1. **Problem Statement**

* A software has to be developed that not only allows you to plan your day but also provides you feedback on how productive you were throughout the day, week or month.
* Noteffy solves this problem by providing following functionalities:

1. Note : It allows the user to quickly note down anything important anytime he/she wants.
2. Tasks : It allows the users to plan their day ahead of time. It also allows the users to set an alarm for the task and receive a notification at the right time.
3. To-do : Tasks for the current day are converted into a checklist so that the user can mark the tasks as they are completed.
4. Kudos : This function gives the user a graphical representation of how many tasks were completed by the user in a month and helps the user keep track of their productivity.
5. **PROCESS MODEL**

The selection of a process model for a project is based on the nature of the application, its requirements, technical knowledge of the development team and the users. The model used to build the Noteffy website is the Incremental Model.

This model is based on linear process flow where the output is produced in a number of deliverables. It is considered because there is a need to provide a limited set of software functionality to users and it can be refined and expanded in later software releases.

In this case, the first increment will be considered as the core product i.e., the basic requirements will be fulfilled but many supplementary features remain undelivered. The core product which is used by the customer undergoes detailed evaluation as a result of which a plan will be developed for the next increment. This plan will consist of the modification of the core product to meet the customer needs and deliver more functionality. Prototyping model can also be used in this case.

1. **USE CASE DIAGRAM**

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. A key concept of use case modeling is that it helps us design a system from the end user's perspective.

**NOTEFFY USE CASES :**

1. **SIGN UP:**
   1. **Actors** :- user
   2. **Introduction** :- This use case documents the procedure that register’s the user into the system .
   3. **Special requirements** :- none
   4. **Basic Flow** : -
      1. User is asked to enter username, email and password
      2. Server validates the email by sending an OTP to the user and the user is asked to enter the OTP.
      3. If valid OTP is entered by the user then the user’s details are stored in a JSON file and the user is also signed into his/her account.
   5. **Alternate Flow:-**
      1. If invalid OTP is entered by the user then the user is asked to re-enter their details.
   6. **Pre - Conditions :-** User should be signed into the account.
   7. **Post – Conditions :-** JSON file is updated
   8. **Related use case :**- sign up
2. **LOGIN :**
   1. **Actors :-** User
   2. **Introduction :**- This use case documents the procedure that allows a user to log in to their account.
   3. **Special Requirements :**- none
   4. **Basic Flow :**-
      1. User is asked to enter username and password.
      2. Server checks whether the username and password match the details stored in the JSON file.
      3. If a match is found then the user Is logged into their account
      4. username and user-number of the user is stored as a cookie on the browser.
   5. **Alternate Flow :-**
      1. Invalid username or password is entered by the user
      2. User is redirected to the sign - up page
   6. **Pre – Conditions :-** User has already registered with Noteffy
   7. **Post – Conditions :-**  User is logged into their account.
   8. **Related use cases :-** sign - up
3. **COMPOSE NOTE :**
   1. **Actors :-** User
   2. **Introduction :**- This use case documents the procedure that allows a user to write and store a note.
   3. **Special Requirements :**- none
   4. **Basic Flow :-**
      1. User enters the title of the note and content of the note.
      2. The system stores all these details in JSON file
   5. **Alternate Flow :-** none
   6. **Pre –** **Conditions : -** User should be logged into his/her account.
   7. **Post –** **Conditions :-** JSON file is updated
   8. **Related use cases :-** none
4. **DELETE NOTE:**
   1. **Actors :**- User
   2. **Introduction :**- This use case documents the procedure that allows a user to delete a particular note.
   3. **Special Requirements :-** none.
   4. **Basic Flow :-**
      1. User selects the note to be deleted.
      2. Server takes the note number of that note , deletes the note and updates the JSON file.
   5. **Alternate Flow :-**
      1. Note number is not found then error page is loaded
   6. **Pre – Conditions :-** User should be logged into his/her account.
   7. **Post –** **Conditions :-** JSON file is updated
   8. **Related use cases :**- none.
5. **UPDATE NOTE :**
6. **Actors :**- User
7. **Introduction :**- This use case documents the procedure that allows a user to update a particular note.
8. **Special Requirements :-** none.
9. **Basic Flow :-**
   * 1. User selects the note to be updated.
     2. Server takes the note number of that note fetches the details of the note and allows the user to make changes to those details.
     3. Once the user makes changes to the note the server updates the JSON file to reflect those changes.
10. **Alternate Flow :-**
    * 1. Note number is not found then error page is loaded
11. **Pre – Conditions :-** User should be logged into his/her account.
12. **Post –** **Conditions :-** JSON file is updated
13. **Related use cases :**- none.
14. **COMPOSE TASK :**
    1. **Actors :-** User
    2. **Introduction :**- This use case documents the procedure that allows a user to write and store a task.
    3. **Special Requirements :**- none
    4. **Basic Flow :-**
       1. User enters the title of the task, date and time at which the user would like to be notified and the content of the task.
       2. The system stores all these details in JSON file.
    5. **Alternate Flow :-** none
    6. **Pre –** **Conditions : -** User should be logged into his/her account.
    7. **Post –** **Conditions :-** JSON file is updated
    8. **Related use cases :-** none.
15. **DELETE TASK**
    1. **Actors :**- User
    2. **Introduction :**- This use case documents the procedure that allows a user to delete a particular task.
    3. **Special Requirements :-** none.
    4. **Basic Flow :-**
       1. User selects the task to be deleted.
       2. Server takes the task number of that task , deletes the note and updates the JSON file.
    5. **Alternate Flow :-**
       1. Task number is not found then error page is loaded
    6. **Pre – Conditions :-** User should be logged into his/her account.
    7. **Post –** **Conditions :-** JSON file is updated
    8. **Related use cases :**- none.
16. **UPDATE TASK :**
    1. **Actors :**- User
    2. **Introduction :**- This use case documents the procedure that allows a user to update a particular task.
    3. **Special Requirements :-** none.
    4. **Basic Flow :-**
       1. User selects the task to be updated.
       2. Server takes the task number of that task fetches the details of the task and allows the user to make changes to those details.
       3. Once the user makes changes to the note the server updates the JSON file to reflect those changes.
    5. **Alternate Flow :-**
       1. Task number is not found then error page is loaded
    6. **Pre – Conditions :-** User should be logged into his/her account.
    7. **Post –** **Conditions :-** JSON file is updated
    8. **Related use cases :**- none.

