# Introduction to Analysis

Analysis can be referred to as the process of defining what the user wants and expects from a system. After gathering enough information on project requirements, the specified requirements should be analyzed, well-documented and validated. The utmost need for analysis is to identify the needs of respective stakeholders and align them as per the need of business.

# Analysis Methodology

Out of a pool of analysis methodology, I found SWOT analysis to be the best fit for this project. Since, the platform is limited to providing tutorials on programming only, there are always rooms for improvement in every aspect of the project. It stands for Strengths, Weakness, Opportunities and Threats (SWOT).

Compared to other analysis methodologies, SWOT analysis often focuses on improvement and expansion of any project in the long run. Since, technology industry is an ever changing and evolving, it is necessary to keep up with latest trends and practices.

The following SWOT framework has been used for analyzing strengths, weaknesses, opportunities and threats.

## Strengths

The advantageous corners of the project that provide precedence over the others can be classified as strengths. The following aspects has been classified under strengths.

* **Distribution of resources in various platforms:** The course contents are available in various platforms such as mobile phone and desktop without compromising the user experience.
* **Accessibility:** The course contents are accessible 24/7. Meaning, learners can learn anytime from anywhere.
* **Promotes active and independent learning:** Learners can have their own space for assimilating the course contents to their convenience.
* **Removes time constraints:** Learners can assimilate course contents at their own pace without any deadlines or time specific constraints.
* **Interactive community for learners:**  A separate community for learners to share and learn concepts beyond contents.
* **Open-source platform:** An entirely free distribution and to access course contents.

## Weaknesses

* **Limited and inadequate**: The contents are not always fulfilling and are inadequate to certain degree.
* **Contents stick to being knowledge-based rather than practicality-based:** Appropriate practical environment for accessing concepts might be absent.
* **Low technical support:** Low technical support if technical issues and problem arise.

## Opportunities

* **Ever expanding platform with latest technology trends:** With changing and evolving technology trends and our dependency, it provides up to date materials.
* **Online certification system:** In order to provide a sense of achievement to learners, certification is provided on completion of a course.
* **Increasing market demand for e-learning services:** Almost every aspect of our life is dependent on technology. Utilizing this opportunity, number of online learners are increasing at the same rate.

## Threats

* **Security risk:** User data might be at risk while surfing online.
* **Chances of students losing motivation:** Since, there is no constrains binding a learner to access the course contents, the learner might lose motivation due to procrastination and lethargy.
* **Copyright infringement on contents: C**ontents derived from other sources might have copyright act embedded within them, it is necessary to have copyright checks in order to avoid unforeseen copyright infringements.

# Feasibility Study

It is necessary to keep track of the analysis process as well as the feasibility of thought-out strategies. With strong reference to factors affecting the system development, the pros and cons has been weighted out with appropriate explanations.

## Technical feasibility

Form technical point of view, the overall application focus on delivering course contents to a wide range of configuration devices, from low end to high end. Since, it is light weight, it is easily accessible even in low end network and low configuration devices.

## Operational feasibility

The overall application focuses mainly on user experience. The navigation system is furbished with navigation bar and appropriate navigation buttons and links. The user will not loose their way while surfing the application contents. The course contents can be download by the user to access it offline in case of unforeseen connection surges.

## Schedule

A system must have a specific schedule to undergo timely maintenance. In case of this application, before undergoing maintenance, a blog post on maintenance break will be delivered to all learners to avoid rooms for inconvenience.

## Legal and social feasibility

Since, contents derived from other sources can have copyright act embedded within them, it is necessary to have copyright checks and cite or credit the rightful owner of derived content in order to avoid unforeseen copyright infringements. The course contents, if derived from other sources will be cited credited along with respective content links.

## Economic feasibility

Since, the goal is to provide open-source platform, initial investments should be handled thoughtfully. With time and increased traffic of web-application, certain sponsors like educational offers and events can uplift the economic and financial requirements.

