_View_ |

## 1.6 Requirements Trace-ability Matrix

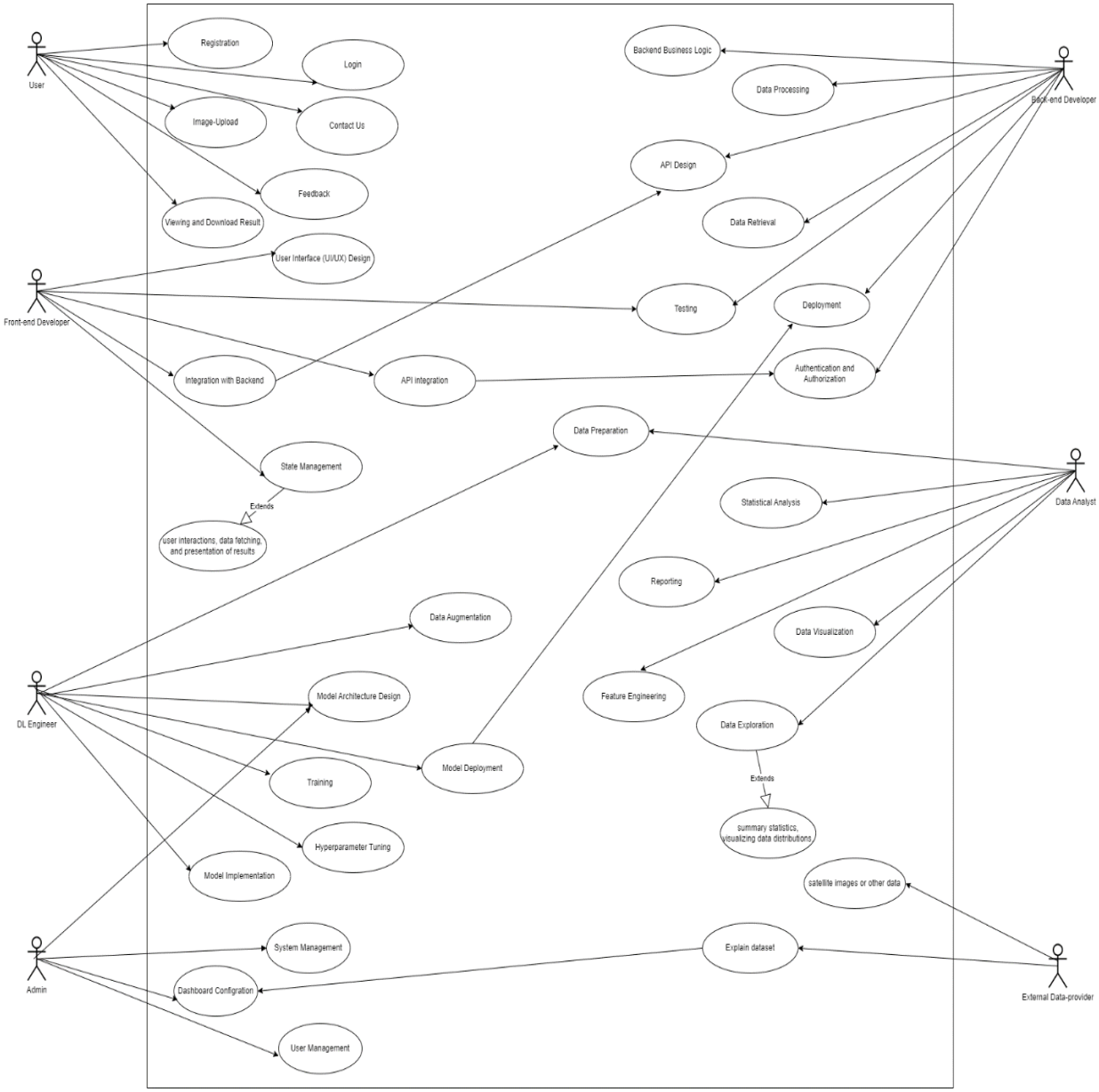
| Sr# | Para # | System Specification Text (Ice Age) | Build | Use Case Name | Category |
| --- | --- | --- | --- | --- | --- |
| 1 | 1.0 | End-User "shall" upload geospatial data (satellite imagery) for glacier debris cover analysis. | B1 | UC_Upload_Imagery | Business |
| 2 | 1.0 | End-User "shall" register with the system to create an account (standard or privileged based on approval). | B1 | UC_Register_End_User | Business |
| 3 | 1.0 | System "shall" provide options for standard and privileged user registration. | B1 | UC_Register_Standard_User | Business |
| 3 | 1.0 | | B1 | UC_Register_Privileged_User (Optional) | Business |
| 4 | N/A | N/A | B1 | UC_Review_Analysis_Request (Optional) | Business |
| 4 | N/A | N/A | B1 | UC_Approve_Analysis_Request | Business |
| 4 | N/A | N/A | B1 | UC_Reject_Analysis_Request | Business |
| 5 | 1.0 | End-User "shall" log in to the system using their credentials. | B1 | UC_Login | Business |

| | | | | | |
| --- | --- | --- | --- | --- | --- |
| **6** | 1.0 | System "shall" allow End-Users to reset their passwords if forgotten. | B1 | UC_Reset_Password | Business |
| **7** | 1.0 | System "shall" update user information based on their requests (details below). | B1 | UC_Update_User_Profile | Business |
| **7** | | | B1 | UC_Change_Personal_Details | Business |
| **7** | | | B1 | UC_Request_Account_Upgrade (Optional) | Business |
| **7** | | | B1 | UC_Change_Payment_Method | Business |
| **8** | 1.0 | End-User "shall" be able to view and update their profile information. | B1 | UC_Manage_User_Profile | Business |
| **9** | (redundant) | System "shall" search any customer details | N/A | (removed) | N/A |
| **10** | 2.1 (Optional) | End-User (Optional) "shall" select pre-trained glacier debris detection models (if applicable). | B1 | UC_Select_Model (Optional) | Business |
| **11** | 2.2 (Optional) | System "shall" provide a user interface for selecting and applying appropriate glacier debris detection models (if applicable). | B1 | UC_Provide_Model_Selection (Optional) | Business |

| | | | | | |
| --- | --- | --- | --- | --- | --- |
| **12** | 3.1 | System "shall" automatically process the uploaded imagery using the chosen model(s). | B1 | UC_Process_Imagery | Business |
| **13** | 3.2 | System "shall" display the analysis results within a reasonable timeframe. | B1 | UC_Display_Results | Business |
| **14** | 3.3 | System "shall" present the results in a user-friendly format, including maps with glacier debris extent highlighted and relevant statistics (e.g., area, volume). | B1 | UC_Format_Results | Business |
| **15** | 3.4 | System "shall" offer data visualization options (e.g., zoom, layer control) for further exploration of results (e.g., debris thickness variations). | B1 | UC_Explore_Results | Business |

| | | | | | |
| --- | --- | --- | --- | --- | --- |
| **16** | 4.1 | End-User "shall" be able to download the analysis results (maps, data) in various formats (e.g., GeoTIFF, CSV) for further analysis or reporting. | B1 | UC_Download_Results | Business |
| **17** | 4.2 | End-User "shall" be able to generate custom reports incorporating analysis results and user annotations. | B1 | UC_Generate_Report | Business |

## 1.9 High Level Usecase Diagram



# 1 Requirements Analysis and System Design

The Ice Age project has reached a pivotal stage! Having thoroughly analyzed the challenges of glacier debris cover assessment, we now embark on crafting a solution. This deliverable delves into the realm of object-oriented design, laying the foundation for a robust and efficient system.

By leveraging object-oriented principles, we can create a software architecture that closely mirrors the real-world entities and processes involved in glacier debris analysis. This approach fosters modularity, maintainability, and scalability, ensuring the system can effectively address current needs and seamlessly adapt to future requirements.

## 1.1 Usecase Description

### User Interaction: Initiate Glacier Debris Analysis

**Actors:**

- User
- Secondary Actors: Database Engineer, Backend Developer (Data Scientist - Optional)

**Brief Description:**

This interaction allows a registered user to initiate a comprehensive glacier debris cover analysis on their uploaded geospatial data (satellite imagery). The system offers functionalities for focused analysis and provides detailed results for informed decision-making.

**Preconditions:**

- User has a valid account and is logged in.
- User has a stable internet connection.
- User possesses the required geospatial data (satellite imagery) in a supported format (e.g., GeoTIFF, JPEG).

**Basic Flow:**

- User navigates to the dedicated section for "Glacier Debris Analysis" within the Ice Age system.
- User uploads their satellite imagery file, ensuring compatibility with the system's accepted formats.
- The system validates the uploaded file for size and format compatibility.
- (Optional) The system might present a selection of pre-trained glacier debris detection models developed by the Data Scientist (if applicable). The user can

choose the most suitable model based on their specific needs or leave the selection to default settings.

- The user reviews and confirms all analysis request details, including any chosen model selection and potentially specifying desired debris characteristics (e.g., size, type).
- The user submits the request, initiating the analysis process.

**Alternate Flows:**

- **Invalid data format:** The system prompts the user to upload a file in a supported format, offering suggestions if available.
- **Insufficient storage space:** The system informs the user about potential storage limitations and suggests alternative actions (e.g., contacting an administrator or upgrading storage plans).
- **System error:** The system displays an error message and attempts to gracefully recover, potentially offering the user options to retry or report the issue.
- **User cancellation:** The user has the option to abandon the analysis request before completion.

**Postconditions:**

- The user's analysis request is successfully submitted to the system.
- The uploaded data is stored securely within the system's database.
- Upon completion, the user receives notification and can access detailed analysis results through the Ice Age system, including:

    - Maps highlighting the identified glacier debris cover.
    - Relevant statistics like debris area, volume, or other user-specified parameters.
    - (Optional) Additional insights based on the chosen model (e.g., debris composition estimates).