**Noteffy**

**Sign - Up**

**Login**

**Delete a Note**

**Delete a task**

**Compose a Note**

**Update a Note**

**Update a task**

**Compose a task**

1. **SEQUENCE DIAGRAMS**

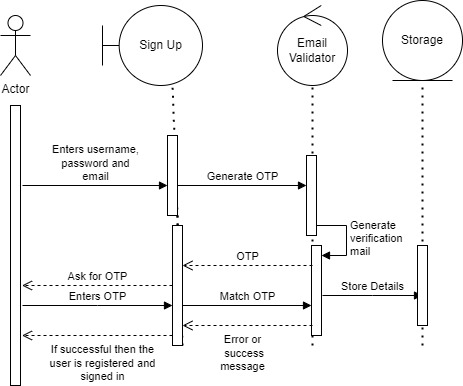
The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. Purpose of a Sequence Diagram

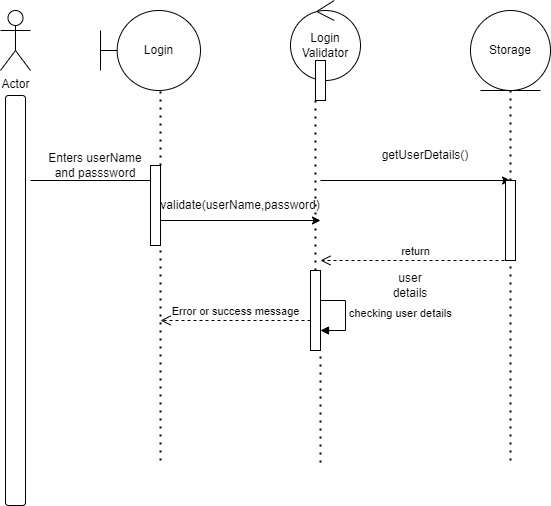
**1**. To model high-level interaction among active objects within a system.

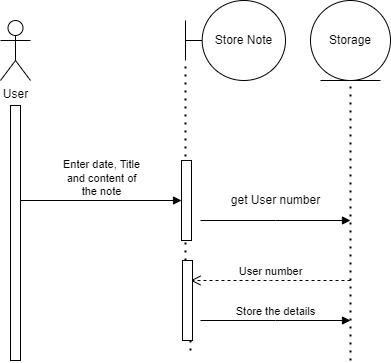
2. To model interaction among objects inside a collaboration realizing a use case.

3. It either models generic interactions or some certain instances of interaction

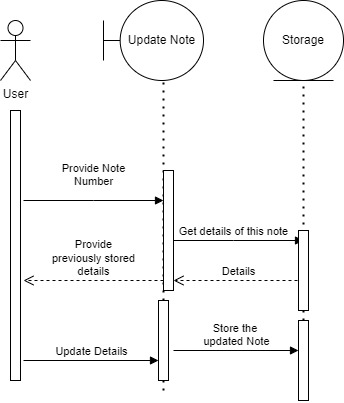
1. SIGN-UP :



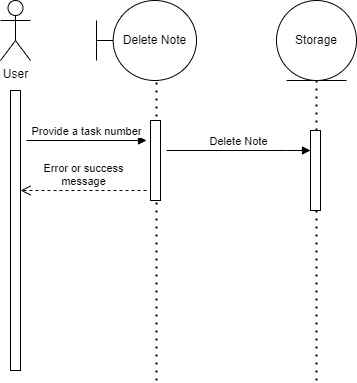
1. LOGIN :
2. COMPOSE NOTE:



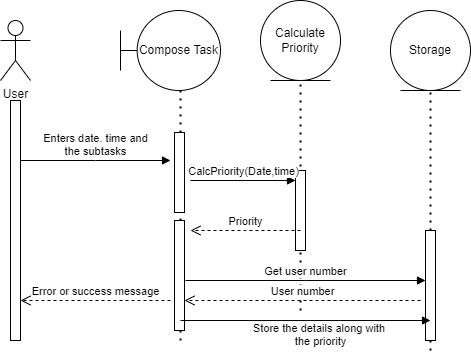
1. UPDATE NOTE :



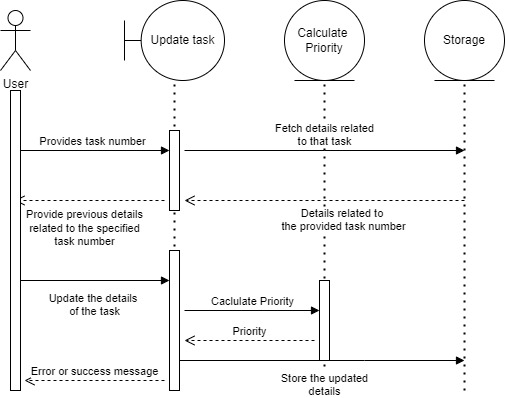
1. DELETE NOTE :



1. COMPOSE TASK :



1. UPDATE TASK :



1. DELETE TASK :

