2. Analysis

2.1 Introduction to Analysis

Analysis is the systematic and descriptive process of evaluating the complex domain and illustrating the complex domain as a simple module or sub-system by breaking them further apart. It is the first stage where all the primary requirements are identified and gathered for further analysis and problem solving. Without analysis, further improvement cannot be imagined.

Analysis phase helps in the identification and formulation of **Proof of concept (PoC)**. The identified requirements are then analyzed using a chosen analysis methodology.

2.2 Analysis Methodology

The preferred analysis methodology is **Object Oriented Analysis (OOA).** Object Oriented Analysis (OOA) is process of discovery where a development team understands and models the requirements of the system. In OOA requirements are organized as objects. It integrates all the process and data. But in others or traditional structural analysis both process and data are considered independently/separately. They use flow chart/structure charts for process and ER diagrams for data.

But in OOA some advance models are used. The common models used in OOA are: Use cases, Object models. Use cases describe pictures or overview for standard domain functions that the system must achieved. Object models describe the names, class relations, operations, and properties of the main objects. User-interface prototypes can also be created for better understanding (Cybarlab, 2017).

OOA Modellings

In OOA, there various modelling methodologies: Static Modelling, Dynamic Modelling and Functional Modeling.

* Static Modelling

It shows the relationship between the static constituents of the system. Static modelling are usually represented by Class diagram. Hence, the overall system’s structure is modelled in this modelling. In this analysis phase, initial class diagram is modelled with the identification of primary requirements.

* Dynamic / Functional Modelling

This modelling represents the behavioral aspects of the system. Dynamic and functional modelling can be generalized as a single modelling method where the control information are represented by series of events and operations that happen in the objects. Use case diagram, Sequence diagram, Activity diagram represents the dynamic model of the system.

2.3 Feasibility Study

In technical term, feasibility study is an assessment measure taken by the project owner in order to determine positive and negative outcomes, perform cost-benefit analysis and identify relevant factors that affect the project for its completion. There are various factors to consider in order to declare the project to be feasible. And they include social, operational, technical, legal, economical, etc.

|  |  |  |
| --- | --- | --- |
| **Factors** | **Summary** | **Feasibility Study** |
| Economic | Will it reduce organizational costs? | Yes |
| Will it provide economic benefits to the organization? | Yes |
| Can small organization afford the system? Is the product scalable as per the organization size? | May be |
| Social | Will there be a significant effect to the product by branding of similar other products? | May be |
| Will there be any need for further management of development time-box due to cultural and environmental factors? | May be |
| Is the proposed product sustainable in future market as per the organization trends? | Yes |
| Technological | Can rapid development of the product be achieved? | Yes |
| Can the product be compatible with emerging technologies? | Somewhat |
| Can we quickly mend the vulnerable dependency in the system? | Yes |
| Legal or Political | Can change in government regulation affect the product market value? | May be |
| Is there any chance of possible affect to the product due to change in trade regulations? | No |

2.4 Software Requirement Specification (SRS)

Software Requirement Specification is the document that describes the software system that is going to be developed with inclusion of functional and non-functional requirements along with relevant other use cases that finalizes the quality product. It is sometime called as Software or Requirement Engineering as well.

As the development methodology is preferred to be PXP (Personal Extreme Programming) which is endorsed by Agile methodology, the requirements are collected and recorded as **Product Backlog Items** where the user’s story are appropriately and concisely presented. In case of this project, the requirements are solely identified by me through the research of current academic organization situation.

