



**ATMA RAM SANATAN DHARMA  
COLLEGE ,  
DELHI UNIVERSITY**

**PROJECT FILE FOR SOFTWARE ENGINEERING  
KEEP NOTES**



**SUBMITTED BY : PIYUSH SOLANKI**

**ROLL NO. : 21/18092**

**BSC(HONS) COMPUTER SCIENCE**

**SUBMITTED TO :**

**MRS. UMA OJHA**

**DEPARTMENT OF COMPUTER SCIENCE**

## INDEX

SNO.	DESCRIPTION
1.	<b>PROBLEM STATEMENT</b>
2.	<b>ACKNOWLEDGEMENTS</b>
3.	<b>PROCESS MODEL</b>
4.	<b>DFD ( LEVEL 1 AND LEVEL 0 )</b>
5.	<b>ESTIMATION MODEL – FUNCTION POINT AND COCOMO II MODEL</b>
6.	<b>GANTT CHART</b>
7.	<b>RISK TABLE</b>
8.	<b>PROTOTYPE OF PROJECT OR DEPLOYMENT LINK</b>
9.	<b>TESTING ON ONE MODULE</b> <b>a. WHITE BOX TESTING USING CYCLOMETRIC COMPLEXITY AND FIND THE INDEPENDENT PATHS.</b> <b>b. BLACK BOX TESTING ON ONE SCREEN</b>
10.	<b>USE CASE DIAGRAM</b>
11.	<b>USE CASES</b>
12.	<b>DATA DICTIONARY</b>
13.	<b>SEQUENCE DIAGRAMS</b>
14.	<b>SRS (Software Requirements Specification )</b>
15.	<b>STRUCTURE CHARTS</b>

## **1. PROBLEM STATEMENT**

With Each New day we get several new thoughts or ideas in our mind , which we as humans tend to forget . So this Project titled ‘Keep Notes ‘ gives you exactly the space to keep your notes safe for a longer period of time , just like the name suggests.

This a web based application.

This web apps provides you to create your on account on the server for storing your notes along with the title of your note. You can view your own notes here as well as the notes shared or kept by other users as well , which gives you a feel like a blog sharing website .

Technologies used for the application :

For frontend :

Html , Css and Bootstrap 4 , JQuery , Ajax , JavaScript

For backend :

Java Core , Java Servlets and JSP pages ,MySQL .

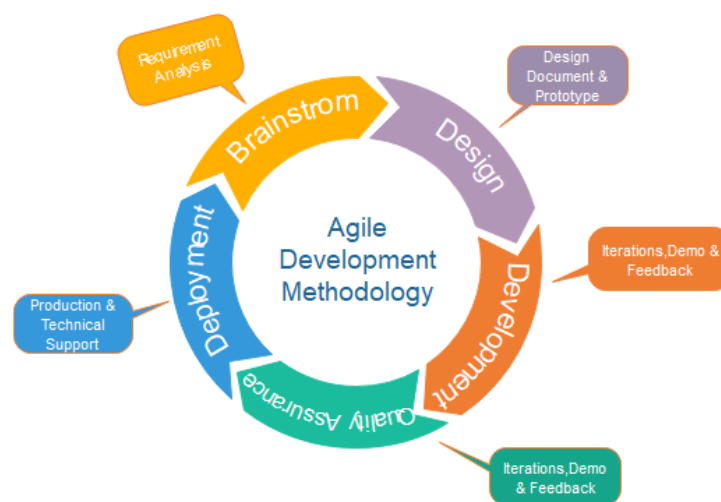
## 2.Acknowledgements

The contentment that is achieved on the successful completion of any task is incomplete without mentioning the names of the people who made it possible with their consistent guidance, support and indispensable encouragement.

The Keep Notes Project was undertaken by Piyush Solanki as his fourth semester Software Engineering Project, under the proper guidance and supervision of “Mrs. Uma Ojha”. My primary thanks to her, who poured over every inch of my project with painstaking attention and helped me throughout the working of the project. It’s my privilege to acknowledge my deepest sense of gratitude to her for her inspiration which has helped me immensely. I am extremely grateful to her for unstilted support and encouragement in the preparation of this project

### 3. PROCESS MODEL

The Project is based on 'Agile Model' because as the name of the model suggests it is swift and easy to work with. With every iteration in the development cycle of the Project new features can be added and older be improved with feedbacks coming in regularly through different means .



**Fig. Agile Model**

"**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks.

The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

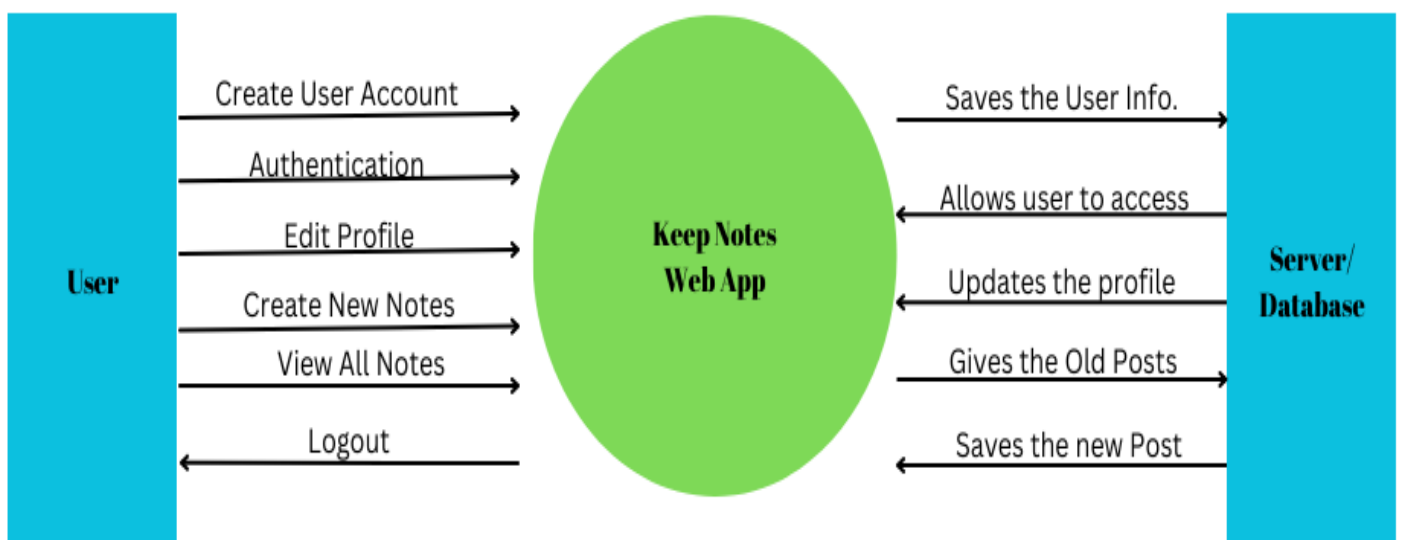
Following are the phases in the Agile model are as follows:

- 1. Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
- 2. Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
- 3. Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
- 4. Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
- 5. Deployment:** In this phase, the team issues a product for the user's work environment.
- 6. Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

#### 4. DFD ( Level 0 , Level 1)

##### DFD – Level 0

- DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



##### DFD Level 1

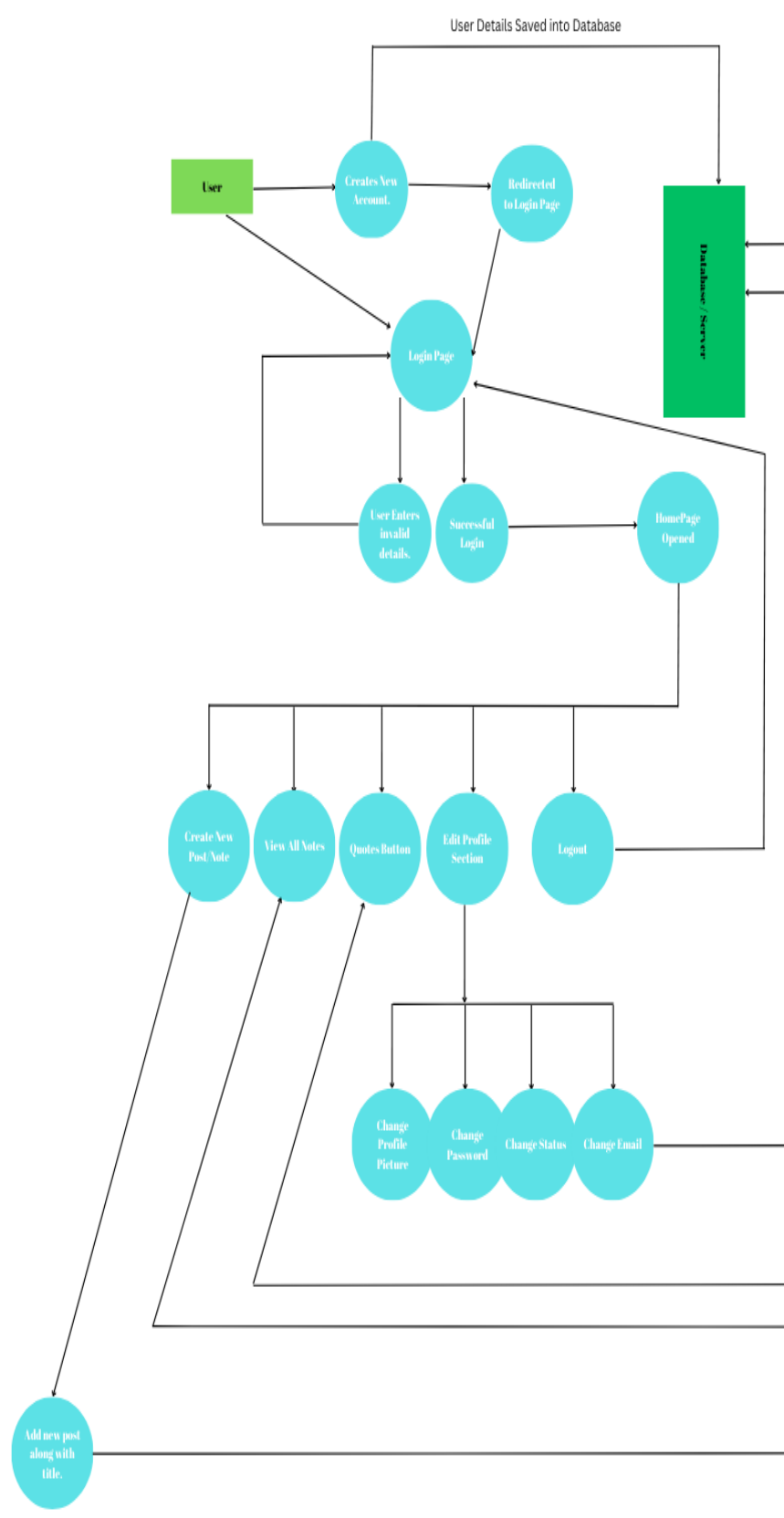
- DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

The New post is saved on the database/server

All the Posts are displayed on the webpage when the button is clicked.

Dynamic quotes are generated everytime user clicks on the button .

Changes applied to Profile added to database.





## 5. Estimation Model – Function Point and Cocomo II Model

Functional Point Analysis:

FPA provides a standardized method to functionally size the software work product. This work product is the output of software new development and improvement projects for subsequent releases. It is the software that is relocated to the production application at project implementation. It measures functionality from the user's point of view i.e. on the basis of what the user requests and receives in return.

- **Transactional Functional Type –**
  - **External Input (EI):** EI processes data or control information that comes from outside the application's boundary. The EI is an elementary process.
  - **External Output (EO):** EO is an elementary process that generates data or control information sent outside the application's boundary.
  - **External Inquiries (EQ):** EQ is an elementary process made up of an input-output combination that results in data retrieval.
- **Data Functional Type –**
  - **Internal Logical File (ILF):** A user identifiable group of logically related data or control information maintained within the boundary of the

application.

- **External Interface File (EIF):** A group of users recognizable logically related data allusion to the software but maintained within the boundary of another software.

#### Counting Function Points :

The five functional units are ranked according to their complexity i.e. Low ,Average or High using a set of prescriptive standards. After classifying each of the five function types , the Unadjusted Function Points ( UFP) are calculated using predefined weights for each function type as given in the table below.

Functional Units	Weighting Factors		
	Low	Average	High
External Inputs ( EI )	3	4	6
External Output ( EO )	4	5	7
External Inquiries ( EQ )	3	4	6
Internal Logic Files ( ILF )	7	10	15
External Interface files ( EIF )	5	7	10

## The Unadjusted Function Point factor value for the

Keep Notes				
Info . Domain Values	Count	Complexity	Weighting Factor	Count total
External Inputs	3	Average	4	12
External Output	3	Average	5	15
External Inquiries	2	Simple	3	6
Internal Logic files	2	Average	10	20
External Interface files	0	-	-	-
			Count Total	53

software is calculated now , using the above table.

