

# **SOFTWARE REQUIREMENT ANALYSIS**

**Project Name: Noteffy**

**Team Members:**

**Gaurang Tyagi (21058570017)**

**Ravish Ranjan (21058570040)**

**Rishi Mahajan (21058570043)**

**Shivang Shukla (21058570054)**

**Noteffy Project Link**

**Github link**

**<https://github.com/Ravish-Ranjan/Noteffy>**



## A.) 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.

## B.) 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.

## C.) 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:

- a. **Actors** :- user
- b. **Introduction** :- This use case documents the procedure that register's the user into the system .
- c. **Special requirements** :- none
- d. **Basic Flow** : -
  - i. User is asked to enter username, email and password
  - ii. Server validates the email by sending an OTP to the user and the user is asked to enter the OTP.

- iii. 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.

**e. Alternate Flow:-**

- i. If invalid OTP is entered by the user then the user is asked to re-enter their details.

f. **Pre - Conditions :-** User should be signed into the account.

g. **Post – Conditions :-** JSON file is updated

h. **Related use case :-** sign up

## 2.) LOGIN :

a. **Actors :-** User

b. **Introduction :-** This use case documents the procedure that allows a user to log in to their account.

c. **Special Requirements :-** none

d. **Basic Flow :-**

- i. User is asked to enter username and password.
- ii. Server checks whether the username and password match the details stored in the JSON file.
- iii. If a match is found then the user is logged into their account
- iv. username and user-number of the user is stored as a cookie on the browser.

e. **Alternate Flow :-**

- i. Invalid username or password is entered by the user
- ii. User is redirected to the sign - up page

f. **Pre – Conditions :-** User has already registered with Noteffy

g. **Post – Conditions :-** User is logged into their account.

h. **Related use cases :-** sign - up

## 3.) COMPOSE NOTE :

a. **Actors :-** User

b. **Introduction :-** This use case documents the procedure that allows a user to write and store a note.

- c. **Special Requirements** :- none
- d. **Basic Flow** :-
  - i. User enters the title of the note and content of the note.
  - ii. The system stores all these details in JSON file
- e. **Alternate Flow** :- none
- f. **Pre – Conditions** : - User should be logged into his/her account.
- g. **Post – Conditions** :- JSON file is updated
- h. **Related use cases** :- none

#### 4.) DELETE NOTE:

- a. **Actors** :- User
- b. **Introduction** :- This use case documents the procedure that allows a user to delete a particular note.
- c. **Special Requirements** :- none.
- d. **Basic Flow** :-
  - i. User selects the note to be deleted.
  - ii. Server takes the note number of that note , deletes the note and updates the JSON file.
- e. **Alternate Flow** :-
  - i. Note number is not found then error page is loaded
- f. **Pre – Conditions** :- User should be logged into his/her account.
- g. **Post – Conditions** :- JSON file is updated
- h. **Related use cases** :- none.

#### 5.) UPDATE NOTE :

- a. **Actors** :- User
- b. **Introduction** :- This use case documents the procedure that allows a user to update a particular note.
- c. **Special Requirements** :- none.
- d. **Basic Flow** :-
  - i. User selects the note to be updated.
  - ii. Server takes the note number of that note fetches the details of the note and allows the user to make changes to those details.

- iii. Once the user makes changes to the note the server updates the JSON file to reflect those changes.
- e. **Alternate Flow :-**
  - iv. Note number is not found then error page is loaded
- f. **Pre – Conditions :-** User should be logged into his/her account.
- g. **Post – Conditions :-** JSON file is updated
- h. **Related use cases :-** none.

## 6.) COMPOSE TASK :

- a. **Actors :-** User
- b. **Introduction :-** This use case documents the procedure that allows a user to write and store a task.
- c. **Special Requirements :-** none
- d. **Basic Flow :-**
  - i. 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.
  - ii. The system stores all these details in JSON file.
- e. **Alternate Flow :-** none
- f. **Pre – Conditions :-** User should be logged into his/her account.
- g. **Post – Conditions :-** JSON file is updated
- h. **Related use cases :-** none.

## 7.) DELETE TASK

- a. **Actors :-** User
- b. **Introduction :-** This use case documents the procedure that allows a user to delete a particular task.
- c. **Special Requirements :-** none.
- d. **Basic Flow :-**
  - i. User selects the task to be deleted.
  - ii. Server takes the task number of that task , deletes the note and updates the JSON file.
- e. **Alternate Flow :-**
  - i. Task number is not found then error page is loaded
- f. **Pre – Conditions :-** User should be logged into his/her account.
- g. **Post – Conditions :-** JSON file is updated

**h. Related use cases :-** none.

## **8.) UPDATE TASK :**

**a. Actors :-** User

**b. Introduction :-** This use case documents the procedure that allows a user to update a particular task.

**c. Special Requirements :-** none.

**d. Basic Flow :-**

- i. User selects the task to be updated.
- ii. Server takes the task number of that task fetches the details of the task and allows the user to make changes to those details.
- iii. Once the user makes changes to the note the server updates the JSON file to reflect those changes.

**e. Alternate Flow :-**

- i. Task number is not found then error page is loaded

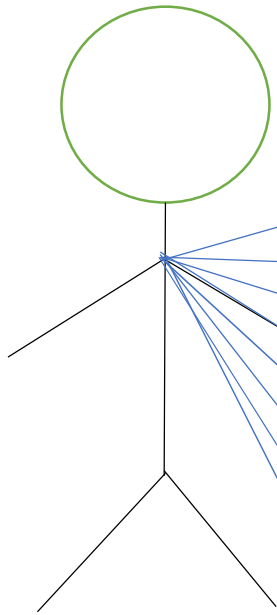
**f. Pre – Conditions :-** User should be logged into his/her account.