Functional Requirements

These requirements are the ones that are very essential for the product to meet the behavioral requirement. It is the backbone of the proposed system. These requirements are focused with major priority.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| FR-01 | User Registration | Administrator should be able to add new user. | To entry new user so that different academic stakeholders can access their custom profiles. | N/A |
| FR-02 | User Login and Authentication | Registered users should be able to login after authentication is passed. | To validate and authorize the user login. | FR-01 |
| FR-03 | Organization’s details Entry | Users with granted permission should be able to add organization detail. | To add details in organization CMS | FR-02 |
| FR-04 | Organization’s detail Update | Permitted user should be able to edit existing details. | To edit existing details of organization. | FR-03 |
| FR-05 | View Organization’s details | All users should be able to view organization’s information | To display organization details. | FR-03 |
| FR-06 | Organization details Deletion | Allow granted user to delete existing organization information. | To remove obsolete organization details. | FR-03 |
| FR-07 | Student’s CMS | Granted user can add, edit, view and delete student records. | To create and manipulate student’s portfolio. | FR-02 |
| FR-08 | Academic Detail CMS | Permitted user (faculty staffs) should be able perform CRUD operation for academic details of the students. | To manage the academic information of the students. | FR-07 |
| FR-09 | Data Validation | Both client-side and server-side validation must be implemented. Form validation and entity (bean) validation must be integrated. | To maintain data integrity and correctness. | N/A |
| FR-10 | Centralized Student Information | Permitted user should be able to view every student details and transactions in the same web page. | To provide centralized summary details of every student. | FR-07, FR-08 |
| FR-11 | Access Control | Moderation of the system as per various authorization levels and permissions. | To filter the view page as per granted roles. | FR-02 |
| FR-12 | Tabular Representation | Representation of resource usage of the organization in tabular format. | To summarize the usage in tables. | FR-03 |
| FR-13 | Student’s analysis | Graphical representation of student’s analyzed information | To view the performance of the student. | FR-10 |
| FR-14 | Student Report Generation | Automated report generation of student details. | To trigger reports generation automatically. | FR-08 |
| FR-15 | Exam and Lecture notices | Automate notices in the organization. | To automate flow of notices within organization. | FR-03 |
| FR-16 | E-invoices | Automate creation and forward invoices through emails. | To automate invoice generation and delivery. | FR-07 |
| FR-17 | Multiple Sessions | Allow multiple sessions to run in runtime. | To allow multiple user use system concurrently. | FR-02 |
| FR-18 | Session Management | Invalidate stale sessions. | To nullify inactive sessions. | FR-02 |
| FR-19 | Intuitive UI | Functionality of the system should be easily achieved through convenient user interface. | To increase user’s productivity. | N/A |

Non-functional Requirements

These are the non-essential requirements which strengthen the developed system with additional features.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| NFR-01 | Access Security | Deny deliberate and intrusive access faults | To implement access control as per authentication and authorization. | N/A |
| NFR-02 | Integrity | Assure data stored by the software is authentic, accurate and not corrupted. | To guarantee data authenticity. | N/A |
| NFR-03 | Flexibility | Easy software modification to adapt varying environments, and user needs. | To provide flexible configuration of the system. | N/A |
| NFR-04 | Portability | Easy transfer of software from current hardware or software environment to other one. | To ease software portability | N/A |
| NFR-05 | Reliability | Perform specified function consistently without failure. | To maintain consistency in the system. | N/A |
| NFR-06 | Maintainability | Easy detection and fix of a software system. | To easily maintain the software while in need. | N/A |
| NFR-07 | Scalability | Able to expand the processing and data handling capabilities. | To cope with business growth and needs. | N/A |
| NFR-08 | Availability | Active system in normal operating times. | To maintain system availability. | N/A |
| NFR-09 | Confidentiality | Protection of sensitive data from unauthorized users. | To safeguard confidential credentials. | N/A |
| NFR-10 | Usability | Convenient interaction of the user with the system. | To ease learning, operating process of the user. | N/A |
| NFR-11 | Efficiency | Effective handling of program load, throughput, response time. | To handle higher resource usage by the system. | N/A |
| NFR-12 | Interoperability | Able to facilitate and communicate the interface with other system. | To operate the system with other compatible system. | N/A |

2.4.3 MoSCoW Prioritization

|  |  |  |
| --- | --- | --- |
| **ID** | **Requirement** | **MoSCoW** |
| FR-01 | User Registration | M |
| FR-02 | User Login and Authentication | M |
| FR-03 | Organization’s detail Entry | M |
| FR-04 | Organization’s detail Update | M |
| FR-05 | View Organization’s detail | M |
| FR-06 | Organization details deletion | M |
| FR-07 | Student’s CMS | M |
| FR-08 | Academic Detail CMS | M |
| FR-09 | Data Validation | S |
| FR-10 | Centralized Student Information | M |
| FR-11 | Access Control | S |
| FR-12 | Tabular Representation | S |
| FR-13 | Student’s analysis | S |
| FR-14 | Student Report Generation | M |
| FR-15 | Exam and Lecture notices | W |
| FR-16 | E-invoices | M |
| FR-17 | Multiple Sessions | M |
| FR-18 | Session Management | M |
| FR-19 | Intuitive UI | W |
| NFR-01 | Access Security | M |
| NFR-02 | Integrity | S |
| NFR-03 | Flexibility | S |
| NFR-04 | Portability | S |
| NFR-05 | Reliability | M |
| NFR-06 | Maintainability | M |
| NFR-07 | Scalability | M |
| NFR-08 | Availability | M |
| NFR-09 | Confidentiality | M |
| NFR-10 | Usability | M |
| NFR-11 | Efficiency | M |
| NFR-12 | Interoperability | C |