# Software Requirement Specification

## Functional Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Req.ID | Title | Description | Purpose | Dependences |
| FR1 | User Authentication | Assigning privileges as per the user’s authority | Security and integrity against a particular user’s action. | **N/A** |
| FR2 | Add course | Addition of new course by the authority | Including new trends and practices | **FR1** |
| FR3 | Update course | Update the contents of an existing course | Keeping up with the latest contents. | **FR1, FR2** |
| FR4 | Delete course | Delete the contents of an existing course | Excluding unwanted additions | **FR1, FR2** |
| FR5 | Display available course | Exhibit existing courses to the user | Interact and assimilate contents of an existing course | **FR1, FR2** |
| FR6 | Post comments on course | Leave comments on an existing course | Comment about additional queries and answers. | **FR1, FR2** |
| FR7 | Update comment | Update an existing comment | Edit or alter existing comment | **FR1, FR2, FR3** |
| FR8 | Delete comment | Delete an existing comment | Delete unwanted comments | **FR1, FR2, FR3** |
| FR9 | View comment | Display and read comments on courses | Display and read comments | **FR1, FR2, FR3** |
| FR10 | Create user profile | Create new learner profile | To save the details and progress of user | **FR1** |
| FR11 | Delete user profile | Delete existing user | Delete existing user | **FR1, FR10** |
| FR12 | Edit user profile | Update an existing user profile | Edit or alter existing user details | **FR1, FR10** |
| FR13 | Display user details | Display selected user details | Display details of a user profile | **FR1, FR10** |
| FR14 | Mark course as read | Mark selected course as read | Mark selected course as read | **FR1, FR5** |
| FR15 | Mark course as unread | Mark selected course as unread | Mark selected course as unread | **FR1, FR5** |
| FR16 | Blog CMS | Perform CRUD operations on blog posts | Perform CRUD operations on blog posts | **FR1** |
| FR17 | Repot issues | Report issues related to course | Report issues related to course | **FR1, FR5** |
| FR18 | View reported issues | View issues related to course | View issues related to course | **FR1, FR17** |
| FR19 | Delete reported issues | Delete reported issues related to course | Delete reported issues related to course | **FR1, FR17** |
| FR20 | Feedback CMS | Perform CRUD operations on course feedbacks | Perform CRUD operations on course feedbacks | **FR1, FR5** |
| FR21 | Request courses | Learners can request for courses | Requesting for addition of courses if not available. | **FR1** |
| FR22 | View requested courses | Display requested courses by learners | Display requested courses by users | **FR1, FR21** |
| FR23 | Delete requested courses | Delete requested courses by users | Delete requested courses by users | **FR1, FR21** |

## Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| Req.ID | Category | Description |
| NFR1 | Modifiability | The contents of the application should be open to updates and be able to adapt changes as per the latest technology trends. |
| NFR2 | Interface | The interface should be user-friendly and easy to navigate since, majority users might not have experience using such technologies. |
| NFR3 | Usability | The effectiveness of an application is reflected upon its degree of usability. The learning environment should be user-friendly so that learners can focus more on learning rather than navigating. |
| NFR4 | Flexibility | Updating contents means having dynamicity within the system. Data exchange and reliability ought to be flexible. |
| NFR5 | Interoperability | The system ought to provide centralized learning environment with evaluation capabilities. |
| NFR6 | Security | Sensitive data contained and exchanged within the system should be monitored and inspected regularly. |
| NFR7 | Performance | Performance as in convenient response time and casting away bugs and glitches at bay. |
| NFR8 | Data integrity | The once deployed and received should have strong received set. |
| NFR9 | Modularity | In order to make projects manageable and extendable, we make the project modular |
| NFR10 | Scalability | The project ought to be scalable and open to any changes. |
| NFR11 | Load and Concurrency | The system ought to be able to log multiple user as one given time. |