**g. Post – Conditions :-** JSON file is updated

**h. Related use cases :-** none.



# Noteffy



**Login**

**Sign - Up**

**Compose a task**

**Update a task**

**Delete a task**

**Compose a Note**

**Update a Note**

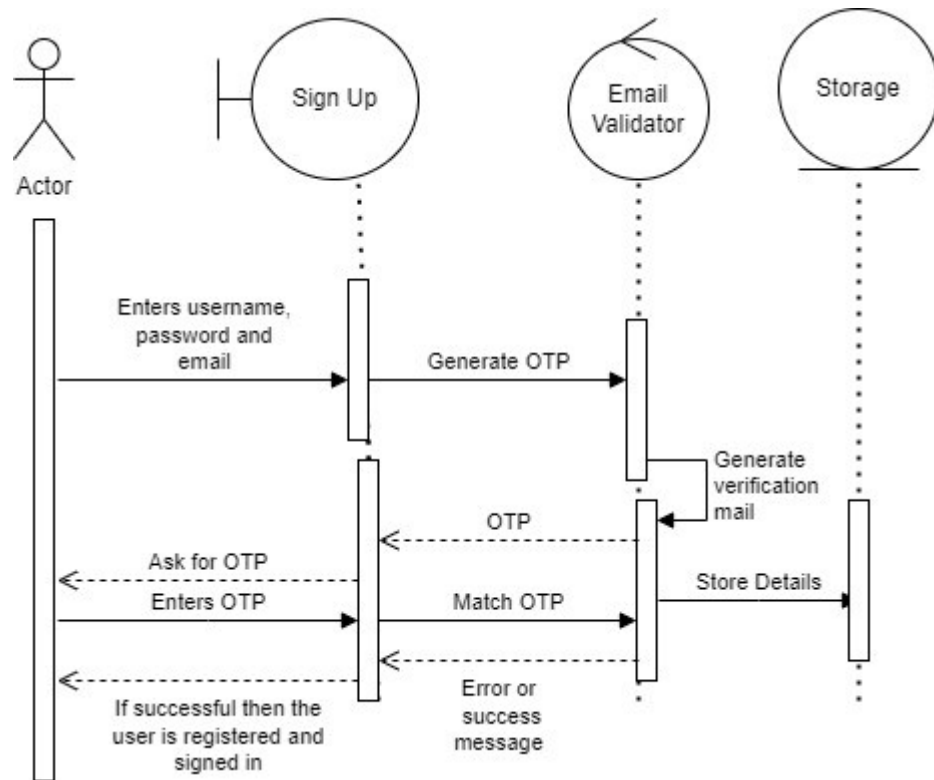
**Delete a Note**

## **D.)SEQUENCE DIAGRAMS**

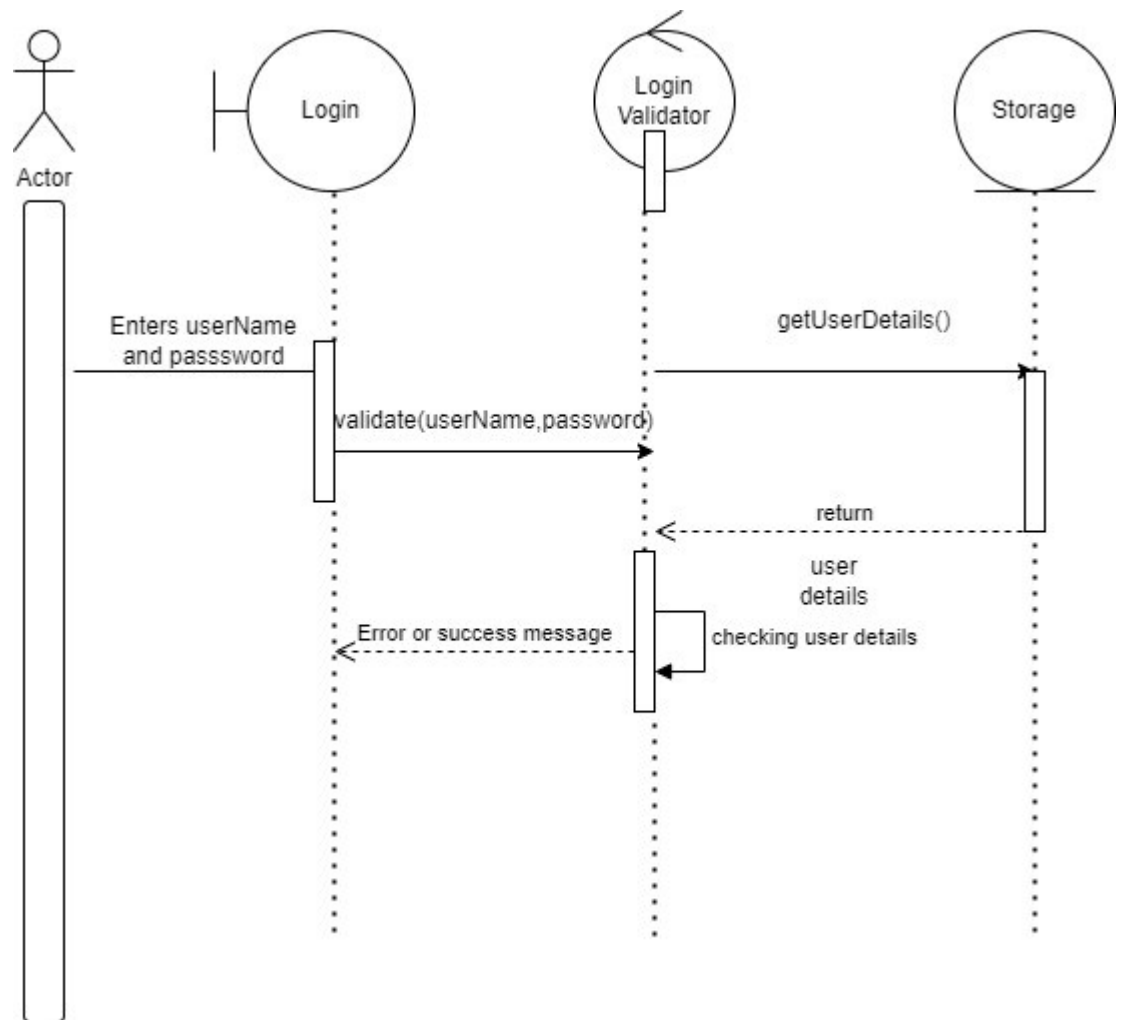
The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time.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