## *1.2 Usecase Diagram (refined and updated)*

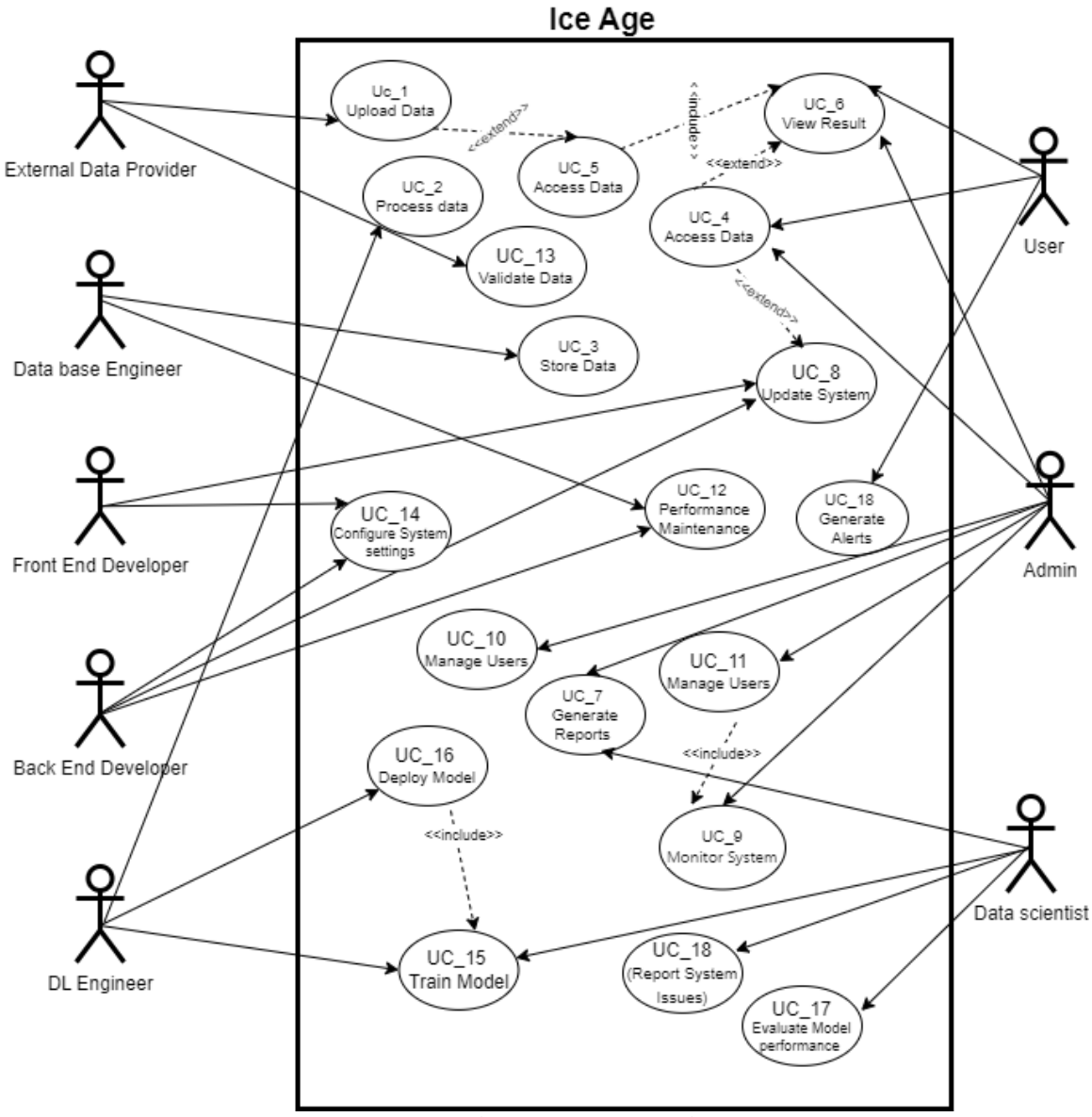



**Figure 7: Use Case Diagram**

## *1.3 Domain Model*

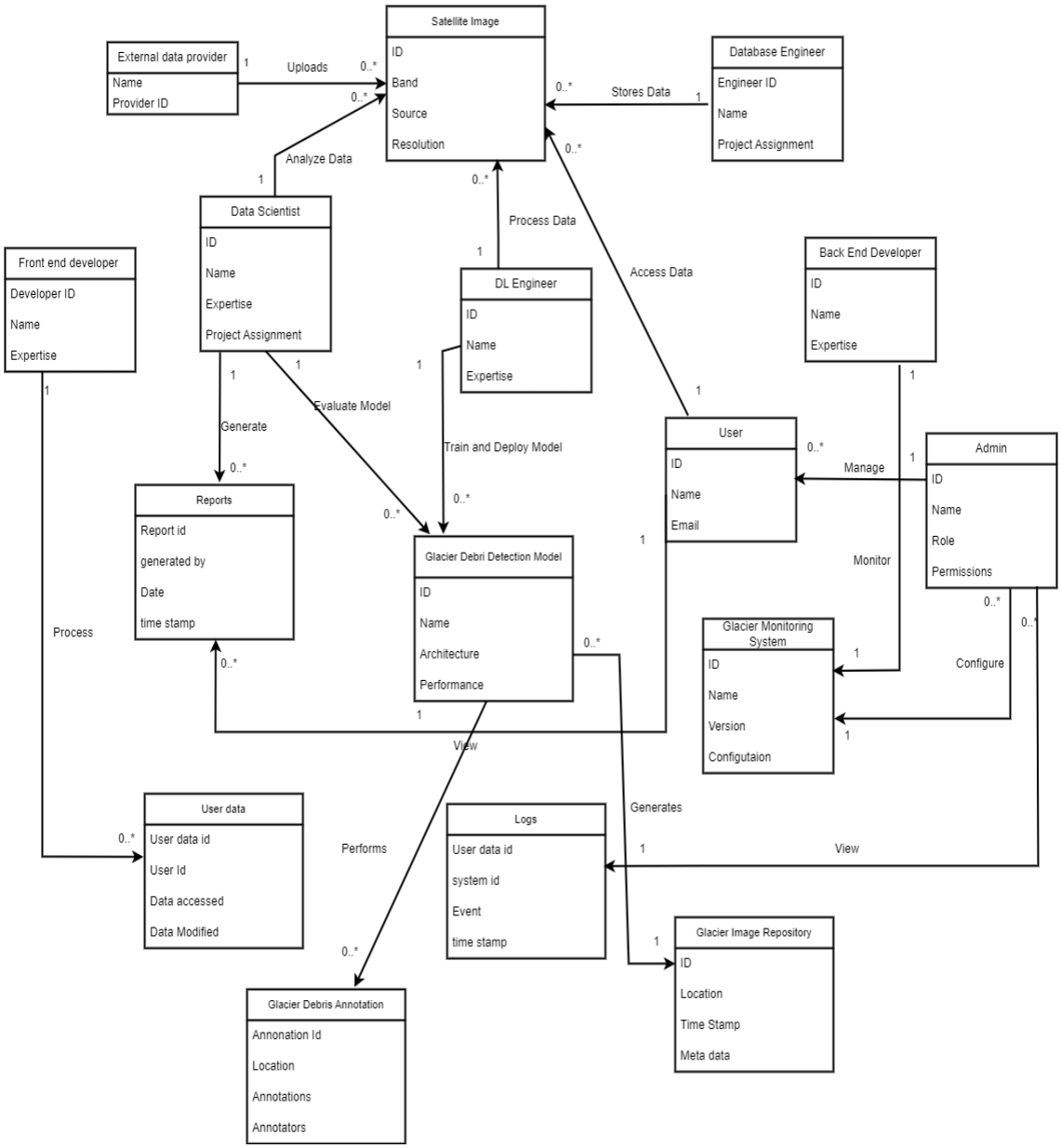


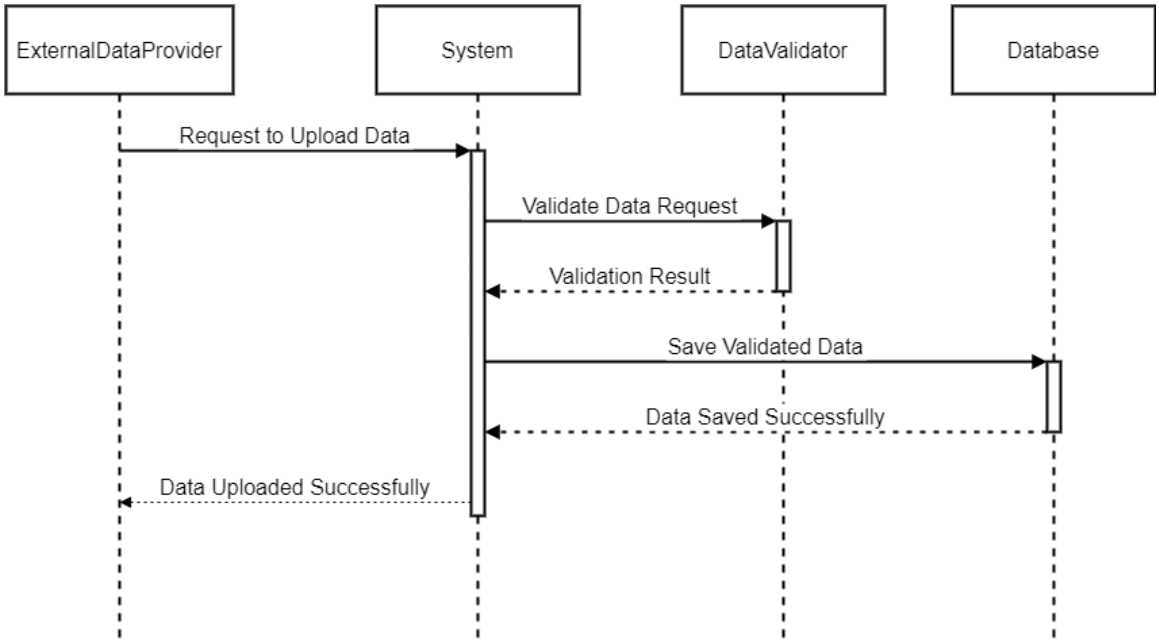

**Figure 8: Domain Model**

## *1.4 Sequence Diagram*

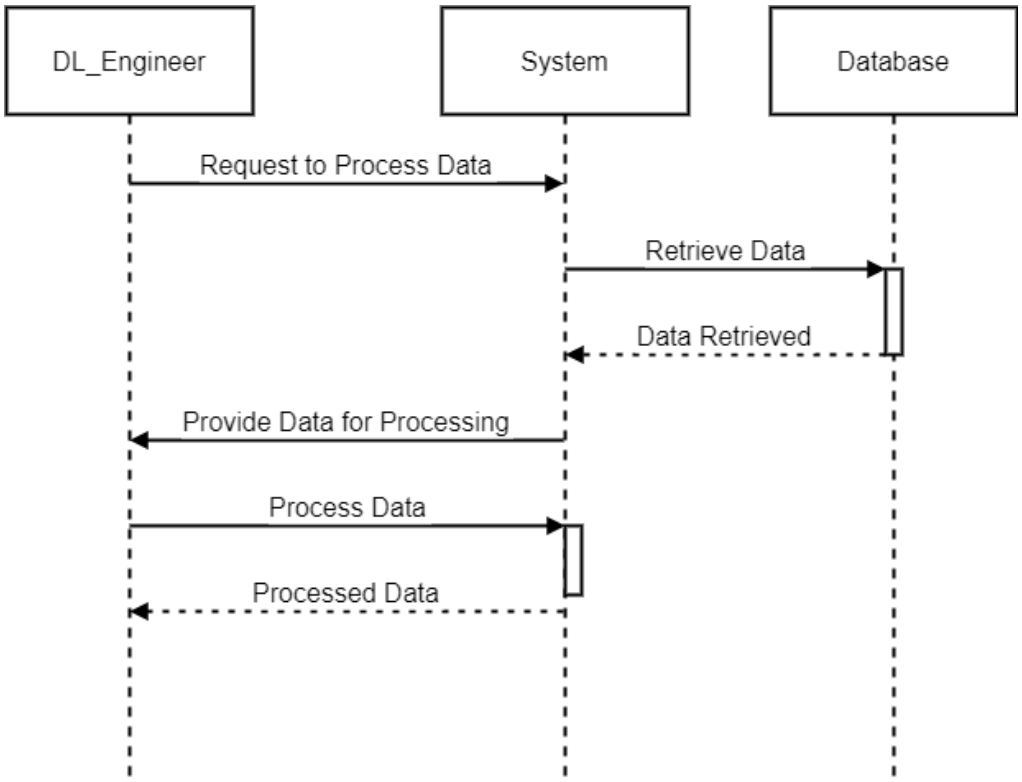
The Ice Age project's sequence diagram depicts the message flow during "Glacier Debris Analysis." The user submits a request with data (imagery) through the UI. The backend developer stores the data and interacts with the data scientist for model selection. Data is then processed, and results are presented to the user.

- **Objects:** User, UI, Backend Developer, Database Engineer, Data Scientist
- **Messages:** User submits request, UI forwards request, data is stored, model selection (optional), data processing, results returned, results displayed.
- **Benefits:** Visualizes interaction flow, clarifies object collaboration.