We answer the following questions pertaining to the General System Characteristics on a scale of 0-5 as follows, where a 0-pointer implies the characteristic has no influence, 1-pointers denote an incidental influence, 2-pointers denote moderate influence, 3-pointers denote average influence, 4-pointers denote significant influence, and 5-pointers denote that the aspect is essential to the system:

Sno.	Description	Points
1	Are specialize data communications required to transfer information to or from the application ?	1
2	Are there distributed processing functions?	3
3	Is performance critical ?	4
4	Will the system run in an existing , heavily utilized operational environment ?	2

5	Does the online data entry require the input transaction to be built over multiple screens or operation ?	1
6	Does the system require online data entry ?	0
7	Does the system require reliable backup and recovery?	5
8	Are the ILFs updated online ?	0
9	Is the internal processing complex?	2
10	Is the code designed to be reusable?	3
11	Are conversion and installation included in the design ?	0
12	Are the inputs , outputs , files or inquiries complex ?	3
13	Is the system designed to for multiple installations in different organizations?	3
14	Is the application designed to facilitate change and ease of use by the user?	2

Now , the Function Point (FP) for the project can be calculated , taking the Cost Adjustment Factor (CAF) or Value Adjustment factor (VAF) in account , as follows:

$$FP = \text{Count Total} * VAF$$

$$= \text{Count Total} \times \left( 0.65 + \left( 0.01 * \sum_i^{14} Fi \right) \right)$$

The Count Total computed above was = 53 and The  $\sum$  can be computed from the table above as 29 . Using the information given :

$$FP = \text{Count Total} * \left( 0.65 + \left( 0.01 * \sum_i^{14} Fi \right) \right)$$

$$FP = 53 * (0.65 + (0.01 * 29))$$

$$FP = 53 * (0.65 + 0.29)$$

$$FP = 53 * 0.94$$

$$FP = 49.82 \sim 50$$

Therefore the Function Point for the Project is 50. Assuming organizational average productivity for systems of this type with low developer experience and environment maturity is 4 FP/person-month . Based on a burdened labour rate of Rs 8000 per month, we estimate the cost and effort needed to do the project in this section.

### *Effort Estimation*

The Estimated Effort in person-months using Function Point Analysis can be calculated as,

$$\text{Effort Estimate} = FP / PROD$$

$$\text{or, Effort Estimate} = 50 \text{ FP} / 4 \text{ FP} / \text{pm}$$

$$\text{or, Effort Estimate} = 12.5 \text{ pm} \approx 13 \text{ pm}$$

Therefore, using FPA, the Estimated Effort for the project is 13 person-months.

### *Cost Estimation*

As the burdened labour rate is Rs 8000 per month, the Cost per Function Point roughly translates to Rs 8000 per month / 4 FP per person-months or Rs 2000 per FP. As the Function Points for the project are 50, the Estimated Cost of the project can be calculated as,

$$\text{Cost Estimate} = FP \cdot \text{Cost per FP}$$

$$\text{or, Cost Estimate} = \text{Rs } (50 \cdot 2000) \approx \text{Rs } 100,000$$

Therefore, using FPA, the Estimated Cost for the project is Rs 100,000.

## Cocomo II Model :

Constructive Cost Model (COCOMO II) is a more comprehensive estimation model. COCOMO II is actually a hierarchy of estimation models that address the following areas:

- Application composition model - Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
- Early design stage model - Used once requirements have been stabilized and basic software architecture has been established.
- Post-architecture-stage model - Used during the construction of the software.

The COCOMO II model require sizing information.

Three different sizing options are available as part of model hierarchy:

1. Object points
2. Function points
3. Lines of source code

Like function points the object point is an indirect software measure that is computed using counts of the number of screens, reports and components likely to be required to build the application. Each object instance is classified into one of the three complexity levels (simple, medium or difficult) as shown in the following table:

Object type	Weighing factors		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Component			10

The object count is determined by multiplying the total number of object instances by weighing factor in the figure and summing to obtain a total object point count. When component based development or general software reuse is to be applied, the percent of reuse is estimated (%reuse) and the object point count is adjusted.

$$\text{NOP} = (\text{object points}) * [(100 - \% \text{ reuse})/100]$$

where NOP is defined as new object points.

To derive an estimate of effort based on the computed NOP value, a “productivity rate” must be derived.

$$\text{PROD} = \text{NOP} / \text{person-month}$$

Productivity rate for different level of developer experience and development environment maturity:

Developer's experience/capability	Very low	Low	Nominal	High	Very high
Environment maturity/capability	Very low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

Once the productivity rate has been determined, an estimate of project effort is computed using –

$$\text{Estimated effort} = \text{NOP} / \text{PROD}$$

COCOMO estimation for the project:

- 1.) Number of screens =6
- 2.) Number of reports = 2  
( Profile Details , All Posts)
- 3.) Number of 3GL components used =0
- 4.) Person-month=NOP/ PROD  
Person-month = 6.5

5.)Object Point Count=( Screen\*weighing factor + reports\*weighing factor + 3GL Componenets\*weighing factor)

Object Point Count =26

- 6.)NOP = OPC x [1- (%reuse) / 100]  
= Taking reusability to be 0  
= NOP = OPC = 26

7.) Prod =4

8.) Estimated Effort =NOP/PROD

Estimated Effort = 6.5



## 6.Gantt Chart

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time.

Project Name		Project Duration	Project Start Date	Project End Date
Keep Notes – Notes Storing Web App				
Task Id.	Task Description	Task Duration	Start Date	End Date
1.	Define Project Scope	4 Days	2/2/23	6/2/23
2.	Creating Login Page for the Project	2 Days	7/2/23	9/2/23
3.	Creating Account Webpage	2 Days	10/2/23	12/2/23
4.	Setting up Database for the project	1 Day	13/2/23	14/2/23
5.	Create Homepage for the project	4 Days	15/2/23	19/2/23
6.	Adding api to Homepage to display dynamic quotes at the end of page.	1 Day	20/2/23	21/2/23
7.	Designing Profile Section in Homepage	3 Days	22/2/23	25/2/23
8.	Adding edit Profile Modal	2 Days	26/2/23	28/2/23
9.	Create a webpage to add new post	2 Days	1/3/23	3/3/23
10.	Create a webpage to show all posts posted by all the users using app.	4 Days	4/3/23	8/3/23
11.	Connecting to database for new post and all posts webpage.	3 Days	9/3/23	12/3/23
12.	Perform Testing and QA	3 Days	13/3/23	16/3/23
13.	Bug fixing	1 Day	17/3/23	18/3/23
14.	Release the web app .	4 Days	19/3/23	23/3/23

### 7.Risk Table

<b>Risks identified</b>	<b>Risk type</b>	<b>Probability</b>	<b>Impact</b>
Maintenance Problem	Technical Risk	0.6	Negligible
High work load	Project Risk	0.3	Critical
Design and implementation	Technical Risk	0.95	Marginal
Hardware failure	Technical Risk	0.4	Critical

Inexper ienced Instruct or	Project Risk	0.8	Critical
-------------------------------------	-----------------	-----	----------