## MoSCoW Prioritization

|  |  |  |  |
| --- | --- | --- | --- |
| Req.ID | Requirement | MoSCoW | Justification |
| FR1 | User Authentication | **M** |  |
| NFR1 | Modifiability | **M** |  |
| NFR2 | Interface | **M** |  |
| NFR3 | Usability | **M** |  |
| NFR4 | Flexibility | **M** |  |
| NFR5 | Interoperability | **M** |  |
| NFR6 | Security | **M** |  |
| NFR7 | Performance | **M** |  |
| NFR8 | Data integrity | **M** |  |
| NFR9 | Modularity |  |  |
| NFR10 | Scalability |  |  |
| NFR11 | Load and Concurrency |  |  |
| FR2 | Add course |  |  |
| FR3 | Update course |  |  |
| FR4 | Delete course |  |  |
| FR5 | Display available course |  |  |
| FR6 | Post comments on course |  |  |
| FR7 | Update comment |  |  |
| FR8 | Delete comment |  |  |
| FR9 | View comment |  |  |
| FR10 | Create user profile |  |  |
| FR11 | Delete user profile |  |  |
| FR12 | Edit user profile |  |  |
| FR13 | Display user details |  |  |
| FR14 | Mark course as read |  |  |
| FR15 | Mark course as unread |  |  |
| FR16 | Blog CMS |  |  |
| FR17 | Repot issues |  |  |
| FR18 | View reported issues |  |  |
| FR19 | Delete reported issues |  |  |
| FR20 | Feedback CMS |  |  |
| FR21 | Request courses |  |  |
| FR22 | View requested courses |  |  |
| FR23 | Delete requested courses |  |  |

# Software and Hardware Specification

|  |  |
| --- | --- |
| Tools | Specified tools |
| IDE | IntelliJ IDEA 2018.3.5 |
| Programming language | Java Servlet (Back-end) |
| Frameworks used | Bootstrap (Front-end) |
| Database | MySQL |
| Server solution stack | XAMPP |
| Modelling tool | Visual Paradigm |

# Use Case Diagram

## Login and Registration

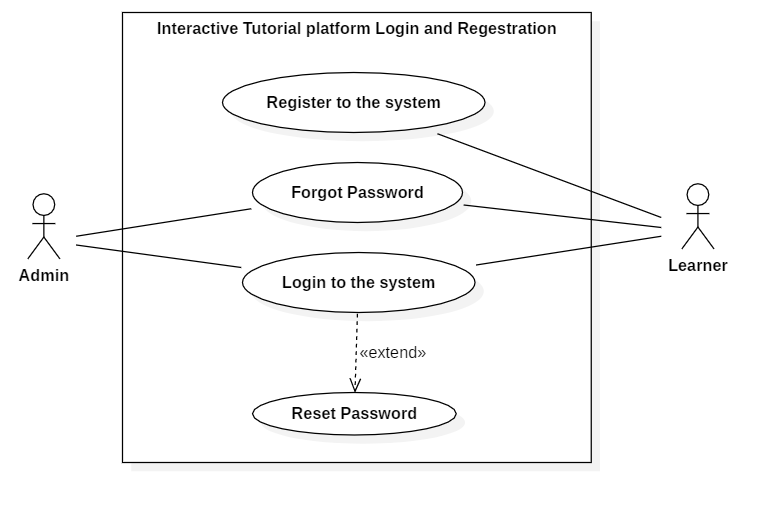


Figure : Login and registration

|  |  |
| --- | --- |
| **Justification** | The following use case depicts the flow of how a user logs or registers into the system viable for both admin and learners. |
| **Primary actor(s)** | Admin/Learners |
| **Secondary actors(s)** | *NA* |
| **Primary flow** | 1. Normal users initially register into the system leaving genuine profile information. 2. The user logs into the system via genuine profile information |
| **Alternative flow** | 1. In case the user forgets or loses their password, an option for resetting password is available. |

