Table of Contents

Table of Figures

Chapter 1 Introduction

Project introduction

N&T is a web application for capturing the notes. It is a platform where user will be able to capture and save their important notes for a whole lifetime. This application is specially targeted for group of people like students who can fully take advantages from it. Even though it is targeted for a certain group, every people are free to use this application and make benefits.

The main role of this application is to provide a platform where you can store your precious notes.

To solve the problems that every notebook has which is availability. This web application is introduced to solve that problem and to provide extra features to the users. This project also helps to solve for planning activities.

Overview of the project

N&T provides platform to users where users can use the platform for their personal as well as professional use. Here users can create their own notebook in which they can add as many notes as they want. They can also mark notes as important for fast access to those notes. They can create tasks list, the incomplete tasks will be notified so they can complete the tasks without forgetting it. More important they can back up their notes for easy transfer of data to another medium. Users data will be totally safe since to access their data they need to login into their account. Also there is a section call “Trash” in which all deleted note data will be stored. They can delete those data permanently or can recover it.

Some main features are

- Create and save unlimited notebooks and notes
- Export your data as back up
- Search note easily
- Share notes
- Pin important notes foe easy access
- Create tasks
- Reminder of uncompleted tasks

Aims and objectives

The aim of the application is

- Provide a secure and easy to access platform where user can save their notes.

The objectives of this application are

- Make users easy to add their content.
- Make easy to manage their note data.
- Make them easy to transfer or back up their data.
- Make users easy to organize their notes by placing into notebooks.

Chapter – 2 Analysis

Introduction to analysis

Analysis is way of dividing a whole topic into suitable parts for understanding. Analysis provides support to task. Analysis covers all the areas that is need to be search deep down and provide better understanding of task.

Why to perform analysis?

This is an important part of task, which helps to make better decision to perform the task. Here I am preparing a web application for which analysis helps to complete this task by providing necessary information. This information will help this project to give better performance and better solution of project.

Analysis methodology

This project is small-scale project. For this project, I have used Waterfall technique for developing and completing this project. In this project, I have determined to use ‘Hard approach’ methodology.

Hard approach methodology rigid techniques and procedures to provide unambiguous solutions to well-defined data and processing problems, focused on computer implementations. This methodology contains many processes, which are explained briefly.

Problem definition

Problem definition gives what are problems or opportunity. This shows the problem that we are facing and opportunity to solve (improve) the current existing system. The initial step is to identify and describe problem or opportunity.

Analysis of situation

Analysis of situation defines the current state and performance level of the system. Here system boundary is defined.

Identification of objectives and constraints

Identification of objectives and constraints is about defining where we would like to be and the constraints that make affect our ability to achieve the new state. It is an important step because it forces stakeholders to clarify what they hope to achieve, but also to understand the external factors and constraints that will restrict our change choices and therefore the level of change.

Measures of performance

Measures of performance is about defining measurable means of assessing the efficacy of any definite possibility. It’s really asking and answering the question “how will we know if the change has occurred?

Implementation

Implementation is about design, development, deployment and installation tasks required to get the agreed proposal. This part also for collecting data on the measures of performance to show that the required change as occur. Here all planned phases will be carried out to reach the project aim.

Data flow diagram (DFD)