2.4.4 Hardware Software Specification

Development Environment Specification

The specification of the system used while developing the product is mentioned below.

**Hardware Specification**

|  |  |
| --- | --- |
| **Item** | **Specification** |
| CPU | Intel® Core™ i5-4200U CPU @ 1.60GHz × 4 |
| RAM | 12 GB |
| Hard Disk | 500 GB |
| Screen Resolution | 1366 x 768 (16:9) |

**Software Specification**

|  |  |
| --- | --- |
| **Item** | **Specification** |
| Operating System (OS) | Ubuntu 18.04.2 LTS |
| Java | Oracle JDK 1.8.0\_201, JRE 1.8.0\_201 |
| Web browsers | Google Chrome v74.0.3729.131 (64-bit), Mozilla Quantum v66.0.3 |
| Integrated Development Environment | IntelliJ IDEA ULTIMATE 2019.1.1 |
| Database | MySQL Ver 14.14 Distrib 5.7.26, for Linux (x86\_64) |

Product Configuration/Deployment Specification

The hardware and software requirements for the product is specified below:

**Hardware Specification**

|  |  |
| --- | --- |
| **Item** | **Minimum Requirement** |
| CPU | Intel® Pentium™ 4 with a clock speed of 2 GHz or higher |
| RAM | 4 GB or higher |
| Hard Disk | 100 GB or higher |
| Screen Resolution | 1024 x 720 or higher |

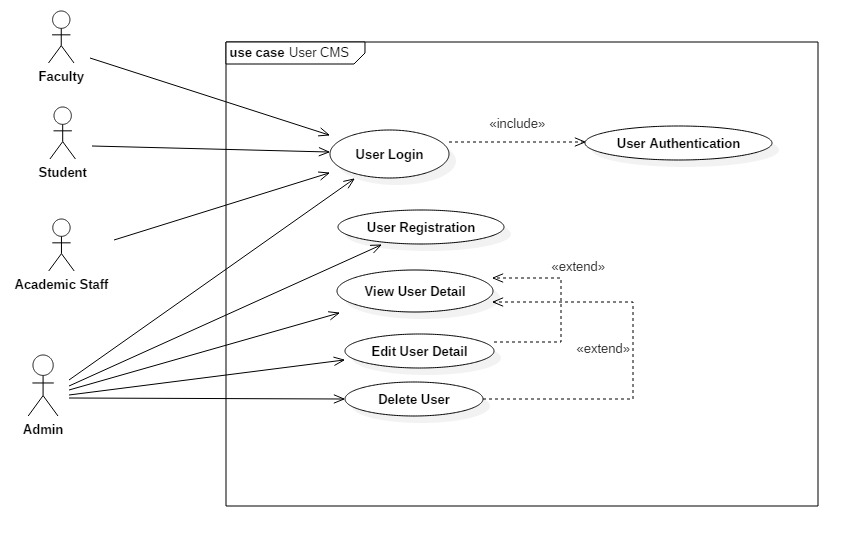
**Software Specification**

|  |  |
| --- | --- |
| **Item** | **Minimum Requirement** |
| Operating System (OS) | Microsoft® Windows® 7 or Windows® 10 or other Linux distros |
| Java | JAVA(TM) SE Runtime Environment (Oracle JRE 1.8.0\_201) |
| Web browsers | Microsoft® Internet Explorer® v11 or Google Chrome v74 (64-bit) or Mozilla Quantum v66 |
| Web Server | Apache Tomcat 9.0.2 |
| Database | mysql Ver 14.14 Distrib 5.7.26, for Linux (x86\_64) |