1. Have top software and customer managers formally committed to support the project?

YES

2. Are end users enthusiastically committed to the project and the system product to be built?

YES

3. Are requirements fully understood by the software engineering team and its customers?

YES

4. Have customers been involved fully in the definition of requirements?

NO

5. Do end users have realistic expectations?

YES

6. Is the project scope stable?

YES

7. Does the software engineering team have the right mix of skills?

YES

8. Are project requirements stable?

YES

9. Does the project team have experience with the technology to be implemented?

YES

10. Is the number of people on the project team adequate to do the job?

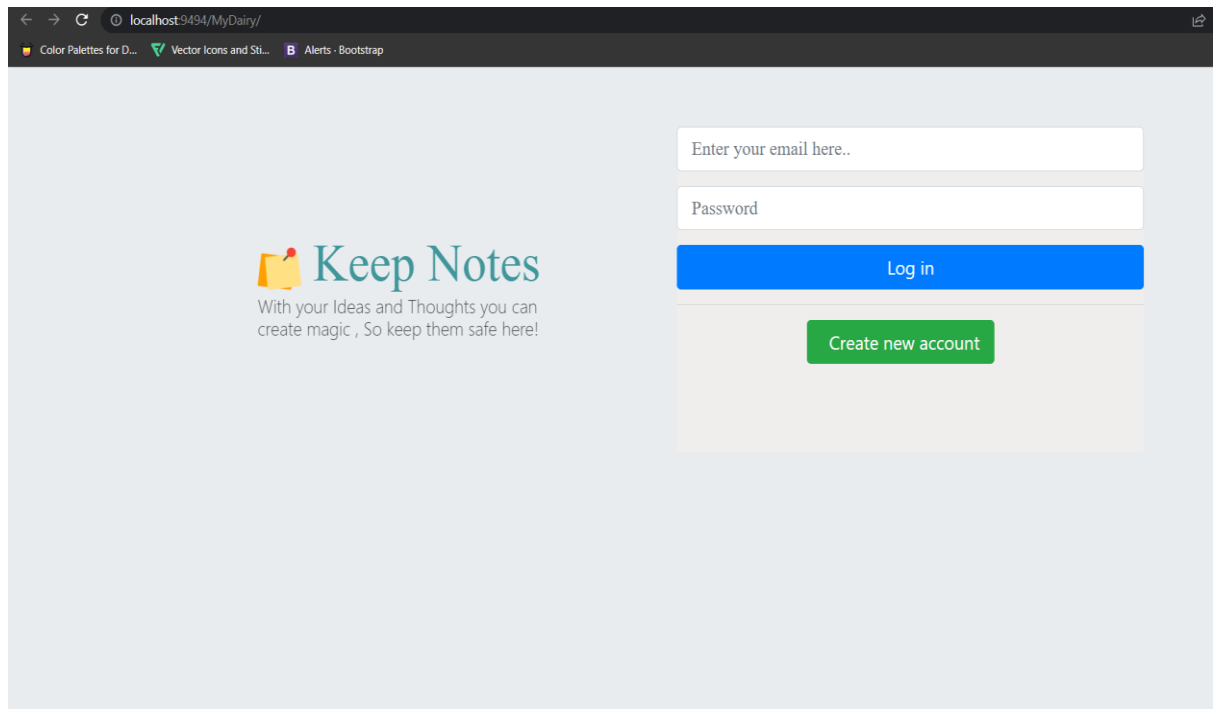
YES

11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

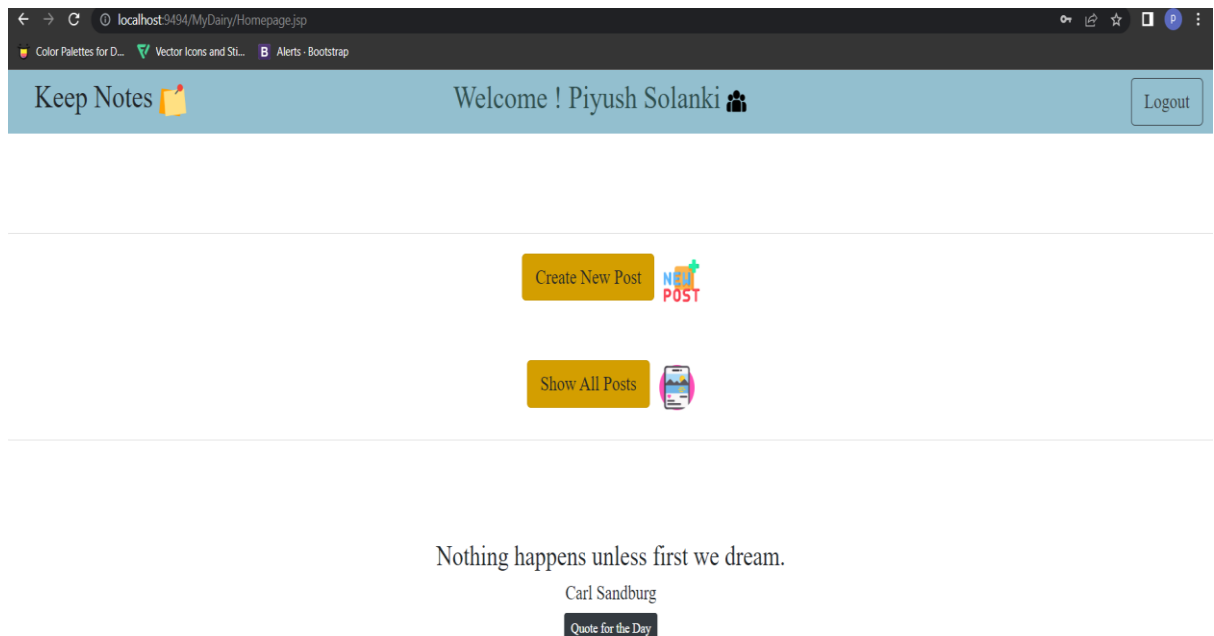
YES

## 8.PROTOTYPE OF PROJECT OR DEPLOYMENT LINK

### 1. Login Page



### 2. HomePage



### 3.Create New Post

localhost:9494/MyDairy/NewPost.jsp

Keep Notes

Welcome ! Piyush Solanki

Logout

Enter post title

Enter your notes...

Post it

Back to Home

You'll see it when you believe it.  
Wayne Dyer

Quote for the Day

### 4. Show all post

localhost:9494/MyDairy/OldPosts.jsp

Keep Notes

Welcome ! Piyush Solanki

Back to Home

Logout

#### First Post

this is the moment, tonight is the night

#### Random Musing

The Journey of my heart is now over but what's the use of a journey without you ?

#### Random

People will quit on you. You got to get up everyday make sure you never quit on yourself.