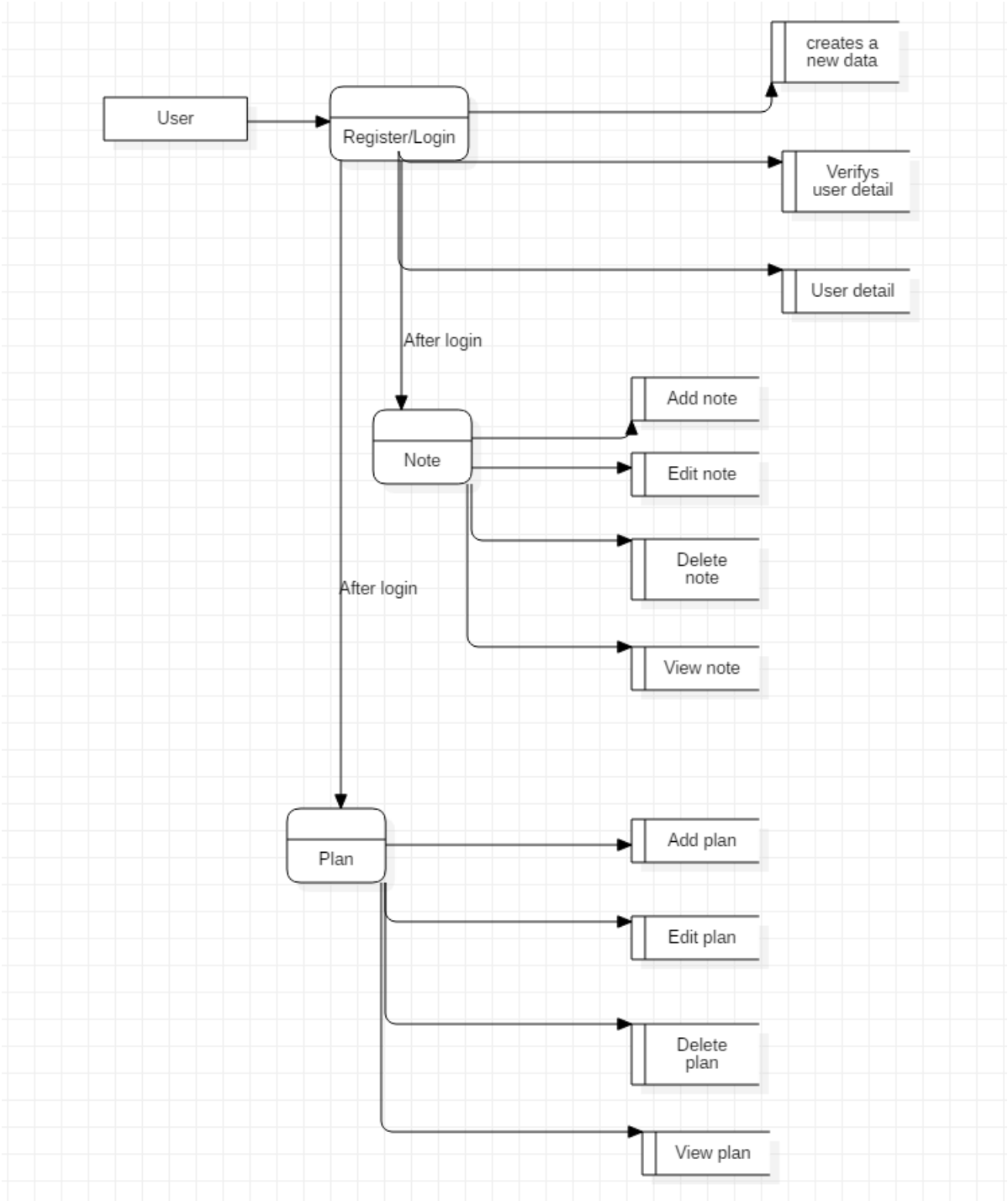


Figure 1 Data flow diagram

Feasibility study

Feasibility study is topic that comes under analysis part. Feasibility study uncovers weakness and strength of an existing business or purposed one. This shows opportunities and threats which exists in the surrounding. A proper feasibility study evaluates the project potential success rate and helps to gain best outcome.

MARKETING ASPECT

This project is designed to replace all the physical (hard copy) note entry into digital form. The demand of this kind of idea is very huge for people who are into converting from physical to digital form of data. To attract all the prospective users, this system will be shared into social media and other various media. This process will make this project feasible to produce.

TECHNICAL ASPECT

The proposed project is suitable for the entire web platform. This platform is suitable for producing the system. All the required hardware resources are available so make feasible to deploy this project.

FINANCIAL ASPECT

This project is form academic purpose, which does not need any base cost for production. This make this project feasible to create.

SOCIO-ECONIMC ASPECT

This projects aims to contribute benefits to the people of any group age because people who are interested into shifting to digital form will get many benefits. One of the main benefits is that they will save cost to buy physical copy, that saved cost can be beneficial into another form.

Software Requirements Specification

Functional requirements

ID	Title	Description	Rational	Dependencies
FR-01	User creation	Users should able to register account.	To recognize users to perform system function.	-
FR-02	User authentication	Users	To maintain authenticity, confidentially and integrity to access the system.	FR-01
FR-03	Add notes	Users should be able to add notes.	To add notes details for manipulating in future.	FR-02
FR-04	Update notes	Users should be able to edit all existing notes.	To edit existing notes.	FR-03
FR-05	Delete notes	Users should be able to delete any notes created by that specific user. Confirmation dialog message box must be shown before deleting.	To delete notes which are no longer in need.	FR-03
FR-06	Display notes	Users should be able to display all their existing notes.	To display all note and for other functionality.	FR-03
FR-07	Search notes	Users should be able to search their notes as from their note titles.	To search notes which have been created.	FR-03
FR-08	Sort notes	Users should be able to order the view of all notes based on all attributes.	To sort all searched notes base on the attribute.	FR-03, FR-06
FR-09	Create notebook	Users should be able to create notebooks in which they can add new or existing notes.	To create notebooks which contains many notes.	FR-02
FR-10	End user session	Users should be able to log-out form system.		FR-02
FR-11	Add to do list	User should be able to add new planning.	To add planning details.	FR-02
FR-12	Edit to do list	User should be able to edit existing planning.	To edit existing planning for any changes.	FR-12
FR-13	Delete to do list	User should be able to delete existing planning.	To delete planning if not necessary.	FR-12

FR-14	View to do list	User should be able to view existing planning.	To display all planning which are complete or not complete.	FR-12
FR-15	View progress	Users should be able to view progress of planning.	To display current planning progress by showing complete and uncompleted plans.	FR-14
FR-16	User manual	Users must be able to view manual for any kind of help or to use system.	To provide guidelines to users. To show how to use system.	FR-01 to FR-19
FR-17	Change theme	Users should able to change theme.	To switch between light and dark theme for eye-comfort.	FR-01
FR-18	Text layout	User should be able to change text layout inside notes.	To change text layout according to user's choice.	FR-03,FR-11
FR-19	Change font	User should be able to change fonts.	To change fonts according to users choice.	FR-03,FR-11

Non-functional requirements

ID	Category	Description
NFR-01	Interface	Interface of application must be easy and proper amount of guide should be provided.
NFR-02	Performance	The system performance should be smooth and without any high latency. In normal condition, system should not lag more than 3 seconds.
NFR-03	Security	Only valid users should be given access to system. All users must be authenticated before entering into system.
NFR-04	Integrity	All the data of users must be kept safely and data must be as user saved.
NFR-05	Usability	Application must be easy to use. Proper guidelines and instruction should be provided.
NFR-06	Accessibility	The system must be accessible all the time when user's needs.
NFR-07	Pre-installed templets	Provide pre-installed templets to help write down note easy.
NFR-08	Share-note	Share note between users.

