Chapter 1: Introduction

Project Introduction

**Automated Academic Organization System (AAOS)** is the automated information system featured collectively with content management system, automated data-driven systems and data-analysis system. This information system is useful for every academic institutions as it provides centralized information of all the stakeholders and helps in decision making from appropriate analysis of institutions’ data.

Project Background and Justification

This software provides digital features for all the stakeholders including the business stakeholders, academic staffs, pupils in the educational institutions. This information system will appropriately centralize the voluminous data of the educational institution by using the digital technology instead of traditional method.

Project Overview

This complete system is designed to address the traditional problems in an educational institution. The three major parts of the system are content management system, automated information system and data-analysis system.

Aims and Objectives of the project

* A centralized and robust information system that can manage the educational sectors and provide convenient way to perform transactions.
* Computerize the transactions in an educational institution.
* Visually represent the information of different sectors in an organization.
* Automate trivial activities related to a particular transaction or activity.

Chapter 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. These modelling methodologies of OOA are specifically applied in **Design phase** of this project.

* 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.

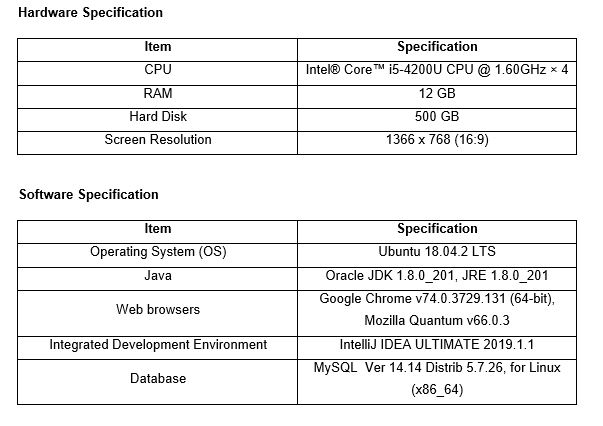
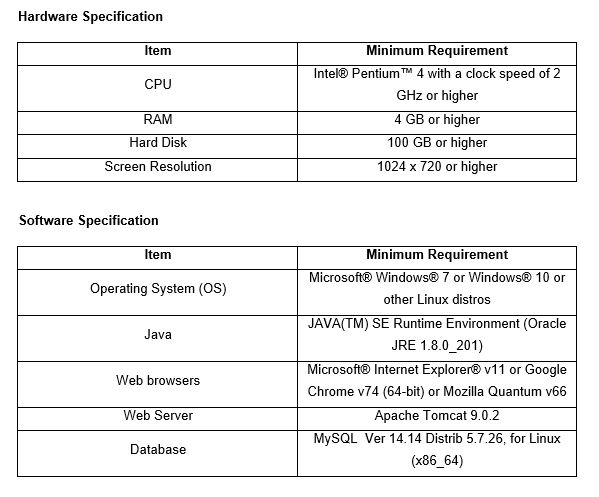


Image- 1: Development Environment Specification

Product Configuration/Deployment Specification

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

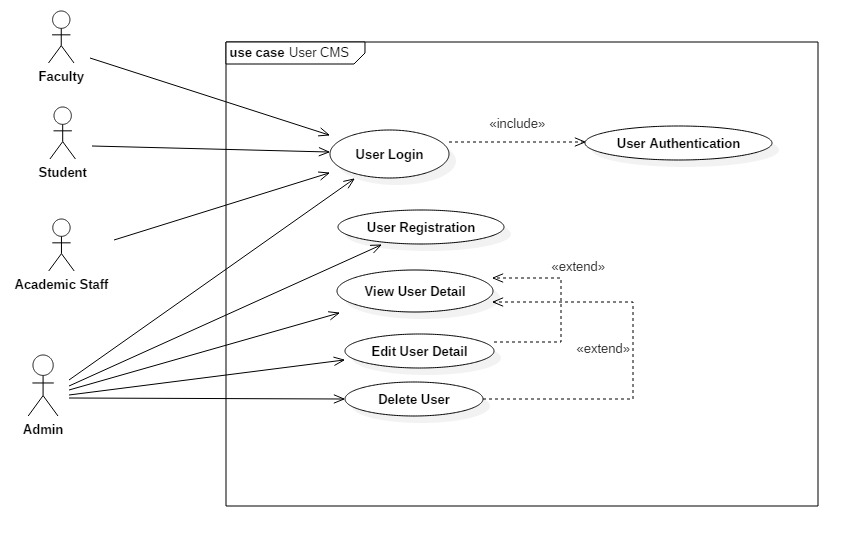
****

Img 2:Product Configuration Specification

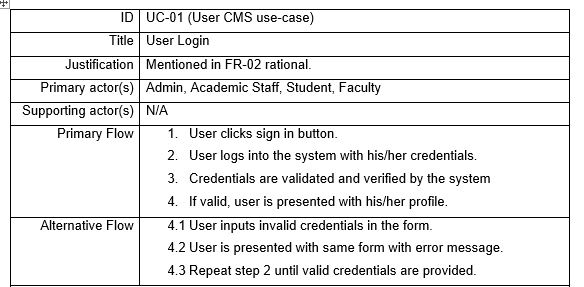
2.5 Use-case diagram

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.

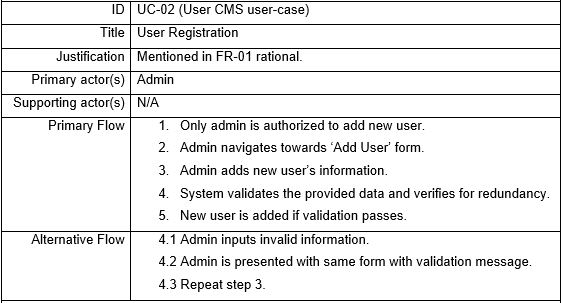
The diagrams below show the use-case for Automated Academic Organization System where four different users are presented with different access in the system.



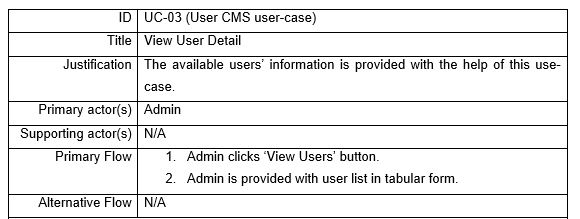
Img 3: User Use-case diagram



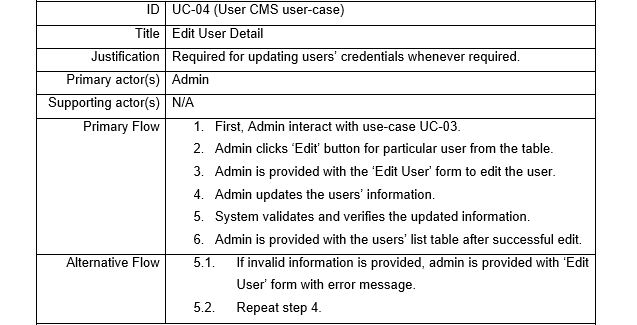
Img 4: UC-01



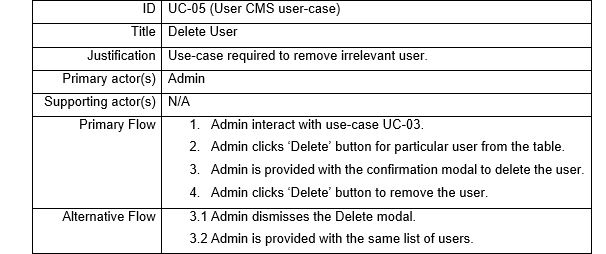
Img 5: UC-02



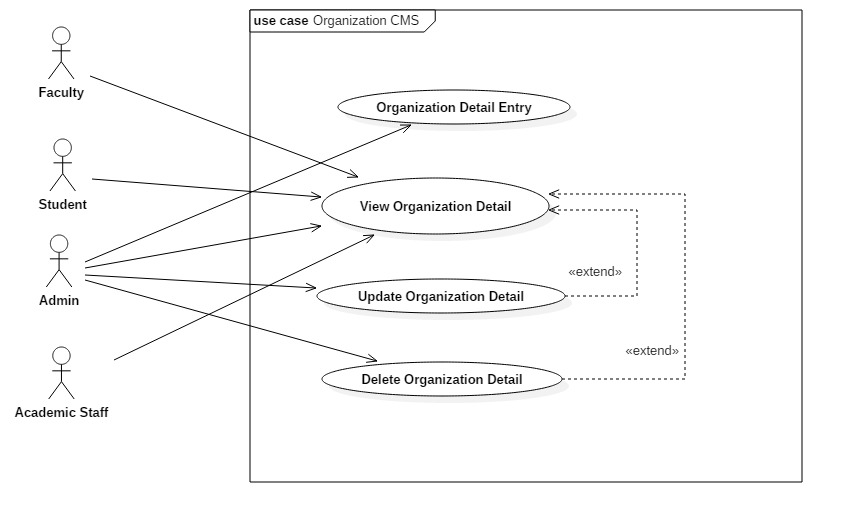
Img 6: UC-03



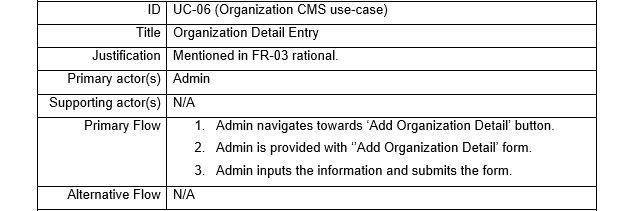
Img 7: UC-04



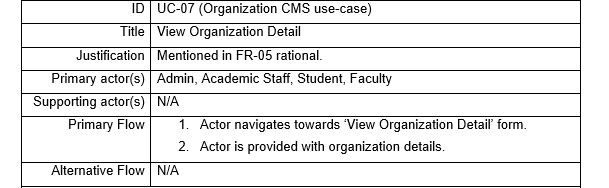
Img 8: UC-05



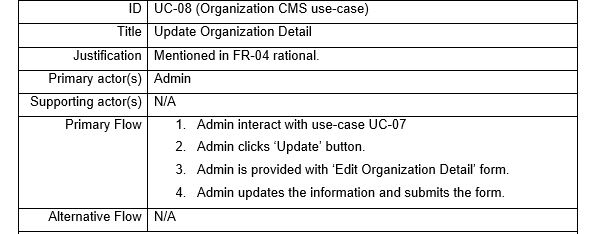
Img 9: Organization CMS Use-case diagram