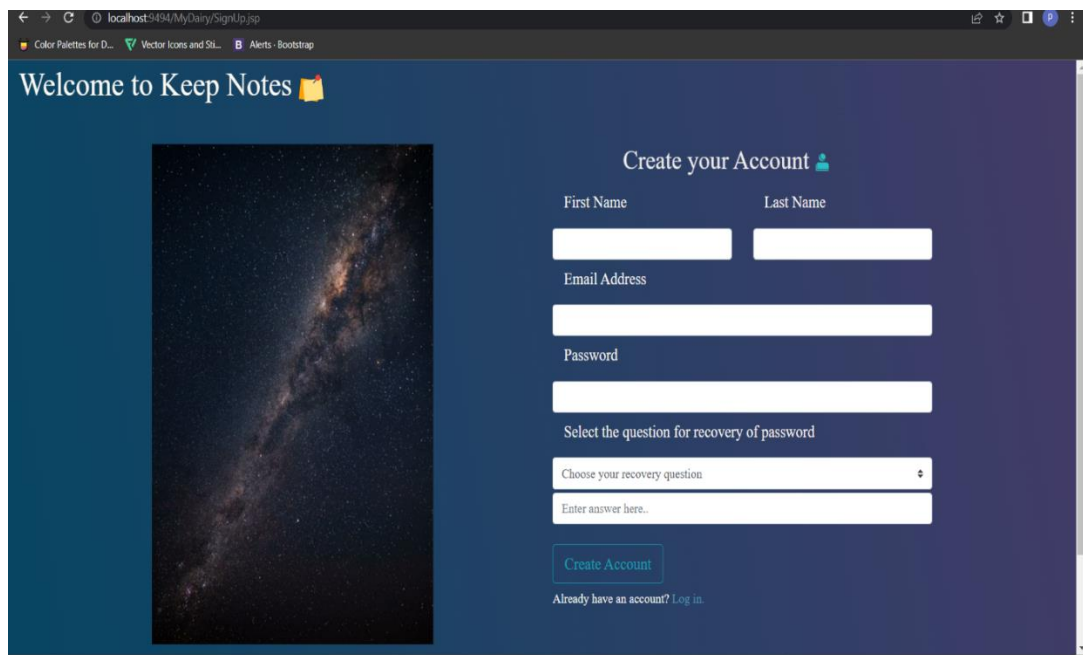
#### My Thoughts

People remember you only if you care for them .

It's easier to see the mistakes on someone else's paper.

Quote for the Day

## 5. Create New account



Welcome to Keep Notes 📖

### Create your Account 👤

First Name

Last Name

Email Address

Password

Select the question for recovery of password

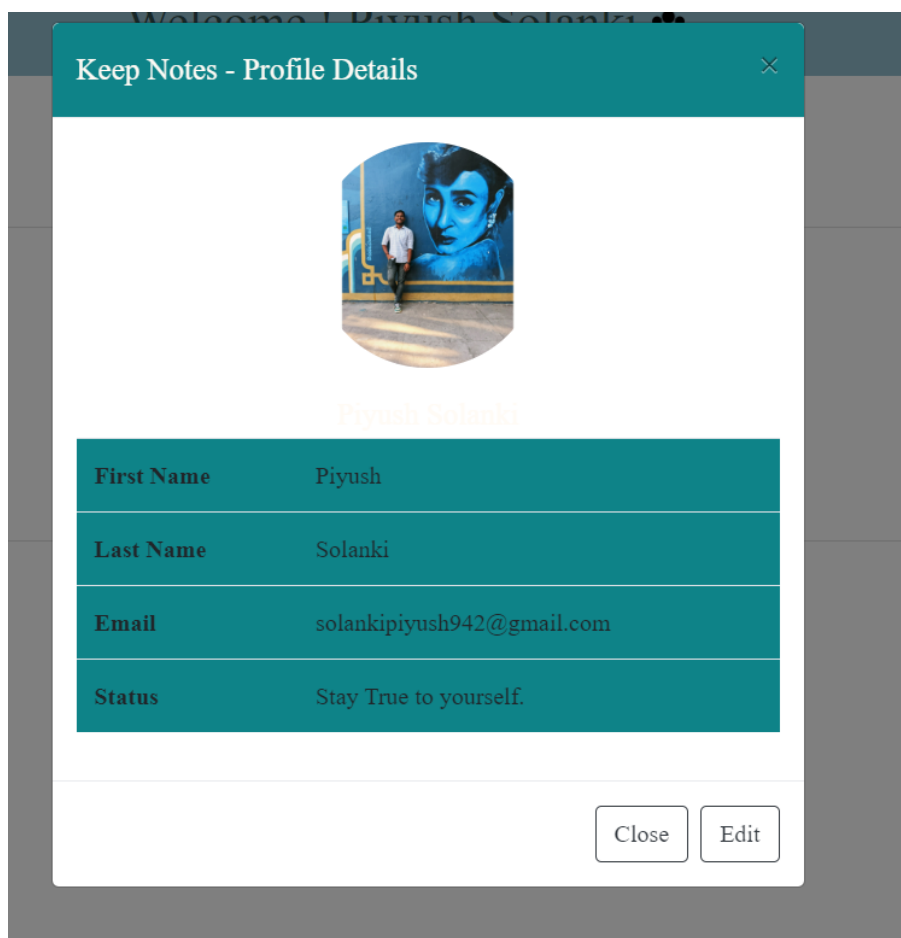
Choose your recovery question

Enter answer here...


[Create Account](#)

Already have an account? [Log in](#)

## 6. Profile



### Keep Notes - Profile Details



Piyush Solanki

First Name	Piyush
Last Name	Solanki
Email	solankipiyush942@gmail.com
Status	Stay True to yourself.

[Close](#) [Edit](#)

**9. TESTING ON ONE MODULE**  
**a. WHITE BOX TESTING USING CYCLOMETRIC**  
**COMPLEXITY AND FIND THE INDEPENDENT PATHS.**  
**b. BLACK BOX TESTING ON ONE SCREEN**

**WHITE BOX TESTING**

Using white box testing technique, we can derive test cases that:

1. Guarantee that all the independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries and within their operational bounds and
4. Exercise internal data structures to ensure their validity.