## Learner Profile

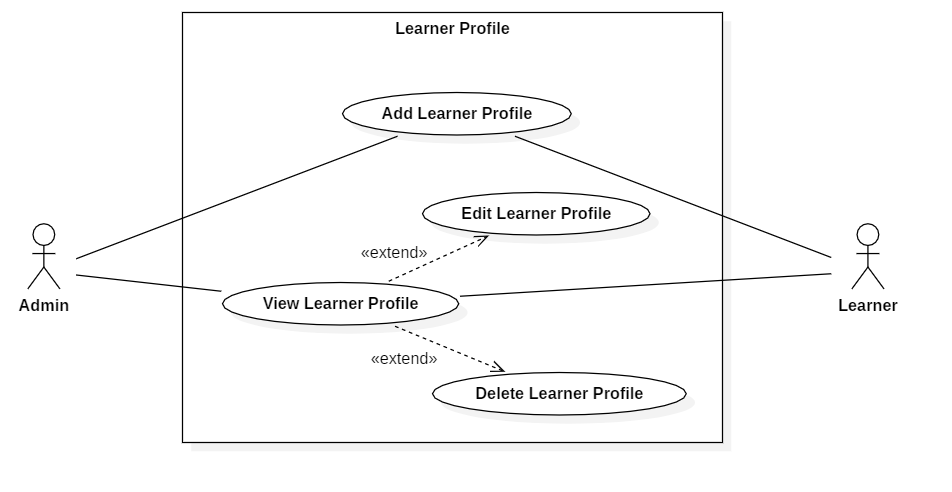


Figure : Learner Profile

|  |  |
| --- | --- |
| **Justification** | The following use case depicts the flow of how a user adds, views, edits and deletes learner profile. |
| **Primary actor(s)** | Admin/Learners |
| **Secondary actors(s)** | *NA* |
| **Primary flow** | 1. Normal users create a learner profile via registration (Admin user adds learner profile in some cases). 2. Both Admin and normal users can view learner profile with normal users having access only to their specific profile. 3. Once the profile has been viewed, an option for editing and deleting the profile credentials is available. |
| **Alternative flow** | 1. While adding or editing profile, users might input invalid data into the forms field. For this, via proper validations, users are to be asked to fill valid information. |

## Course – Admin

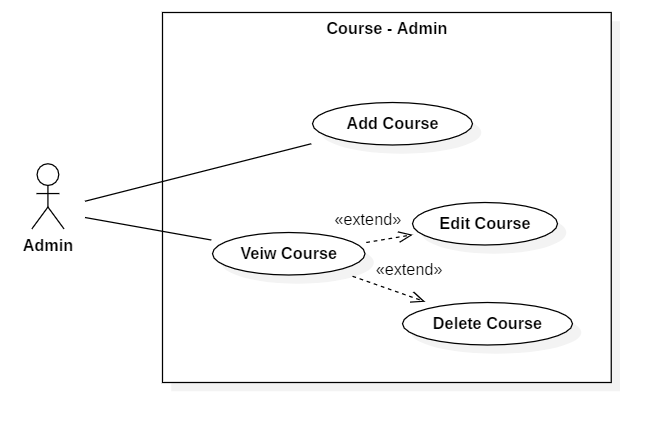


Figure : Course – Admin

|  |  |
| --- | --- |
| **Justification** | The following use case depicts the flow of how an admin user adds, views, edits and deletes a course. |
| **Primary actor(s)** | Admin |
| **Secondary actors(s)** | *NA* |
| **Primary flow** | 1. The admin user adds course to the system making it available for normal learner users. 2. Once the course has been added, the admin has the authority to edit or delete the content. |
| **Alternative flow** | 1. While adding or editing course, the admin might input invalid data into the field. For this, via proper validations, the admin is to be asked to fill valid information. |

