# Chapter 2 – Analysis

## 2.1 Introduction to analysis

Analysis is the 2nd phase of the SDLC (Software Development Life Cycle). The main objective of analysis phase is requirements determination. This is the most important phase in software development as it works to understand the requirement of the system and helps to develop a system meeting those requirements. This is done by working through the information like who will use the product, how the user will use the product, specific information included with any special users’ requirement and studying and analyzing each requirement of the current system and defining and prioritizing users’ requirements. Analysis is the least technical phase of the SDLC and needs effective social, communication and managerial skills to be successful.

## 2.2 Methodology

A methodology is the steps that are followed when working on a project and in what order those steps should be taken. There are different types of analysis methodology like Hard System methodology, Soft System Methodology, combined soft/hard methodology, organization-oriented methodology etc. from those methodology I have chosen **Hard System methodology** for this project. Within Hard System methodology there are different methods among them I picked **Structured System Analysis and Design Methodology (SSADM).**

SSADM divides an application development project into stages, steps, modules, tasks, and provides a framework for describing projects in a way suited to managing the project.

Following are the stages of SSADM:

1. **Feasibility Study:** Feasible study of a project is to determine if the project is technically feasible, if it is feasible within the estimated cost or not, and will be profitable or not.
2. **Analysis and Requirement Specification:** All aspect of the system like hardware and software are analyzed and also investigate the current environment and then requirement specification is generated.
3. **Design:** Database design and set of program specifications are created using the logical system specification and technical system specification.
4. **Implementation:** The system is introduced to the company either directly in a phase transformation, or running in parallel with the existing system until the new system is working successfully.
5. **Testing:** This stage is done after the development of the system. In this stage all aspect of the new system is tested in order to find errors in the system and fix them to ensure robustness and reliability of the system.
6. **Maintenance:** To ensure there are no big problems in system regular maintenance is necessary. It also ensures robustness and reliability of the system.

A **data-flow diagram (DFD)** is a way of graphically representing a flow of a data of a process or an information system. And a good data-flow diagram makes it easy to understand the flow of data through the system.

Data Flow diagram of admin is given below:

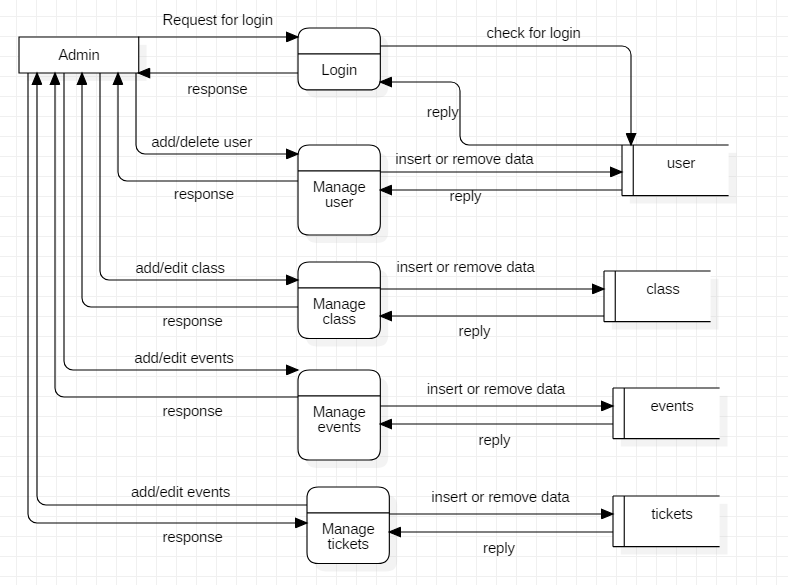


Figure 1: Data Flow Diagram of Admin

Data Flow diagram of user is given below:

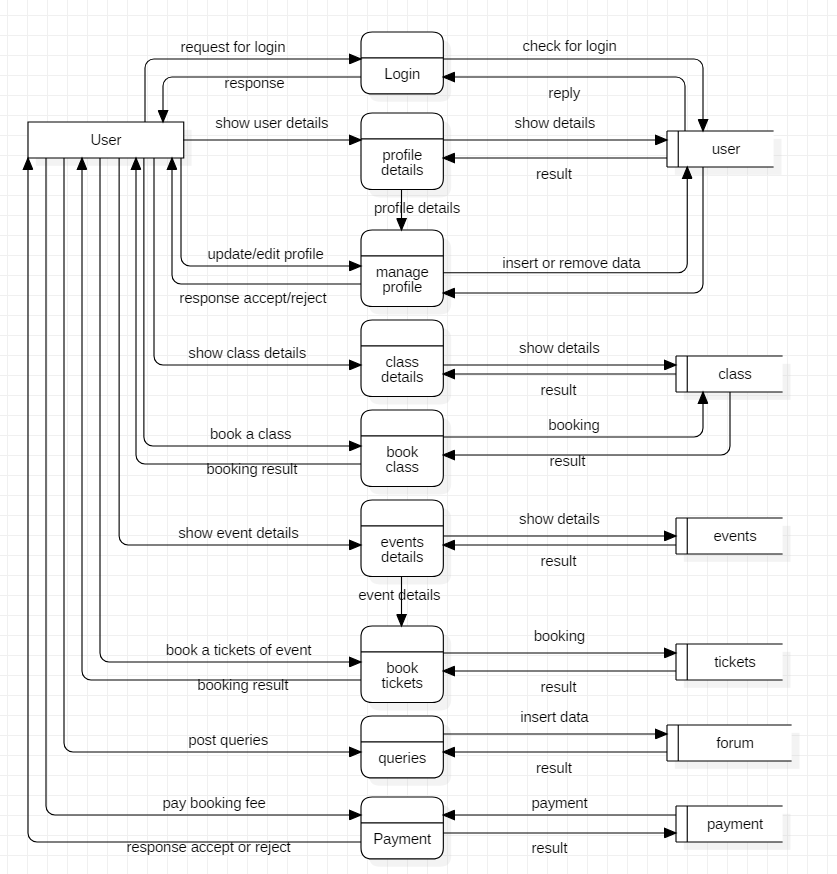


Figure 2: Data Flow Diagram user

## 2.3 Feasibility Study

A feasibility study is an analysis that study all of a project's relevant factors including economic, technical, legal, and scheduling considerations to ensure the likelihood of completing the project successfully. In every project, project manager uses feasibility study to find out the pros and cons of the project before they invest their time and money in that project. Feasibility study can provide important information that could prevent the company entering into risky business.

The different types of feasibility that are performed in this analysis are given below in the table:

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Feasibility Study | What does this study find out? | How it is related to my project. |
| 1 | Schedule Feasibility | Can this project be completed as schedule? Is there enough time to undertake the project? | I used WBS, milestone and Gantt chart for project schedule and its going as planned. |
| 2 | Technical Feasibility | Is current skill and technical resources sufficient for completing the system? | Technical resource needed to complete the system is sufficient with me and about skill it’s depends on situation. |
| 3 | Social Feasibility | Will the system feasible for todays’ society? Will it get negative reaction? | This system is web-based application for dance studio which makes busy society’s work easy. |
| 4 | Economic Feasibility | Is this project feasible to be completed within the given budget? Does it benefit to complete the project with that cost? | It helps to complete project within budget. And saves from loss. |
| 5 | Market Research | Is this project feasible with market? Is this system going to be on same lane with market? | Help to know if market need it or not. |
| 6 | Ethical Feasibility | Is this project ethically right for the users? | This system gives easy access to dance studio by giving them all needed information ethically. |

Table 1: feasibility study

## 2.4 Requirement Analysis

### 2.4.1 Functional Requirement

Functional requirements are concerned with system services such as the scope of the system, the necessary business functions and the required data structures.

The different types of functional requirement of system is given below in the table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Title** | **Description** | **Rational** | **Dependencies** |
| FR1 | Database Connection | It makes connection between system and database. | To handle user information. | N/A |
| FR2 | Registration | Registering new user for getting access to privilege. | To create an account. | FR1 |
| FR3 | Login | Function to let user to get access to the system. | To login to the system. | FR2 |
| FR4 | Authentication | Authentication is required to check if account is correct one. | To avoid unknown user to get access. | FR2 |
| FR5 | Add class | If there is some new class in studio are added then admin can add those classes in the system. | To add new class. | FR3 |
| FR6 | Delete class | If some of the existing class are going to be close then admin can remove those class from the system. | To remove class. | FR3 |
| FR7 | Edit profile | User can edit their profile information.  Change their information. | To edit profile. | FR3 |
| FR8 | Delete profile | User themselves can delete their profile if they are closing contact with the studio. | To delete profile. | FR3 |
| FR9 | Add events | Any new events that the studio is going to organize or participate in are posted here by admin. | To add events. | FR3 |
| FR10 | Edit events | Only admin can edit existing events from system. | To edit events | FR3 |
| FR11 | Delete events | Admin can delete events that are not needed. | To delete events | FR3 |
| FR12 | Post tickets | Tickets of events organize by studio are posted in the system by admin. | To post tickets. | FR3 |
| FR13 | Tickets booking | User can book events tickets from home. | For booking event tickets. | FR3 |
| FR14 | Class booking | User can book available dance class anytime they want. | To book dance class. | FR3 |
| FR14 | Cancel booking | User can cancel their class and tickets booking before class or events starting day. | To cancel tickets and class booking. | FR3 |
| FR15 | Payment | Users who books tickets and class pays for them with the help of payment function | To pay for booking of tickets and class. | FR3 |
| FR16 | Add user | Only admin can add other user in the system. | To add user. | FR3 |
| FR17 | Delete user | Only admin can delete other user from the system | To delete user accounts. | FR3 |
| FR18 | Advertise performance | Admin can advertise other user or another artists’ performance. | To advertise. | FR3 |
| FR19 | Queries | user who has login access can ask questions in this section. | To post question and answer. | FR3 |
| FR20 | Log out | User can logout when they finish their work. | To logout the account. | FR3 |