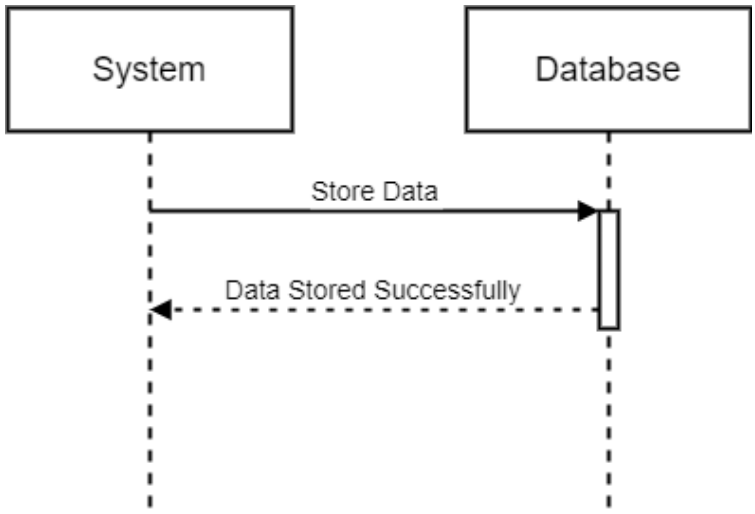
### 1.4.1. << UC-1>> UC_ Upload Data



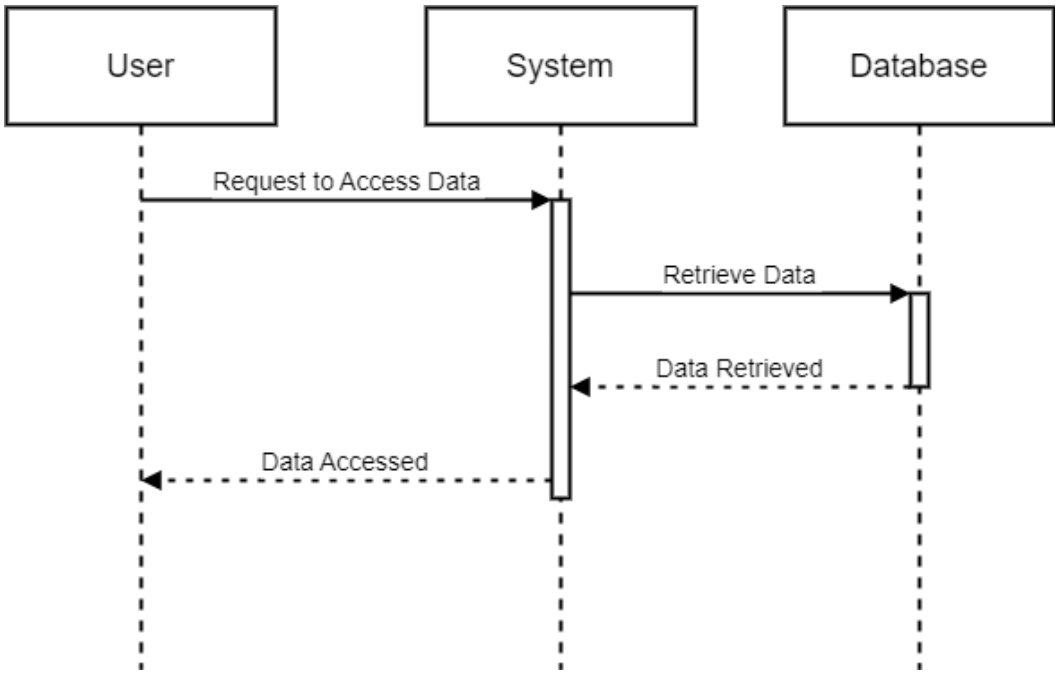
### 1.4.2. << UC-1>> UC_ Process Data



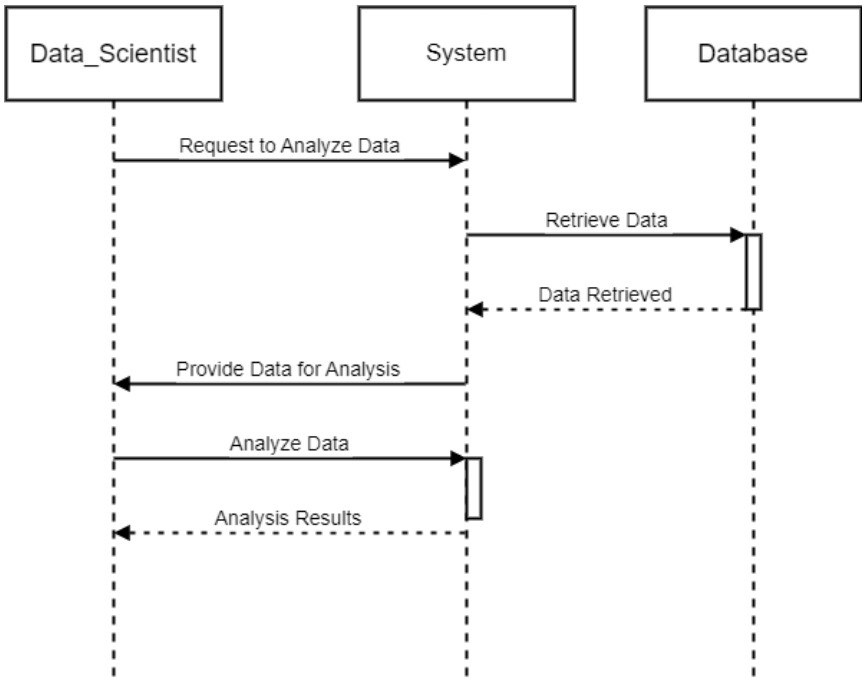

### 1.4.3. << UC-1>> UC_ Store Data



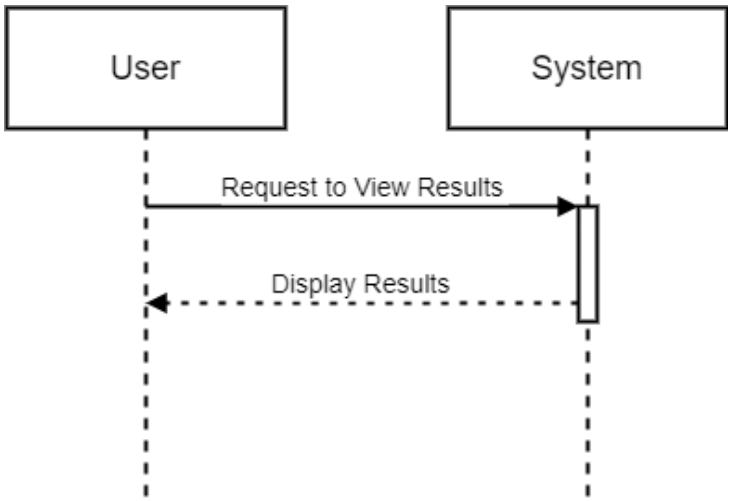
### 1.4.4. << UC-1>> UC_ Access Data



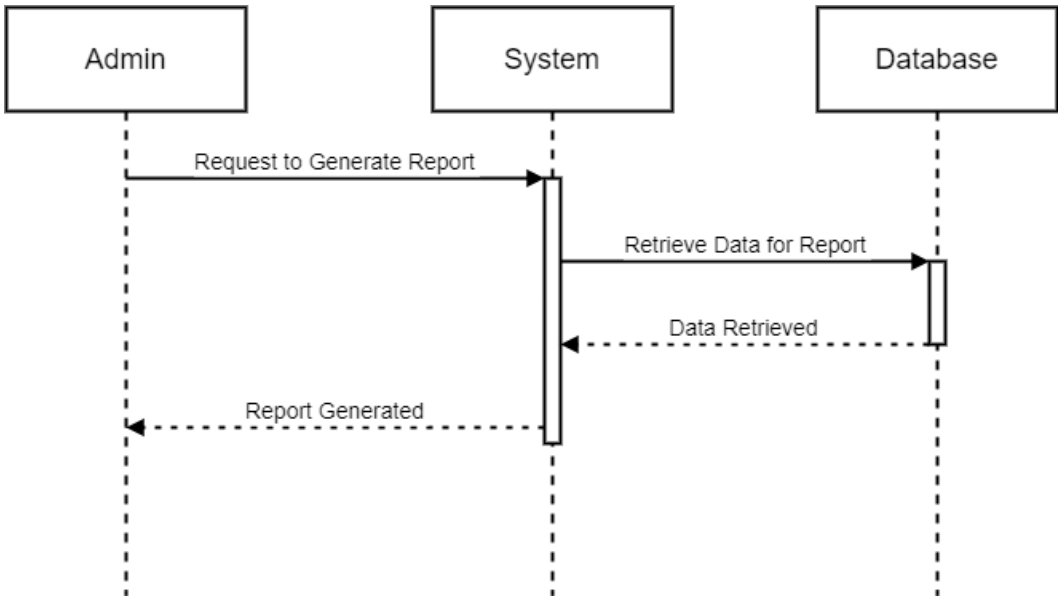

### 1.4.5. << UC-1>> UC_ Analyze Data



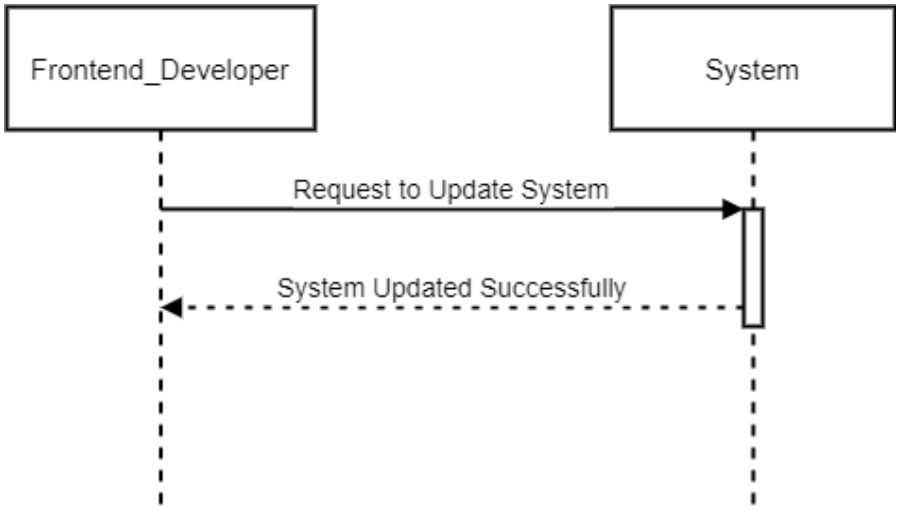
### 1.4.6. << UC-1>> UC_ View Results



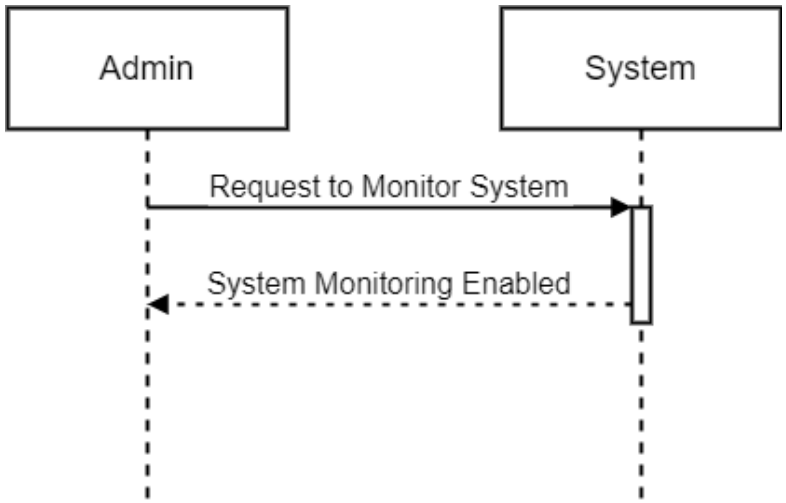

### 1.4.7. << UC-1>> UC_ Generate Reports



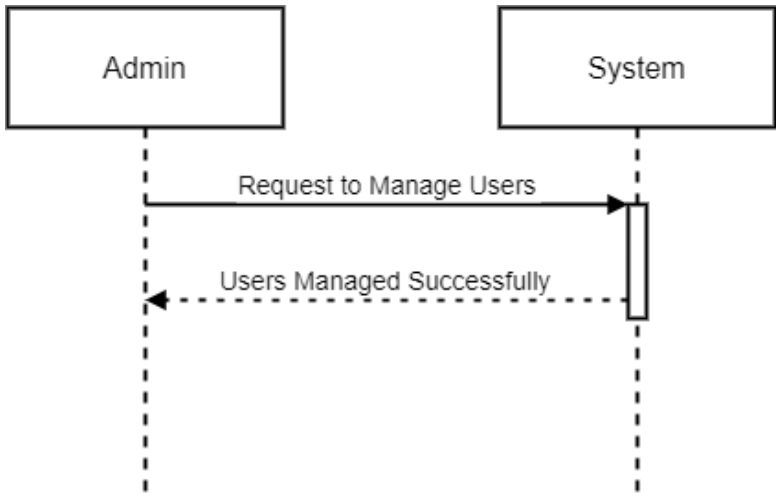