## Course – Learner

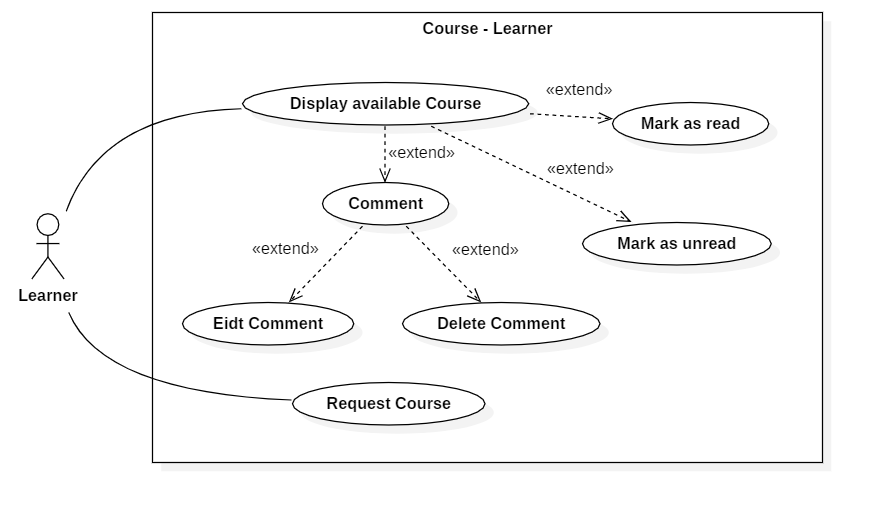


Figure : Course – Learner

|  |  |
| --- | --- |
| **Justification** | The following use case depicts the flow of how a normal user or learner accesses the available course. |
| **Primary actor(s)** | Learners |
| **Secondary actors(s)** | *NA* |
| **Primary flow** | 1. The user views one of the available courses. 2. The user has the privilege to leave comments on the course. 3. The comments left by a user can be edited and deleted by that particular user. 4. The courses, once completed or has fulfilled its purpose, can be marked as read or marked as unread at user’s will. |
| **Alternative flow** | 1. In case the user does not find what he/she is looking for, a field for requesting courses for future addition is available. |

## Blog

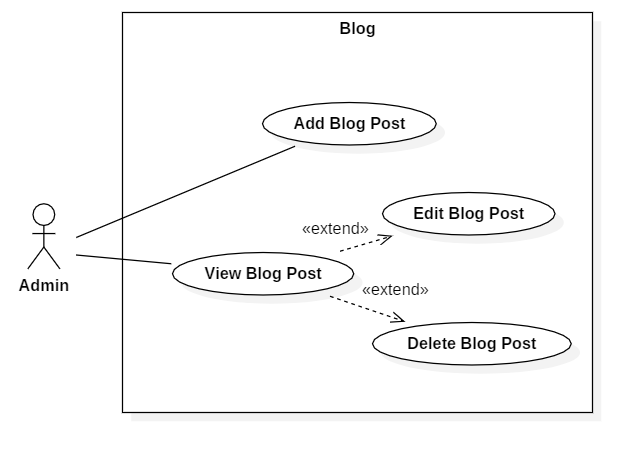


Figure : Blog use case

|  |  |
| --- | --- |
| **Justification** | The following use case depicts the flow of how an admin adds, views, edits and deletes blog posts. |
| **Primary actor(s)** | Admin |
| **Secondary actors(s)** | *NA* |
| **Primary flow** | 1. The admin adds blog post to the system dashboard making it possible for normal learners to view. 2. Once the blog post has been added, the admin has the authority to edit or delete an existing blog post. |
| **Alternative flow** | 1. While adding or editing blog post, the admin might input invalid data into the field. For this, via proper validations, the admin is to be asked to fill valid data. |

## Issues and Feedbacks

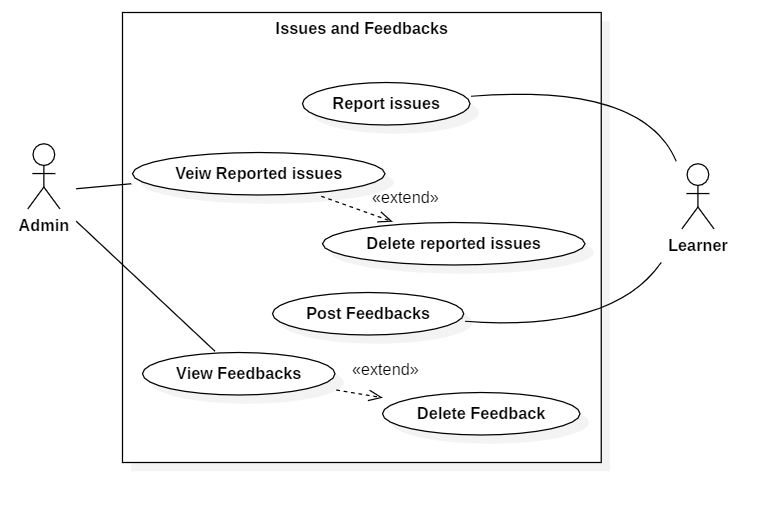


Figure : Issues and Feedbacks

|  |  |
| --- | --- |
| **Justification** | The following use case depicts the flow of how a user reports issues and post feedbacks for the admin to view and examine them. |
| **Primary actor(s)** | Admin/Learners |
| **Secondary actors(s)** | *NA* |
| **Primary flow** | 1. Normal users initially report some issues or bugs and post feedbacks for the admin to view. 2. The admin can view reported issues and posted feedbacks from normal learner users. 3. The viewed issues and feedbacks can be deleted upon rectification. |
| **Alternative flow** | 1. While reporting or posting issues and feedbacks, the user might input invalid data into the field. For this, via proper validations, the user is to be asked to fill valid information. |

# Initial Class Diagram via NLA

## Project scenario

Interactive tutorials platform, an interactive web-based application provides a viable platform for learners yearning to learn techniques and skill contents necessary for programming. The platform includes a widespread pool of features packed with course materials. The contents provided in this platform promises to deliver up to date course resources with high quality evaluation contents such as quiz and competitions.