Table 2: functional requirements

### 2.4.2 Non- Functional Requirement

Whereas nonfunctional requirements deal with system constraints such as operation-how easy is it to use, performance and security, these are also known as supplementary requirements.

The different types of non-functional requirement of system is given below in the table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Title** | **Description** | **Rational** | **Dependencies** |
| NFR1 | Security | System should be secure by authentication method to secure data and information. | To secure data and information of users. | FR2 & FR3 |
| NFR2 | Performances | System should perform quick with better response time. | To make program run quick and smooth. | N/A |
| NFR3 | Availability | System should be available anytime user wants. | To ensure user can access the system anytime. | N/A |
| NFR4 | Privacy | System should be able to provide privacy. Privacy make system trustable. | To ensure user can feel safe. | N/A |
| NFR5 | Maintainability | System should be able to maintain itself and repair and replace faulty components. | To ascertain there will be not many errors. | N/A |
| NFR6 | Usability | System should be usable by user with effectiveness, efficiency, and satisfaction. | To make user satisfied. | N/A |
| NFR7 | Scalability | Increase in user shouldn’t be problem to system. It should work with ease even with more user. | To run system smoothly even with large number of users. | N/A |
| NFR8 | Reliability | System should perform consistently well. | To build trust. | N/A |
| NFR9 | Legal | System should be made within legal terms. So, it can operate in society. | System inside legal boundary keep problem away. | N/A |
| NFR10 | Capacity | System must be capable of storing large data effectively and inform before storage runs out of space. | To ensure system run efficiently. | N/A |

Table 3: non-functional requirements

### 2.4.3 MoSCoW Prioritization

MoSCoW prioritization is a popular prioritization technique use for managing requirements. This method is commonly used for making key stakeholders understand the significance of initiatives in a specific release.

There are four categories of MoSCoW and they are must have, should have, could have, and won’t have. Sometimes, “wish” is used in MoSCoW instead of “won’t have”.

* Must have

It describes those requirements that must be in the system. These requirements are non-negotiable. And without them project will fail.

* Should have

It describes the requirements which occupy second place in the priority list. It is important but not critical to launch but it has high value to users.

* Could have

This point describes requirements that are desirable but not necessary.

* Won’t have

This point describes the requirements which will not be implemented in current release but it may be included in future release.

The prioritization of different functional requirements of my project are shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Title** | **MoSCoW** | **Remarks** |
| FR1 | Database Connection | Must have |  |
| FR2 | Registration | Must have |  |
| FR3 | Login | Must have |  |
| FR4 | Authentication | Must have |  |
| FR5 | Add class | Must have |  |
| FR6 | Delete class | Must have |  |
| FR7 | Edit profile | Must have |  |
| FR8 | Delete profile | Should have |  |
| FR9 | Add events | Must have |  |
| FR10 | Edit events | Could have |  |
| FR11 | Delete events | Must have |  |
| FR12 | Post tickets | Should have |  |
| FR13 | Tickets booking | Should have |  |
| FR14 | Class booking | Must have |  |
| FR14 | Cancel booking | Must have |  |
| FR15 | Payment | Should have |  |
| FR16 | Add user | Should have |  |
| FR17 | Delete user | Should have |  |
| FR18 | Advertise performance | Won’t have |  |
| FR19 | Queries | Must have |  |
| FR20 | Log out | Must have |  |

Table 4: MoSCoW Prioritization of functional requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Title** | **MoSCoW** | **Remarks** |
| NFR1 | Security | Must have |  |
| NFR2 | Performances | Should have |  |
| NFR3 | Availability | Should have |  |
| NFR4 | Privacy | Must have |  |
| NFR5 | Maintainability | Should have |  |
| NFR6 | Usability | Should have |  |
| NFR7 | Scalability | Must have |  |
| NFR8 | Reliability | Should have |  |
| NFR9 | Legal | Must have |  |
| NFR10 | Capacity | Should have |  |
| NFR1 | Security | Must have |  |

Table 5: MoSCoW Prioritization of non- functional requirements.