Requirement Prioritization

ID	Requirement	MoSCoW	Rational
FR-01	User creation	Must	For recognizing users
FR-02	User authentication	Must	For security purpose and for valid users to enter into system
FR-03	Add notes	Must	Important function of the system
FR-04	Update notes	Must	Important function of the system
FR-05	Delete notes	Must	Important function of the system
FR-06	Display notes	Must	Important function of the system
FR-07	Search notes	Must	Important function of the system
FR-08	Sort notes	Would	For sorting displayed notes
FR-09	Create notebook	Must	For organizing the notes
FR-10	End user session	Must	For logging out from system of current logged in users
FR-11	Add to do list	Must	Important function of the system
FR-12	Edit to do list	Must	Important function of the system
FR-13	Delete to do list	Must	Important function of the system
FR-14	View to do list	Must	Important function of the system
FR-15	View progress	Must	Important function of the system
FR-16	User manual	Must	Important function of the system
FR-17	Change theme	Would	For changing themes
FR-18	Text layout	Would	Change layout of text
FR-19	Change font	Would	Change fonts of text
NFR-01	Interface	Must	Important function of the system
NFR-02	Performance	Must	Important function of the system
NFR-03	Security	Must	Important function of the system
NFR-04	Integrity	Must	Important function of the system
NFR-05	Usability	Must	Important function of the system
NFR-06	Accessibility	Would	Important function of the system

NFR-07	Pre-installed templets	Would	Functionality of system to make note easy to add.
NFR-08	Share-note	Would	Functionality of system to share note between users.

Hardware software specification

Minimum hardware requirements

- Processor (CPU) with 2 gigahertz (GHz) frequency or above
- A minimum of 2 GB of RAM
- Monitor Resolution 1024 X 768 or higher
- A minimum of 10 GB of available space on the hard disk
- Internet Connection Broadband (high-speed) Internet connection with a speed of 2 Mbps or higher
- Keyboard and a Mouse or some other compatible pointing device
- Sound card

Minimum software requirements

Operating system:

- Windows 7/8/10/Vista
- Mac OS X or Linux

Browsers

- Chrome* 36+
- Edge* 20+
- Mozilla Firefox 31+
- Internet Explorer 11+ (Windows only)
- Safari 6+ (MacOS only)

Use case diagram

Use case diagram is diagram representing dynamic diagram in UML. Use case diagram shows functionality of a system using use cases and actors. Here actors refer to people or sub-system operating the particular system. System refers to something that is developed. (Yuwantoko, Daniel and Ahmadiyah, 2017)

Why make Use Case diagrams?

Use case diagrams are very important for visualizing the functional requirements of a particular system. They also may provide aid for external or internal factors that may influence the system. They also provide high-level analysis from outside the system. They show how the system will interact with actors.

Use of Use Case Diagram for my project

All the necessary Use case diagram needed for my project are shown below. The description of each use case diagram scenarios are explained into table showing the necessary explanation.