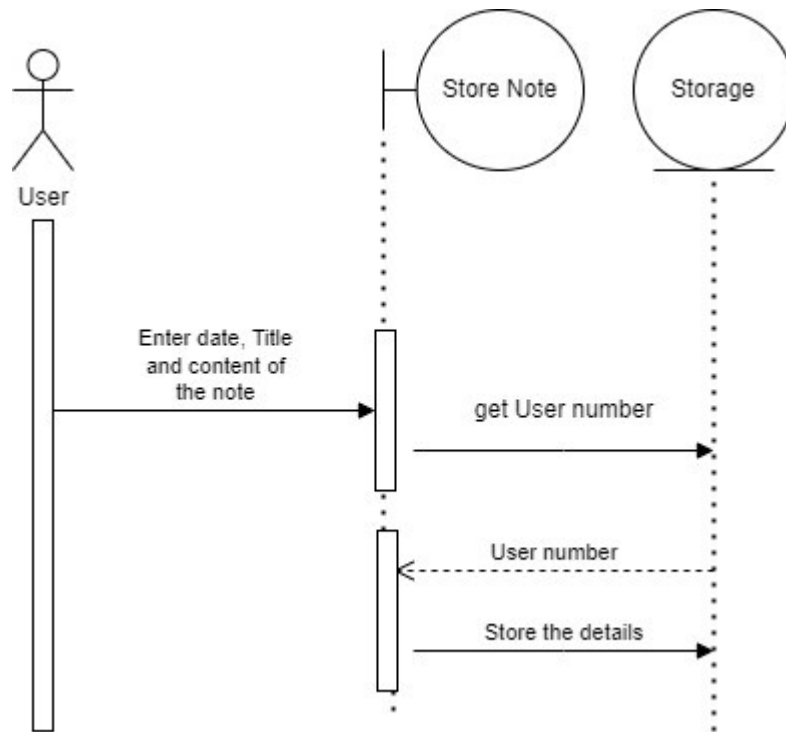
**1.) SIGN-UP :**



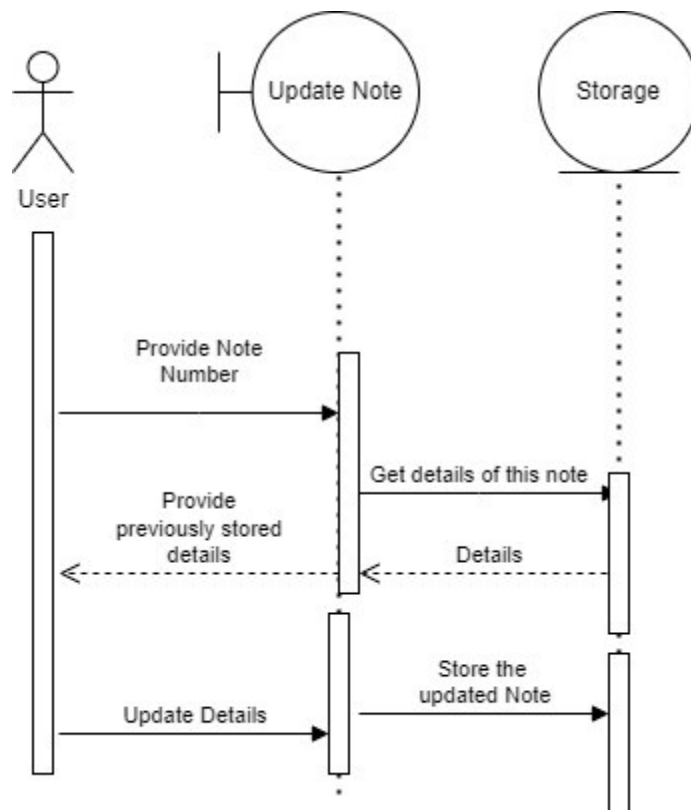
## 2.) LOGIN :



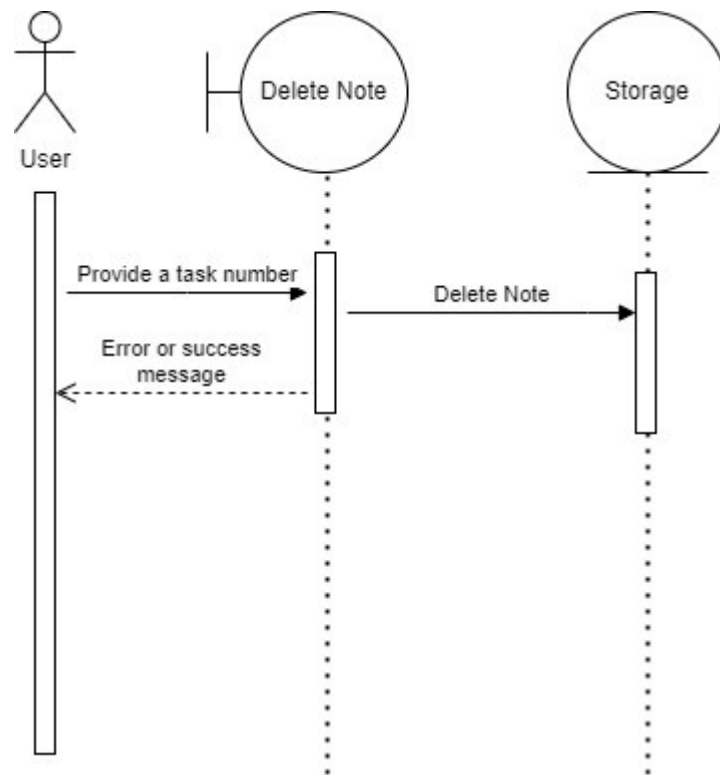
## 3.) COMPOSE NOTE:



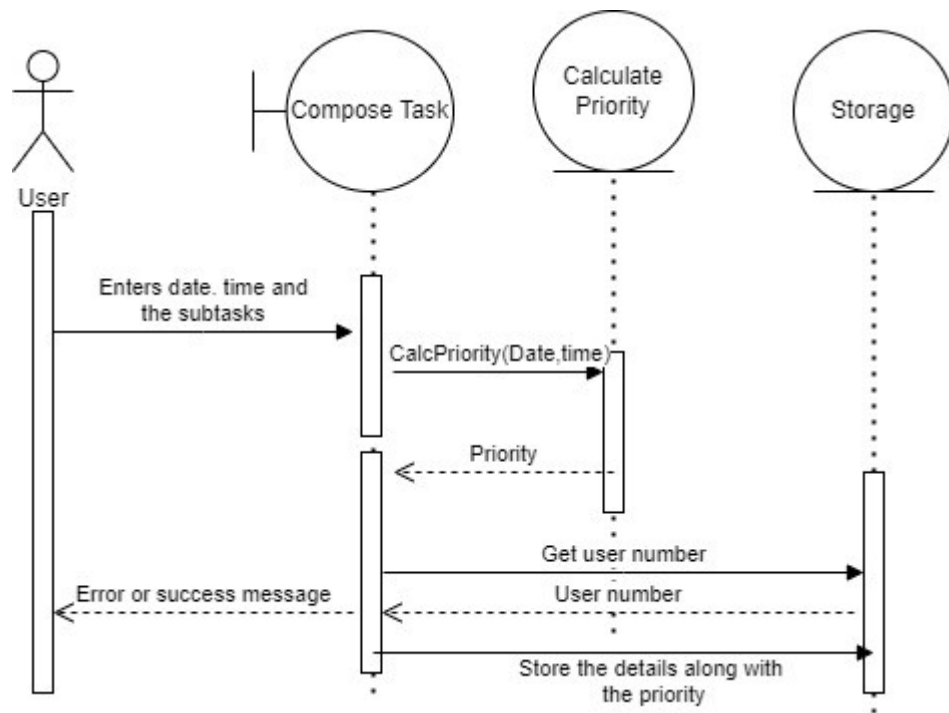
#### 4.) UPDATE NOTE :



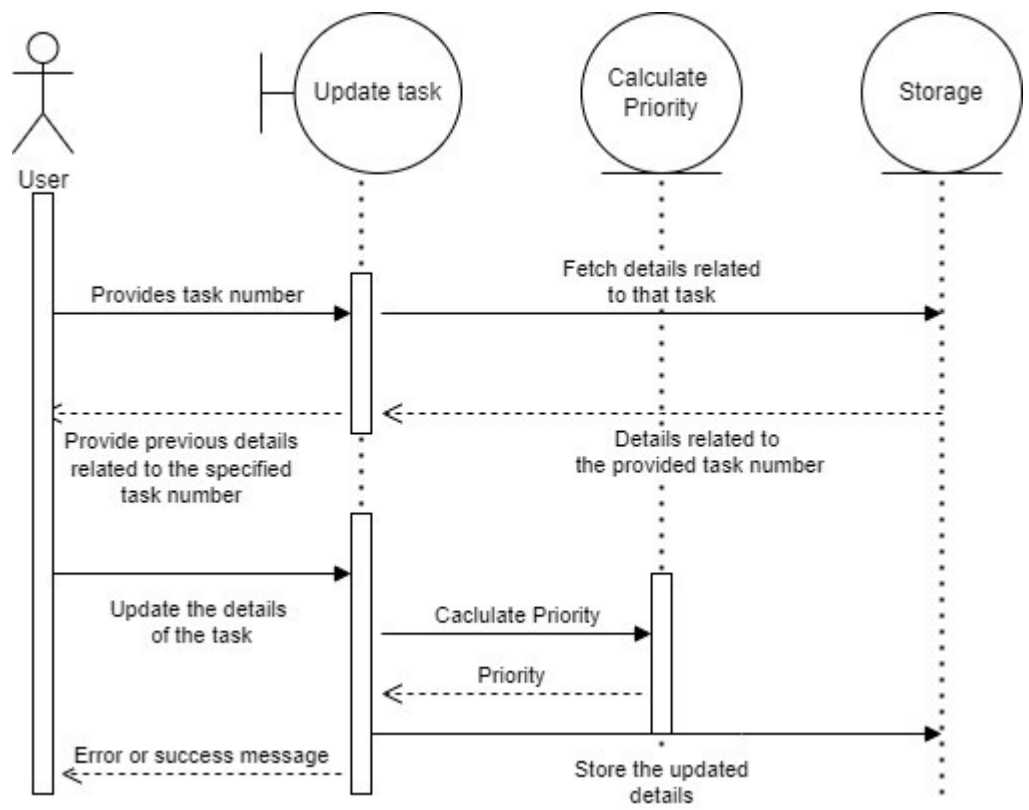
### 5.) DELETE NOTE :