### 2.4.4 SRS (System Requirement Specification)

Hardware and software requirement specification are given below:

|  |  |
| --- | --- |
| **Hardware** | **Software** |
| * RAM: 2 GB minimum * Processor: core 2 duo and higher * Disk space: 400GB | * Operating system: Windows 7, 8, 10 * Browser: Mozilla Firefox, google chrome, opera. * Database: SQL |

Table 6: hardware and software specification

## 2.5 Use Case diagram

Use Case is a graphical representation of interaction between actor and system. It shows how user can work in system with following notations:

* Actor (user): stick man figure which represents user.
* System boundary: rectangle shape box.
* Use Case: it is in oval shape where actions that actor might perform.
* Associations: it is a line which shows relation between actor and use case.

Use case diagram of admin is given below:

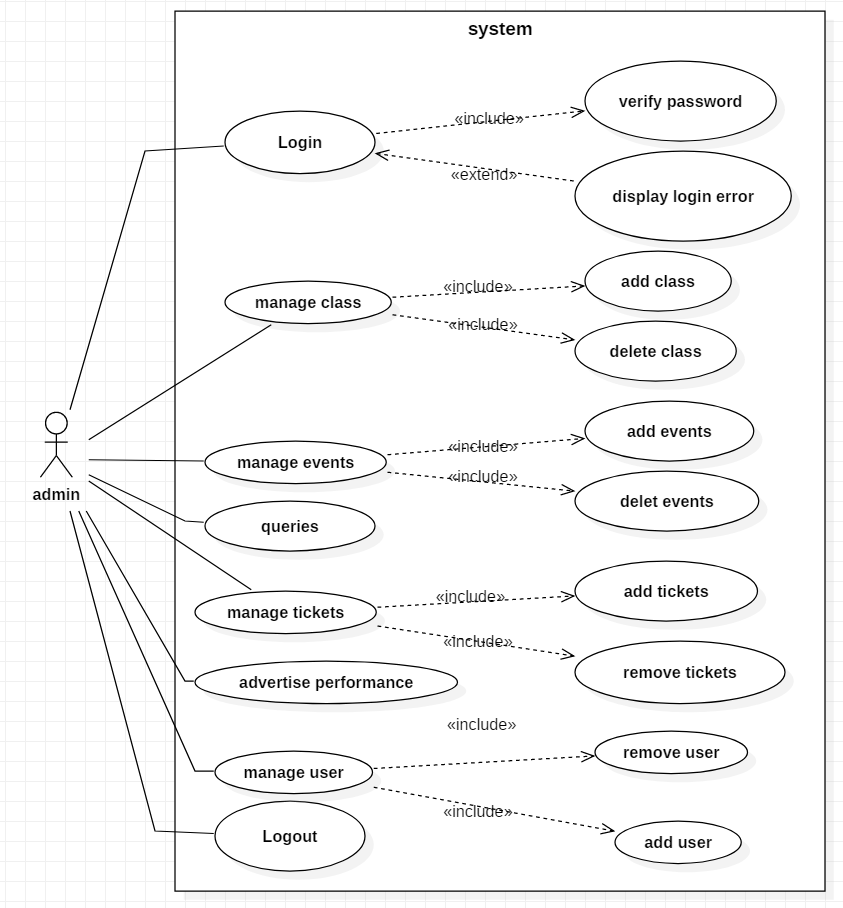


Figure 3: Use Case of admin

Description

Actor: Admin

Admin is a user who supervise the system with its super privilege. Admin can use following function:

* Admin can login to the system with correct id and password.
* Admin can manage classes like adding new class and deleting existing class.
* Admin can manage events like add events and delete events.
* Admin can use queries to ask question or give answer to questions.
* Admin can add tickets of events and remove tickets.
* Admin can advertise performance.
* If there is some user who needed to be added in the system or remove from the system then admin can do that.
* Admin can logout from the system so other can’t use the same id to misuse the privilege of admin.