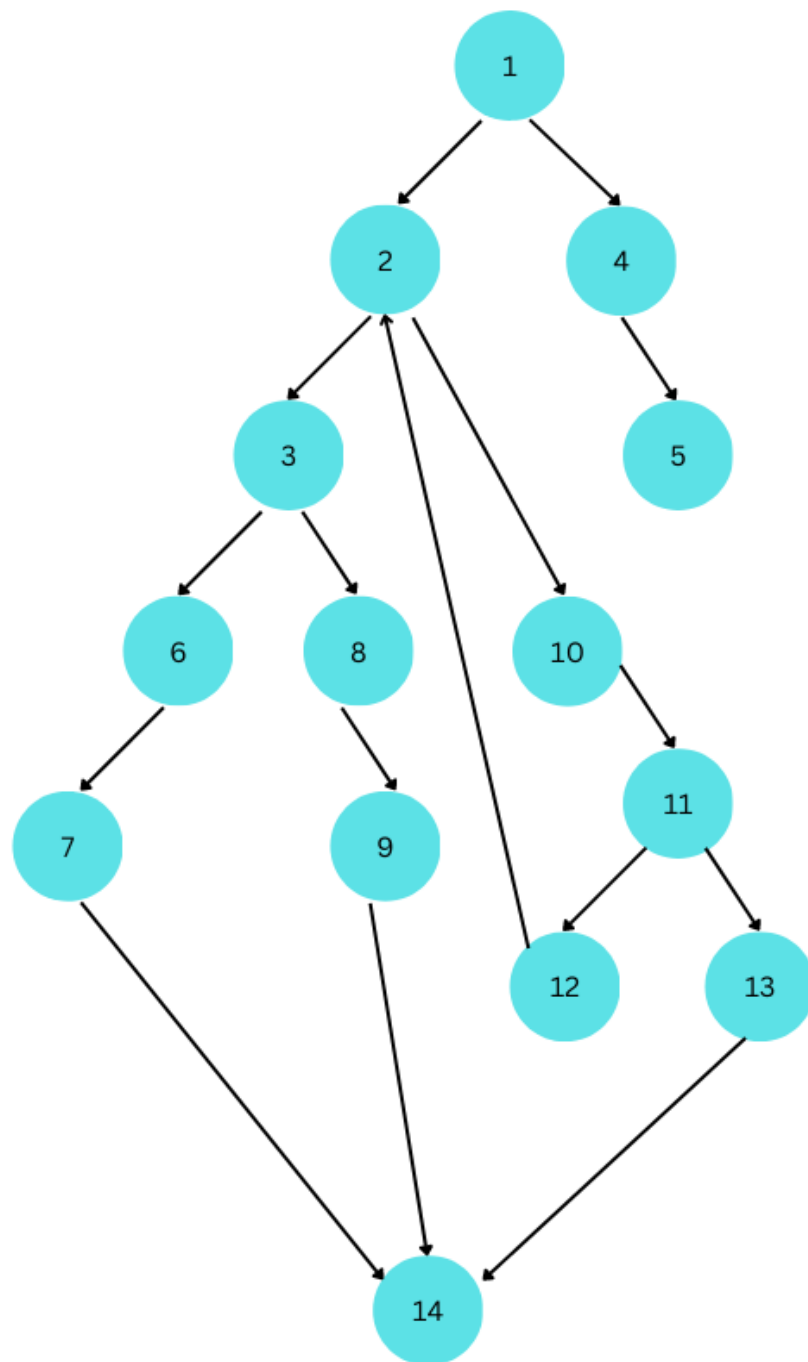
**BASIS – PATH TESTING**

It is a white-box testing technique which enables the test case designer to derive a logical complexity measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least once during testing.

We are doing white box testing for Homepage screen:



1. User try to login in
2. if Credentials Correct
3. homepage opens
4. else
5. Ask user to re-enter the details
6. if User selects create new post
7. Newpost webpage opens
8. else if User selects show all posts
9. All posts webpage opens
10. else if User clicks on username and choose edit
11. A modal opens to edit profile
12. else if user clicks on close
13. Homepage stays
14. else if user clicks on logout buttons
15. Sign in page occurs



Determining the Cyclometric Complexity:

$$V(G) = E - N + 2 * P$$

WHERE E=Edges , N = Nodes and P =Predicate Nodes

Here E=14 , N=14 and P =2

$$V(G) = 14 - 14 + 2 * 2 = 4$$

$$V(G) = P + 1 \text{ , Here } P = 3$$

$$V(G) = 3 + 1 = 4$$

$V(G)$  = Number of Regions

Here R= 4

$$V(G) = 4$$

## BLACK BOX TESTING

Test cases for Boundary value analysis:

Experience shows that test cases that are close to boundary conditions have a higher chance of detecting an error. Here, boundary conditions means an input value may be on the boundary, just below the boundary (upper side) or just above the boundary (lower side).

We, here, perform Black Box Testing on the LOGIN module.

The username should follow email pattern and password should be strictly combination alphabets , digits and special characters and in order to login the username and password will be checked with the database.

Best Case:

Test case	username	password	Expected output
1	Solankipiyus h942@gmail. com	grapes	Invalid input- Incorrect username or password
2	krishna	home	Invalid input- Incorrect username or password
3	harry	white	Invalid input- Incorrect username or password
4	Ron123@gm ail.com	morgan	Invalid input- Incorrect username or password
5	<a href="#">Solankijatin4 9@gmail.com</a>	Harrysol	Valid input- hello
6	<a href="#">Solankipiyus h942@gmail. com</a>	piyush	Valid input- hello
7	manna	tiara	Invalid input- Incorrect username or password

8	<a href="mailto:stonecold@gmail.com">stonecold@gmail.com</a>	snake	Invalid input- Incorrect username or password
9	<a href="mailto:beast@123gmail.com">beast@123gmail.com</a>	hope	Invalid input- Incorrect username or password.

As we know, with the single fault assumption theory,  $4n + 1$  test cases can be designed and which are here in this case equal to 9. The boundary value test cases are:

$n=2$ (number of inputs)

Worst Case:

If we reject the “single fault” assumption theory of reliability and may like to see what happens when more than one variable has an extreme value. Worst case testing for a function of  $n$  variables generates  $5^n$

$n$  test cases. Our login module takes 2 variables input and will have  $5^2=25$  cases.

Test case	username	password	Expected output
1	manna	grapes	Invalid input- Incorrect username or password
2	piyush	home	Invalid input- Incorrect username or password
3	mount	white	Invalid input- Incorrect username or password