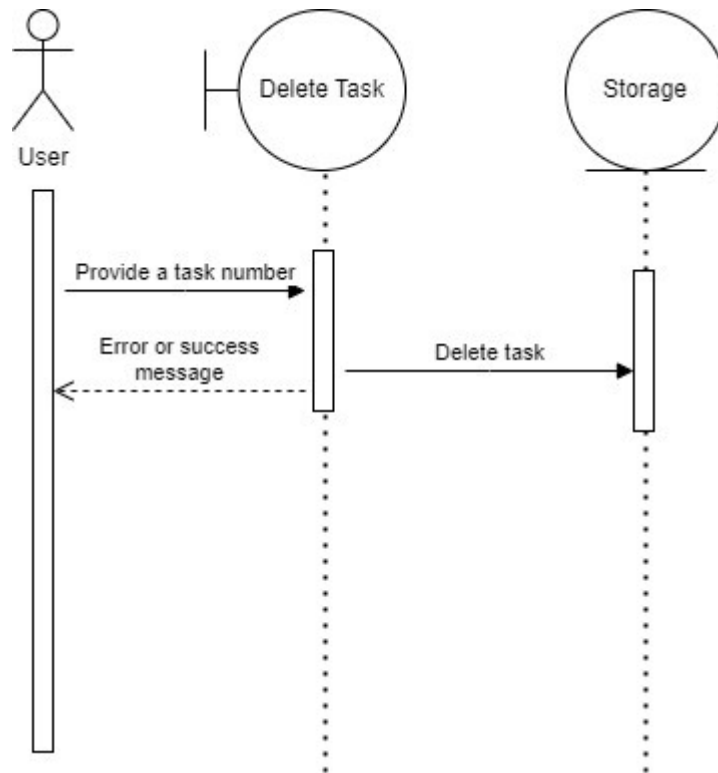
### 6.) COMPOSE TASK :



## 7.) UPDATE TASK :



## 8.) DELETE TASK :





The primary function of requirement analysis is that it translates the ideas in the mind of the clients into a formal document. Thus, the output of this phase is a set of precisely specified requirements that are complete and consistent. This document is called Software Requirement Specification.

## **1) Software Requirements Specification**

The following document provides complete and consistent software requirements for Noteffy, a productivity management website. The final product should provide the requirements mentioned in the following document and a formal change request will need to be raised if it requires some additional features, and a new release of this document and product will be produced.

### **1.1) Purpose :**

Today we are so busy in our lives that we don't pay attention to the work and tasks we were assigned or we are so procrastinating that we delay our work on and on.

To handle this our project will help you to out of this. Here you can assign yourself some task give them priorities and we will make sure that you we get it done on time. To make sure you remember your tasks we will notify you based on your task your tasks priority and never let you forget your work. You can give your tasks high priority if you got to complete the task asap or low priority for not so urgent once. We will **Noteffy** you as you like.

We also maintain a record of tasks you complete on/before the time and those you don't to give you some social credits and we will share it with you at the end of the month for your self-analysis.

Along with this you can also maintain your normal notes when you just got to write something down or make yourself a shopping list and much more.

The objective is to build a website that meets the following criteria:

- a.) Quickly Accessible – The user should be able to access the website as quickly as possible.

b.) Simple – The website should be simple to use for every user.

## 1.2) Scope :

Noteffy is a productivity management website that allows users to keep track of their productivity by allowing them to plan their days ahead of time. It also allows users to maintain notes anytime and anywhere they want.

Noteffy can be accessed anywhere and by anyone like students, teachers, clerks, software developer or casual users etc.

It can be used in colleges, schools, offices, in homes, etc.

## 1.3) Definitions, Acronyms, Abbreviations :

1. Definitions – none
2. Acronyms –
  - a. JSON : JavaScript Object Notation
  - b. HTML : Hypertext Markup Language
  - c. PHP : Hypertext Preprocessor
  - d. CSS : Cascading Stylesheets
3. Abbreviations – Zombie task :tasks which have passed there deadline but are not completed.

## 1.4) References :

- 1.) DataFlowDiagram.pptx : Data flow diagram for the project.

## 1.5) Overview :

The rest of the SRS includes the following:

- 1.) Overall description of the requirements for Noteffy
- 2.) Specific requirements for Noteffy.

## **2. General Requirements**

### **2.1) Product Perspective**

#### **2.1.1) System Interfaces:**

- **Noteffy is a completely self-contained software that does not belong to any larger ecosystem of products**

#### **2.1.2) User Interfaces:**

- **Users interface with Noteffy through a website:**
  - New users register their details through the authentication page described in section 3.1
  - Recurring users register their details through a personal dashboard page described in section 3.1
  - Recurring users can view their monthly productivity charts and other statistics through a personal scoreboard page described in section 3.1

#### **2.1.3)Hardware Interfaces**

- **OS: Windows 7 / 8/ 10 , Android**
- **Monitor: At least 1280x800 pixels in 256 colours.**
- **A mouse or other pointing device & a keyboard**
- **Active internet connection**

#### 2.1.4) Software Interfaces

- **Visual Studio Code:** IDE developed by Microsoft, used for making this web-app.
- **WAMP server:** PHP based local server for hosting the web-app.
- **JS:** JavaScript (ES13) used for dynamic updation, controlling multimedia, UI and API communication. The libraries used were the standard library with JSON and fetch support and the chart API.
- **PHP:** Hypertext Preprocessor used for developing the main back-end logic. Only the standard library was used.
- **Python:** Python Programming languages used for mailing system. Libraries used will be the standard library and smtplib(for mailing).
- **JSON:** JavaScript Object Notation used for storing data.
- **HTML5:** Hypertext Markup Language used for laying the basic foundation of the UI of the web-app.
- **CSS6:** Cascading Style Sheet used for styling the elements rendered by HTML.

#### 2.1.5) Communication Interfaces

- To implement the notification functionality, the mailing API in python will use the Gmail SMTP relay server through the smtplib library.

#### 2.1.6) Memory constraints

- No such constraint

### 2.1.7) Operations

- **There are 2 modes of operation for the user:**
  - New users(with no cached data) interact through authentication page
  - Returning users(with cached data) interact with the dashboard and create, delete and edit the notes and tasks that they make.
  - Returning users also get to complete and edit to-do lists and view their monthly productivity charts.