use case diagram of note

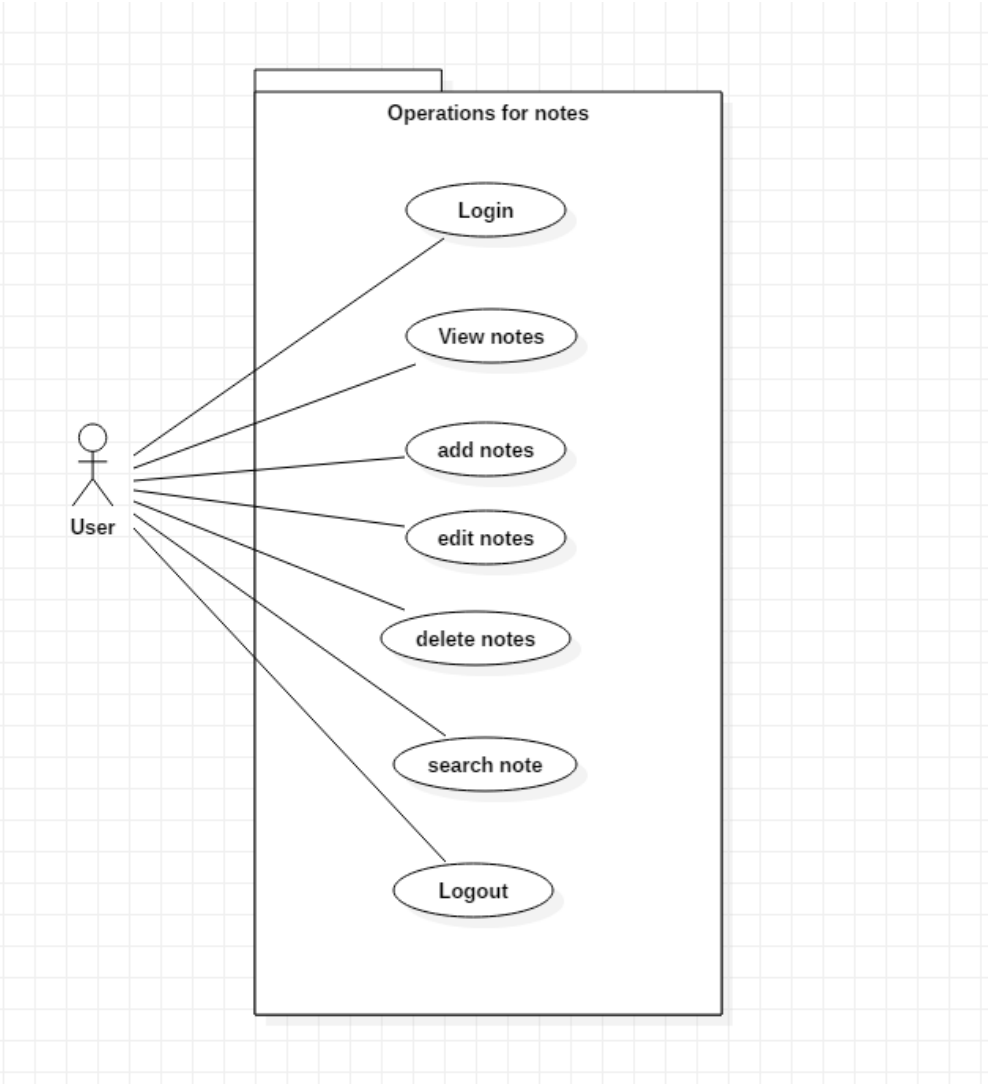


Figure 2Use case diagram for notes

Title: Add note

ID	FR-03
Justification	An important fundamental use case where data can be stored in form of note.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User goes into 'add new note section'. 3. User inputs all the data. 4. System stores the data into database.
Alternative flow	

Title: Edit note

ID	FR-04
Justification	One of an important function which edit the data from existing notes.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User goes for note to edit. 3. Users edits note. 4. User click on save button. 5. System process the data and saves into database. 6. System redirect outside the edited note.
Alternative flow	

Title: Delete note

ID	FR-05
Justification	One of an important function which delete the data from existing notes.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User searches the particular note to delete. 3. System displays all the possible note. 4. User selects the note and click on delete button. 5. System displays a dialog box for confirmation. 6. User confirms and system deletes note. 7. System redirect to dashboard.
Alternative flow	

Title: Search note

ID	FR-07
Justification	One of an important function which search the data from existing notes.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User goes into search bar where user types the note title. 3. System search the title given by user into database. 4. System displays all the related data if found. If not found message is displayed to user that notes is not found.
Alternative flow	

Title: Sort note

ID	FR-08
Justification	One of an important function which sort the data from existing notes.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User click on button to sort data accordingly. 3. System takes data from user input and sorts all the notes as accordingly.
Alternative flow	1. User enters into the system from login. 2. User search the note title. 3. System displays the notes if found. 4. System sorts the note accordingly.