Use case diagram of user is given below:

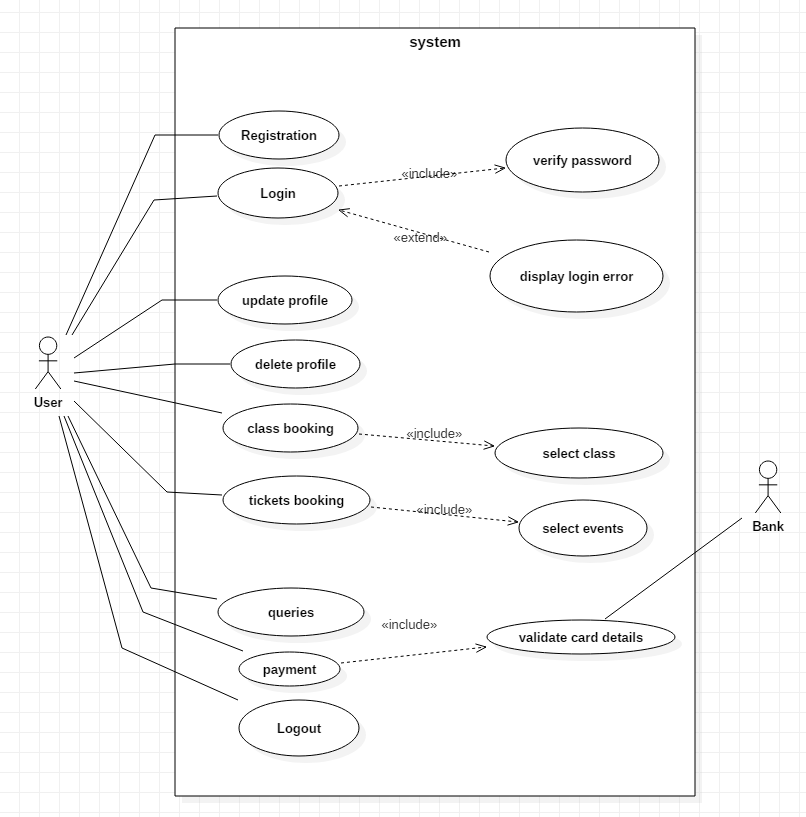


Figure 4: Use Case of user

Description

Actor: general user

General users are user for whom this system is made. They can perform following activities:

* Unregister user can register as new user in the system.
* User can login to the system after they register in the system.
* User has privilege to edit their own profile.
* If user is leaving studio, they can delete their profile.
* User can book classes they want to take part in.
* User can book events that they want to visit.
* Users can post question and answer in queries.
* User can logout from the system so, their account remains safe.

## 2.6 NLA & Initial Class Diagram

Dance studio is a dance school in Lalitpur which provide platform for learning and performing dance for children. Its been doing work like booking class, booking tickets of events, advertising events and performance manually. Now every company works digitally and it makes work easier and efficient so, Dance studio wants to develop the online website for themselves.

In this system there are two types of users with different access of function. They are admin and general user. General user needs to register into the system to get access to the functions of the website with information like name, address, phone, email, password, etc. with the access to system user must be able to view the class they want and book them to enroll in that class. With login access of general users, they can also update and delete their profile. Since there is event section so, general users can also book events tickets online. There is a queries section in which user can ask question or provide answer.

|  |  |  |
| --- | --- | --- |
| **Candidate Classes** | **Candidate Method** | **Candidate attribute** |
| users, class, events, tickets, payment | Book, register, view, add, delete, update | name, address, email, phone, password |

Initial Class Diagram is given below:

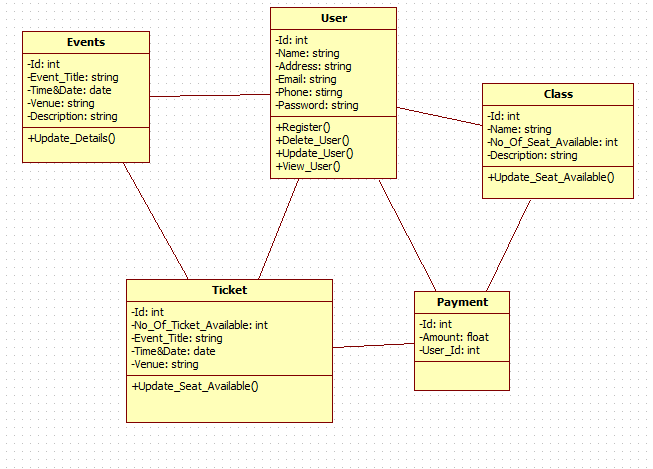


Figure 5: initial class diagram