Furthermore, it provides its learners with additional features such as leaving comments and queries on certain course topic, posting possible feedbacks and issues related to specific course content etc. The user initially creates a learner profile to access the available contents where the user is asked to provide their name, email, DoB, username and a password. The learner is facilitated with an option to alter the status as in mark a course as read or unread. Likewise, adding, upgrading and deleting courses is performed by respective admin members. The learners are updated via blogs posted by admins about new changes and additions.

## Generating class diagram via NLA

Primarily, we begin with listing out potential candidate classes which generally poses as nouns. In case of this project, we consult the scenario mentioned above.

Potential candidate classes (Nouns):

Tutorial, Web, Application, Platform, Technique**,** Skill, Content, Course, Pool, Features, Quiz, Comments, Queries, Feedback, Issues, Profile, Admin, Member, etc.

Filtering the above potential candidate classes, we end up with final candidate classes.

|  |
| --- |
| **Final candidate class** |
| Blog |
| Comment  User |
| User Type |
| Course |
| Issue |
| Feedback |
|  |

### Potential candidate for attributes (Adjective):

Interactive application, course status, admin members, course content, new course, possible feedback, possible issues, updated blog, User name, User DoB, User email, User Password.

Filtering the above potential attribute candidates, we end up with following attributes.

|  |
| --- |
| **Final candidate attributes** |
| User name |
| Member type (user)  User DoB, |
| User email,  User Password |
| Available Course  Course content |
| Possible Issue |
| Possible Feedback |
| Updated Blog  Course status |

### Potential candidates for behavior (Verbs):

Add, update, delete, leave, provide, yearn, include, deliver, post, ask, create, make.

Filtering the potential behaviors listed above, we end up with following behaviors.

|  |
| --- |
| **Final candidate methods** |
| Add |
| Update  Delete |
| Mark |
| Post |
| Create |

# Generated initial Class diagram

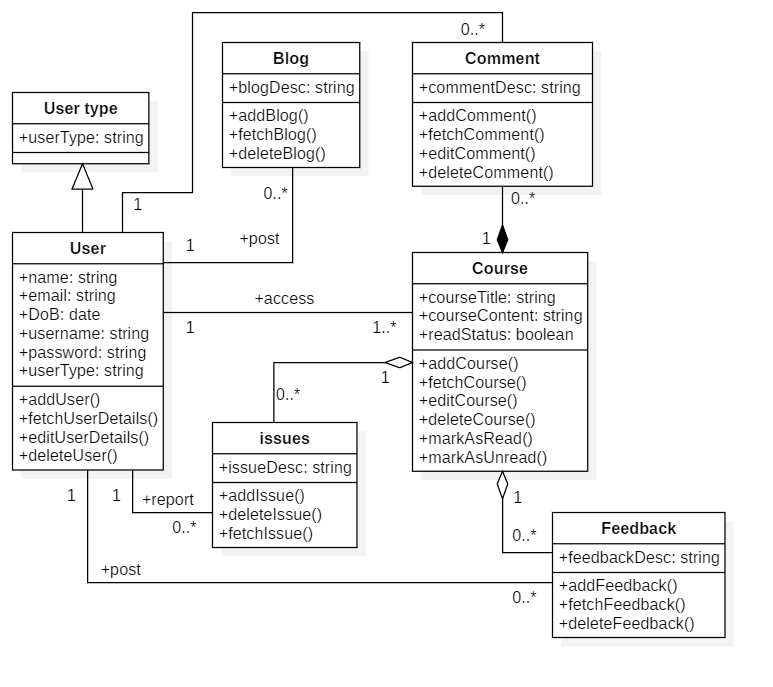


Figure : Initial Class diagram