Use case diagram of todo

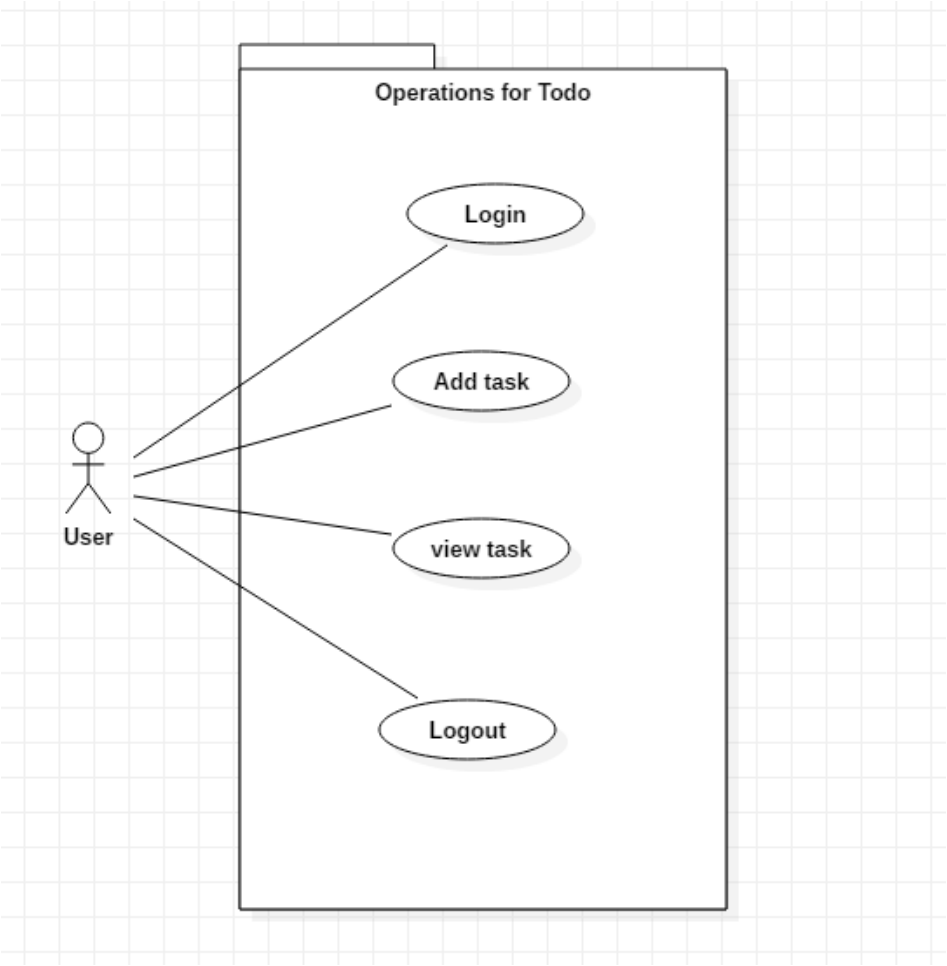


Figure 3Use case diagram for plan

Title: Add plan

ID	FR-11
Justification	One of an important function which adds plan.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User goes into plan content. 3. User click on add plan. 4. System redirect into a new window. 5. Users inputs plan data and click on save button. 6. System saves into database.
Alternative flow	

Title: View plan

ID	FR-14
Justification	One of an important function which view the data from existing plans.
Primary actor(s)	User
Supporting actors(s)	NA
Primary flow	1. User enters into the system from login. 2. User search the plan from title. 3. System takes user input and displays plan. 4. Users opens plan for viewing in detail.
Alternative flow	

Initial class diagram

Brief description about project

Todolist & Note management is a platform into which users can take notes and plan. This system allows users to create a new user. Users can also edit their profile. They can add, edit, delete and view their notes. They can view the existing notes by searching and sorting them. Users can also create a notebook where they can add one or more notes. In this notebook, users can add notes from existing notes or they can add new fresh notes. They can edit and delete notebook as per user's view.

Users can make plan where they can add plan. They also view, edit and delete plans. Users can also view plan progress by showing how much have is completed and how much is left.

Natural language analysis (NLA)

Natural language analysis is a method of finding out classes and functions that a program could have. This is made by reviewing the documents. In this case, description of project is a document. (Chirkova, 2002)

Nouns from document

All the nouns are noted and listed in a single place for identifying candidate class.

- Platform
- System
- Note
- Plan
- User
- Profile
- Notebook
- Progress

Selection of candidate classes

- System
- Note
- Plan
- User
- Profile
- Notebook
- Progress

Verbs from documents

All verbs are identified and listed

- Take
- Allows
- Create
- Edit
- Add

- Delete
- View
- Search
- Sort
- Show
- Completed
- Left

Selection of methods

Since all the verbs are useful in this software so all the listed verbs are selected for methods used in this software.

Class diagram (initial)