### 2.1.8) Site adaptation requirements

- **No such constraint**

## **2.2) Product Functions**

The functional requirements of this web-application are:-

- **Register new users.**
- **Record the Notes/Tasks created by the user.**
- **Convert a task into a to-do list over a time-bound constraint.**
- **Generate notification alert of current tasks/to-do to be completed by the user.**

- **Generate various statistical reports based on productivity of the user.**

## **2.3) User characteristics**

The ideal users for the web-application are:-

- **Casual users looking to increase their productivity and have better time management**
- **Professionals seeking to better schedule their workload and keep track of it across platforms (mobile phone, workstations, etc.)**
- **Young to medium aged users who want to socially interact and positively compete with their friends, colleagues or acquaintances online on the basis of productivity**

## **2.4) Constraints**

- **The product shall meet accessibility standards to make the application inclusive for all potential user groups and the developer agnosticism to make the development process robust and sustainable.**
- **The user database has to be securely encrypted to prevent data breach and violation of personal privacy.**
- **System criticality is medium level as only email id is provided by the user that can be misused but safety parameters have still been put in place**

## **2.5) Assumptions and Dependencies**

- The user database has to be securely encrypted to prevent data breach and violation of personal privacy.
- The application interacts with cronjob api which is only compatible with Linux operating system. For windows OS, correlating api would be task scheduler

## **2.6) Apportioning of Requirements**

- The quirky theme has to be added to workspace and dashboard segments in order the maintain UI consistency.
- OTP functionality has to be reorganized to ensure user data security
- Graveyard and administrative panel modules have to be constructed to store expired tasks and provide control and maintenance functions.
- The web application is not responsive and so behaves visually as intended on desktop environments.

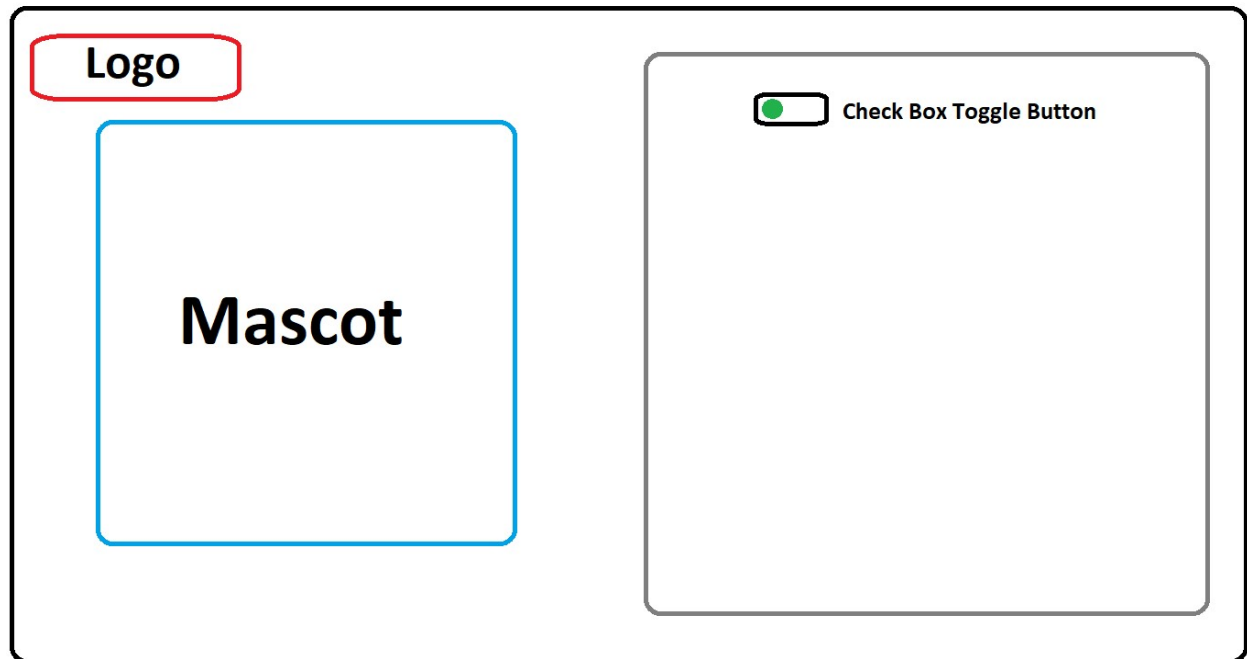
### **3. Specific Requirements :**

#### **3.1) External Interfaces**

User Interfaces

- 1) User Log in / Sign up



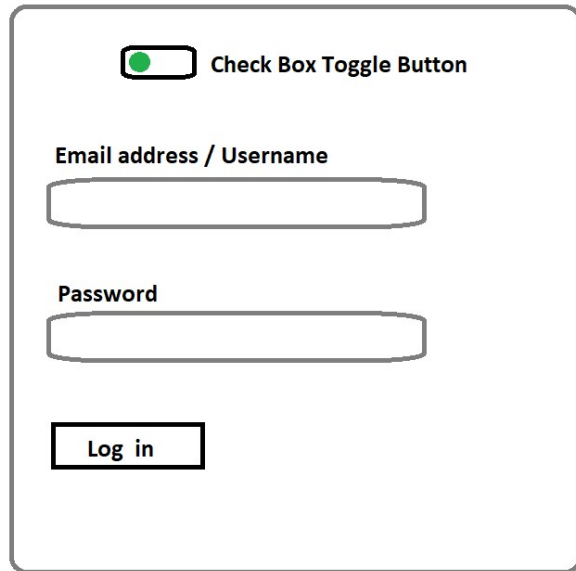


(Fig 1)

Description of each UI Button:-

- **Check Box Toggle Button:** Common div in HTML to switch between Sign in & Sign up.

*IF check Box value == 0, the div displays Sign in parameters.*



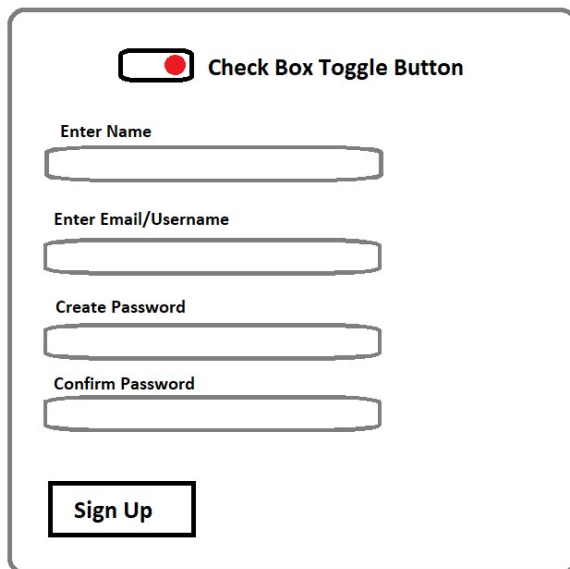
A login form with a rounded rectangular border. At the top left is a 'Check Box Toggle Button' with a green circle. Below it are two input fields: 'Email address / Username' and 'Password'. At the bottom is a 'Log in' button.

(Fig 2)

Description of each UI Buttons:

- **Email address/Username:** User can enter his/her registered username or email address in the input field.
- **Password:** User can enter his/her registered password in the input field.
- **Log in:** Logs the user into Noteffy.