### 1.4.8. << UC-1>> UC_ Update System



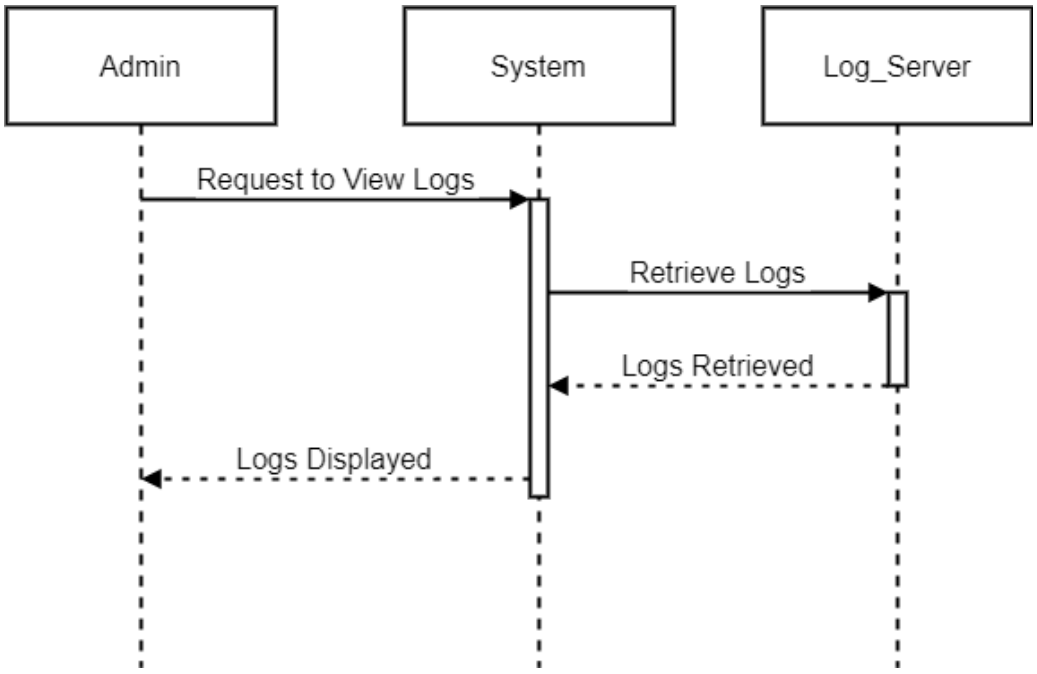

### 1.4.9. << UC-1>> UC_ Monitor System



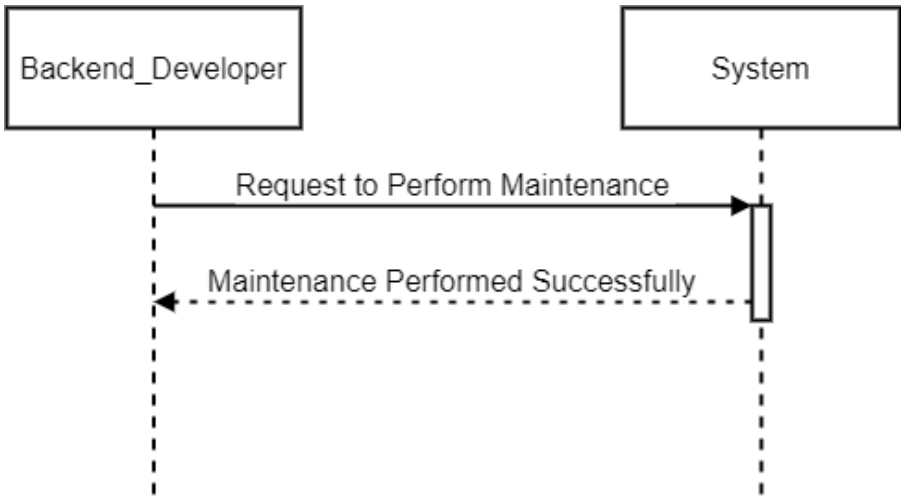
### 1.4.10. << UC-1>> UC_ Manage Users



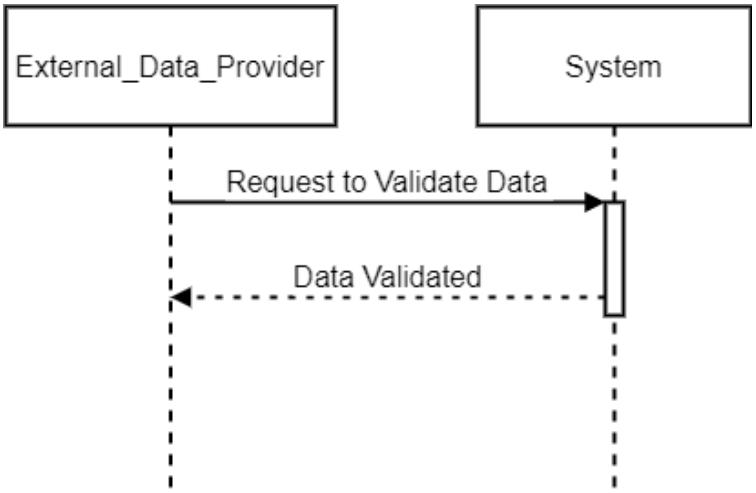

### 1.4.11. << UC-1>> UC_ View Logs



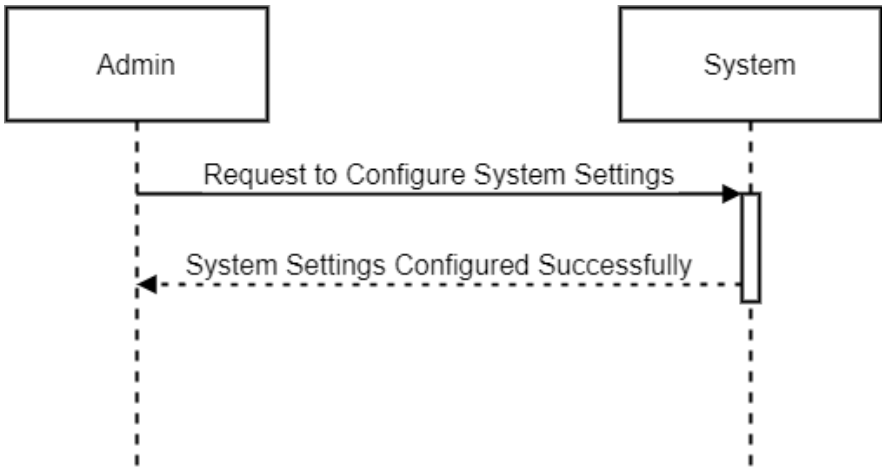
### 1.4.12. << UC-1>> UC_ Perform Maintenance



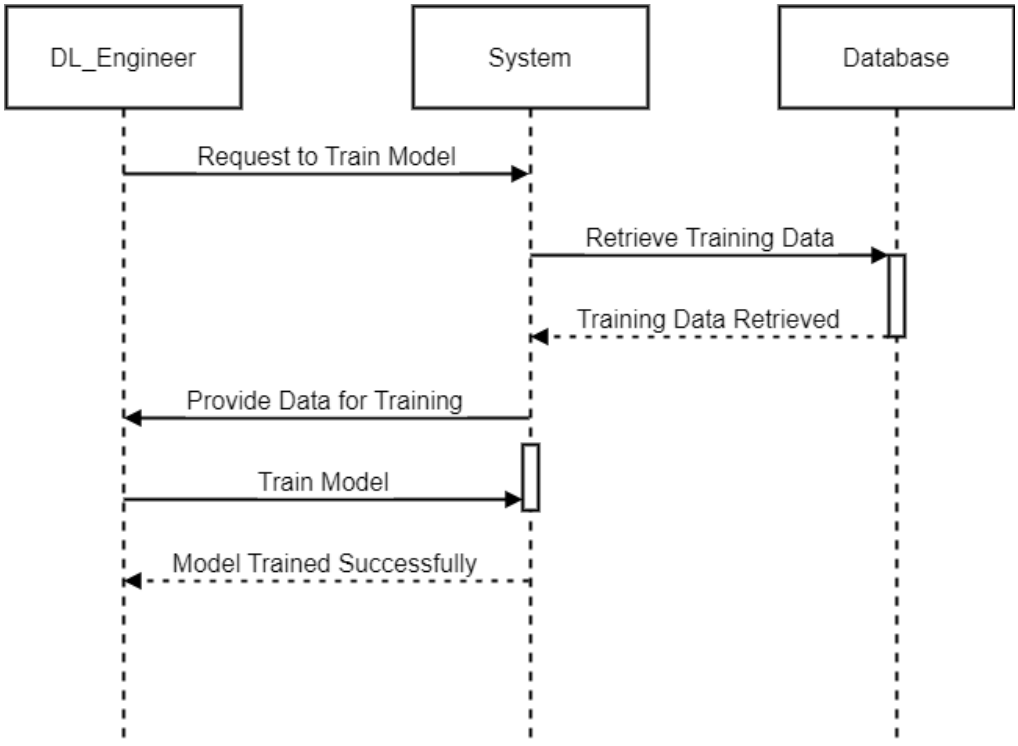

### 1.4.13. << UC-1>> UC_ Validate Data



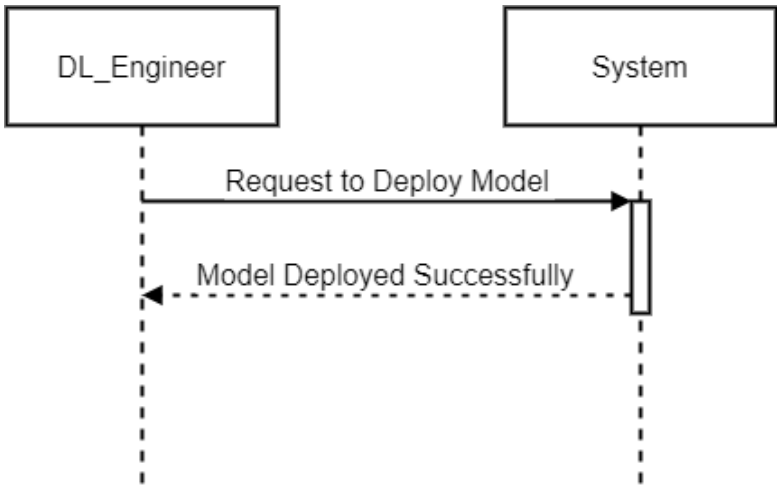
### 1.4.14. << UC-1>> UC_ Configure System Settings