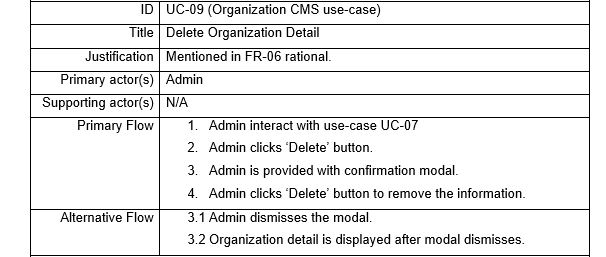
Img 10: UC-06



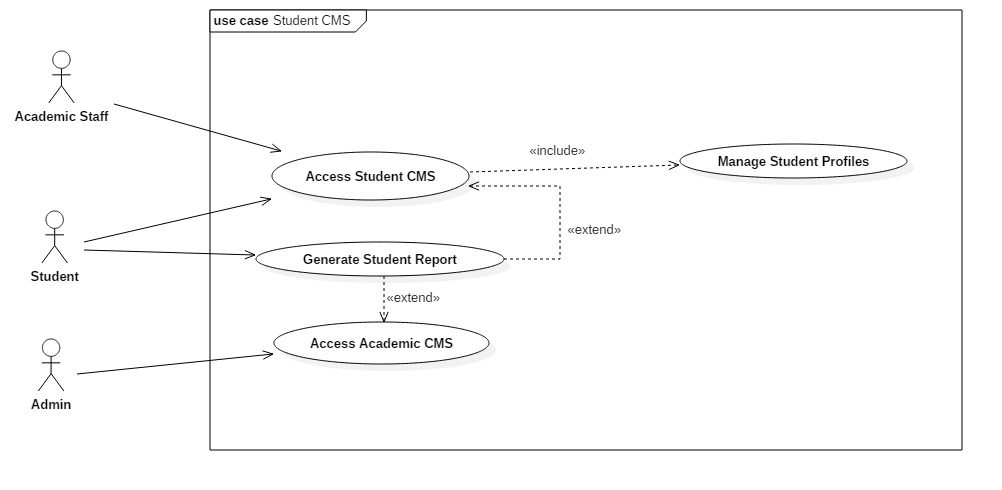
Img 11: UC-07



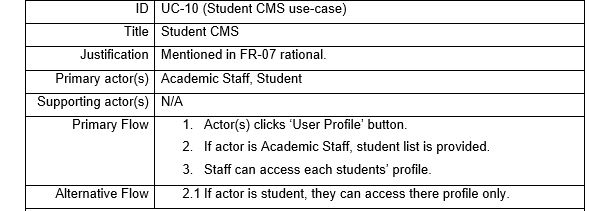
Img 12: UC-08



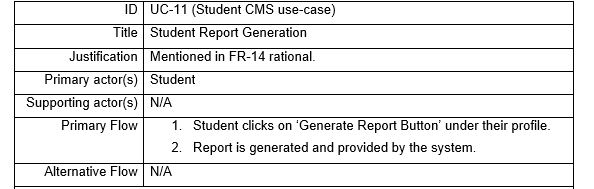
Img 13: UC-09



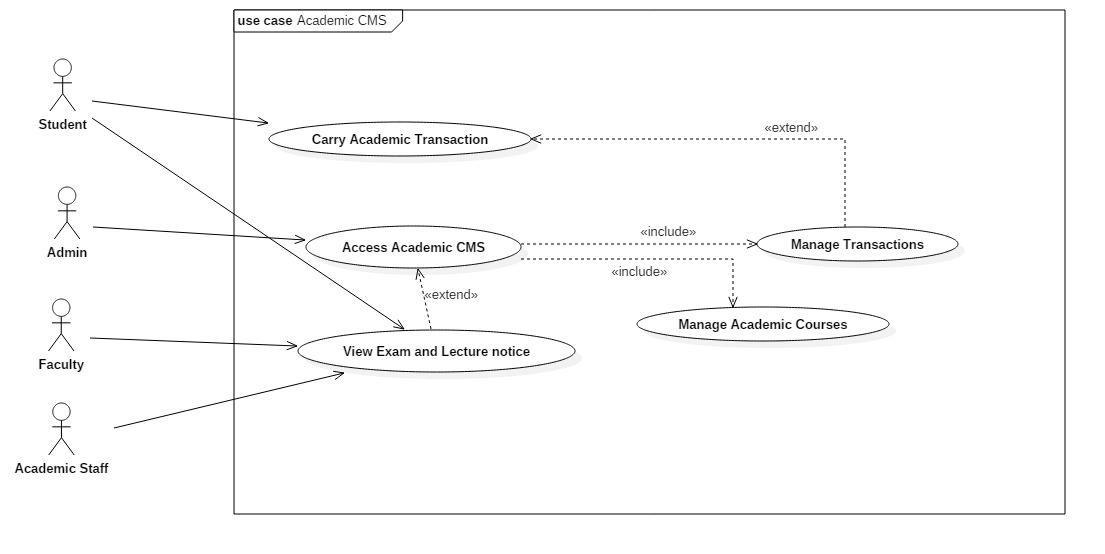
Img 14: Student CMS Use-case diagram

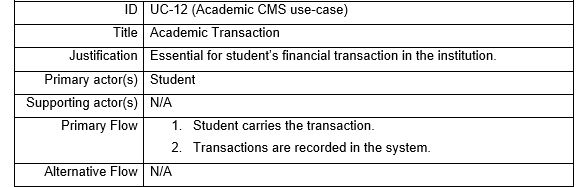


Img 15: UC-10

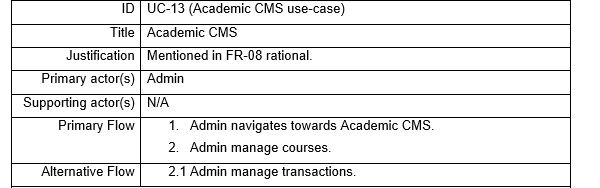


Img 16: UC-11

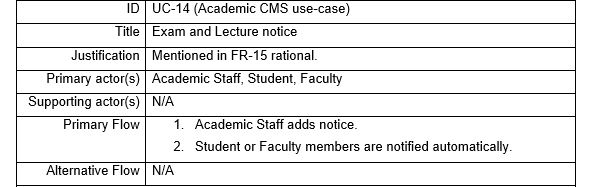


Img 17: Academic CMS Use-case diagram

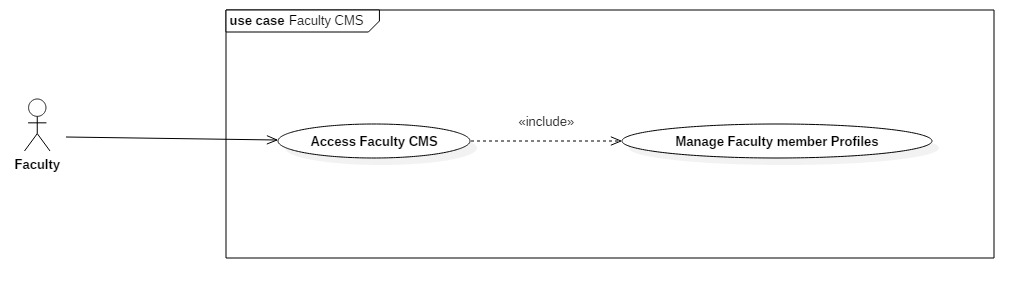
Img 18: UC-12



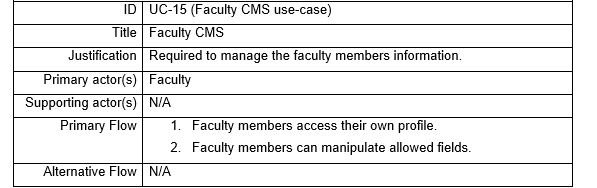
Img 19: UC-13



Img 20:UC-14



Img 21: Faculty CMS Use-case diagram



Img 22:UC-15

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 |

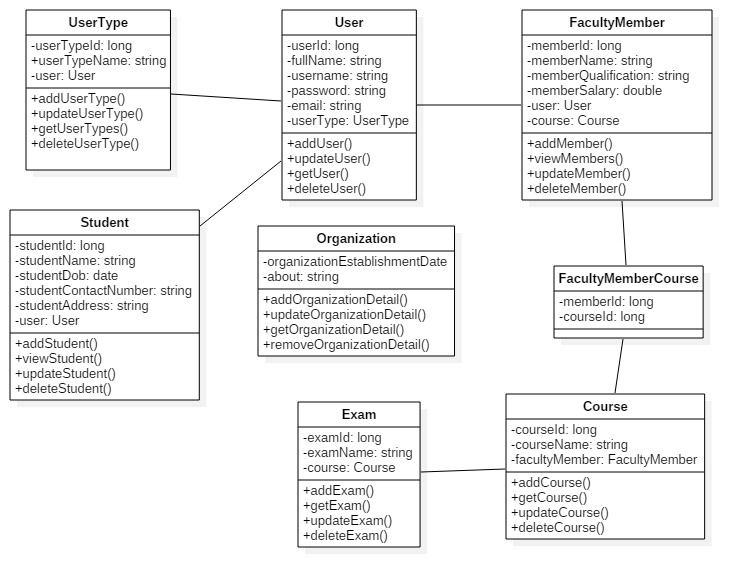
**Remove ambiguous words and those that are out of scope**

|  |  |
| --- | --- |
| **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 23: Initial Class Diagram

Chapter 3: Design

About Design

Design is a work process which has a user perspective and drives development based on your specific customers’ needs. (SVID, 2005)

Project Design Plan

The design phase has been planned by implementing various design methodologies. With the making of different design models, the design phase will be completed. The implemented design models are mentioned below as:

* Structural Modelling
* Behavioral Modelling
* Database Modelling
* Architectural Modelling
* User Interface Modelling

Structural Modelling

This modelling methodology represents the system in various components along with their communication. Class Diagram and Data Flow Diagram has been applied in order to produce the structural model of the system.

Final Class Diagram

Definition

It is a structure diagram that falls in the category of static diagram or static modelling where the system’s structure is represented by the classes, their attributes, operations and relationship among different class or objects.

Justification

The reasons for approaching this modelling methodology are mentioned below:

* It clearly states the major components of the system which may be classes or interfaces.
* The type of entity can be distinguished easily with the study of attributes of the class.
* The relationship among different classes is represented clearly with the help of different notations.
* Class Diagram also lists out possible operations of the entity or class in a system.

Notations

Actual Diagram

Diagram Description

Data Flow Diagram

Definition

Data Flow Diagram is also a diagram that falls in structural modelling. It shows the flow of a data from a process to another process or from a system to another system.

Justification

The reasons for approaching this modelling methodology are mentioned below:

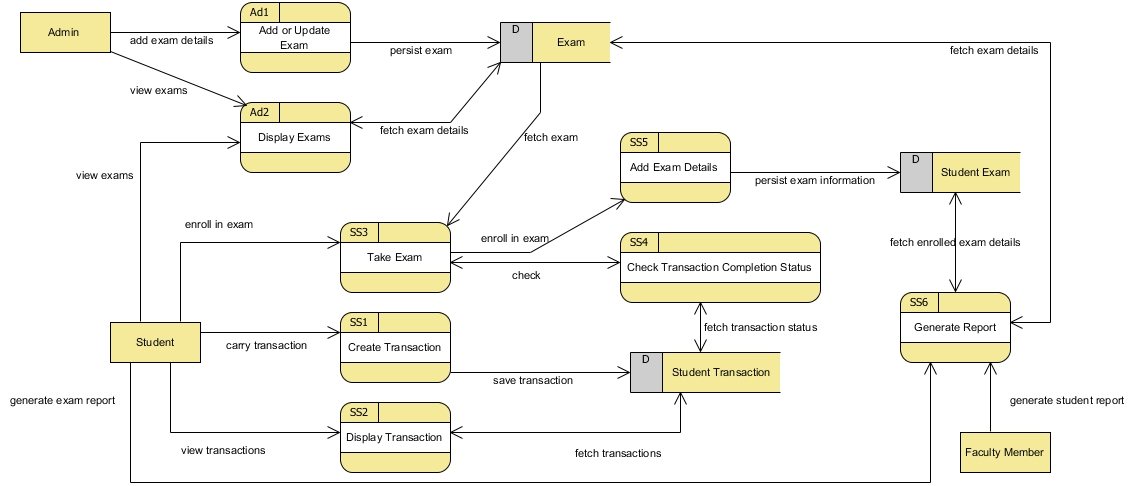
* It helps to identify the processes involved to transfer data.
* DFD provides logical information flow of the system.
* Its diagramming notations are simple and easily understandable.