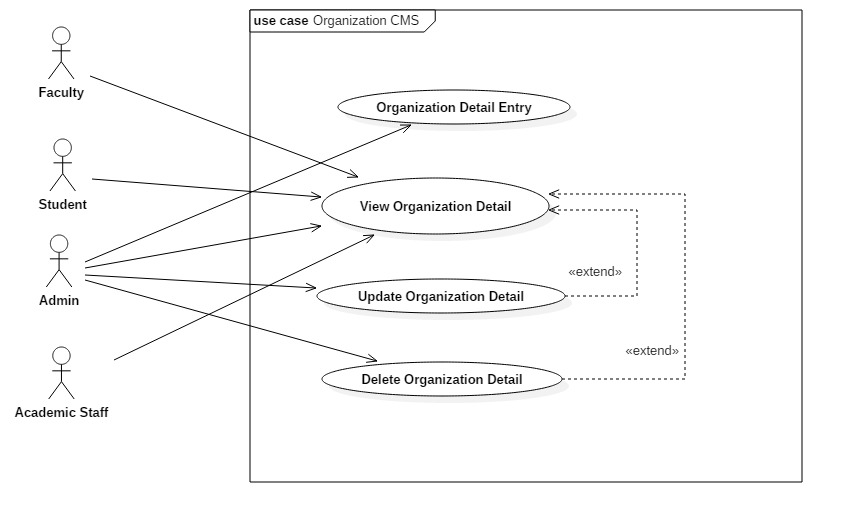
2.5 Use-case diagram

Use-case diagram also known as behavioral diagram presents the system’s high level requirement analysis by clearly displaying the interaction of the user (known as the actor) with different parts of the system.

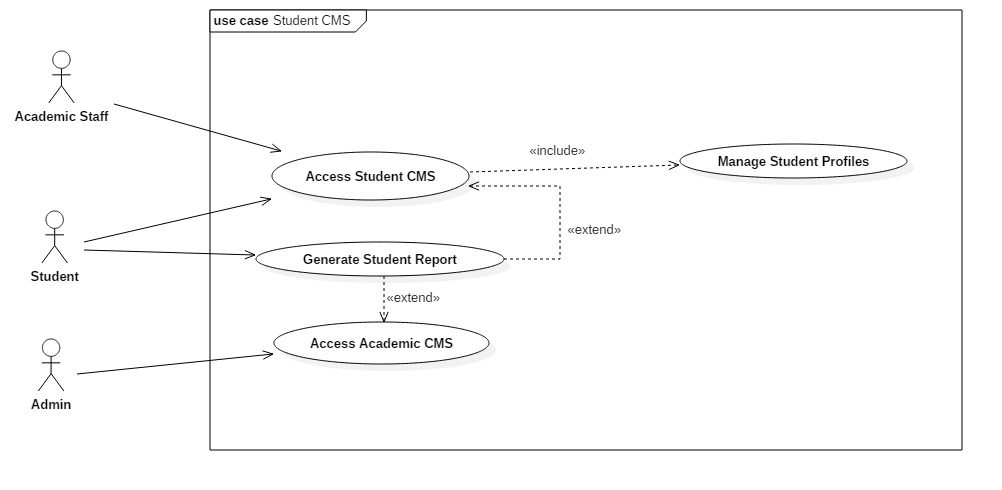
The diagram below shows the use-case for Automated Academic Organization System where four different users are presented with different access in the system.