*IF check Box value == 1, the div displays Sign up parameters.*



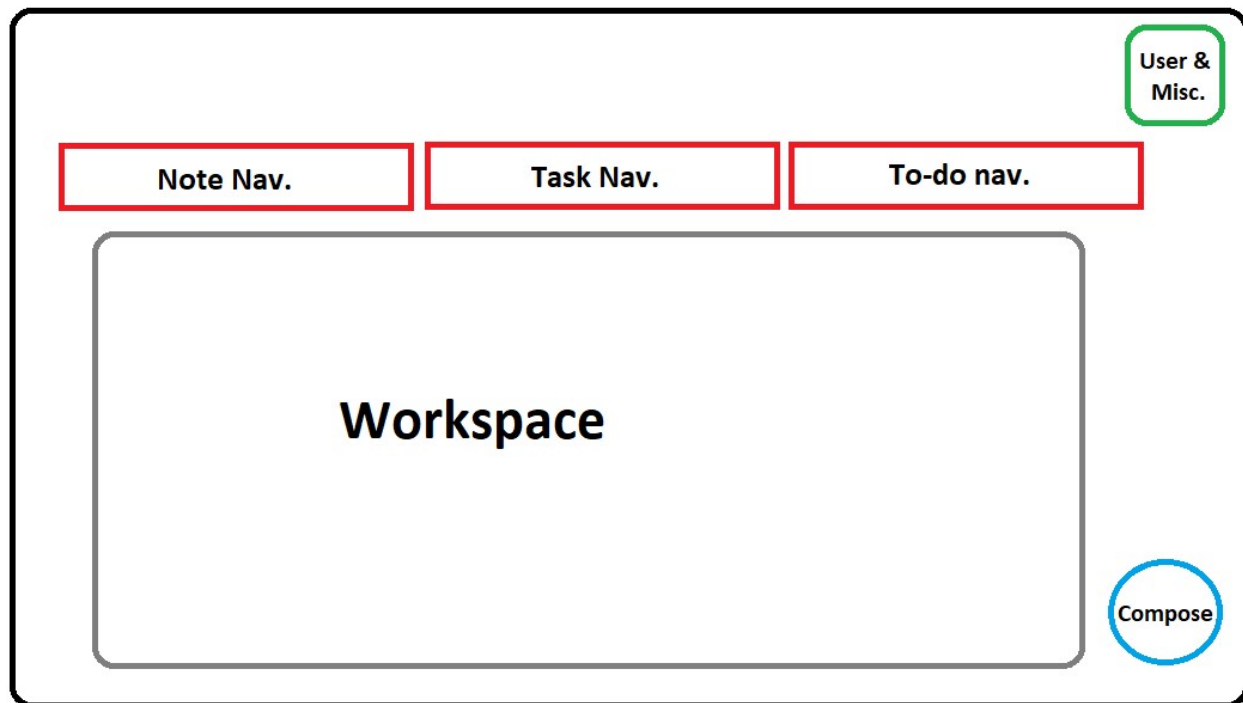
A sign-up form with a rounded rectangular border. At the top left is a 'Check Box Toggle Button' with a red circle. Below it are four input fields: 'Enter Name', 'Enter Email/Username', 'Create Password', and 'Confirm Password'. At the bottom is a 'Sign Up' button.

(Fig 3)

Description of each UI Buttons:

- **Email name:** User can enter his/her name.
- **Enter Email/Username:** User can enter his/her email or desired username.
- **Create Password:** User can create his/her desired password.
- **Confirm Password:** Input field to confirm the desired password set by user.
- **Sign Up:** Registers the user into the database.

## 2) User Workspace

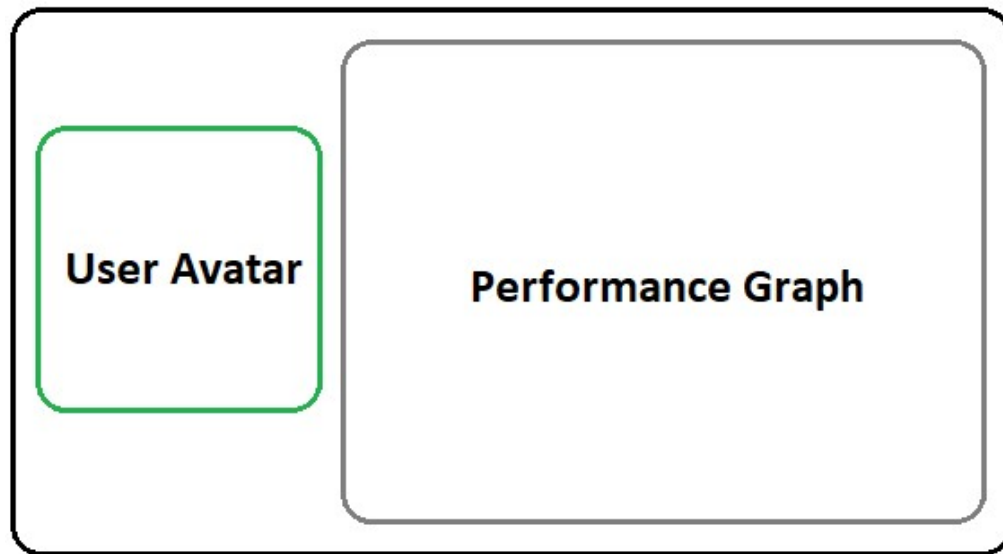


(Fig 4)

Description of each UI Buttons:-

- **Compose:** To compose a new Note/Task/To-do based on current workspace selected.
- **Note Navigation Button:** To show current notes saved by user.
- **Tasks Navigation Button:** To show current tasks set by user.
- **To-do Navigation Button:** To show to-do tasks to be done by user.
- **User & Miscellaneous-** Button to display user details, Completion statistics, Settings, etc.

### 3) Score Board



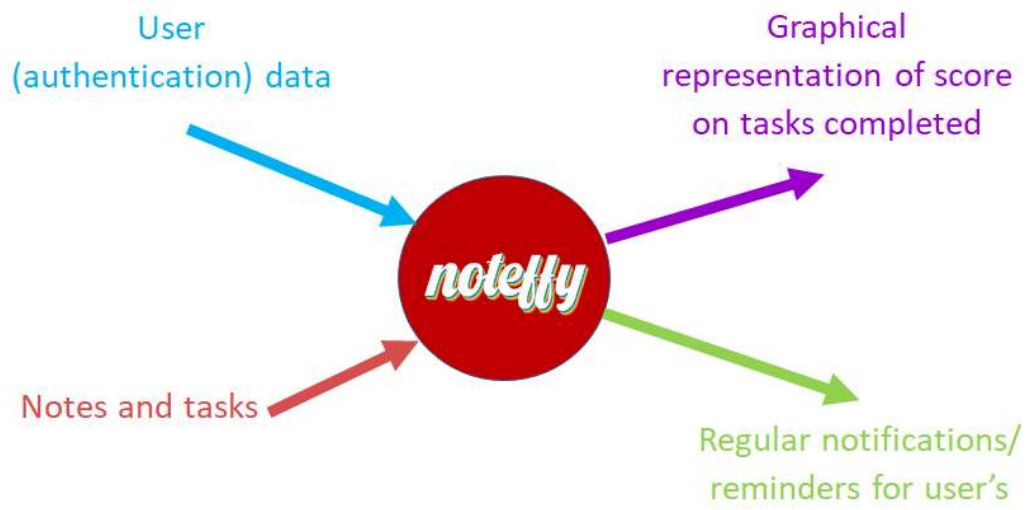
(Fig 5)