Notations

The notations used for this methodology are tabulated below:

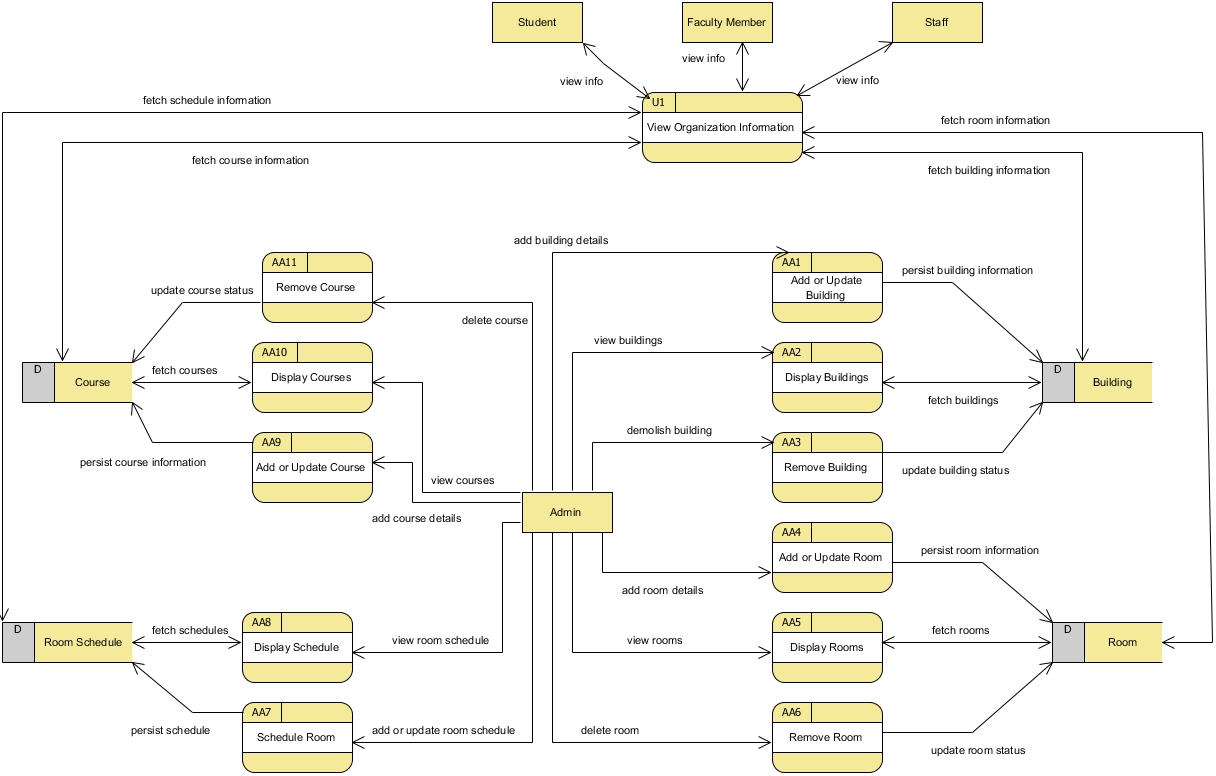
|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| External Entity |  | Represents the entity that provides data or receive output to or from the system |
| Process | DFD Process | Represents a place where input data is processed to produce output |
| Data Flow | DFD Data Store Example | Arrow represents the path for flow of data in the system |
| Data Store |  | Represents a place to persist the data. |

Actual Diagram and Description



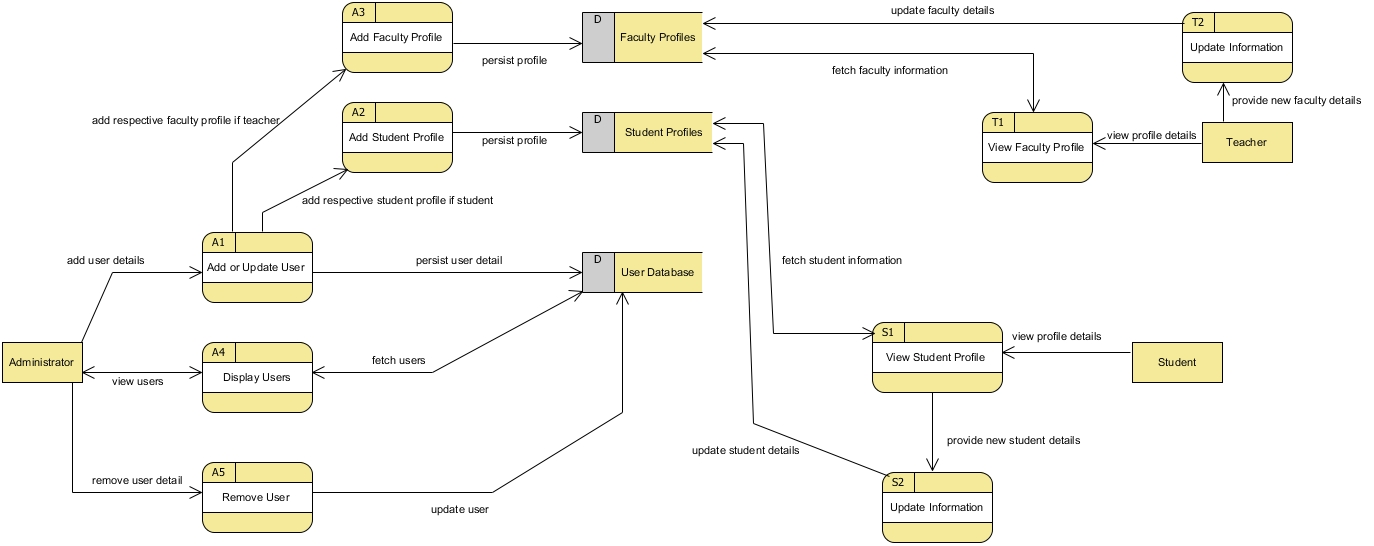
Img 1-Academic Data Flow Diagram

The diagram above shows the flow of data in the system while carrying out different academic operations. The external entities are Admin, Student, Faculty Member who carries out different processes affecting different data store such as Exam, Student Transaction, Student Exam.



Img 2-Organization Data Flow Diagram

The data flow diagram above shows general processes of the system while managing fundamental information of the organization. In the diagram, Admin as an external entity triggers multiple processes affecting multiple data stores. Similarly, external entities like Student, Faculty Member, Staff carry out same single process that retrieves data from multiple data stores.



Img 3-User Profile Data Flow Diagram

The data flow diagram represents the flow of data in the system as per various access level in the system. As per access level of the external entity, the processes are carried out distinctively. The data flow diagram generally represents multiples processes that ultimately manages the user profiles in the system.

Behavioral Modelling

This modelling methodology describes the internal actions of the system with the help of models that represent the business process and change in data.

Activity Diagram

Definition

It is a diagram that falls in dynamic system modelling. Activity diagram shows the flow of program or flow of control in the system.

Justification

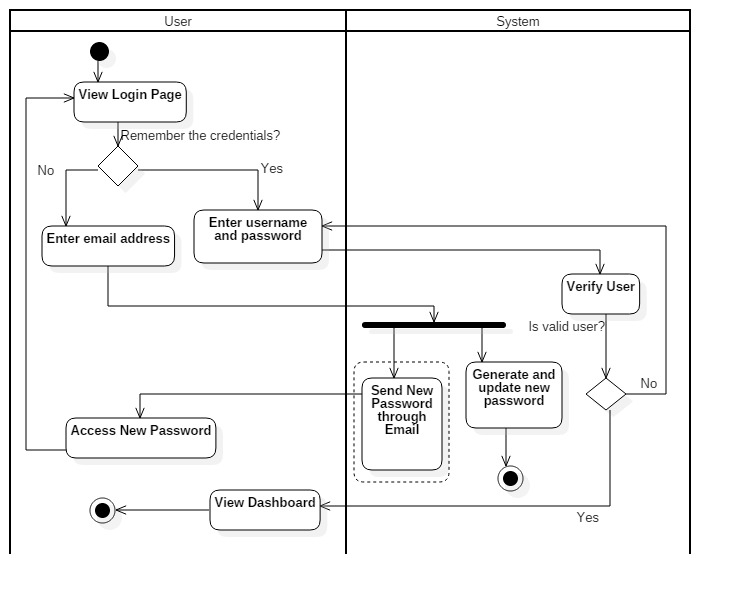
The reasons for approaching this modelling methodology are mentioned below:

* The business processes are clearly represented as each task is shown as a single activity within a diagram.
* It represents the dynamic view of the flow of the system.
* Activity Diagram shows the program flow from various task perspective.

Notations used

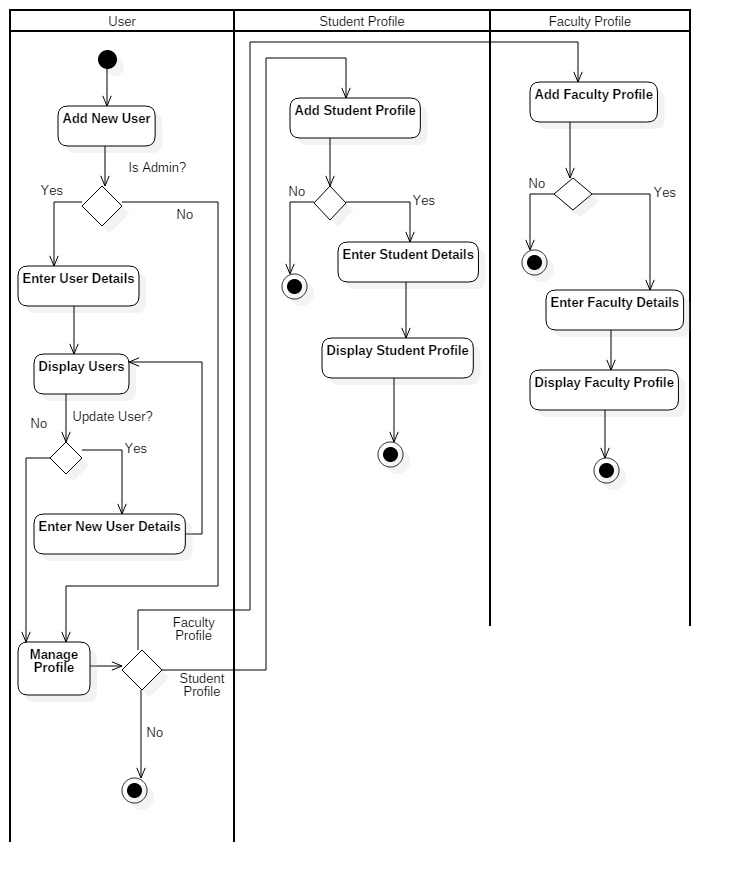
|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Action | Activity Diagram Notation - Action | Represents the action in the system |
| Control Flow | Activity Diagram Notation - Control Flow | Represents the sequence of execution |
| Initial Node | Activity Diagram Notation - Initial Node | Represents the beginning of action. |
| Activity Final Node | Activity Diagram Notation - Activity Final Node | Represents point where all control flows stops. |
| Decision Node | Activity Diagram Notation - Decision Node | Represents a condition so that control flow goes single way. |
| Fork Node | Activity Diagram Notation - Fork Node | Represents control flow split into parallel flows. |
| Join Node | Activity Diagram Notation - Join Node | Represents parallel flows brought together. |
| Swimlane | Activity Diagram Notation - Swimlane and Partition | Represents grouping of activities performed by same actor. |
| Interruptible Activity Region |  | Represents activity region where the activity flow may interrupt |
| Send Signal |  | Represents a component from where the message is send |
| Accept Signal |  | Represents a component where the message is received |

Actual Diagram and Description



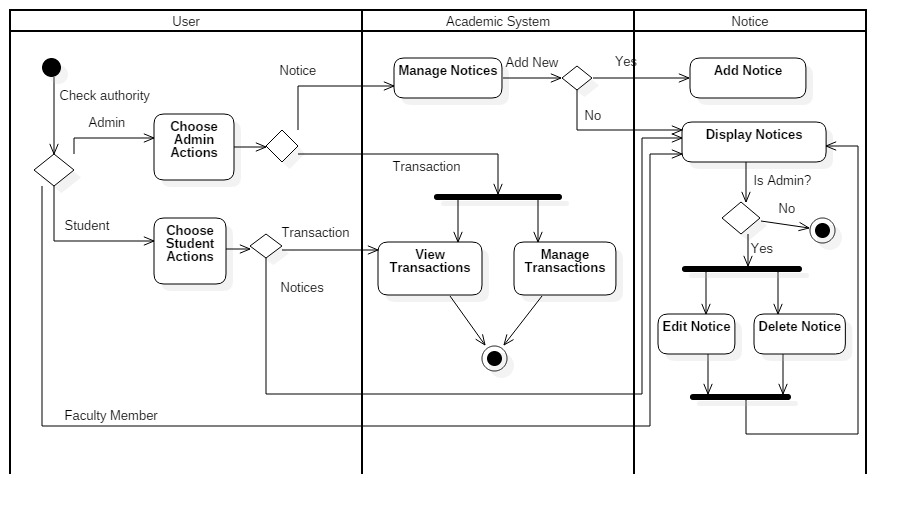
Img 4-User Login Activity Diagram

The activity diagram above models behavior of the system while a user logs into the system. In case of resetting password, a password is reset through email which is an interruptible activity. The activity performed by the user are all composed in swim lane named User. Similarly, the system’s activity is composed in swim lane named System.