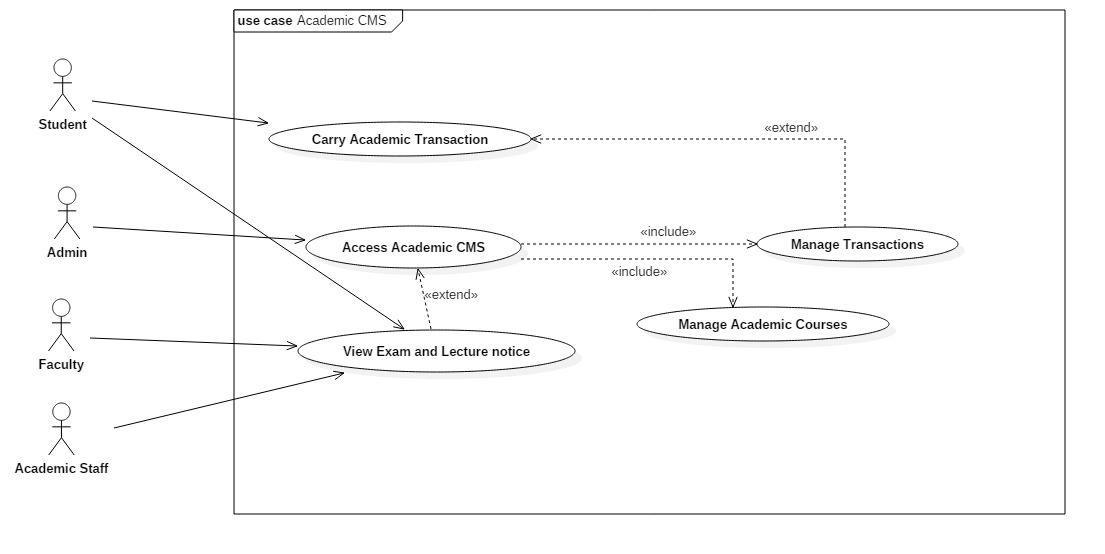
Img 1: User Use-case diagram



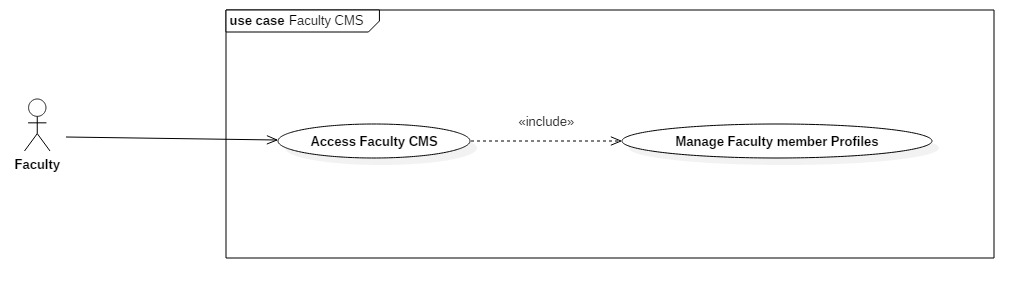
Img 2: Organization CMS Use-case diagram