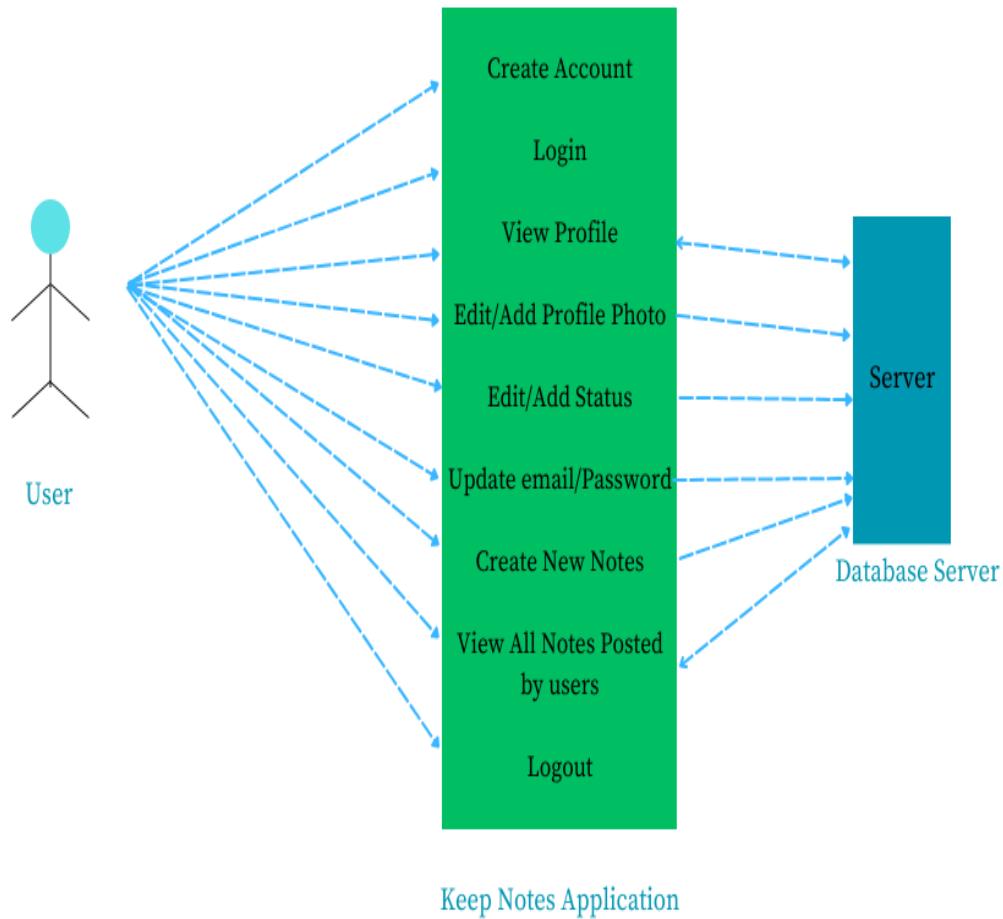
4	josh	morgan	Invalid input- Incorrect username or password
5	Solankipiyus h942@gmail. com	piyush	Valid input- hello
6	Solankipiyus h942@gmail. com	piyush	Valid input- hello
7	manna	tiara	Invalid input- Incorrect username or password
8	kiya	snake	Invalid input- Incorrect username or password
9	Solankiayush 22@gmail.co m	hope	Invalid input- Incorrect username or password
10	<a href="mailto:reed@gmail.com">reed@gmail.c om</a>	red	Invalid input- Incorrect username or password

11	<a href="mailto:john@gmail.com">john@gmail.com</a>	jute	Invalid input- Incorrect username or password
12	asking@gmail.com	maala	Invalid input- Incorrect username or password
13	zaan	begum	Invalid input- Incorrect username or password
14	keep	notes	Invalid input- Incorrect username or password
15	manna	umbrella	Invalid input- Incorrect username or password
16	Solankipiyush942@gmail.com	piyush	Valid input- hello
17	rohit@94gmail.com	ball	Invalid input- Incorrect username or
18	<a href="mailto:99@gmail.com">99@gmail.com</a>	Green	Invalid input- Incorrect



			username or password
19	<a href="#">Gmail12@gmail.com</a>	LIGHTS	Invalid input- Incorrect username or password
20	Salman	camera	Invalid input- Incorrect username or password
21	mona	MONA	Invalid input- Incorrect username or password
22	<a href="#">Solankijatin492@gmail.com</a>	Harrysol	Valid input- hello
23	Krishna	final	Invalid input- Incorrect username or password
24	<a href="#">raj@123@gmail.com</a>	staet	Invalid input- Incorrect username or password
25	samay	kon	Invalid input- Incorrect username

## 10. USE CASE DIAGRAM



## 11. Use Cases

### 1. Login :

This use case enables a user to log in and create a session on the system.

The following actor(s) take part in the use case:

User and the database server to validate the user.

This use case starts when user wish to login into the application , the following flow of action takes place :

I . User is asked to enter his/ her email and password.

II . The entered details are then checked by the database server , if the entered details are incorrect a pop is generated to tell user that something is wrong , enter the details again to log in .

III . If the entered details are correct , the user is logged in and the session starts by taking the user to the homepage .

### 2. Create Account :

This use case enables user to register on the application , if the user is new to it.

The following actor(s) take part in the use case:

User and the database server to register the user.

This use case starts when user wish to Create a new account , the following action takes place :

I . User is asked to input all the necessary details to create his/her account on the web application .

II. The system shows a dialog box , informing the user that his/her account is successfully created , if all the details are correct else it pops up a message to add his/her details again.

III . The User is the redirected to the Login Page.

### 3 . Create a new Post :

This Use case enables the user to create a new post/ notes and save it in the database .

The following actor(s) take part in this use case:

User and the Database server.

This use case starts when the user clicks on Create new notes button on the homepage, the following action occur here:

I . The session is already going on as the user is logged in .

II. The user is currently on homepage , when the user clicks on the Create new post button on the homepage, he/she is redirected to new page to create new note.

III. The user enters the title , content of the post / notes they want .

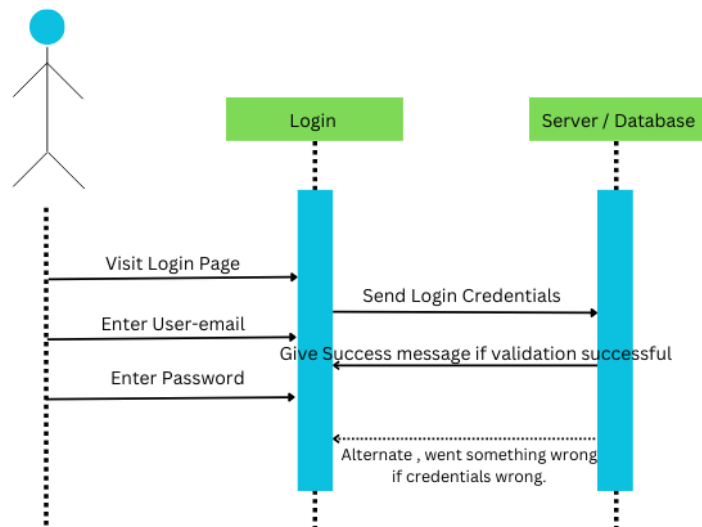
IV. When submit button is clicked , a dialog box informing the success of post is shown to the user .