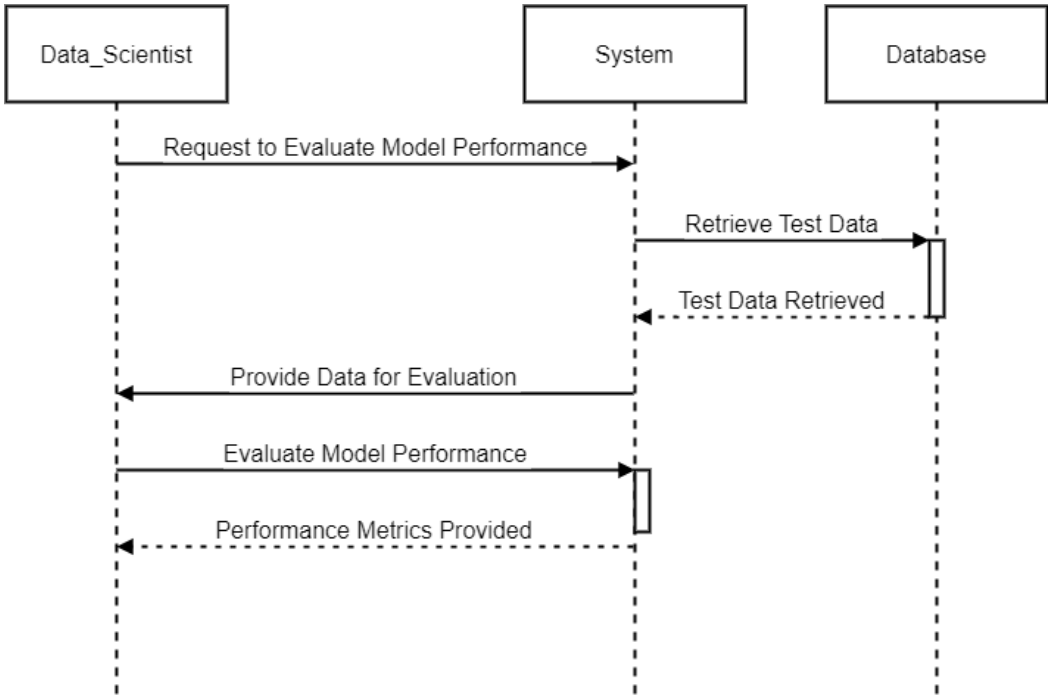

### 1.4.15. << UC-1>> UC_ Train Model



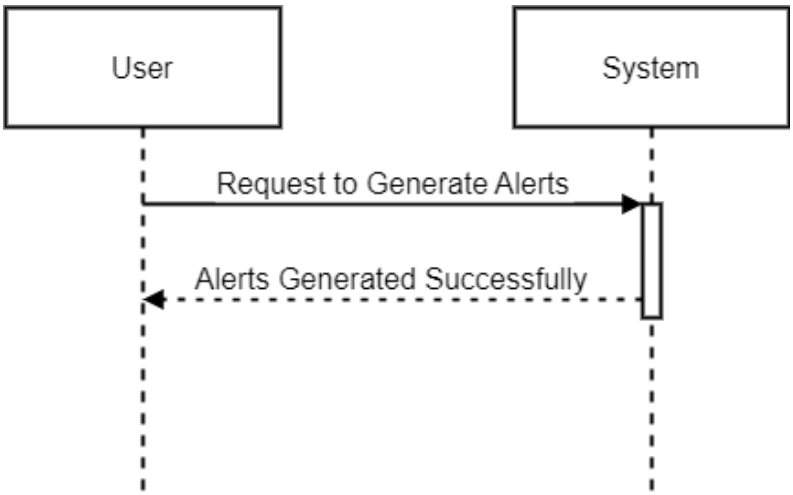
### 1.4.16. << UC-1>> UC_ Deploy Model




### 1.4.17. << UC-1>> UC_ Evaluate Model

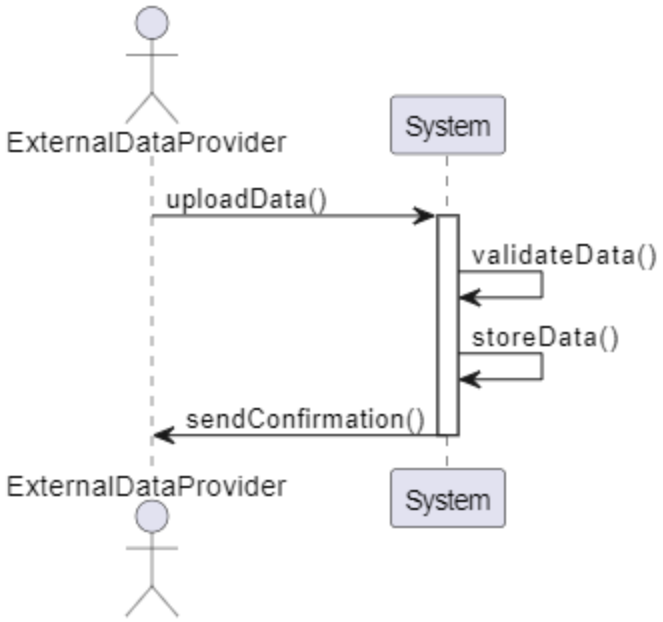


### 1.4.18. << UC-1>> UC_ Generate Alerts

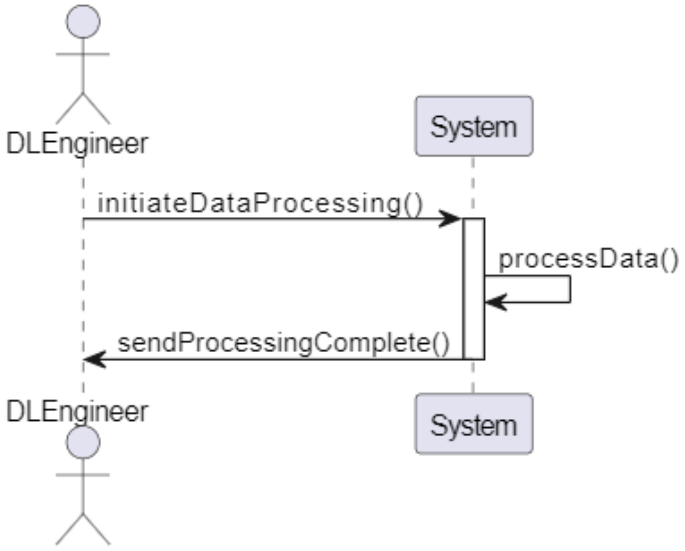


## *1.5 Collaboration Diagram*

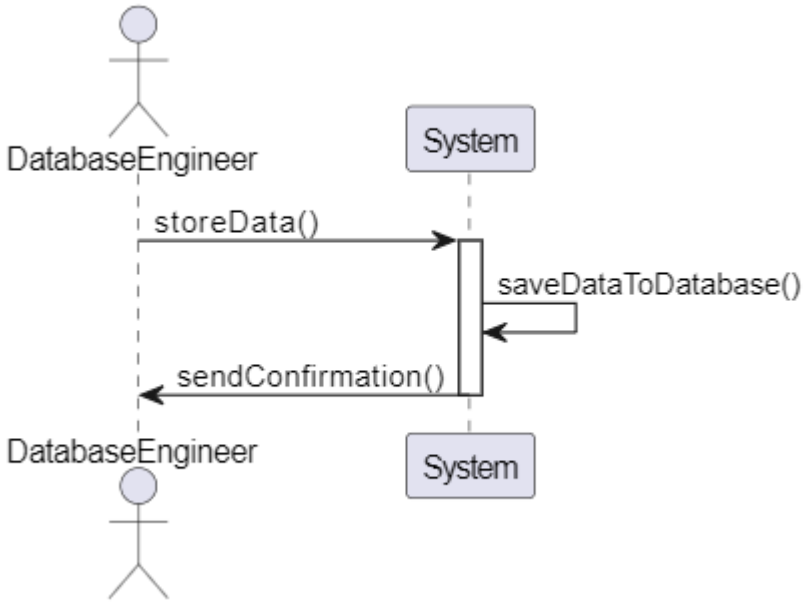
### 1.5.1. << UC-1>> UC_ Upload Data



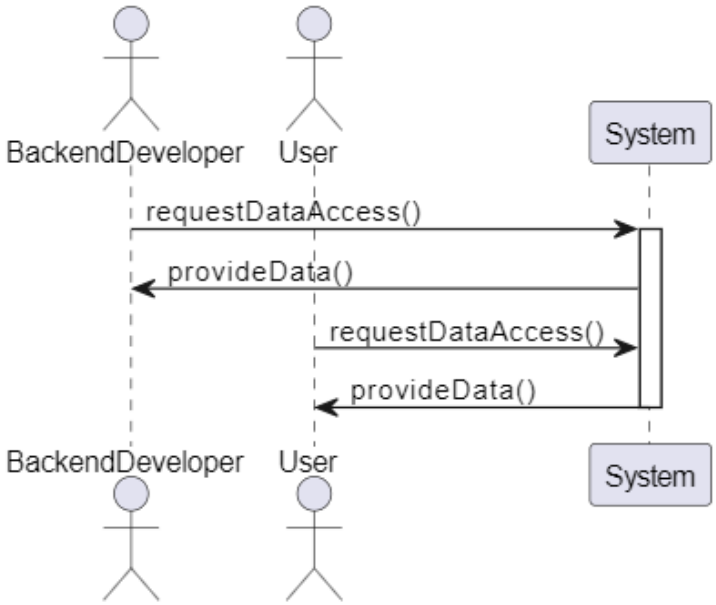

### 1.5.2. << UC-1>> UC_ Process Data



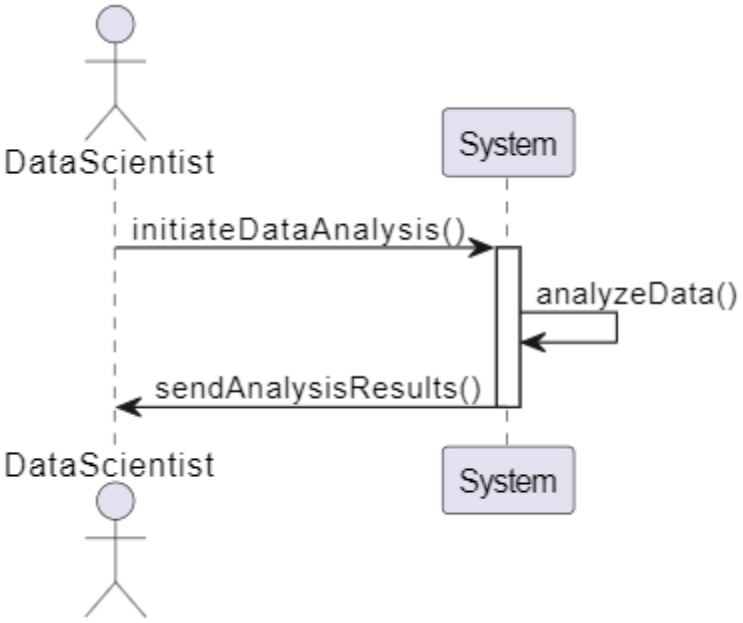
### 1.5.3. << UC-1>> UC_ Store Data



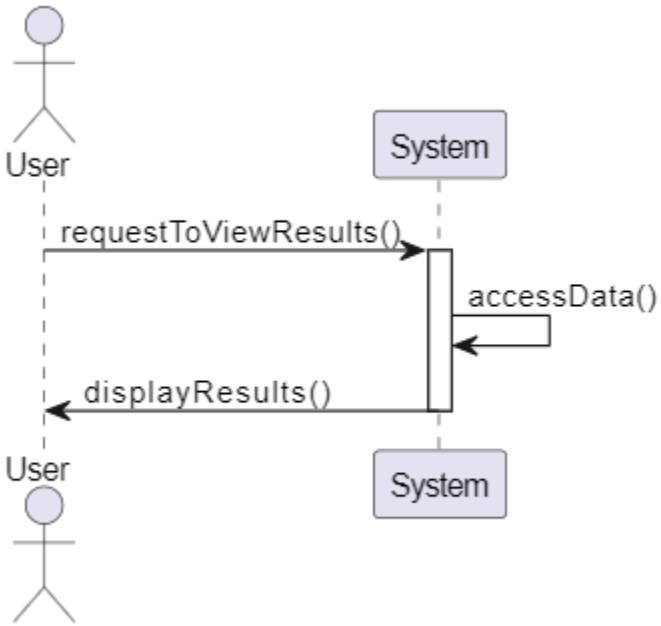

### 1.5.4. << UC-1>> UC_ Access Data



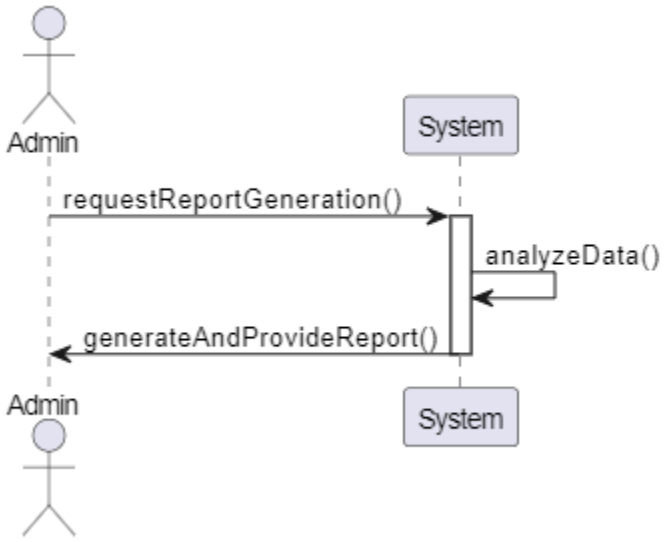
### 1.5.5. << UC-1>> UC_ Analyze Data



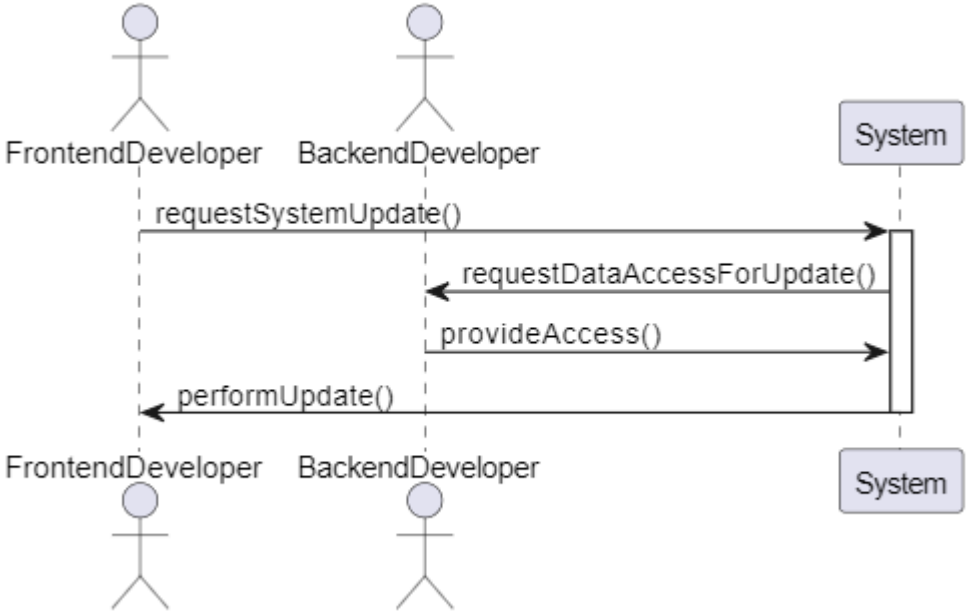

### 1.5.6. << UC-1>> UC_ View Results



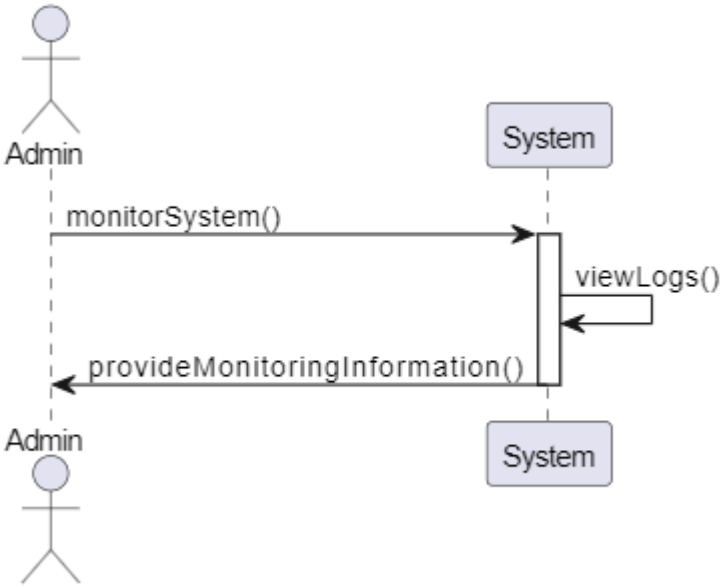
### 1.5.7. << UC-1>> UC_ Generate Reports