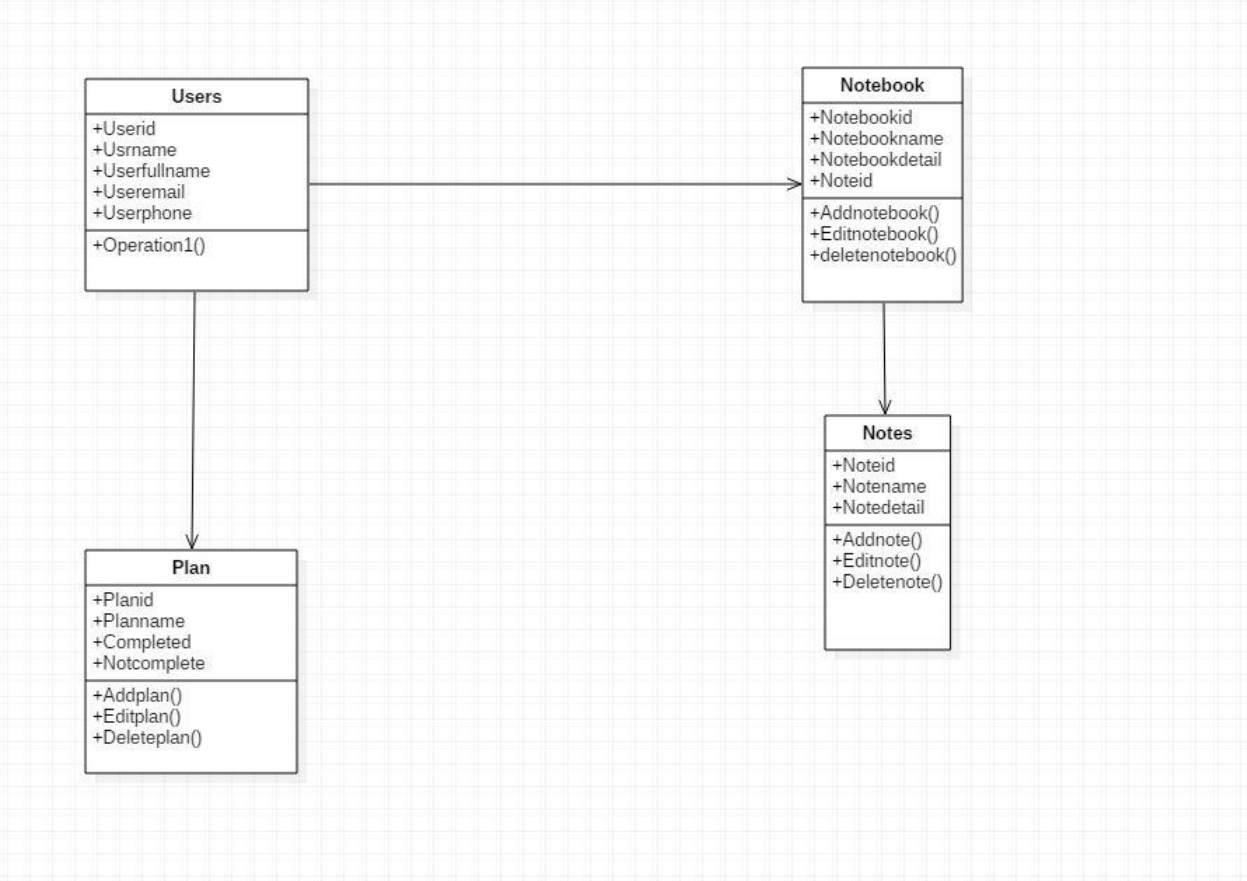


Figure 4 Initial class diagram

Chapter – 4 Design

Design is a way of creating plan or a baseline for implementing of process. System design is the process of designing the elements of system.

The purpose of the system design is to provide detail data and information about system and its functions.

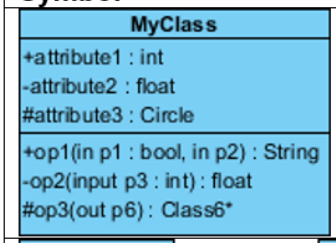
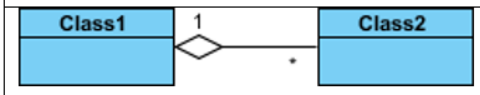
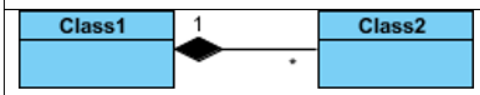
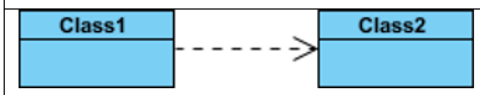
For my project I have prepared many design diagrams for representing my system behaviors.

Structural model

Final class diagram

Class diagram is a diagram that represents the static view of an application. Class diagram is used for visualizing, describing and documenting many aspects of the system along with constructing executable code of the application. (Abdulkareem et al., 2017)

Notation used

Symbol	Description
	Entity classes model long-lived information handled by the system, and sometimes the behavior associated with the information. They should not be identified as database tables or other data-stores.
	An association represents a family of links. A binary association (with two ends) is normally represented as a line. An association can link any number of classes.
	Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship
	The UML representation of a composition relationship shows composition as a *filled* diamond shape on the containing class end of the lines that connect contained class to the containing class.
	A dependency is a semantic connection between dependent and independent model elements.