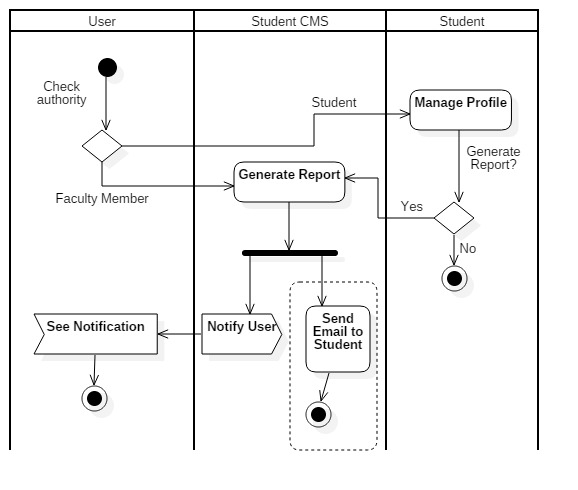
Img 5-User Profile Management Activity Diagram

The activity diagram shows the behavioral model when the user is added. Furthermore, in case of student or faculty member, the activity of the system extends in order to manage the profile which are represented by the decision node.



Img 6-Academic Operation Activity Diagram

The activity diagram above shows three different swim lanes for User, Academic System and Notice. The User swim lane is composed with multiple decision node which allows admin or student or faculty member manage transactions and notices. Similarly, the Academic System swim lane is composed with fork node to run parallel activities while manipulating the transactions. Furthermore, the Notice swim lane comprises of decision node along with both fork node and join node to manage notices.



Img 7-Student CMS Activity Diagram

The diagram above shows the behavioral model when the operations are carried out in student’s system. The swim lane named Student CMS comprises of interruptible region while sending an email to the user. Moreover, the **Notify User** send signal represents that the notification is sent to the user which is received by the user as a **See Notification** accept signal.

Sequence Diagram

Definition

Sequence Diagram shows the interaction of the objects within the program flow which helps to identify the message sequence.

Justification

The reasons for approaching this modelling methodology are mentioned below:

* Besides dynamic view of the system like shown by Activity diagram, it also keeps time sequence as an essential part of the diagram.
* It clearly shows the lifetime of the object in the system during task execution.
* It provides detail visual representation of how objects communicate with each other within the program flow.

Notations

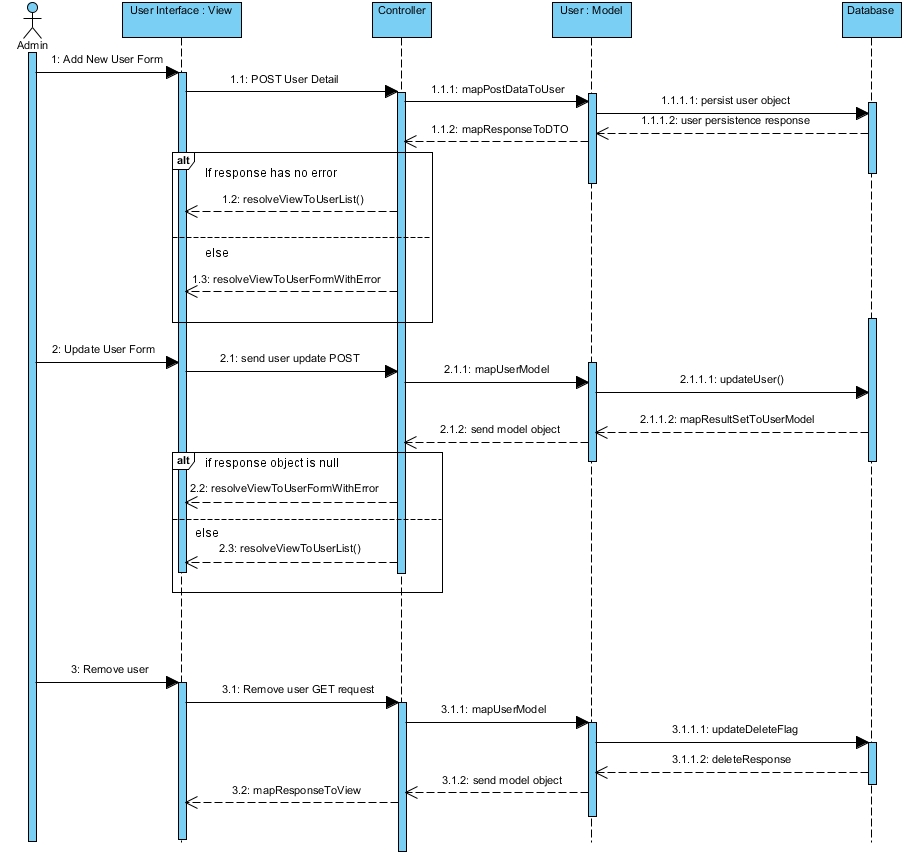
|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Actor | Actor | Represents the entity’s role |
| Lifeline | Lifeline | Represents an individual participant. |
| Activations | Activation | Represents an operation period of an element. |
| Call Message | Call Message | Represents the invocation message that invocate communication between lifelines. |
| Return Message | Return Message | Represents the reply message to the caller. |
| Recursive Message | Recursive Message | Represents the message invocated in same lifeline. |
| Sequence Fragment | Fragment | Loop fragment represents that the fragment may execute multiple times. Alt fragment represents only single execution whenever the condition matches. Par fragment represents parallel execution. |

Actual Diagram and Description



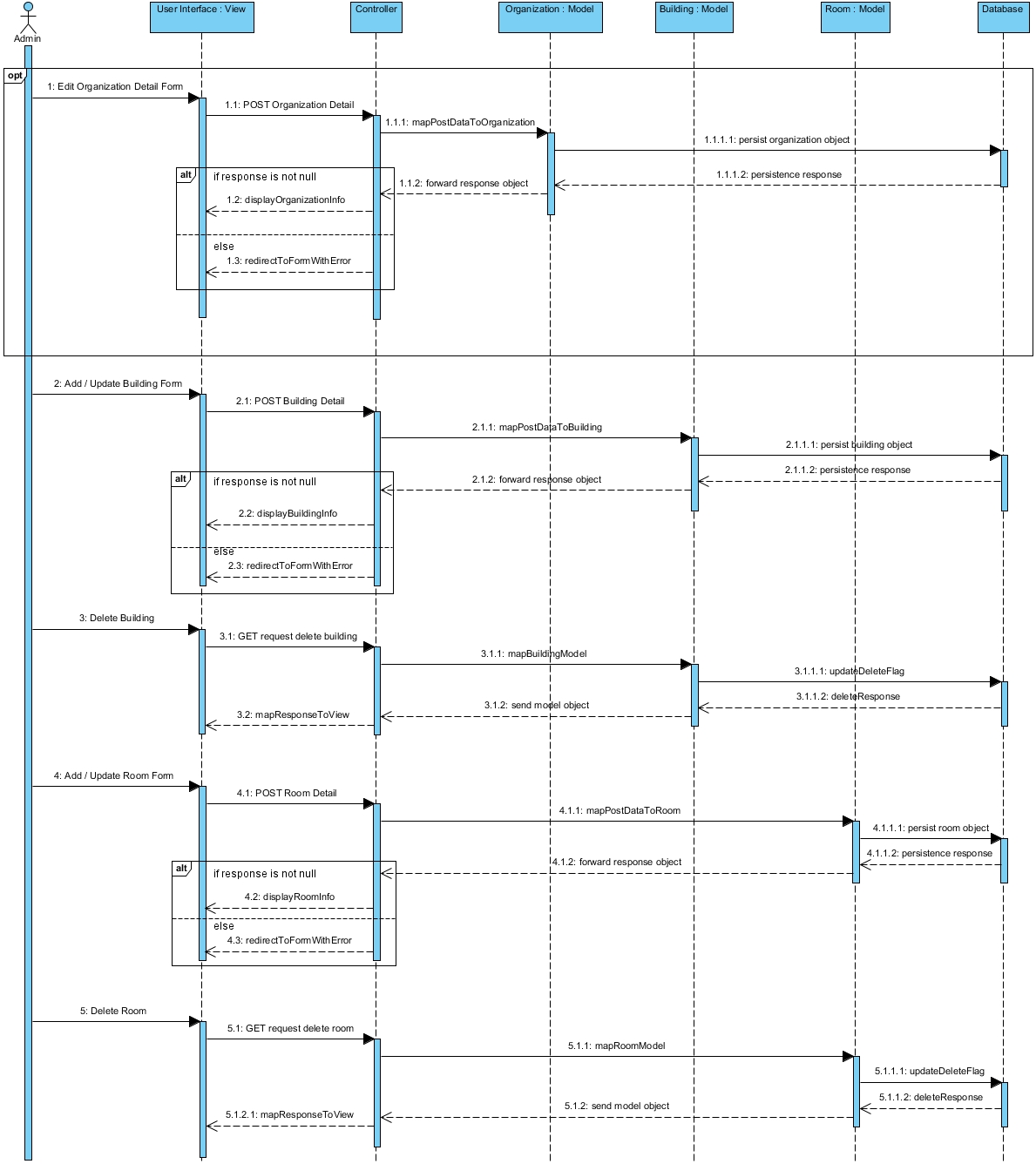
Img 8-Login Sequence Diagram

The sequence diagram above represents a single actor **User** who initiate the communication between different objects within five different lifelines. The optional sequence fragments are for user login and reset password in case needed. In addition, the alternate sequence fragment in case of user login represents that user is redirect into different view as shown in the figure. Similarly, in case of reset password, the view is alternate as per the response.



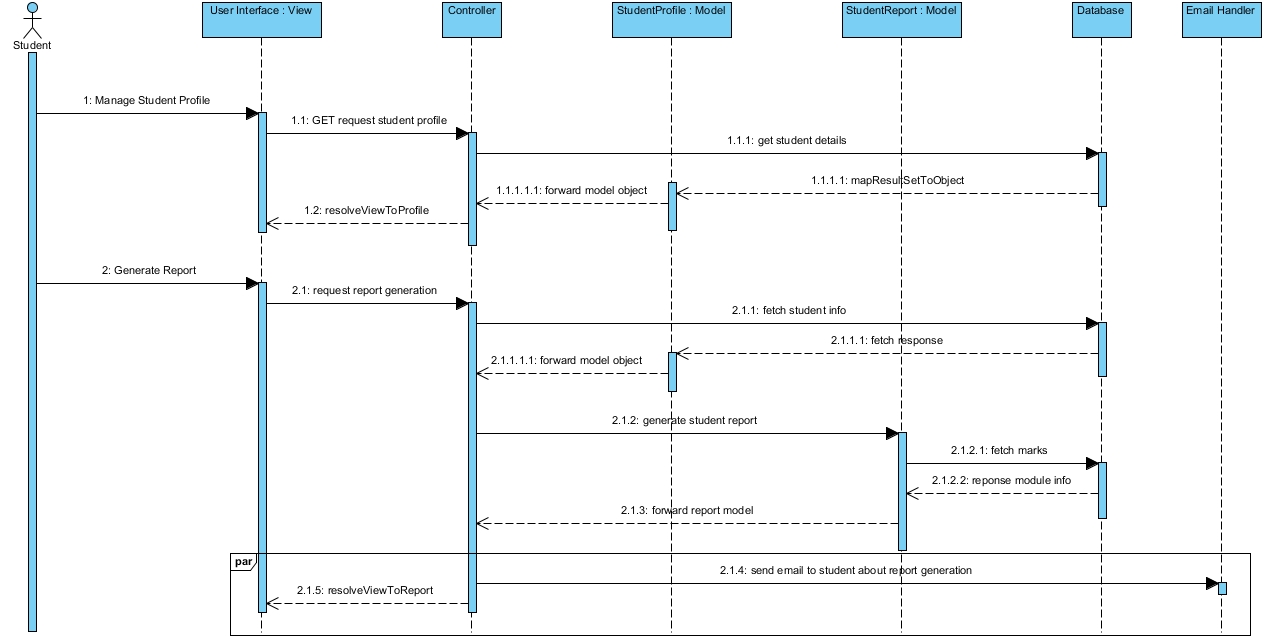
Img 9-User Sequence Diagram

The sequence diagram above shows the communication between different lifelines or objects when the user information is manipulated. In case of adding or updating the user, the alternate sequence fragment represents that the view will be manipulated as per the response object status. Besides, other messages are transmitted in the system as shown in the diagram.