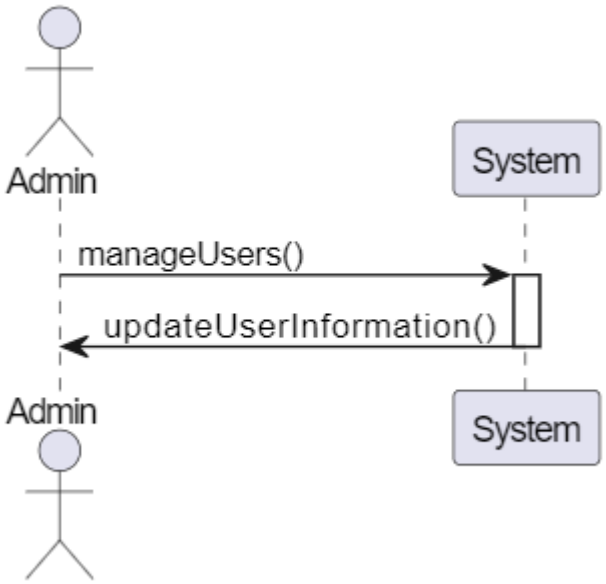

### 1.5.8. << UC-1>> UC_ Update System



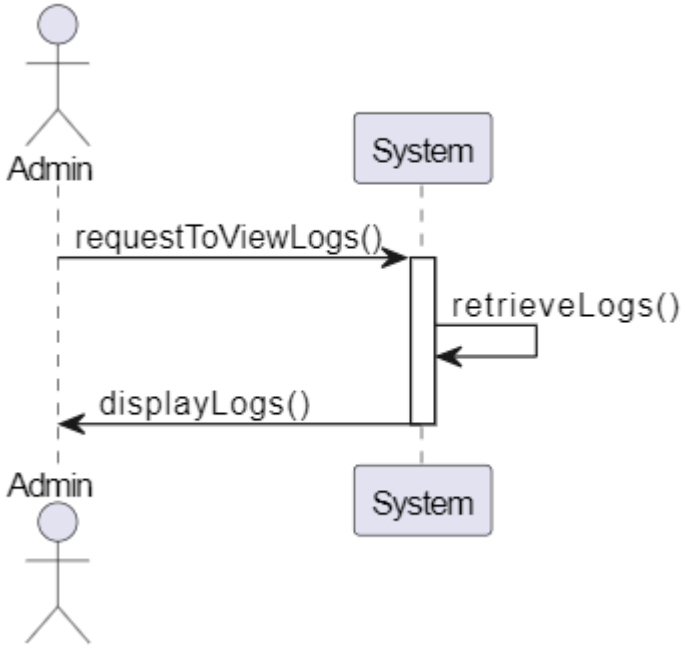
### 1.5.9. << UC-1>> UC_ Monitor System



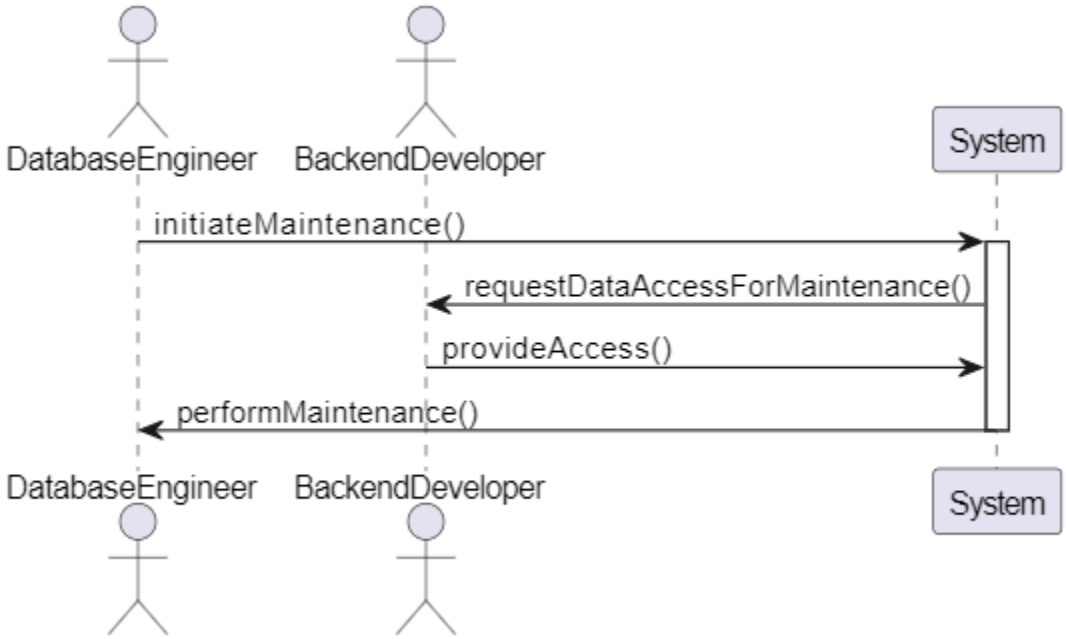

### 1.5.10. << UC-1>> UC_ Manage Users



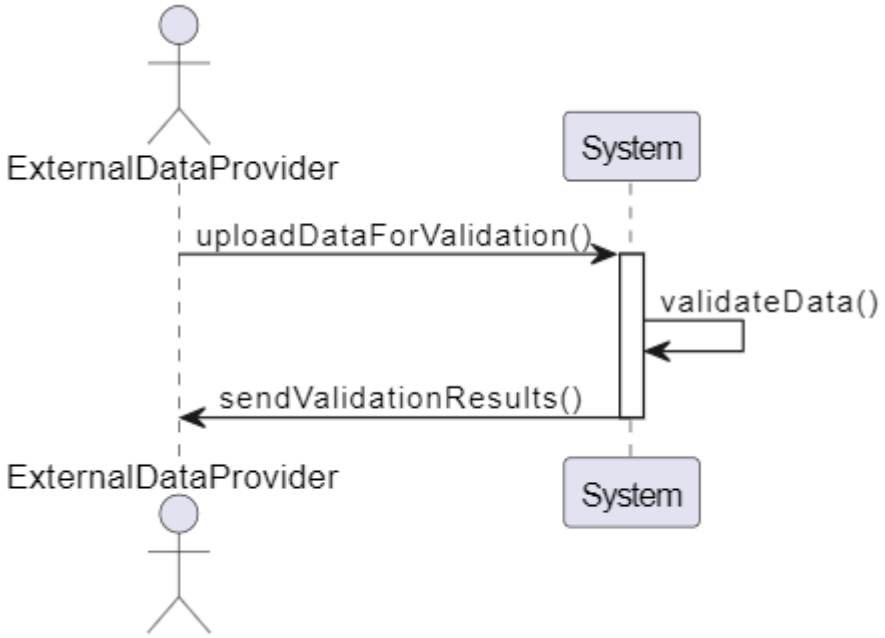
### 1.5.11. << UC-1>> UC_ View Logs



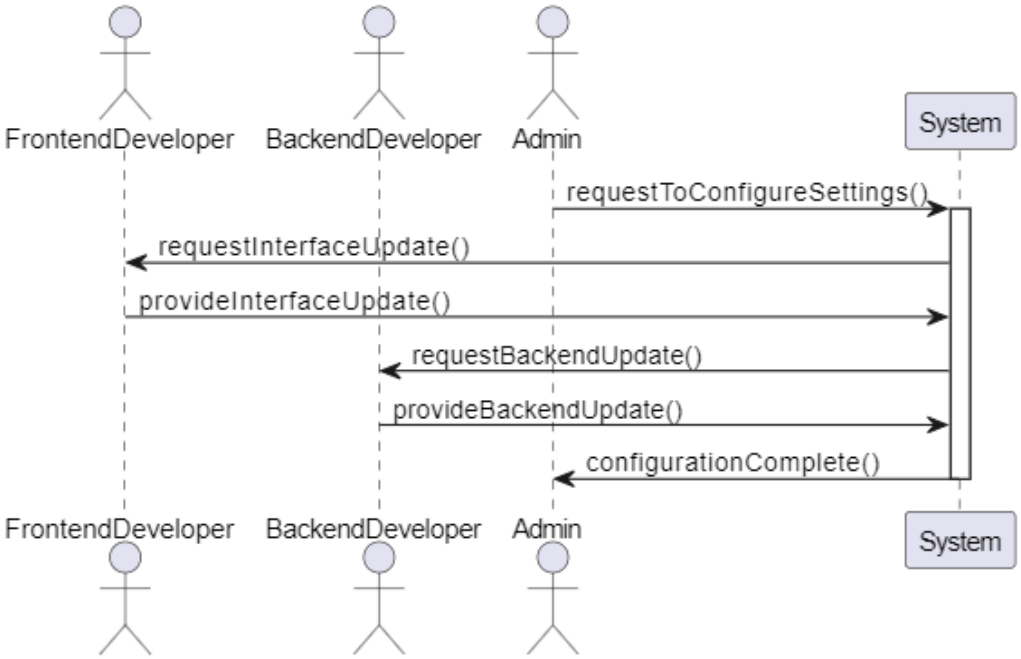

### 1.5.12. << UC-1>> UC_ Perform Maintenance



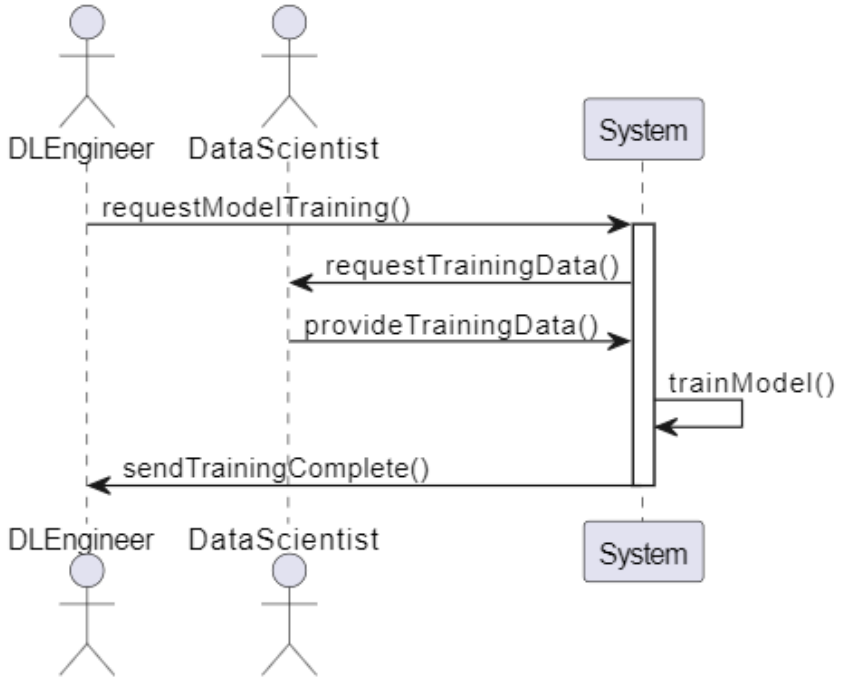
### 1.5.13. << UC-1>> UC_ Validate Data



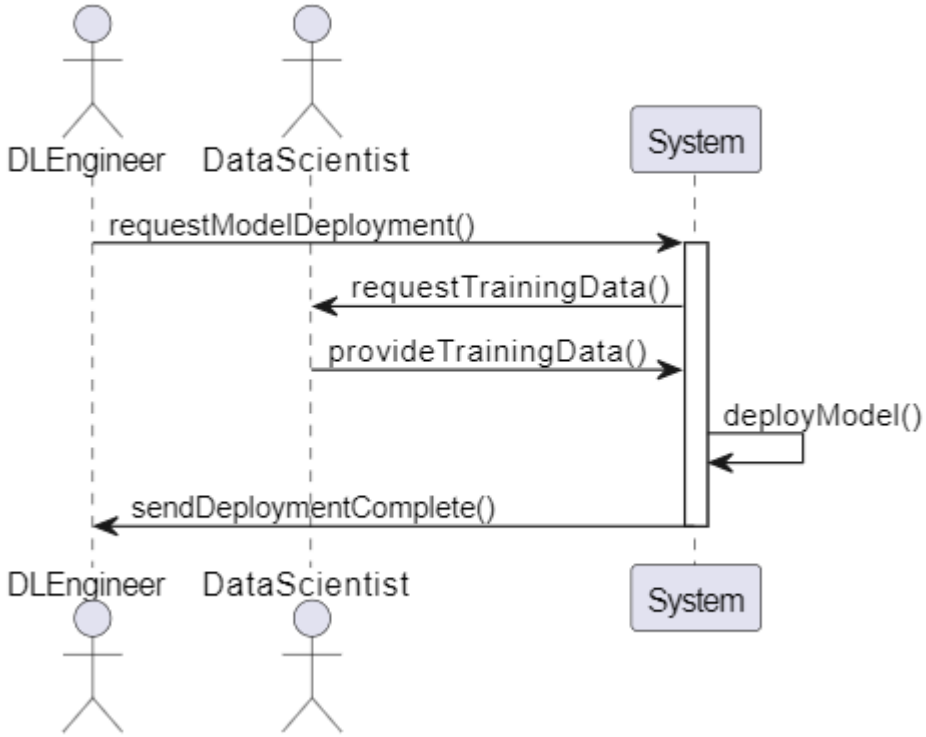

### 1.5.14. << UC-1>> UC_ Configure System Settings



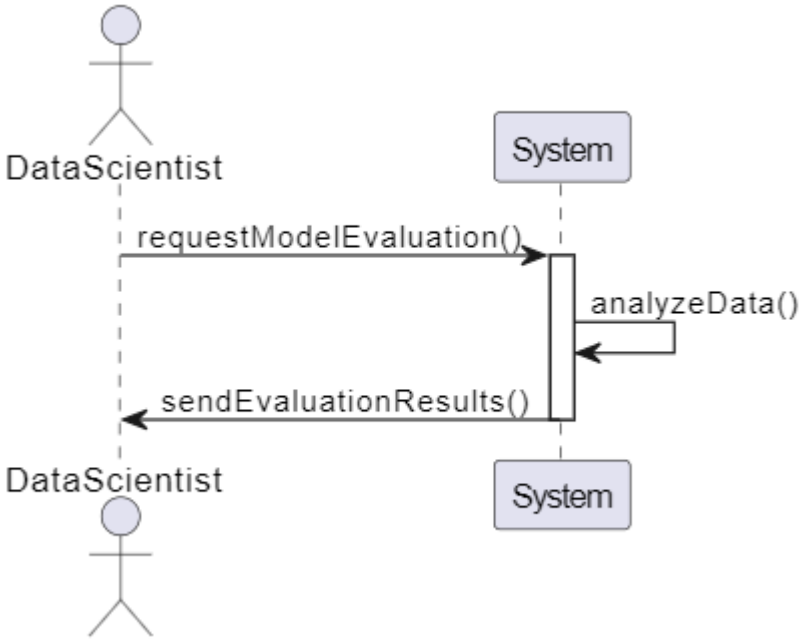
### 1.5.15. << UC-1>> UC_ Train Model



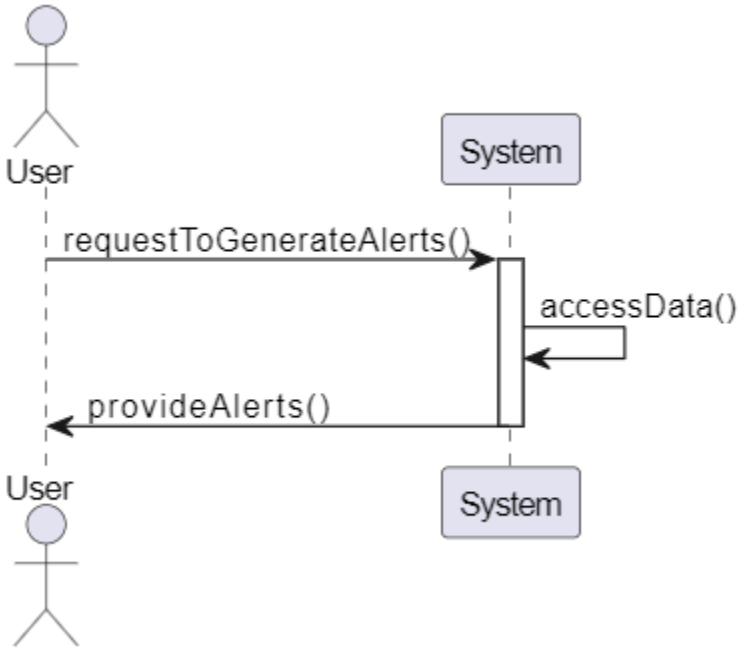

### 1.5.16. << UC-1>> UC_ Deploy Model



### 1.5.17. << UC-1>> UC_ Evaluate Model




### 1.5.18. << UC-1>> UC_ Generate Alerts



## *1.6 Operation Contracts*

### OC1: Upload Data

| Field | Description |
| --- | --- |
| **Name** | uploadData(data, model) |
| **Responsibilities** | • Initiates the process of uploading glacier debris data (provided in the "data" parameter).<br>• Allows for the optional selection of a pre-defined analysis model (provided in the "model" parameter, if applicable). |
| **Cross References** | UC_Upload Data |
| **Exceptions** | • Invalid data format: The uploaded data format is not recognized or is corrupt (e.g., incorrect file type, invalid file structure).<br>• Missing required data fields: Essential data fields within the glacier debris data are missing. |
| **Preconditions** | • A valid User object exists in the system (authenticated and authorized to upload data).<br>• The User object has the necessary permissions for data upload.<br>• The "data" parameter represents a valid glacier debris data object (e.g., image file).<br>• (Optional) The "model" parameter, if provided, represents a valid Model object for analysis. |
| **Postconditions** | • A new Data object representing the uploaded glacier debris data is created and stored securely within the system (association with DB Engineer).<br>• (Optional) The selected Model object is associated with the newly created Data object, linking it for analysis.<br>• The Backend Dev object is responsible for further processing the uploaded data and generating analysis results.<br>• A success or error message is displayed to the User based on the upload outcome (successful upload or details of encountered exceptions). |