Img 3: Student CMS Use-case diagram



Img 4: Academic CMS Use-case diagram



Img 5: Faculty CMS Use-case diagram

Use-case Justification

|  |  |
| --- | --- |
| ID | UC-01 (User CMS use-case) |
| Title | User Login |
| Justification | Mentioned in FR-02 rational. |
| Primary actor(s) | Admin, Academic Staff, Student, Faculty |
| Supporting actor(s) | N/A |
| Primary Flow | 1. User clicks sign in button. 2. User logs into the system with his/her credentials. 3. Credentials are validated and verified by the system 4. If valid, user is presented with his/her profile. |
| Alternative Flow | * 1. User inputs invalid credentials in the form.   2. User is presented with same form with error message.   3. Repeat step 2 until valid credentials are provided. |
|  | |
| ID | UC-02 (User CMS user-case) |
| Title | User Registration |
| Justification | Mentioned in FR-01 rational. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Only admin is authorized to add new user. 2. Admin navigates towards ‘Add User’ form. 3. Admin adds new user’s information. 4. System validates the provided data and verifies for redundancy. 5. New user is added if validation passes. |
| Alternative Flow | 4.1 Admin inputs invalid information.  4.2 Admin is presented with same form with validation message.  4.3 Repeat step 3. |
|  | |
| ID | UC-03 (User CMS user-case) |
| Title | View User Detail |
| Justification | The available users’ information is provided with the help of this use-case. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Admin clicks ‘View Users’ button. 2. Admin is provided with user list in tabular form. |
| Alternative Flow | N/A |
|  | |
| ID | UC-04 (User CMS user-case) |
| Title | Edit User Detail |
| Justification | Required for updating users’ credentials whenever required. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. First, Admin interact with use-case UC-03. 2. Admin clicks ‘Edit’ button for particular user from the table. 3. Admin is provided with the ‘Edit User’ form to edit the user. 4. Admin updates the users’ information. 5. System validates and verifies the updated information. 6. Admin is provided with the users’ list table after successful edit. |
| Alternative Flow | * 1. If invalid information is provided, admin is provided with ‘Edit User’ form with error message.   2. Repeat step 4. |
|  | |
| ID | UC-05 (User CMS user-case) |
| Title | Delete User |
| Justification | Use-case required to remove irrelevant user. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Admin interact with use-case UC-03. 2. Admin clicks ‘Delete’ button for particular user from the table. 3. Admin is provided with the confirmation modal to delete the user. 4. Admin clicks ‘Delete’ button to remove the user. |
| Alternative Flow | * 1. Admin dismisses the Delete modal.   2. Admin is provided with the same list of users. |
|  | |
| ID | UC-06 (Organization CMS use-case) |
| Title | Organization Detail Entry |
| Justification | Mentioned in FR-03 rational. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Admin navigates towards ‘Add Organization Detail’ button. 2. Admin is provided with ‘’Add Organization Detail’ form. 3. Admin inputs the information and submits the form. |
| Alternative Flow | N/A |
|  | |
| ID | UC-07 (Organization CMS use-case) |
| Title | View Organization Detail |
| Justification | Mentioned in FR-05 rational. |
| Primary actor(s) | Admin, Academic Staff, Student, Faculty |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Actor navigates towards ‘View Organization Detail’ form. 2. Actor is provided with organization details. |
| Alternative Flow | N/A |
|  | |
| ID | UC-08 (Organization CMS use-case) |
| Title | Update Organization Detail |
| Justification | Mentioned in FR-04 rational. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Admin interact with use-case UC-07 2. Admin clicks ‘Update’ button. 3. Admin is provided with ‘Edit Organization Detail’ form. 4. Admin updates the information and submits the form. |
| Alternative Flow | N/A |
|  | |
| ID | UC-09 (Organization CMS use-case) |
| Title | Delete Organization Detail |
| Justification | Mentioned in FR-06 rational. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Admin interact with use-case UC-07 2. Admin clicks ‘Delete’ button. 3. Admin is provided with confirmation modal. 4. Admin clicks ‘Delete’ button to remove the information. |
| Alternative Flow | * 1. Admin dismisses the modal.   2. Organization detail is displayed after modal dismisses. |
|  | |
| ID | UC-10 (Student CMS use-case) |
| Title | Student CMS |
| Justification | Mentioned in FR-07 rational. |
| Primary actor(s) | Academic Staff, Student |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Actor(s) clicks ‘User Profile’ button. 2. If actor is Academic Staff, student list is provided. 3. Staff can access each students’ profile. |
| Alternative Flow | * 1. If actor is student, they can access there profile only. |
|  | |
| ID | UC-11 (Student CMS use-case) |
| Title | Student Report Generation |
| Justification | Mentioned in FR-14 rational. |
| Primary actor(s) | Student |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Student clicks on ‘Generate Report Button’ under their profile. 2. Report is generated and provided by the system. |
| Alternative Flow | N/A |
|  | |
| ID | UC-12 (Academic CMS use-case) |
| Title | Academic Transaction |
| Justification | Essential for student’s financial transaction in the institution. |
| Primary actor(s) | Student |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Student carries the transaction. 2. Transactions are recorded in the system. |
| Alternative Flow | N/A |
|  | |
| ID | UC-13 (Academic CMS use-case) |
| Title | Academic CMS |
| Justification | Mentioned in FR-08 rational. |
| Primary actor(s) | Admin |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Admin navigates towards Academic CMS. 2. Admin manage courses. |
| Alternative Flow | * 1. Admin manage transactions. |
|  | |
| ID | UC-14 (Academic CMS use-case) |
| Title | Exam and Lecture notice |
| Justification | Mentioned in FR-15 rational. |
| Primary actor(s) | Academic Staff, Student, Faculty |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Academic Staff adds notice. 2. Student or Faculty members are notified automatically. |
| Alternative Flow | N/A |
|  | |
| ID | UC-15 (Faculty CMS use-case) |
| Title | Faculty CMS |
| Justification | Required to manage the faculty members information. |
| Primary actor(s) | Faculty |
| Supporting actor(s) | N/A |
| Primary Flow | 1. Faculty members access their own profile. 2. Faculty members can manipulate allowed fields. |
| Alternative Flow | N/A |

2.6 Initial Class Diagram (NLA)

Natural Language Analysis (NLA) is a process to identify list of candidate classes by separating out lists of verbs, adjectives, and nouns from the descriptive paragraphs. The possible classes, its attributes and methods are identified from the problem domain where some specific classes will be our ultimate classes of the program.