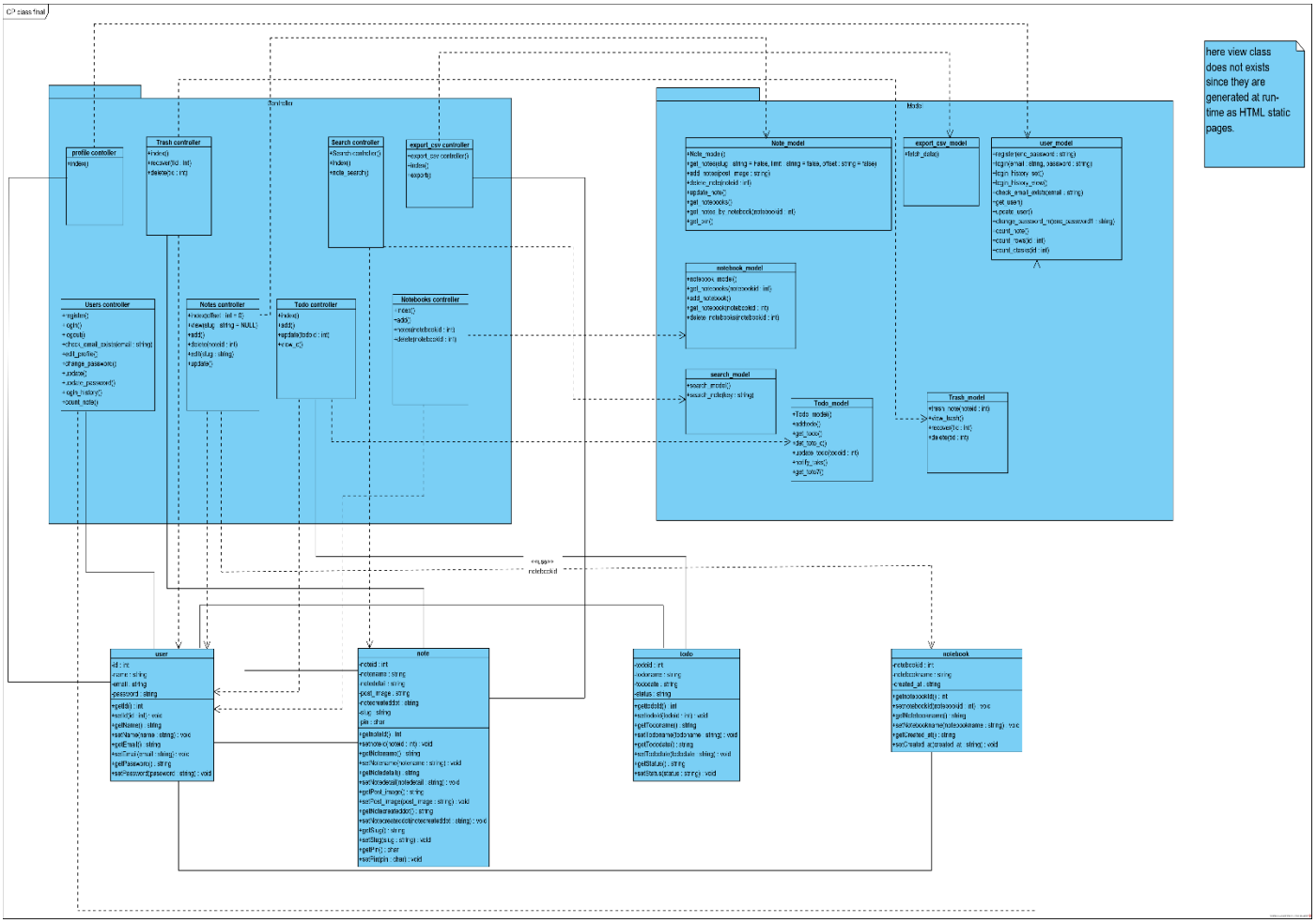


Figure 5 final class diagram

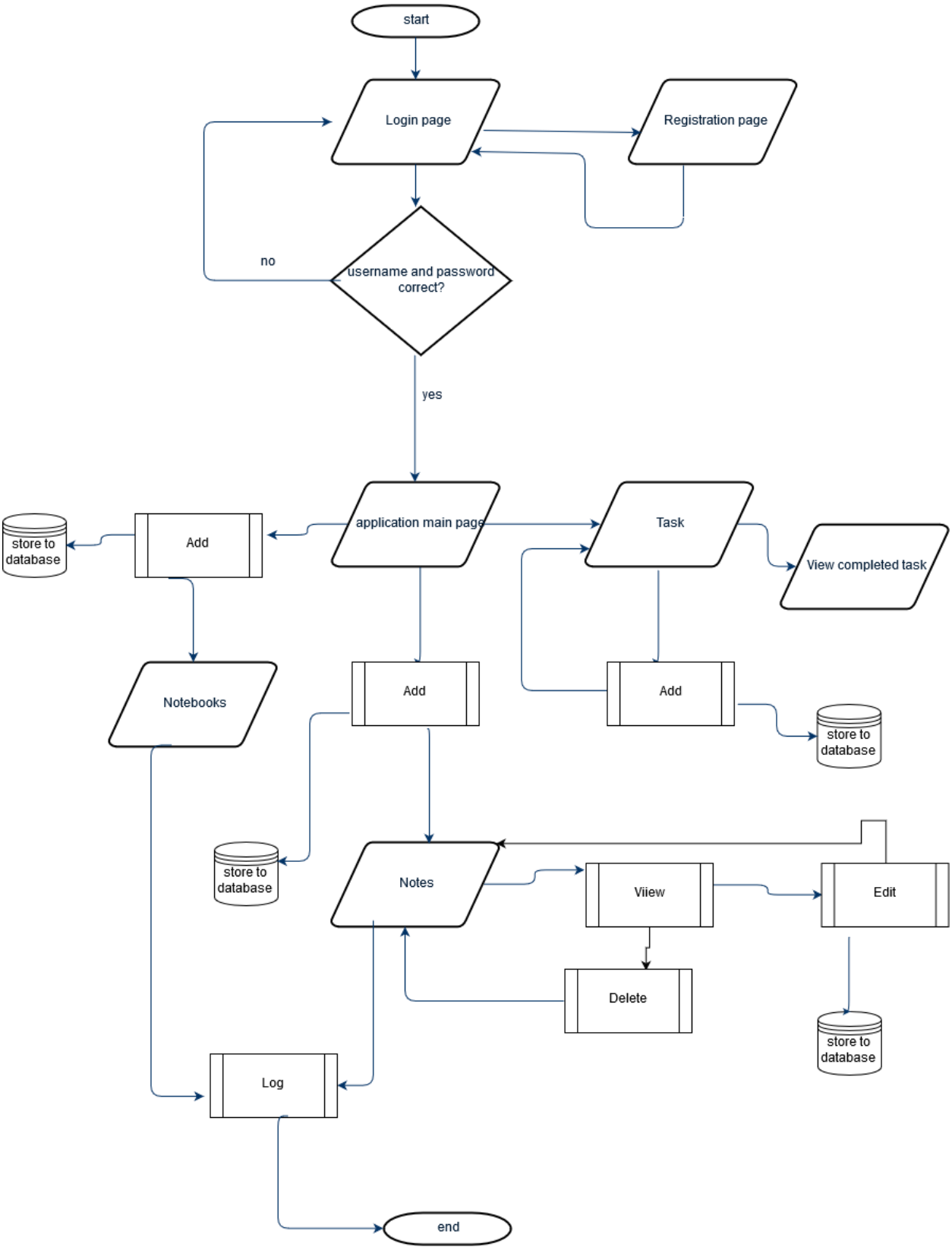


Figure 6 flow chart of the application

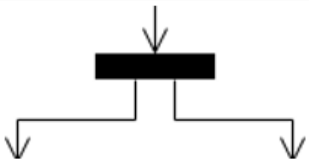
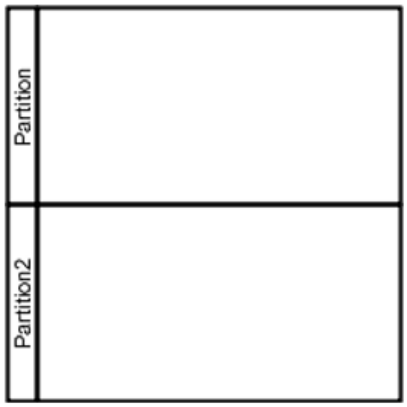
Behavioral model

Activity diagram

Activity diagram is a visually representing diagram in UML to explain the dynamic aspects of the system. It is like flowchart that represents the flow from one to another activity.