## 12. DATA DICTIONARY

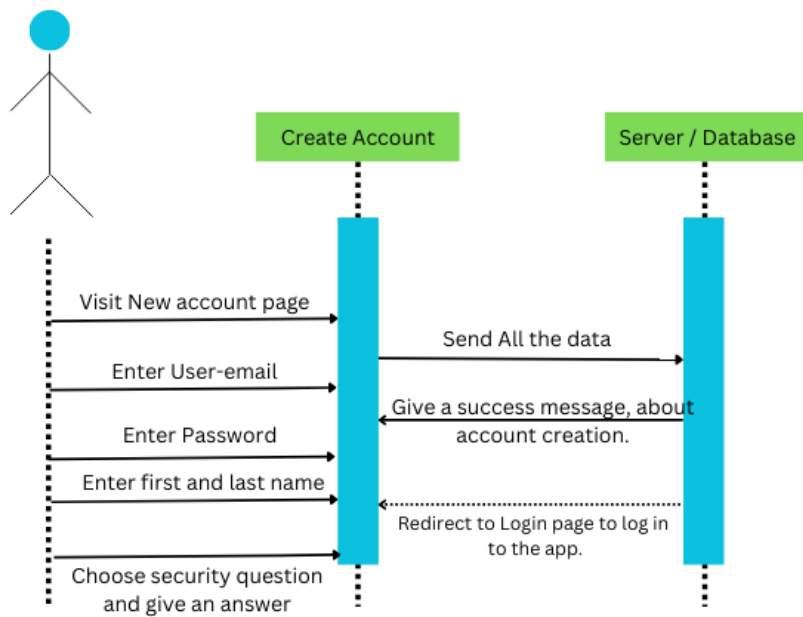
DATA	DESCRIPTION
Login details	{ User-email + password }
New account details	{ First-name , Last-name , User-email , Password, Security question , Answer }
Profile details	{ Name , email , password , status , profile picture }
Old notes	{ User-Id , Notes stored in database }
New note	{ User-Id , Date , Content , title }
Login response	{ Successful   Unsuccessful }

## 13. SEQUENCE DIAGRAMS

Login :



Create Account :



## **14. SRS (Software Requirement Specification)**

### **1.Introduction**

The purpose of this SRS document is to provide a detailed overview of Keep Notes along with its parameters and goals.

This SRS document describes the project's audience and its user interface, hardware and software requirements. It defines how the user should see the product and interact with its functionality. Nonetheless, it helps any designer and developer to assist in Software Development Life Cycle (SDLC) processes.

#### **1.1**

##### **Purpose**

This document provides an overview of the final product and assures the project management stakeholders and client that the product which is being delivered meets the requirement of what they asked for. It is intended for the customer and the developer.

#### **1.2**

##### **Scope**

This web application let the user who use this , create notes of any kind or topic they want , just like memos in our smartphones . It also shows all the notes posted by all the users using this web application.

#### **1.3**

##### **Definitions, Abbreviations and Acronyms**

User- The person who is using the software

Admin - the person who is maintaining the software at the backend.

## 2.Overall description

This project asks the user to first be registered on it so to use it. The registered user can then create posts or short notes of any kind and post them / save them on the server.

### **User interfaces**

The frontend will have a user friendly , responsive interface that provides the following screens.

#### Registration screen

The user will be required to provide their first name , last name , email id , username and password, a security question and an answer to it . After successful registration , the user will see a pop up that the account has been created.

#### Login screen

The users shall use their username and password to login into the system and access the feature according to their role.

#### HomePage

The users will be able to see two options : One of them to create a new post and the other to see all the old posts. The user will also be able to see the profile details placed on the navbar . As the user clicks on the name displayed on the navbar , the profile modal will be opened . Where user gets functionalities to update or edit the profile details.

### **Hardware Interfaces**

Any device with a proper internet connection and a latest web browser with in built java support .



## **Software Interfaces**

MySQL Database version 8.0 to store the data of user and all the posts made by the user.

### **3. Specific Requirements**

This section contains the software requirements to a level of detail sufficient to enable system designers to design the application and testers to test the application.

#### **User interfaces 25**

##### **Registration screen**

The screen will be available to visitors who want to register themselves with the app.

The required fields are

- First name
- Last name
- Email id
- Security question answer
- password

##### **Login screen**

In order to access the app the user will be required to login into the system , the required fields are

- Username
- Password

## Homepage Screen

It displays two buttons which user can use either to create a new note/post or else see all the notes posted by the users using the web application.

## Create new note Screen

Here the user will be able to write a note/ post. The user can choose any title they want along with the content .

## All notes screen

Here the user can see all the notes/posts posted by the users using the web application .By scrolling through the page. User also gets to see dynamic quotes at the bottom of the webpage which are called by an API.

## Performance requirements

### High rate data transfer

The backend should be efficient to allow applications to work well. The system should have a reliable uptime and a service availability at least 95% of the time other than planned maintenance routines.

## Logical database requirements

The app communicates with internal database it used for keeping record of users, admin data and maintaining details regarding the cycle.

It contains the following information is to be placed in the database

- Registration information (Id , firstname ,lastname ,email , profilepic , status , password , sec\_answer)

- Post Information (PostId , UserId , Post\_title , Post\_content , Post\_date )

### **Software system attributes**

- Availability

The data should be backed up timely for easy retrieval in dysfunctional scenarios

- Security

The users data is secured through a strong alphanumeric password .

- Maintainability

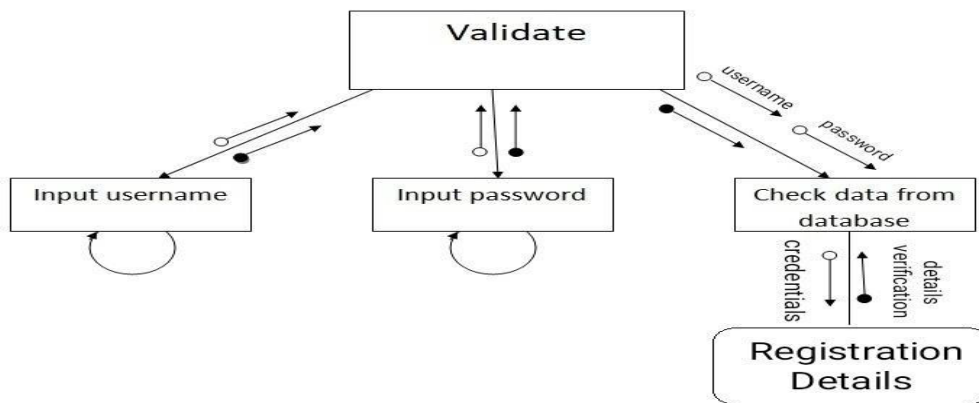
The app will facilitate easy maintenance process. Updates like password changes , username changes , status changes , profile picture changes are easy to do in the Edit section of the profile.

- Portable

The app will be portable.

## 15. STRUCTURE CHART

## Login Screen



## Create Account Screen