### 3.2) Functions:

The functional requirements of this web-application are:-

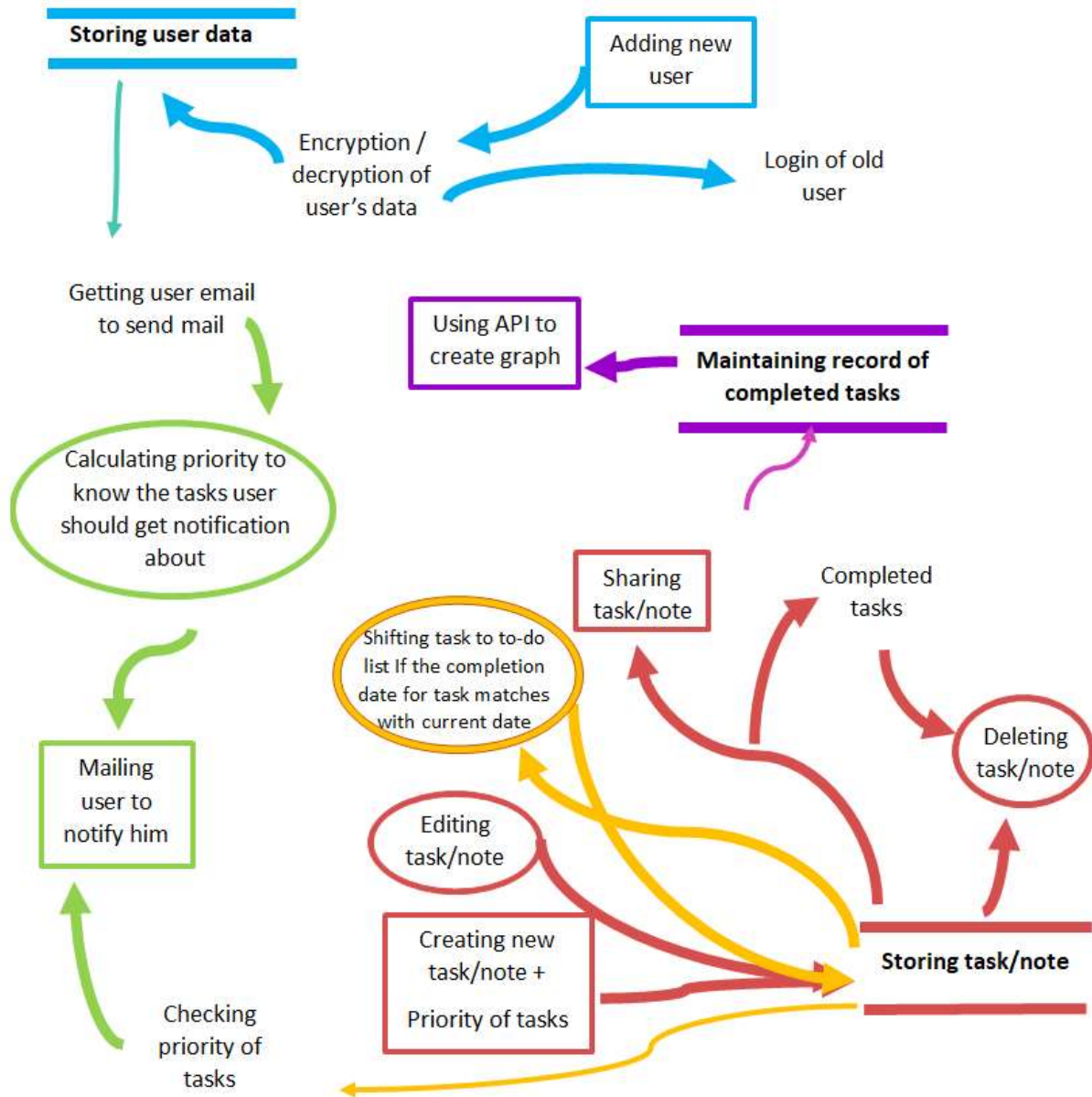
#### **3.2.1) Data Flow Diagram (DFD)**

##### **Level 0**



(Fig 6)

## Level 1



(Fig 7)

### 3.2.1) Registration

The system shall register new users.

#### 3.2.1.1) Entities

User Data

#### 3.2.1.2) Processes

Registration

Encrypt/Decrypt

#### 3.2.1.3) Topology



(Fig 8)

### 3.2.2) Note/Task Management

The system shall store/edit/create note/task made by the user

#### 3.2.2.1) Creation Module

##### 3.2.2.1.1) Entities

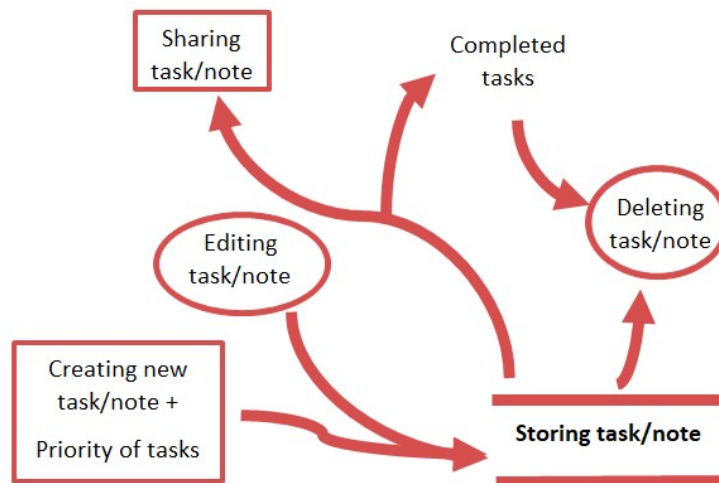
Tasks & Notes

##### 3.2.2.1.2) Processes

Priority Setting of Tasks

Note/Task Storage

##### 3.2.2.1.3) Topology



(Fig 9)

### 3.2.3) Maintenance of To-do list

The system shall convert a task into a to-do list over a time-bound constraint.

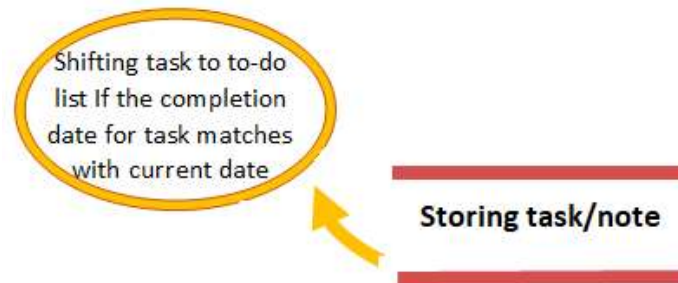
#### 3.2.3.1) Entities

Task

#### 3.2.3.2) Processes

Converting a task into to-do





(Fig 10)

### 3.2.4) Notify

The system shall generate notification alert of current tasks/to-do to be completed by the user.

#### 3.2.4.1) Entities

Task & To-do

User

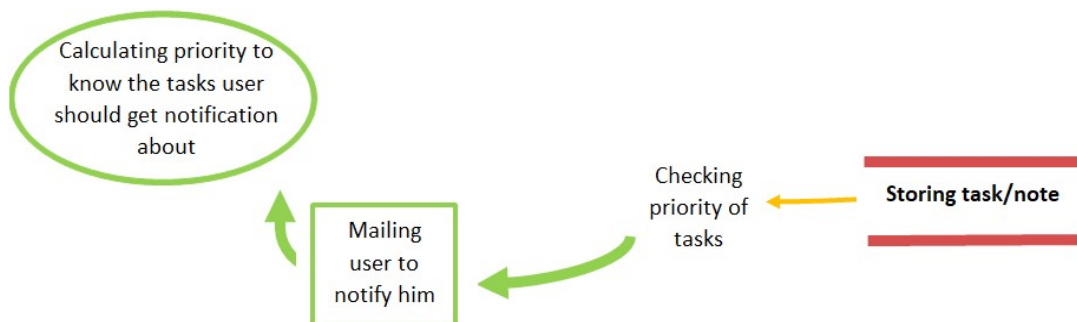
#### 3.2.4.2) Processes

Notification

#### 3.2.4.3) Attributes

Email

Priority



(Fig 11)

### 3.2.5) Performance Statistics

The system shall generate various statistical reports based on productivity of the user.