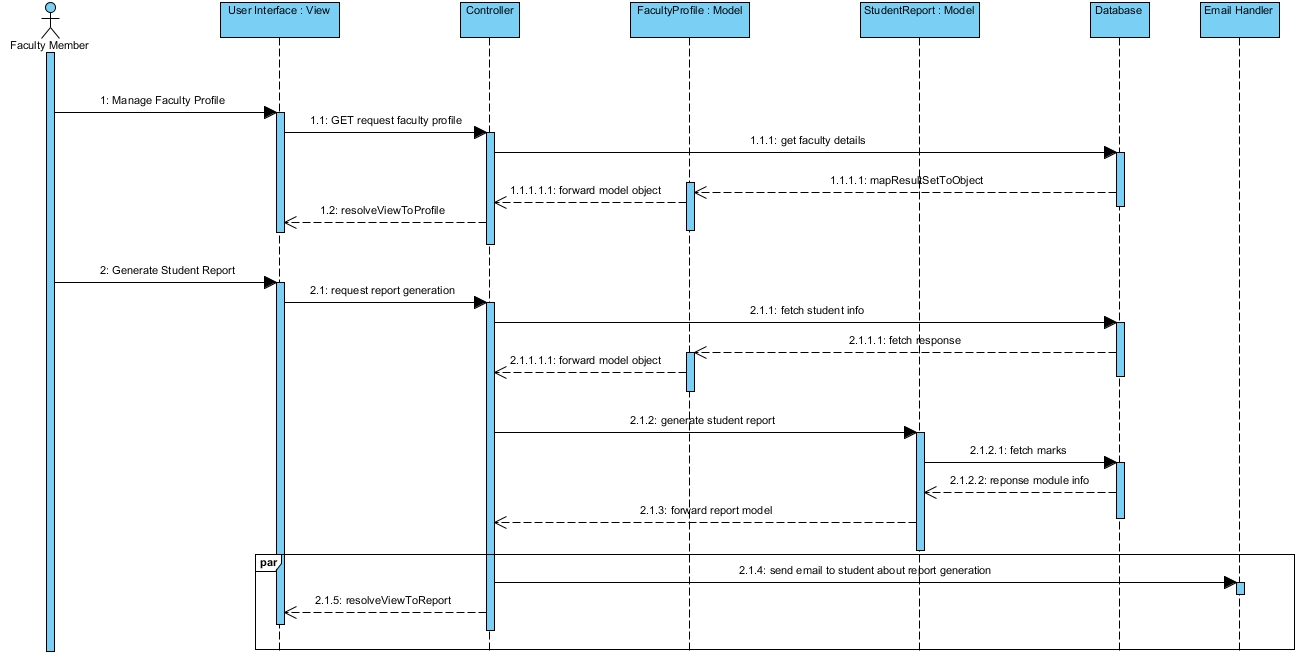
Img 10-Organization Sequence Diagram

The sequence diagram above shows the interaction of the objects while managing the information of the organization. In case of organization information, the whole interaction is optional and is represented by the optional sequence fragment. However, in case of building or room, the flow of message is normal and the flow can alternate as per the response from different lifelines represented by alternate sequence fragment.



Img 11-Student Sequence Diagram

The sequence diagram shows various lifelines and their interaction through lifeline activations. The flow of message is as shown in the diagram. In case of report generation of the student, two parallel activities that are to present the report in the system’s view and send an email is presented by the parallel sequence fragment.



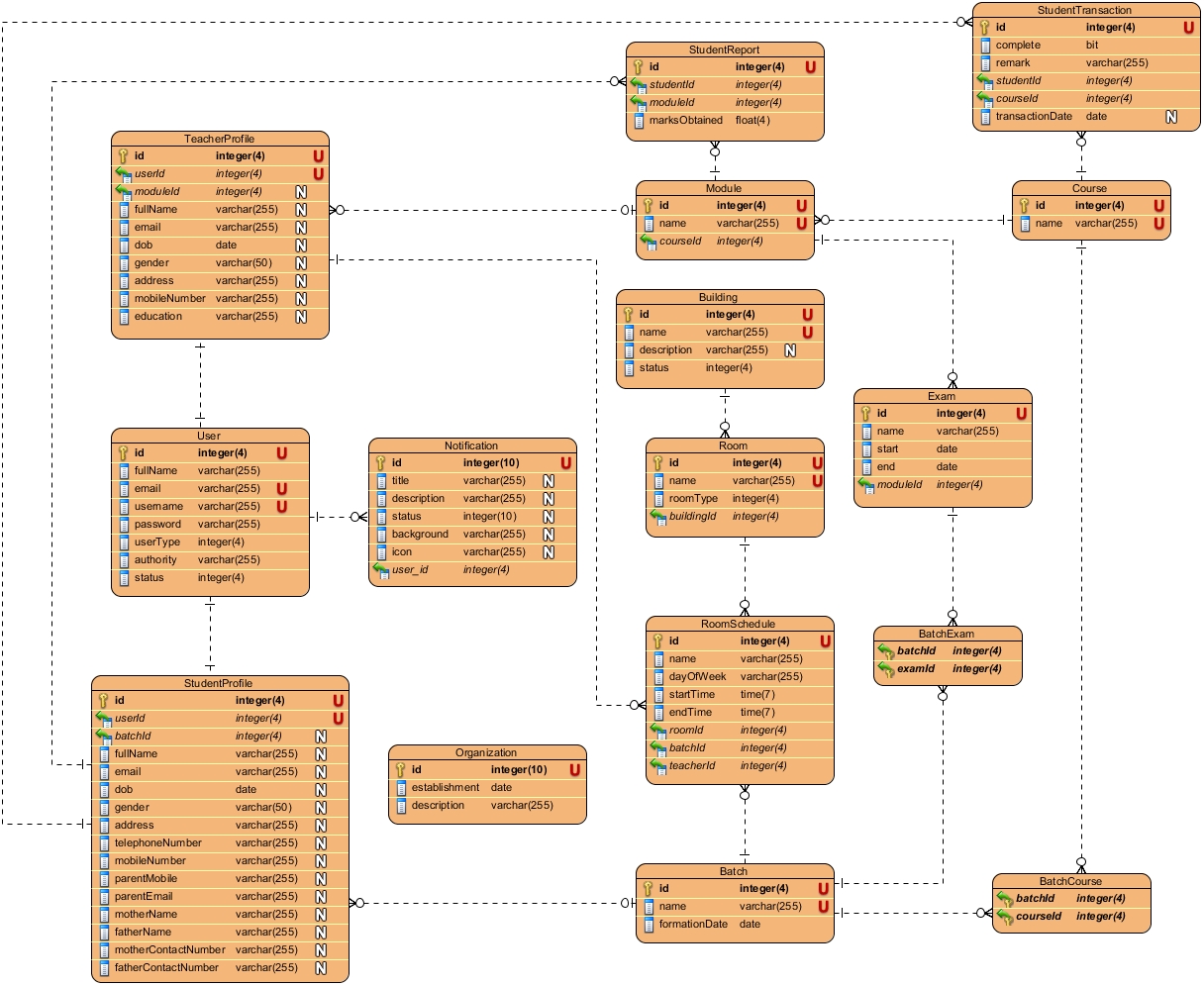
Img 12-Faculty Sequence Diagram

The faculty sequence diagram is similar to the student sequence diagram except the lifelines and messages. The flow of message while managing the profile is identical to the flow of message in student sequence. And, the parallel sequence fragment represents two activities that are report generation by the faculty member and sending email to the student about report generation.

Database Modelling

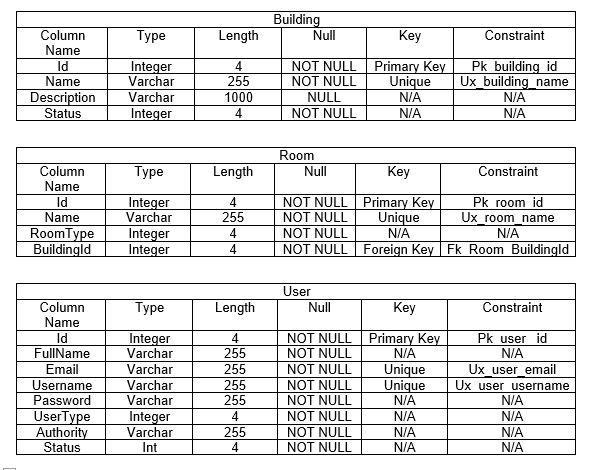
This modelling methodology presents the proposed logical design of the database along with the data dictionary. The logical database design is obtainable through **Entity Relationship Diagram (ER Diagram)**.

ER Diagram

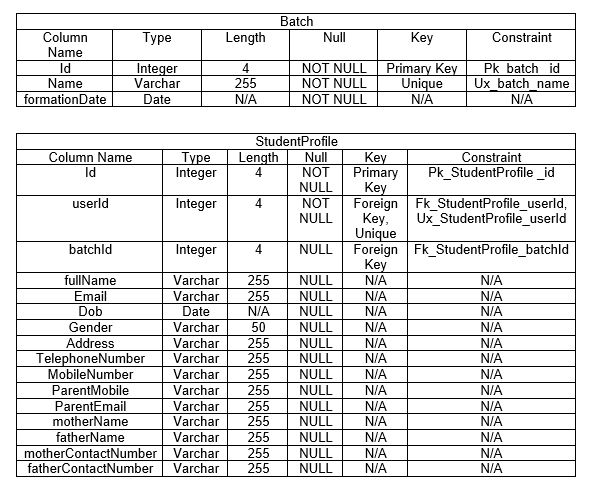


Img 13-ER Diagram

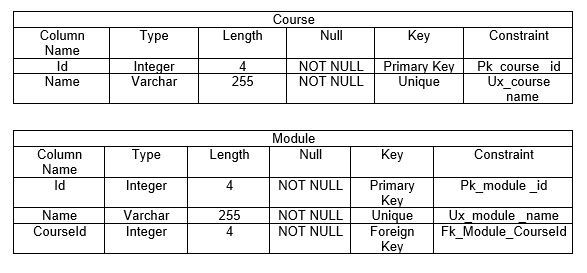
Data Dictionary



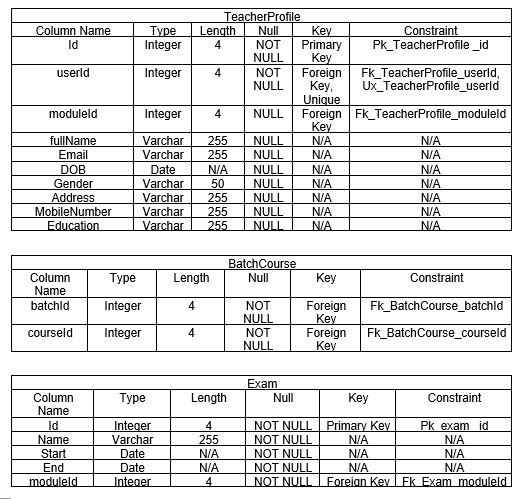
Img 14-Data Dictionary of Building, Room, User tables