Problem Statement/Domain

Automated Academic Organization System is an automated information system for academic institutions. The system will be accessed by various users with different permissions levels. In general, there are four category of users: Administrator, Academic Staff, Faculty members and Students. Admin manage major parts of the system whereas remaining users carry operations on specific level. The system will be featured with user login and registration system along with user authentication. It will also provide a way to manage the registered users in the system. The admin will be able to register, view, edit and delete user details, organization details, academic details. Apart from that, academic staffs will be able to add, view, update and delete exam and lecture information. They will also manage student profiles. Similarly, faculty members will be able to manage their own profiles, view exam notices, lecture notices, view organization details, view student performance. Moreover, student will be able to view their own profile, generate academic reports, view organization details and notices. Every user should be authenticated as per user types at first to access filtered part of the system.

Natural Language Analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **List of nouns give potential candidate classes and attributes** | | | |
| Academic | Administrator | Authentication | Category |
| Detail | Exam | Faculty | Information |
| Institution | Lecture | Level | Login |
| Member | Notice | Operation | Organization |
| Part | Performance | Permission | Profile |
| Registration | Report | Staff | Student |
| System | User | User Type |

|  |  |  |  |
| --- | --- | --- | --- |
| **List of verbs give potential methods** | | | |
| Access | Add | Authenticate | Automate |
| Carry | Delete | Edit | Filter |
| Generate | Manage | Provide | Register |
| View |

The above lists are filtered to reduce number of classes, attributes and methods.

**Remove synonymous words**

|  |  |  |
| --- | --- | --- |
| **Word Type** | **Words** | **Synonyms** |
| Noun | User Type | Administrator, Staff |
| Information | Detail, Notice |
| Organization | Institution |
| User | Member |
| Verb | Authenticate | Access |
| Add | Register |

**Remove words representing high level of abstraction**

|  |  |
| --- | --- |
| **Word Type** | **Words** |
| Noun | Authentication, Information, Permission, Profile, Registration, Login, System, Performance, Report |
| Verb | Automate, Authenticate, Manage, Filter |

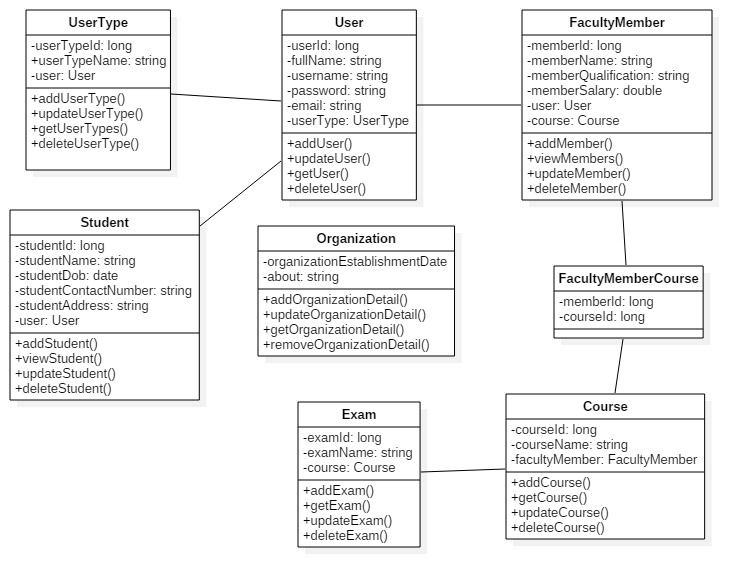
**Remove ambiguous words and those that are out of scope**

|  |  |
| --- | --- |
| **Word Type** | **Words** |
| Noun | Academic, Category, Level, Operation, Part |
| Verb | Carry, Provide |

The final lists of classes and its methods are mentioned below:

|  |  |
| --- | --- |
| **Classes** | **Methods** |
| User  User Type  Organization  Student  Faculty  Lecture/Course  Exam | Add User / Organization Detail / Student / Faculty / Lecture / Exam / Notice  Generate Student Report  Edit User / Organization Detail / Student / Faculty / Lecture / Exam / Notice  View Users / Organization Details / Students / Faculty / Lecture / Exam / Notice |

The initial class diagram generated using NLA is as shown below:



Img : Initial Class Diagram