### OC2: Process Data

| Field | Description |
| --- | --- |
| **Name** | processAndAnalyzeData(data, model) |
| **Responsibilities** | • Initiates the process of analyzing uploaded glacier debris data.<br>• Uses the associated analysis model (if provided) or a default model for the analysis. |
| **Cross References** | UC_ProcessData |
| **Exceptions** | • Data object not found (corrupted upload or missing data).<br>• Model object not found (associated model missing).<br>• Errors during data analysis (algorithmic issues, external dependencies). |
| **Preconditions** | • A valid Data object representing the uploaded glacier debris data exists.<br>• (Optional) A valid Model object representing the chosen analysis model exists (if applicable). |
| **Postconditions** | • The data is queued for processing by a designated processing engine.<br>• Analysis results are generated for the uploaded data using the chosen model (or default).<br>• Results are stored in the system (association with DB Engineer).<br>• An analysis report (e.g., maps, statistics) is generated in a user-friendly format (may involve separate functionalities within the Backend Dev object). |

**OC3: Store Data**

| Field | Description |
| --- | --- |
| **Name** | storeData(data) |
| **Responsibilities** | • Stores the uploaded data object (e.g., glacier debris data) in a secure and persistent manner. |
| **Cross References** | UC_Upload Data |
| **Exceptions** | • Database connectivity issues (inability to connect to the database).<br>• Storage quota limitations reached (system storage capacity is full). |
| **Preconditions** | • A valid Data object representing the uploaded data exists. |
| **Postconditions** | • The data object is stored securely within the system's database (association with DB Engineer).<br>• Metadata associated with the data (e.g., upload time, user information) may also be stored. |

**OC4: Access Data**

| Field | Description |
| --- | --- |
| **Name** | AccessData(dataID, user) |
| **Responsibilities** | • Retrieves the requested data object (e.g., glacier debris data) based on its unique identifier (dataID).<br>• Verifies user authorization to access the requested data. |
| **Cross References** | UC_Access Data |
| **Exceptions** | • Data object not found (corrupted data, deleted object).<br>• User not authorized to access the specific data (permission issues). |
| **Preconditions** | • A valid dataID representing the desired data object exists.<br>• A valid User object requesting access exists.<br>• The User object has the necessary permissions to access the data. |
| **Postconditions** | • If successful, the requested data object is retrieved and made available for viewing (may involve format adjustments for display).<br>• If unsuccessful, an error message is displayed to the User explaining the access issue. |

### OC5: Analyze Data

| Field | Description |
| --- | --- |
| **Name** | analyzeData(data, model) |
| **Responsibilities** | • Analyzes the uploaded glacier debris data using the associated analysis model (if provided) or a default model.<br>• (Optional) Processes and interprets the generated analysis results to extract insights (depends on system design). |
| **Cross References** | UC_AnalyzeData (potential) |
| **Exceptions** | • Data object not found (data missing or inaccessible).<br>• Model object not found (associated model missing).<br>• Errors during data analysis (algorithmic issues if re-analysis is attempted). |
| **Preconditions** | • A valid Data object representing the uploaded glacier debris data exists.<br>• Analysis results for the data are already generated.<br>• (Optional) A valid Model object representing the chosen analysis model exists (if applicable). |
| **Postconditions** | • The uploaded data is analyzed using the chosen model (or default).<br>• (Optional) Insights or interpretations of the results may be generated (depends on system design).<br>• Analyzed results are available for further processing or report generation. |

**OC6: View Results**

| Field | Description |
| --- | --- |
| **Name** | ViewResults(dataID, user) |
| **Responsibilities** | • Retrieves analysis results associated with the uploaded data identified by its dataID.<br>• Presents the retrieved results in a user-friendly format for authorized users. |
| **Cross References** | UC_ViewResults |
| **Exceptions** | • Data object not found (corrupted data, deleted object).<br>• Analysis results not found (analysis not completed or missing results).<br>• User not authorized to access the specific data (permission issues). |
| **Preconditions** | • A valid dataID representing the uploaded data exists.<br>• A valid User object requesting access exists.<br>• The User object has the necessary permissions to access the data and its associated analysis results.<br>• Analysis of the data must be completed successfully and results generated. |
| **Postconditions** | • If successful, the retrieved analysis results are presented to the User in a user-friendly format (e.g., charts, visualizations, reports).<br>• If unsuccessful, an error message is displayed to the User explaining the access issue or missing results. |

### OC7: Generate Reports

| Field | Description |
| --- | --- |
| **Name** | generateAnalysisReport(data, results) |
| **Responsibilities** | • Formats and processes the analysis results (data) into a comprehensive report suitable for user consumption.<br>• May include visualizations, charts, statistics, and textual summaries based on the analysis findings. |
| **Cross References** | UC_AnalyzeData (potential)<br><br>UC_ViewResults (potential) |
| **Exceptions** | • Data object not found (data missing or inaccessible).<br>• Analysis results not found (analysis not completed or missing results).<br>• Errors during report generation (formatting issues, missing data points). |
| **Preconditions** | • A valid Data object representing the uploaded glacier debris data exists.<br>• Analysis results for the data are already generated (potential post-condition of UC_AnalyzeData). |
| **Postconditions** | • A user-friendly analysis report is generated in a specified format (e.g., PDF, HTML).<br>• The report incorporates visualizations, statistics, and interpretations of the analysis findings.<br>• The report is made available for download or viewing by authorized users (potential link to UC_ViewResults). |

### OC8: Update System

| Field | Description |
| --- | --- |
| **Name** | updateSystem(updatePackage) |
| **Responsibilities** | • Downloads and verifies the integrity of the system update package.<br>• Backs up the current system configuration (optional, depending on system design).<br>• Applies the update package to the system software.<br>• Restarts the system services (if necessary). |
| **Cross References** | N/A (standalone Use Case) |
| **Exceptions** | • Update package download failure (network issues, server unavailability).<br>• Update package integrity verification failure (corrupted download, invalid package).<br>• Insufficient storage space for the update package.<br>• Errors during update application (compatibility issues, unexpected bugs). |
| **Preconditions** | • A valid update package file exists (downloaded or provided).<br>• The system is in a stable state and has sufficient resources for the update (storage, processing power).<br>• The user has the necessary permissions to initiate system updates. |
| **Postconditions** | • If successful, the system software is updated to the new version.<br>• The system restarts with the updated software (if necessary).<br>• An update log is generated, detailing the update process and outcome (success or failure).<br>• (Optional) The system configuration is restored from the backup (if created). |

### OC9: Monitor System

| Field | Description |
| --- | --- |
| **Name** | monitorSystemHealth(parameters) |
| **Responsibilities** | • Gathers system health data based on specified parameters (e.g., CPU usage, memory utilization, database connectivity).<br>• Analyzes the collected data to identify potential issues or performance bottlenecks.<br>• Generates alerts or notifications if critical thresholds are exceeded or anomalies are detected.<br>• Logs system health data for historical analysis and future reference. |
| **Cross References** | N/A (standalone Use Case) |
| **Exceptions** | • Errors during data collection (sensor malfunctions, communication issues).<br>• Issues with data analysis or anomaly detection algorithms. |
| **Preconditions** | • The system is operational with data collection mechanisms in place (sensors, monitoring tools).<br>• System health parameters for monitoring are defined. |
| **Postconditions** | • System health data is collected and analyzed at regular intervals.<br>• Alerts or notifications are triggered for critical events or potential issues.<br>• System health data is stored in a central repository for historical analysis and trend identification. |

**OC10: Manage Users**

| Field | Description |
| --- | --- |
| **Name** | manageUser(action, userID, userData, permissionSet) |
| **Responsibilities** | • Performs user account management actions based on the specified action parameter.<br>• Valid actions include: CreateUser, UpdateUser, DeleteUser, ManagePermissions, ResetPassword. |
| **Parameters** | • action (string): Defines the specific user management operation to perform (e.g., CreateUser, UpdateUser, etc.).<br>• userID (optional - string/number): Unique identifier for the target user account (relevant for UpdateUser, DeleteUser, ManagePermissions).<br>• userData (optional - object): User data for creating a new account or updating existing information (relevant for CreateUser, UpdateUser).<br>• permissionSet (optional - object): Set of permissions to assign or revoke for a user account (relevant for ManagePermissions). |
| **Cross References** | N/A (core functionality) |
| **Exceptions** | • Invalid action specified (unsupported user management operation).<br>• User account not found (invalid userID for UpdateUser, DeleteUser, ManagePermissions).<br>• Insufficient permissions to perform the action (e.g., trying to create an admin account without privileges).<br>• Username already exists (duplicate username violation for CreateUser).<br>• Invalid or incomplete user data (missing required fields for CreateUser, UpdateUser).<br>• Invalid or unsupported permission set (ManagePermissions).<br>• System errors during user management operations (database issues, email delivery problems for ResetPassword). |
| **Preconditions** | • The user initiating the action has the necessary permissions for the specified operation.<br>• Valid data is provided based on the chosen action (userData for CreateUser/UpdateUser, userID for update/delete/permissions, permissionSet for permission management). |

**OC11: View Logs**

| Field | Description |
| --- | --- |
| **Name** | viewLogs(filterCriteria, user) |
| **Responsibilities** | • Retrieves log data from the system based on provided filter criteria (e.g., log level, timestamp range, specific modules).<br>• Presents the retrieved log data in a user-friendly format for authorized users. |
| **Parameters** | • filterCriteria (object): Defines filters for selecting specific logs (optional, allows retrieval of all logs if omitted).<br>• user (object): Represents the user requesting access to logs. |
| **Cross References** | N/A (standalone Use Case) |
| **Exceptions** | • User not authorized to access system logs (permission issues).<br>• Errors during log retrieval (database issues, log data corruption). |
| **Preconditions** | • A valid User object requesting access exists.<br>• The User object has the necessary permissions to view system logs. |
| **Postconditions** | • If successful, filtered or complete system log data is presented to the user in a readable format (e.g., table, log viewer).<br>• Log data may include details like timestamps, severity levels, source modules, and relevant messages.<br>• An error message is displayed to the User if access is unauthorized or log retrieval fails. |

### OC12: Perform Maintenance

| Field | Description |
| --- | --- |
| **Name** | performMaintenance(task, parameters) |
| **Responsibilities** | • Executes a specific system maintenance task based on the provided task parameter.<br>• Valid tasks may include: Update System, Restart Services, Clear Cache, Run Diagnostics, etc. (add other relevant tasks specific to your system). |
| **Parameters** | • task (string): Defines the specific maintenance task to perform.<br>• parameters (optional - object): Additional parameters required for specific tasks (e.g., service names for Restart Services). |
| **Cross References** | UC_UpdateSystem (potential) |
| **Exceptions** | • Invalid task specified (unsupported maintenance operation).<br>• Errors during maintenance execution (update failures, service restart issues).<br>• Insufficient permissions to perform the task (e.g., requiring admin privileges for system updates). |
| **Preconditions** | • The user initiating the maintenance task has the necessary permissions.<br>• Valid parameters are provided based on the chosen task (optional parameters for specific tasks).<br>• The system is in a stable state suitable for maintenance activities (consider scheduling downtime). |
| **Postconditions** | • If successful, the chosen maintenance task is executed.<br>• The system state may be updated (e.g., software version change after update, services restarted).<br>• Diagnostic results may be generated (for Run Diagnostics).<br>• An error message is displayed if the task fails or the user lacks permissions. |

### OC13: Perform Maintenance

| Field | Description |
| --- | --- |
| **Name** | configureSetting(settingName, settingValue, user) |
| **Responsibilities** | • Updates a specific system setting based on the provided setting name and value.<br>• Performs validation checks to ensure the setting value is within allowed ranges or adheres to specific formats. |
| **Parameters** | • settingName (string): Name of the system setting to be modified.<br>• settingValue (string/number/object): New value for the specified setting.<br>• user (object): Represents the user requesting the configuration change. |
| **Cross References** | N/A (standalone Use Case) |
| **Exceptions** | • Setting name not found (invalid system setting).<br>• Invalid setting value (e.g., out of range, incorrect format).<br>• User not authorized to modify system settings (permission issues).<br>• Errors during configuration update (system errors, permission limitations). |
| **Preconditions** | • A valid User object requesting the change exists.<br>• The User object has the necessary permissions to modify system settings.<br>• A valid setting name representing an existing system setting is provided.<br>• A valid setting value adhering to the expected format and constraints for the specific setting is provided. |
| **Postconditions** | • If successful, the specified system setting is updated with the new value.<br>• The system configuration may be reflected in system behavior (e.g., language change, logging level adjustment).<br>• An error message is displayed to the User if the setting name is invalid, the value is invalid, or the user lacks permissions. |