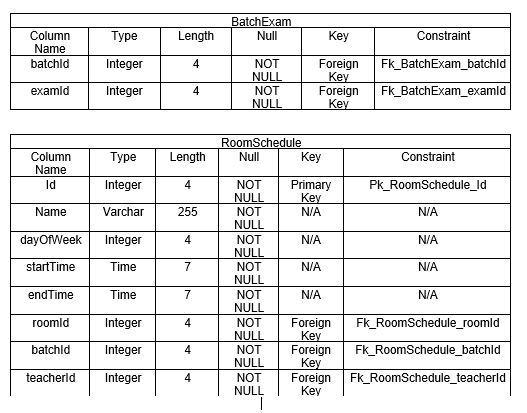
Img 15-Data Dictionary of Batch, Student Profile table



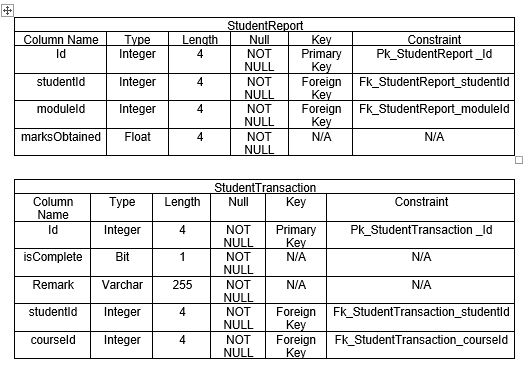
Img 16-Data Dictionary of Course, Module table



Img 17-Data Dictionary of TeacherProfile, BatchCourse, Exam tables



Img 18-Data Dictionary of BatchExam, RoomSchedule tables



Img 19-Data Dictionary of StudentReport, StudentTransaction tables

Architectural Modelling

Definition

An architecture model is a partial abstraction of a system. It is an approximation, and it captures the different properties of the system. It is a scaled-down version and is built with all the essential details of the system. (ScienceDirect, 2018)

Justification

The reasons for approaching this modelling methodology are mentioned below:

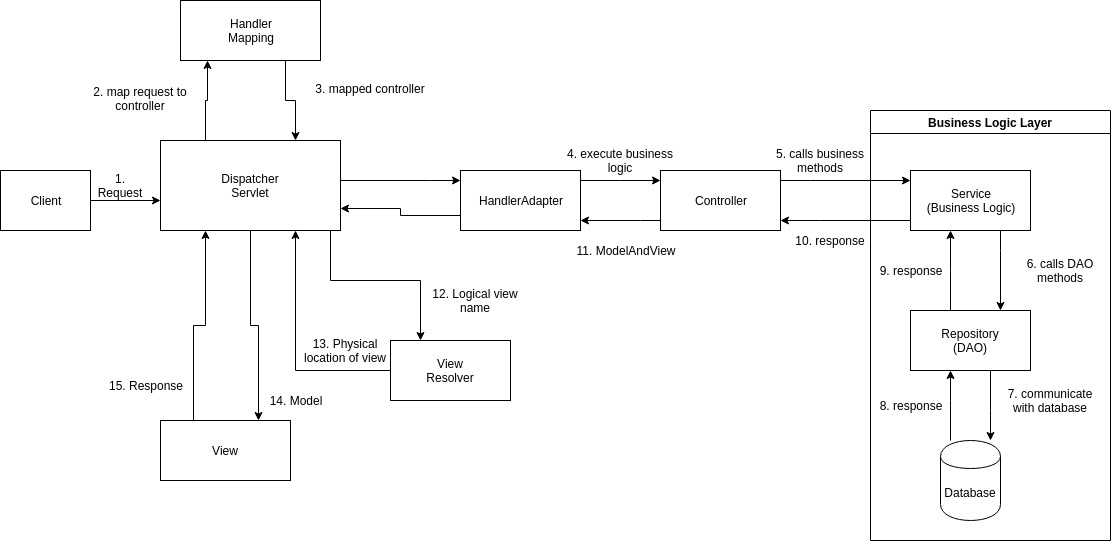
* It helps to easily identify the characteristics of the system and model it.
* It helps to visualize the information with the help of models.
* The basic knowledge of the system can be extracted from this modelling methodology.

Notations used

The notations used for this modelling are tabulated below:

|  |  |  |
| --- | --- | --- |
| Name | Notation | Description |
| Component |  | This notation represents the components of the system. It is generalization of various others components like class, interface, services, repositories. |
| Database |  | This notation represents the database used by the system. |
| Data/Process Flow |  | This notation represents the program flow. |

Actual Diagram



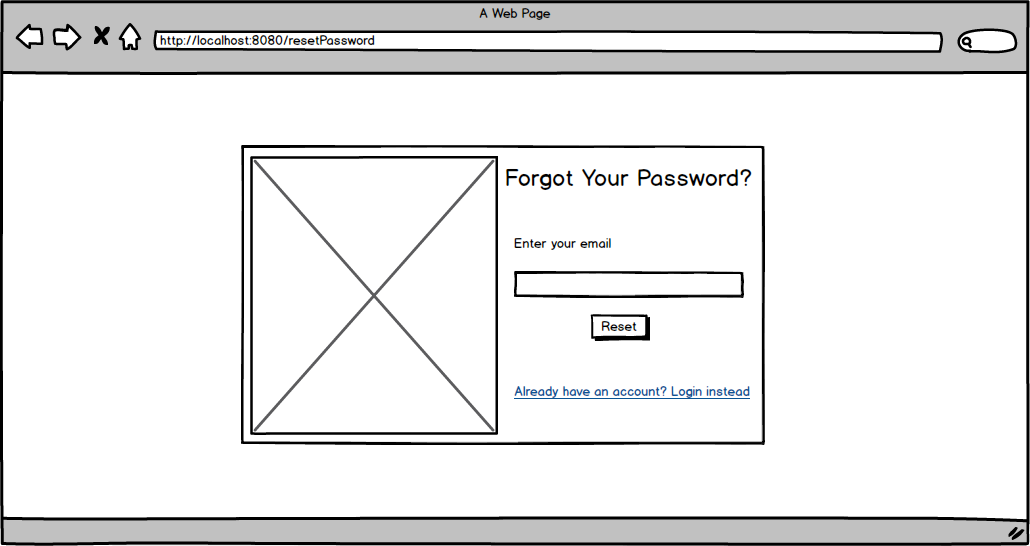
Img 20-Architectural Model

Diagram Description

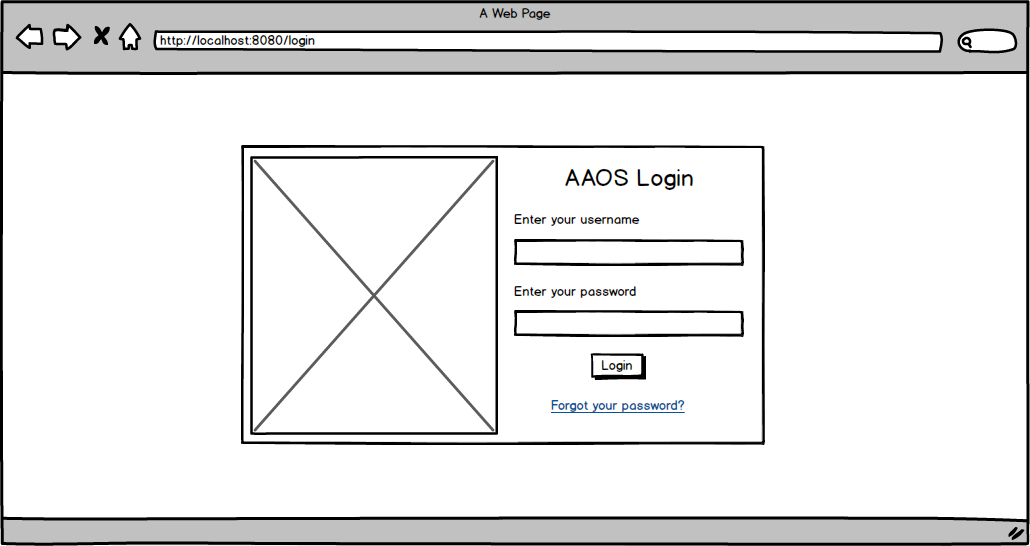
The request of the client is handled by the DispatcherServlet, then appropriate controller is selected by HandlerMapping where the request URL is mapped and finally it returns the controller. DispatcherServlet forwards the process to HandlerAdapter where it calls Controller for executing the business logic. Controller executes the business logic. Business logic are implemented in different layers: Service layer, Repository layer or Data Access Object and the database. After the controller finishes executing business logic, it sets the result in Model and returns the logical view name to DispatcherServlet. With the application of ViewResolver, the physical page is returned into the view. Ultimately, the view is returned as the response.

User Interface Modelling

UI Modelling presents the proposed system with the possible design or layout of the user interface. The user interface is modelling through the prototyping methodology. The prototypes are mentioned below as:



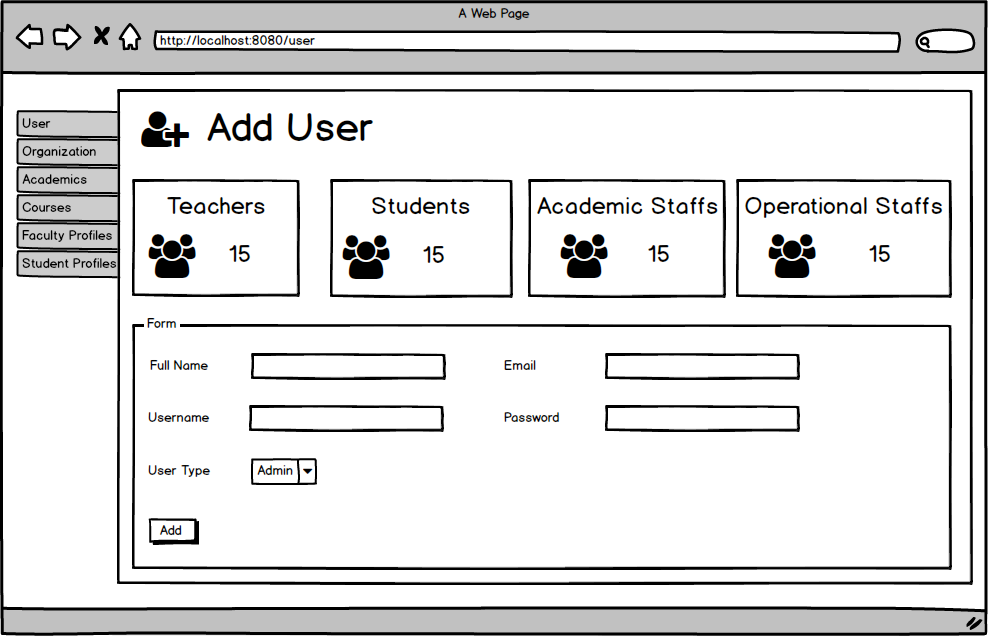
Img 21-Login Page Prototype



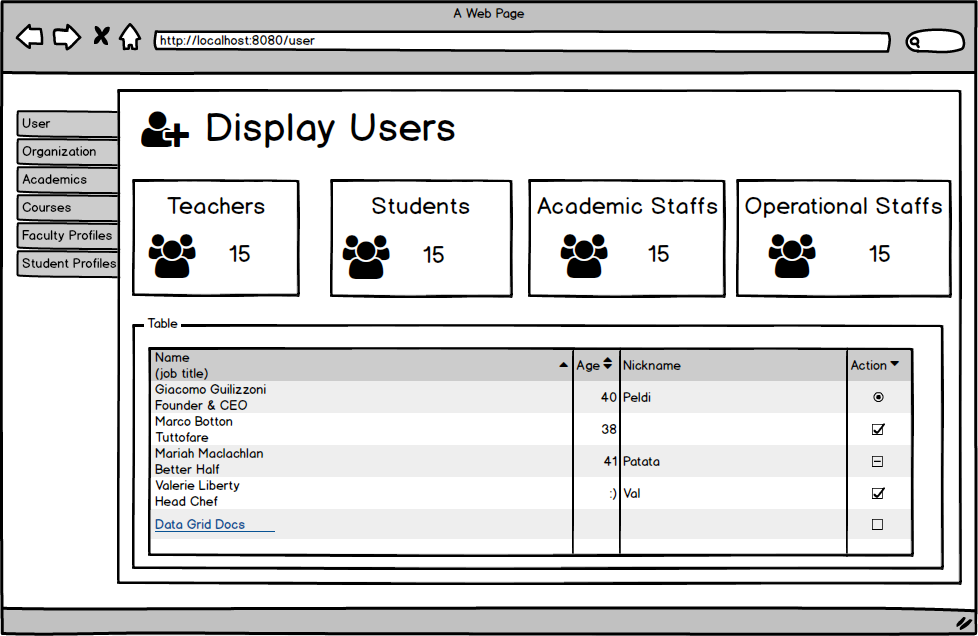
Img 22-Forgot Password Page



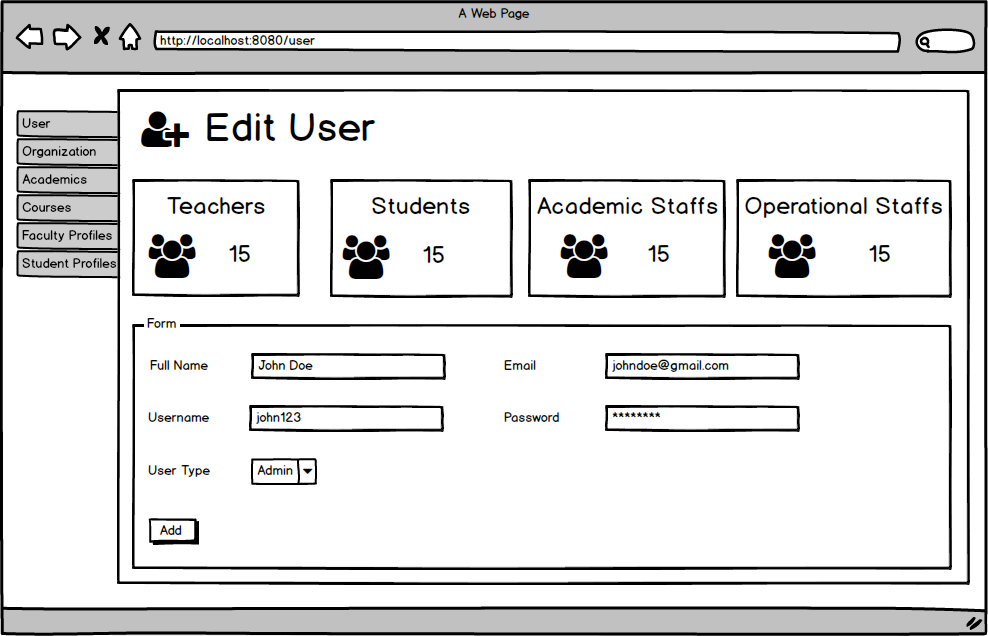
Img 23-Dashboard Page Prototype



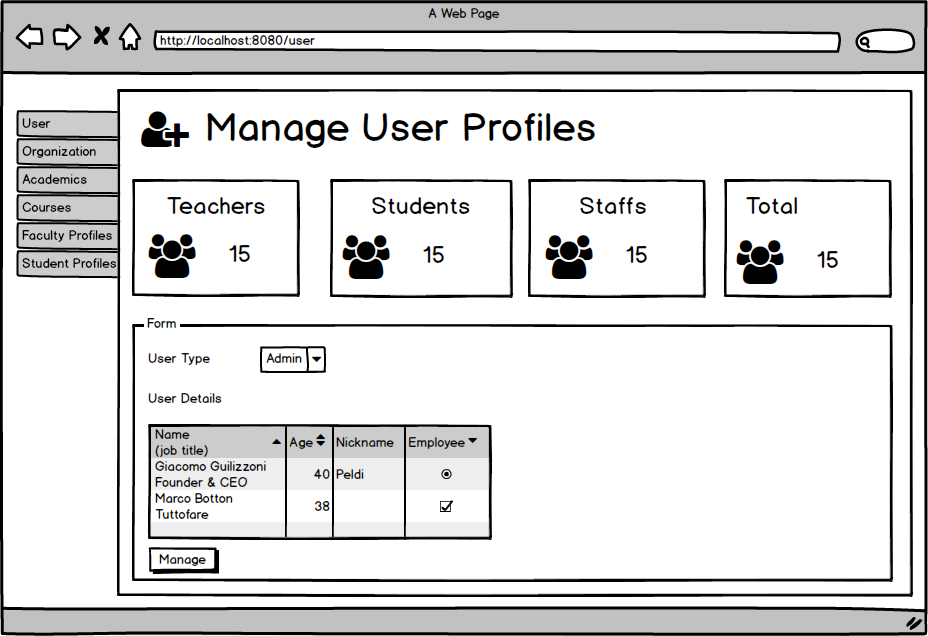
Img 24-Add User Page Prototype



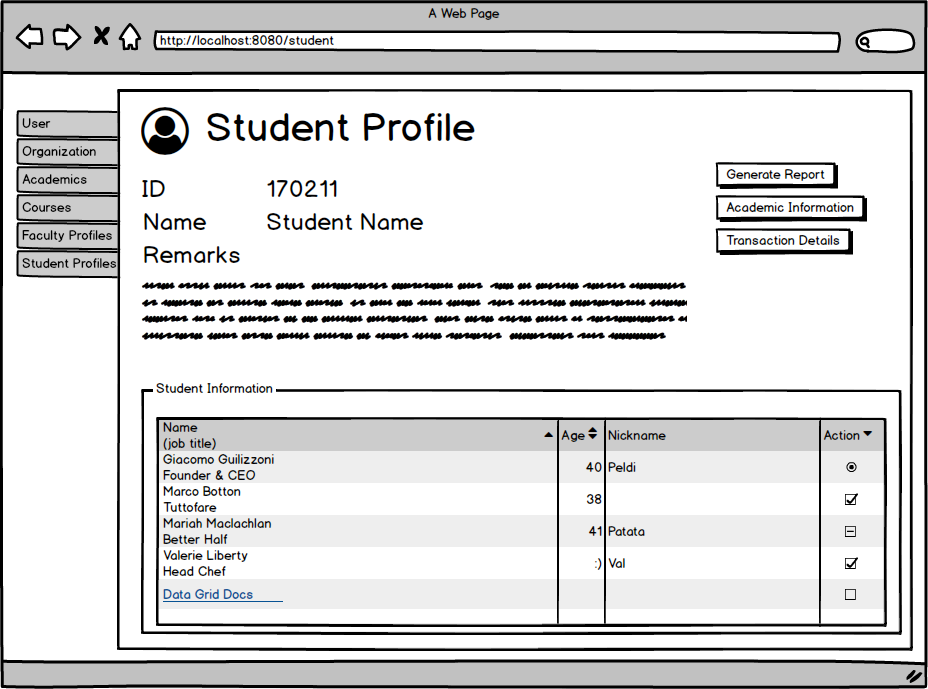
Img 25-Display User Page Prototype



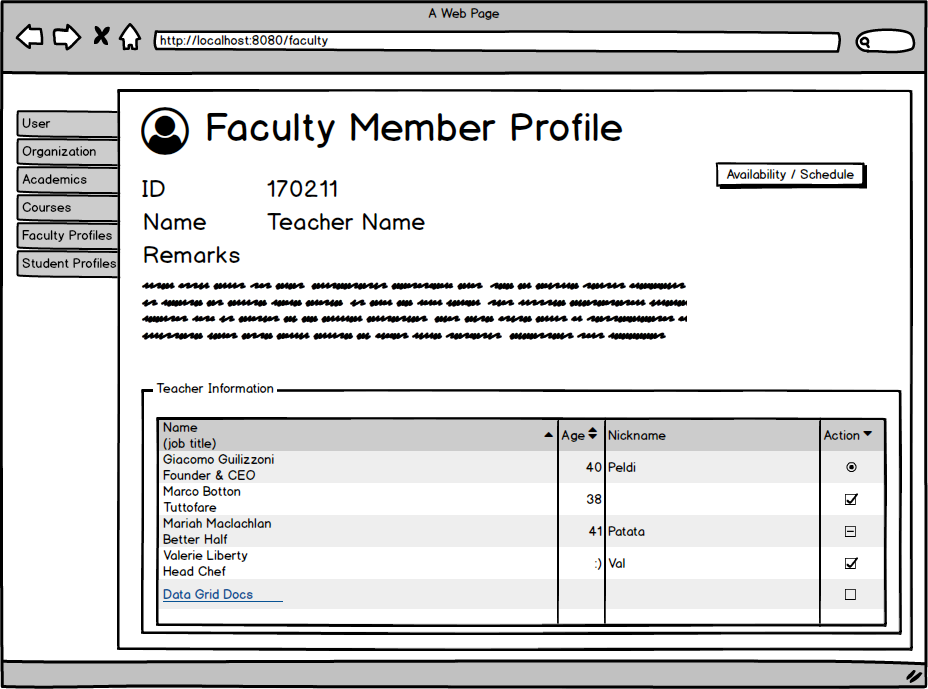
Img 26-Edit User Page Prototype



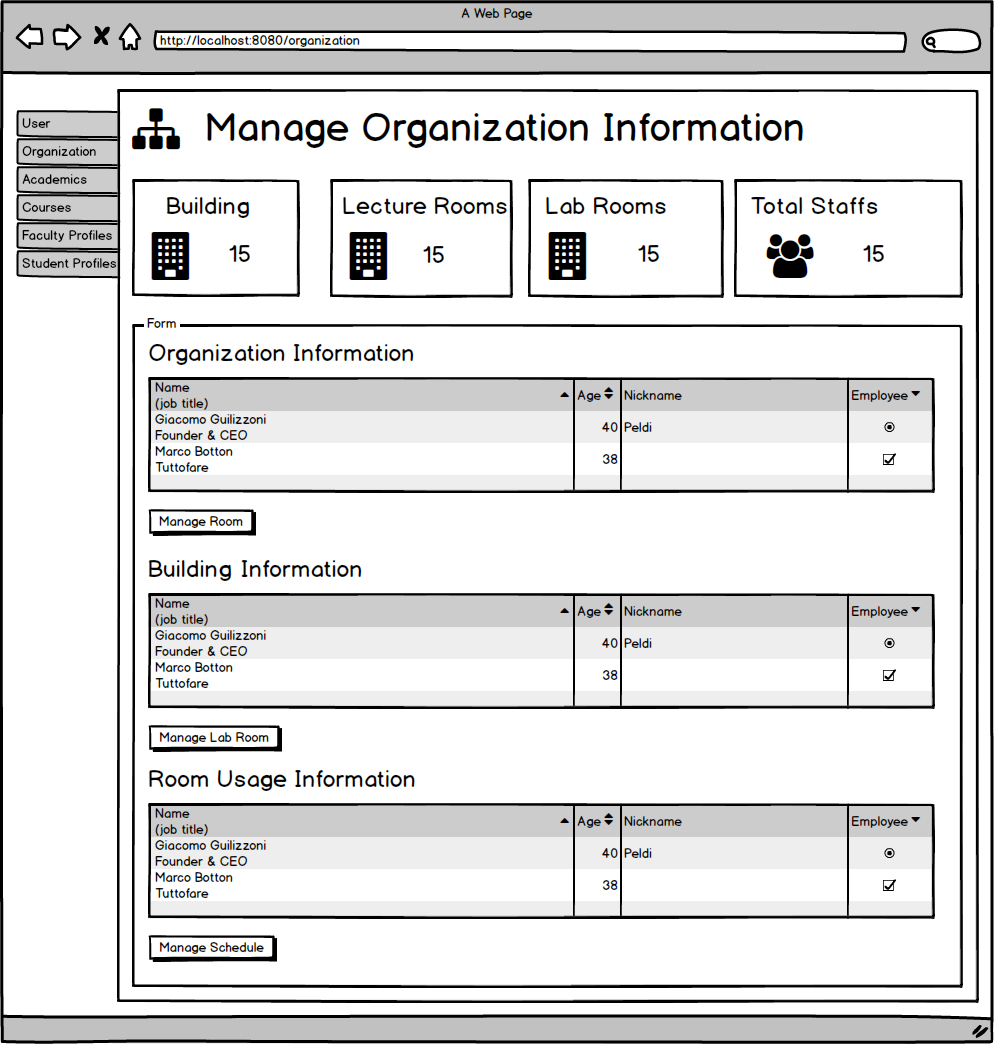
Img 27-Manage User Profile Page Prototype