#### 3.2.5.1) Entities

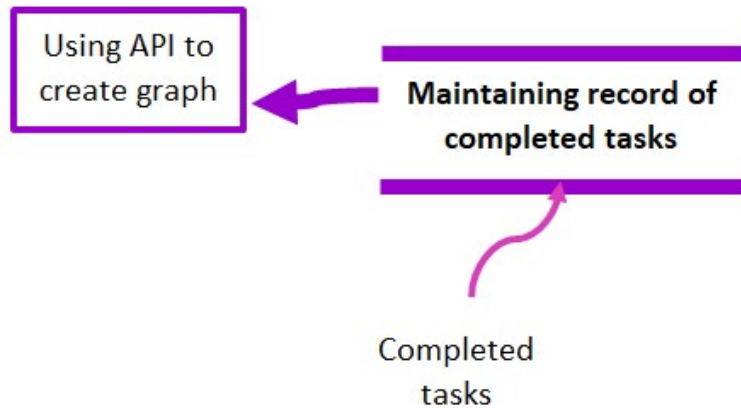
Tasks & To-do

Record

Graph

#### 3.2.5.2) Processes

Graph API



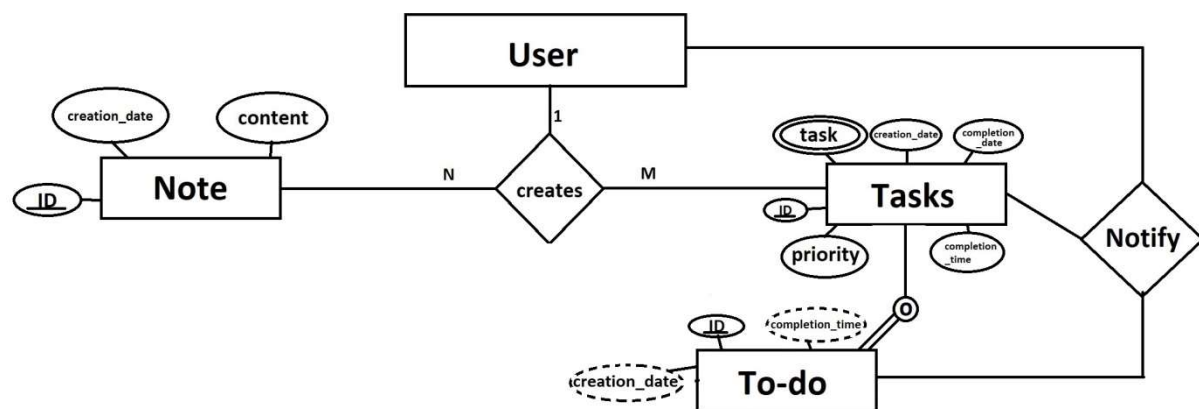
(Fig 12)

### 3.3) Performance Requirements :

As of now, Noteffy has not undergone through a beta-testing, hence analyzing the current performance of the prototype is a dubious task.

### 3.4) Logical Database Requirements :

EER (Enhanced Entity Relationship) Diagram



(Fig 13)

In this ER diagram, there are 4 entities: User, Note, Task & To-do.

The User entity is connected to the Note & Task entities through a "creates" relationship, which indicates that a user can create multiple notes & tasks.

The "To-do" entity is a subclass of the "Task" entity.

Task & To-do entities are connected back to the User entity through a “notify” relationship, which indicates that if a task or to-do is not completed by a completion\_time fixed by user, the user will be notified.

### 3.5) Design Constraints :

#### 3.5.1) Standards Compliance

- **‘Accessibility’ is an important idea behind many web standards, especially HTML. Not only does this mean allowing the web to be used by people with disabilities, but also allowing web pages to be understood by people using browsers other than the usual ones – including voice browsers that read web pages aloud to people with sight impairments, Braille browsers that translate text into Braille, hand-held browsers with very little monitor space, teletext displays, and other unusual output devices.**
- **On grounds of ‘Developer Agnostic’, Noteffy may go through several teams of designers during its lifetime, it is important that those people are able to comprehend the code & edit it easily. Complying to current Web standards, Noteffy should use open-source software**

**for its functioning & current versions of – HTML5, CSS3, PHP8, JS ES15, Python 3.10**

- **The graphical assets used for Noteffy are designed to be perceived with a quirky theme & must be made using latest version of Adobe Photoshop CC 2022.**

### **3.6) Software System Attributes :**

#### **3.6.1) Reliability**

As of now, the prototype of the web-app stores the note/task successfully created by the user through the interface. The back-end successfully notifies the user on his/her incomplete task.

On a small-scale, the reliability is well-intact.

#### **3.6.2) Availability**

The web-app will be available on the internet & anybody can access it via the internet.

The user data is well-intact on the server with full encryption.

In case a user is not able to complete his/her task, the incomplete task will be available in the miscellaneous tab of the web-app for the user's reference- the user can access it any time.

If, any time in the future, the web-app is lost or gets corrupt due to obscure reasons, a backup of the app is well-kept & maintained.

### **3.6.3) Security**

The user data is well-intact on the server & to avoid any case of external breach, the data is encrypted using Advanced Encryption Standard (.aes).

### **3.6.4) Maintainability**

### **3.6.5) Portability**

## **3.7) Organizing the Specific Requirements**

### **3.7.1) Functional Hierarchy**

## **4. Change Management Process:**

## **5. Document Approval:**

This SRS document shall get approved by Dr. Vandana Gandotra.

## **6. Supporting Information:**

### **6.1) Table of content**

- 1) Introduction*
- 2) General / Overall Requirements*
- 3) Specific Requirements*
- 4) Change Management Process*
- 5) Document Approval*
- 6) Supporting Information*

### **6.2) Index**

- 1) Introduction*
  - 1.1) Purpose*
  - 1.2) Scope*

***1.3) Definitions, Acronyms, Abbreviations***

***1.4) References***

***1.5) Overview***

***2) General / Overall Requirements***

***2.1) Product Perspective***

***2.2) Product Functions***

***2.3) User Characteristics***

***2.4) Constraints***

***2.5) Assumptions & Dependencies***

***2.6) Apportioning of Requirements***

***3) Specific Requirements***

***3.1) External Interfaces***

***3.2) Functions***

***3.3) Performance Requirements***

***3.4) Logical Database Requirements***

***3.5) Design Constraints***

***3.6) Software System Attributes***

***3.7) Organizing the Specific Requirements***

***4) Change Management Process***

***5) Document Approval***

***6) Supporting Information***

***6.1) Table of Contents***



## ***6.2) Index***

## ***6.3) Appendices***

### **6.3) Appendices**

**Fig 1: User Log in/ Sign up layout**

**Fig 2: Sign in Layout**

**Fig 3: Sign Up Layout**

**Fig 4: User Workspace Layout**

**Fig 5: Scoreboard Layout**

**Fig 6: Level 0 DFD**

**Fig 7: Level 1 DFD**

**Fig 8: Registration Topology**

**Fig 9: Note/Task Topology**

**Fig 10: Maintenance Topology**

**Fig 11: Notify Topology**

**Fig 12: Performance Statistics Topology**

**Fig 13: Logical EER Diagram**