### OC14: Validate Data

| Field | Description |
| --- | --- |
| **Name** | validateData(data, rules) |
| **Responsibilities** | • Performs a comprehensive validation check on the provided data object using predefined validation rules.<br>• Utilizes a set of predefined validation rules (rules) to assess data integrity, including data type checks, format checks, range checks, and consistency checks. |
| **Parameters** | • data (object): The data object to be validated (e.g., uploaded glacier debris data).<br>• rules (object): A set of validation rules to be applied to the data (may include data type checks, format checks, range checks, consistency checks, etc.). |
| **Cross References** | UC_StoreData (potential), UC_AnalyzeData (potential) |
| **Exceptions** | • Invalid data format (corrupted data or incompatible structure).<br>• Data failing validation checks based on the defined rules (e.g., missing required fields, values outside allowed ranges). |
| **Preconditions** | • A valid data object representing the data to be validated exists.<br>• A valid rules object containing the validation rules to be applied exists. |
| **Postconditions** | • The data is validated against the defined rules.<br>• If successful, a validation success message is generated, indicating the data is ready for further processing.<br>• If unsuccessful, a detailed error report is generated, specifying the validation failures encountered (e.g., missing fields, invalid formats). |

### OC15: Train Model

| Field | Description |
| --- | --- |
| **Name** | trainModel(modelData, trainingParameters) |
| **Responsibilities** | • Trains a machine learning model based on the provided model data and training parameters.<br>• Model data may include training datasets, feature engineering specifications, and model architecture definition.<br>• Training parameters specify training algorithms, hyperparameters (learning rate, epochs), and optimization techniques (e.g., gradient descent). |
| **Parameters** | • modelData (object): Encompasses data and specifications required for model training.<br>• Training Data: Historical or labeled data used to train the model.<br>• Feature Engineering: Techniques for transforming raw data into features suitable for the model.<br>• Model Architecture: Definition of the model's structure (e.g., layers, neurons for neural networks).<br>• trainingParameters (object): Defines settings for the training process.<br>• Training Algorithm: The machine learning algorithm used for training (e.g., decision tree, neural network).<br>• Hyperparameters: Tuning parameters controlling the learning process (e.g., learning rate, epochs).<br>• Optimization Techniques: Methods to minimize the loss function and improve model performance (e.g., gradient descent, Adam optimization). |
| **Cross References** | UC_AnalyzeData (potential) |
| **Exceptions** | • Errors during data loading or preprocessing (e.g., corrupted training data, incompatible formats).<br>• Model training failures (e.g., convergence issues, overfitting, underfitting).<br>• Insufficient computational resources for training (e.g., memory, processing power limitations). |
| **Preconditions** | • Valid model data object containing training data, feature engineering specifications, and model architecture definition exists.<br>• Valid training parameters object specifying the training algorithm, hyperparameters, and optimization techniques exists. |

| | |
| --- | --- |
| **Postconditions** | • The system has access to necessary computational resources for training (e.g., GPU availability for deep learning models).<br><br>• If successful, a trained machine learning model is generated.<br>• The trained model is stored within the system for further use in tasks such as data analysis or prediction.<br>• Training logs or metrics may be generated to track training progress and evaluate model performance (e.g., accuracy, loss curves).<br>• In case of training failures, error messages or reports are generated to facilitate troubleshooting and model improvement (e.g., parameter tuning, data adjustment). |

### OC16: Train Model

| Field | Description |
| --- | --- |
| **Name** | deployModel(model, deploymentParameters) |
| **Responsibilities** | • Packages the trained model for deployment based on system infrastructure and requirements.<br>• Deploys the packaged model to a designated serving environment (e.g., web server, containerized environment).<br>• Integrates the deployed model with the system's workflow for real-time or batch processing of data. |
| **Parameters** | • model (object): The trained machine learning model to be deployed.<br>• deploymentParameters (object): Defines configurations for the deployment process.<br>• Serving Environment: Specifies the target environment for deployment (e.g., web server, container runtime).<br>• Model Serialization Format: Defines the format for storing and loading the model (e.g., joblib, PMML).<br>• Resource Allocation: Configures resources allocated to the deployed model (e.g., CPU, memory). |
| **Cross References** | UC_TrainModel (precursor), UC_AnalyzeData (potential) |
| **Exceptions** | • Errors during model packaging (e.g., incompatibility with deployment environment).<br>• Deployment failures in the serving environment (e.g., resource limitations, security issues).<br>• Integration issues with the system's workflow for processing data. |
| **Preconditions** | • A valid trained model object (refer to UC_TrainModel) is provided.<br>• Valid deployment parameters object specifying the serving environment, model serialization format, and resource allocation exists.<br>• The target deployment environment is configured and accessible. |
| **Postconditions** | • If successful, the trained model is deployed to the designated serving environment.<br>• The deployed model is integrated with the system's data processing pipeline for real-world use (e.g., scoring new data, making predictions).<br>• Monitoring configurations may be set up to track the model's |

| | |
| --- | --- |
| | performance and health in production.<br><br>• In case of deployment failures, error messages are generated to identify and troubleshoot the issue. |

### OC17: Evaluate Model

| Field | Description |
| --- | --- |
| **Name** | evaluateModel(model, data, evaluationMetrics) |
| **Responsibilities** | • Evaluates the performance of the provided model on a separate evaluation dataset (data).<br>• Calculates a set of pre-defined evaluation metrics (evaluationMetrics) to assess the model's effectiveness. |
| **Parameters** | • model (object): The trained machine learning model to be evaluated.<br>• data (object): A separate dataset specifically designated for model evaluation (should not be the training data).<br>• evaluationMetrics (object): A set of metrics to be calculated for evaluation (e.g., accuracy, precision, recall, F1-score, AUC-ROC). |
| **Cross References** | UC_TrainModel (precursor) |
| **Exceptions** | • Errors during data loading or preprocessing for evaluation.<br>• Issues during evaluation metric calculation (unsupported metrics, errors in implementation). |
| **Preconditions** | • A valid trained model object (refer to UC_TrainModel) is provided.<br>• A valid evaluation dataset (data) separate from the training data exists.<br>• A valid evaluation metrics object specifying the desired metrics to be calculated exists. |
| **Postconditions** | • The model is evaluated on the evaluation dataset.<br>• A comprehensive evaluation report is generated, including the calculated values for each chosen evaluation metric.<br>• The report interprets the metrics and provides insights into the model's strengths and weaknesses (e.g., high accuracy but low precision for a classification model).<br>• Based on the evaluation results, a decision can be made regarding model deployment (deploy if performance meets requirements) or further refinement (retrain with adjusted hyperparameters or data augmentation). |

### OC18: Generate Alerts

| Field | Description |
| --- | --- |
| **Name** | generateAlerts(data, alertCriteria) |
| **Responsibilities** | • Monitors data against predefined alert criteria.<br>• Generates alerts if certain thresholds or conditions specified in the alert criteria are met. |
| **Parameters** | • data (object): The dataset or stream of data to be monitored for generating alerts.<br>• alertCriteria (object): A set of conditions or thresholds that trigger alerts (e.g., CPU usage above 90%, disk space below 10%). |
| **Cross References** | N/A (standalone Use Case) |
| **Exceptions** | • Errors in data processing or retrieval.<br>• Issues with evaluating alert criteria (unsupported conditions, incorrect thresholds). |
| **Preconditions** | • Valid data object to be monitored is available.<br>• Valid alert criteria object specifying the conditions for alerts exists.<br>• The system is capable of monitoring the specified data in real-time or at regular intervals. |
| **Postconditions** | • Data is continuously or periodically monitored based on the provided criteria.<br>• Alerts are generated and logged if specified conditions are met.<br>• Notifications (e.g., emails, system messages) are sent to the relevant stakeholders if configured.<br>• A report or log of generated alerts is maintained for historical analysis and auditing purposes. |

## *1.7 Class Diagram*

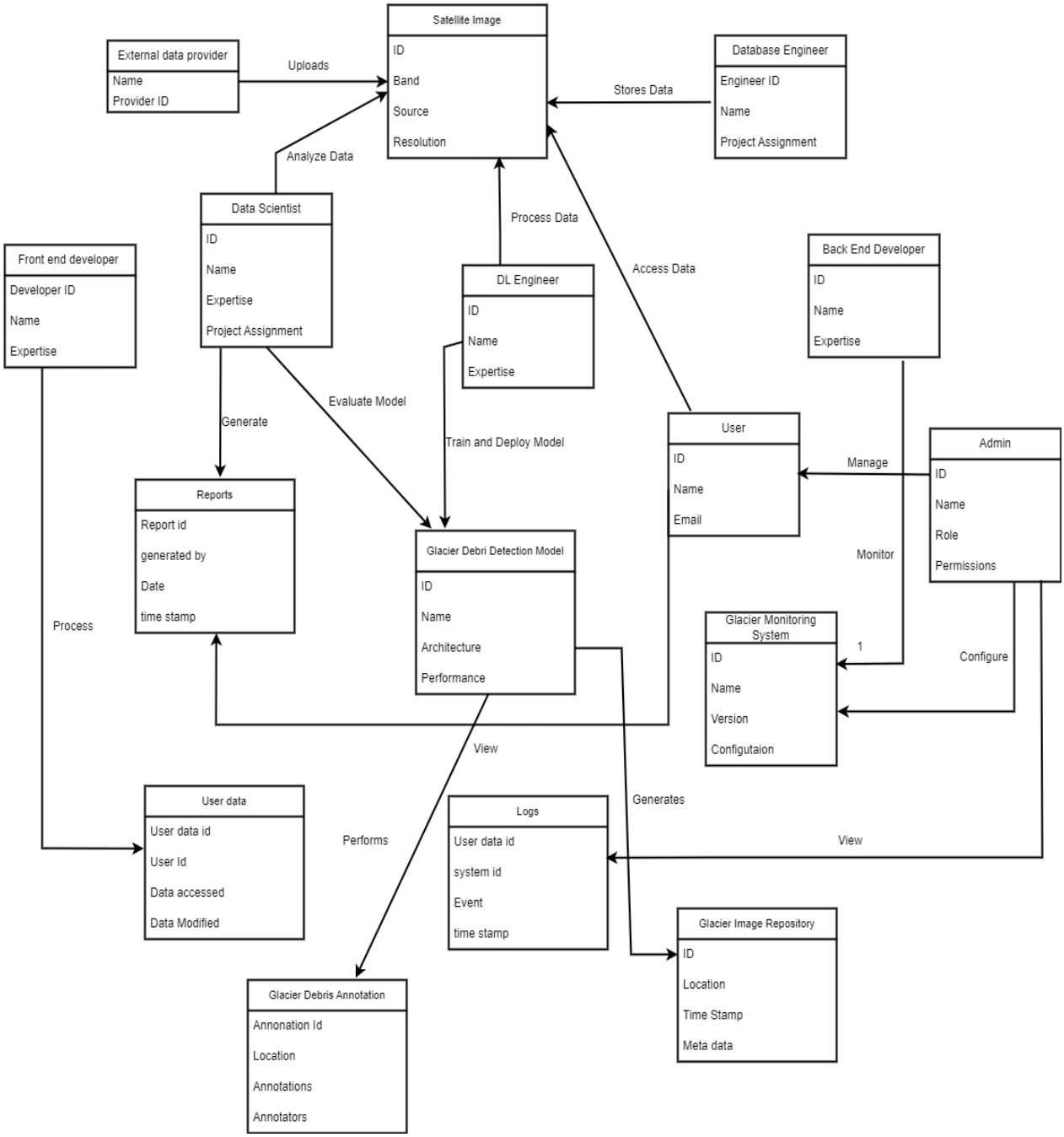



**Figure 9: Class Diagram**

## *1.8 Data Model*

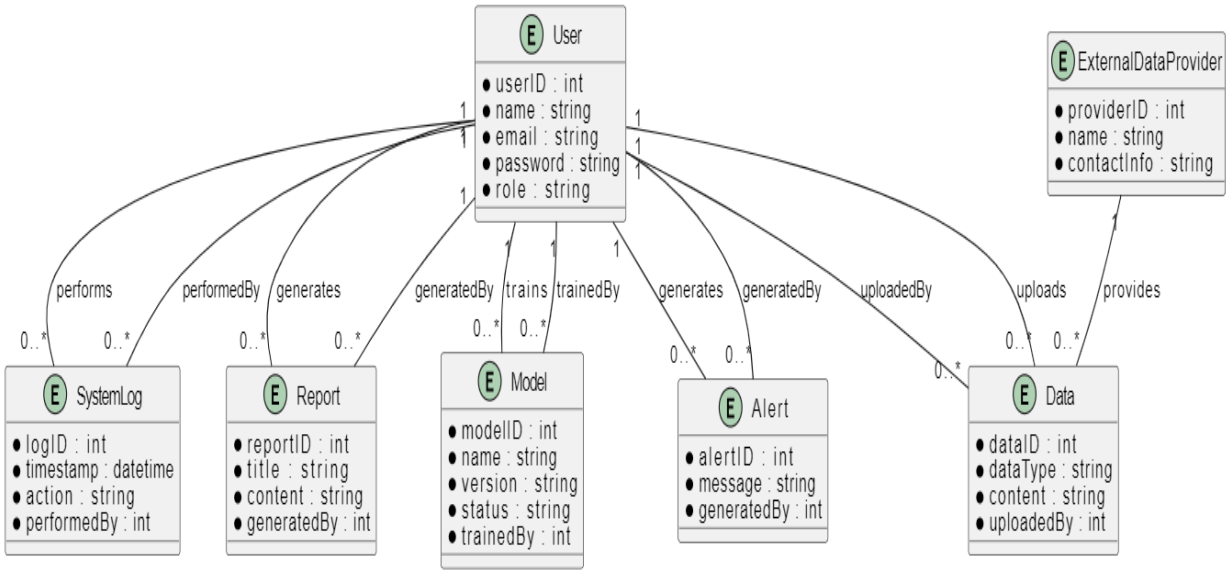



**Figure 10: Data Model**