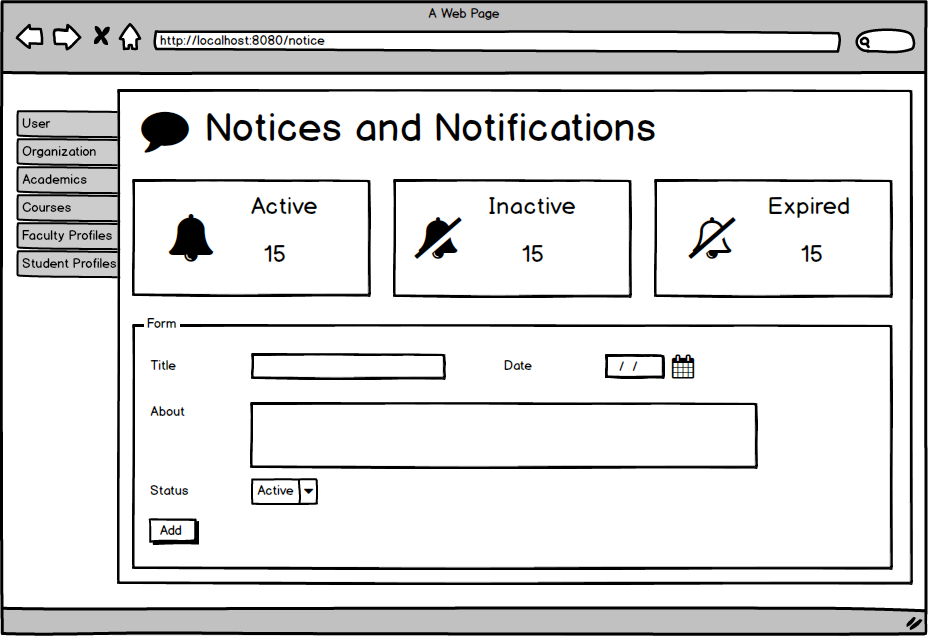
Img 28-Student Profile Page Prototype



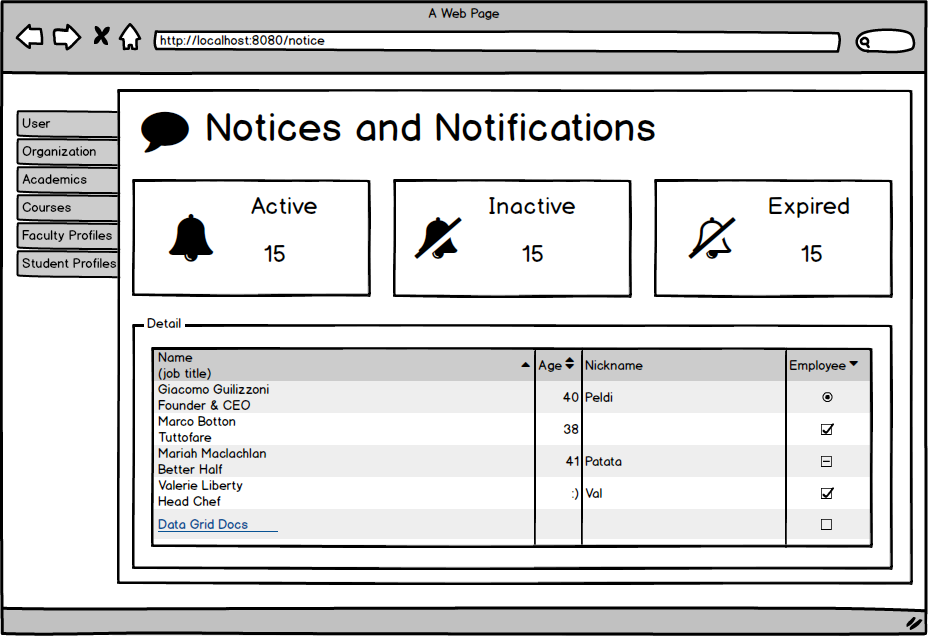
Img 29-Faculty Member Profile Page Prototype



Img 30-Organization Information Page Prototype



Img 31-Add Notice Page Prototype



Img -Manage Notice Page Prototype

Chapter 4: Implementation

Programming Language Used

**JAVA 8** is the programming language used to develop the overall system. On top of this language, Spring framework is used for the foundation of this system. More specifically, this framework comprises of **Spring MVC, Spring Security** and **Spring Test**. In case of view or the user interface, **Java Server Pages (JSP)** renders the complete web page. Along with JSP, for server-side manipulation of the view, **JSTL (JavaServer Pages Standard Tag Library)** is used whereas, for client-side manipulation **JavaScript** is used. Moreover, for asynchronous web request and response**, AJAX (Asynchronous JavaScript and XML)** is used along with **JQuery.**

Integrated Development Environment (IDE)

IntelliJ IDEA ULTIMATE 2019.1.1 is an IDE that is used for the development of this project. For the quick and convenient development work, this IDE was chosen as it is the smart IDE for Java developers.

Screenshots of the code snippets

Screenshots of distinct code snippets are kept here whereas, remaining code snippets that perform similar transactions are placed in appendix.

Chapter 5: Testing

About Testing

Testing in software development can be defined as the evaluation process where its functionalities are evaluated in order to identity any defects in the program and whether the software has meet its requirements.

Unit Testing

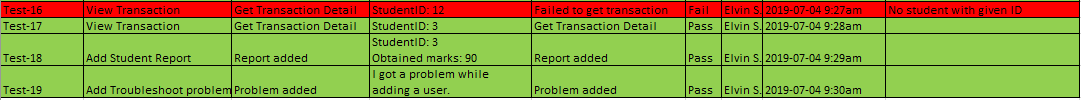
Unit testing is a level of software testing where individual units/components of a software are tested. A unit is the smallest testable part of any software. (Software Testing Fundamentals, 2009)

Black-box Testing

BLACK BOX TESTING, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. (Software Testing Fundamentals, 2009)

Black-box Testing Test Table

Black-box testing is done as per the test cases table mentioned below:



Chapter 6: Other Project Issues

Limitation of the project

* Use of JSP in spite of emergence of front-end technologies such as Angular framework makes the program to consume more resources because the complete webpage must be reserved by the server when a request is made.
* The form templates are not dynamic and this may result in additional work when specific requirement of particular organization is required to meet.

Future work of the project

* Intensive testing must be performed by testing it in real scenario of the educational organization.
* Additional appropriate charts and graphs must be integrated.
* Separate transaction module must be created and integrated, and should replace the current one which is very simple.
* Notification must be real-time and should be build using web-sockets instead of asynchronous request and response.

Risk Management

Project risk management is the process of identifying, analyzing and then responding to any risk that arises over the life cycle of a project to help the project remain on track and meet its goal. Risk management isn’t reactive only; it should be part of the planning process to figure out risk that might happen in the project and how to control that risk if it in fact occurs. (PROJECT MANAGER, 2017)

The lifecycle of the risk management process is as shown in the figure below: (INTLAND SOFTWARE, 2014)



Img 8 Risk Management Lifecycle

For the risk management of this project, the steps are generic compared to the lifecycle mentioned above. The major steps include:

1. Identification of risks
2. Risk and Vulnerability assessment
3. Identification of risk reduction measures
4. Prioritization of risk reduction measures

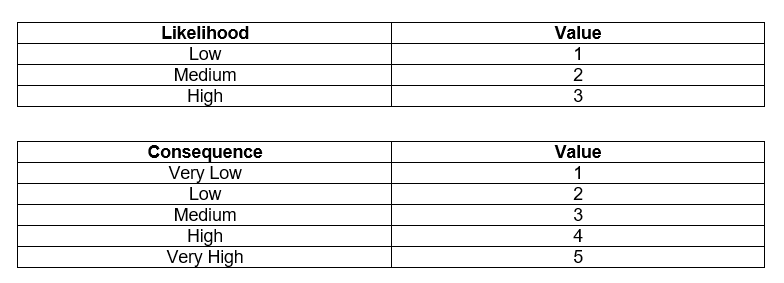
The risk assessment is performed on the basis of following factors:

* Consequence
* Likelihood
* Risk

The relation between these factors is given by,

Risk (Impact) = Consequence x Likelihood

Hence, with the reference of following table, the risk analysis is carried out as:



Img 9 Risk Analysis Lookup Table

The possible risks that were proposed in the proposal are analyzed here in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk** | **Likelihood** | **Consequence** | **Impact** | **Occurred** | **Measures Taken** |
| Addition of extra features that were not included in project scope | 1 | 5 | 5 | No | N/A |
| Deadline Overruns | 2 | 4 | 8 | No | N/A |
| Hardware failure | 2 | 4 | 8 | No | N/A |
| Natural calamities | 1 | 5 | 5 | No | N/A |
| Vulnerabilities in project dependencies | 2 | 4 | 8 | Yes | Performed impact analysis and switched to secure dependency. |
| Difficulty in implementing complex functionalities | 3 | 4 | 12 | Yes | Managed to get help from documentation and online community. |
| Growth of Project Scope | 2 | 5 | 10 | No | N/A |
| Inflexibility in design making change request impossible | 2 | 4 | 8 | No | N/A |

Configuration Management

The complete project directory structure configured as per various stages of Software Development Lifecycle resides in remote repository hosted in GitHub which can be accessed from <https://github.com/elwyncrestha/AAOS>. The root folder of remote repository provides the implementation codes along with the directory named **CP\_Config** where given project is stored as per the directory structure mentioned above.

User Manual

The user manual also resides in the remote repository in the directory named **UserManual** inside directory **5\_Documentation.** Some sample of user manual is as shown in the figure below:

Chapter 7: Conclusion

Chapter 8: References and Bibliography

1. ResearchGate (2009). (PDF) Personal Extreme Programming–An Agile Process for Autonomous Developers. Available at:

<https://www.researchgate.net/publication/229046039_Personal_Extreme_Programming-An_Agile_Process_for_Autonomous_Developers> [Accessed on: March 31, 2019]

1. Alpha Epsilon (2017). Personal Extreme Programming. Available at: <http://www.alpha-epsilon.de/programming/2017/12/06/personal-extreme-programming/> [Accessed on: March 31, 2019]
2. PROJECT MANAGER (2017). The Risk Management Process in Project Management. Available at: <https://www.projectmanager.com/blog/risk-management-process-steps> [Accessed on: April 6, 2019]
3. INTLAND SOFTWARE (2014). Benefits of Applying an Appropriate Risk Management Lifecycle. Available at: <https://content.intland.com/blog/alm/benefits-of-applying-an-appropriate-risk-management-lifecycle> [Accessed on: April 6, 2019]
4. Cybarlab (2017). Object Oriented Analysis (OOA) | Cybarlab. Available from: <https://cybarlab.com/ooa> [Accessed on: 24 April 2019]
5. CISCO (2011). Hardware and System Software Specification for Cisco Unified Web and E-Mail Interaction Manager. Available from: <https://www.cisco.com/c/dam/en/us/td/docs/voice_ip_comm/cust_contact/contact_center/cisco_interaction_manager/cim_441/reference/cim441bom.pdf> [Accessed on: 4 May 2019]
6. SVID (2005). What is design? Available from: <http://www.svid.se/en/What-is-design/> [Accessed on: 2 June 2019]
7. ScienceDirect (2018). Architecture Model - an overview | ScienceDirect Topics. Available from: <https://www.sciencedirect.com/topics/computer-science/architecture-model> [Accessed on: 3 June 2019]
8. Software Testing Fundamentals (2009). Unit Testing – Software Testing Fundamentals Available from: <http://softwaretestingfundamentals.com/unit-testing/> [Accessed on: 4 July 2019]
9. Software Testing Fundamentals (2009). Black Box Testing – Software Testing Fundamentals Available from: <http://softwaretestingfundamentals.com/black-box-testing/> [Accessed on: 4 July 2019]

Chapter 9: Appendix