I have chosen activity diagram to show my part of system activity flows to another activity visually. This diagram shows dynamic nature of my system and may help in execute system by using forward and reverse engineering techniques.

Notation used

Symbol	Description
	Portrays the beginning of a set of actions or activities
	Show the flow of an activity from one activity (or action) to another activity (or action).
	A task to be performed
	Stop all control flows and object flows in an activity (or action)
	Split behavior into a set of parallel or concurrent flows of activities (or actions)
	A way to group activities performed by the same actor on an activity diagram or to group activities in a single thread
	Represent a test condition to ensure that the control flow or object flow only goes down one path

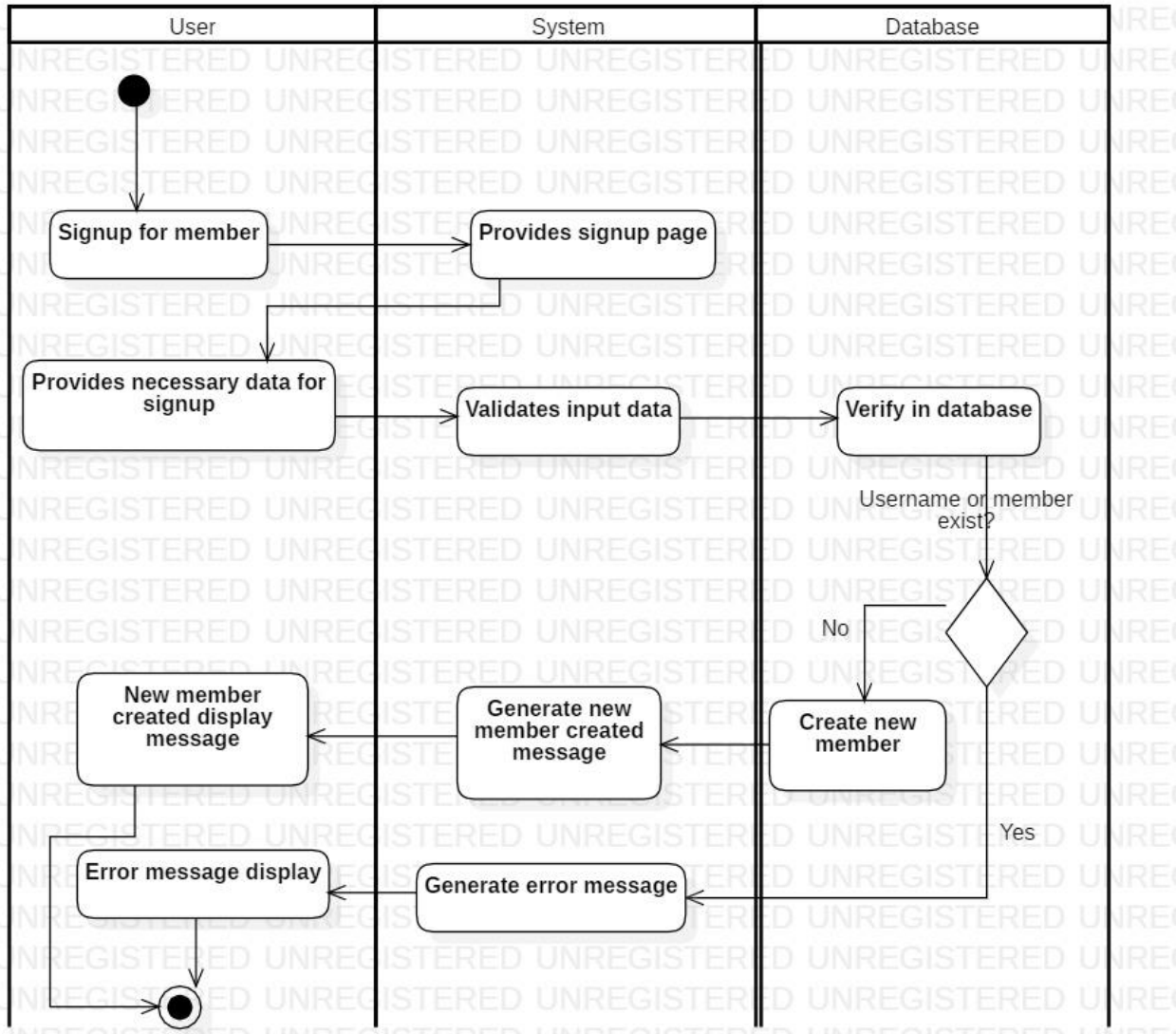


Figure 7Activity diagram of registration

Here user clicks on signup button and application redirect the user to the registration page. Then users provide necessary data into that page. At that time application checks input data and sends to the database for register. Database checks whether that user exists or not. If that user exists in the database, the sends error to system and system provides an error message. Else database creates a new user. The system now provides a message to users that a new user is created. Now all action is finished.

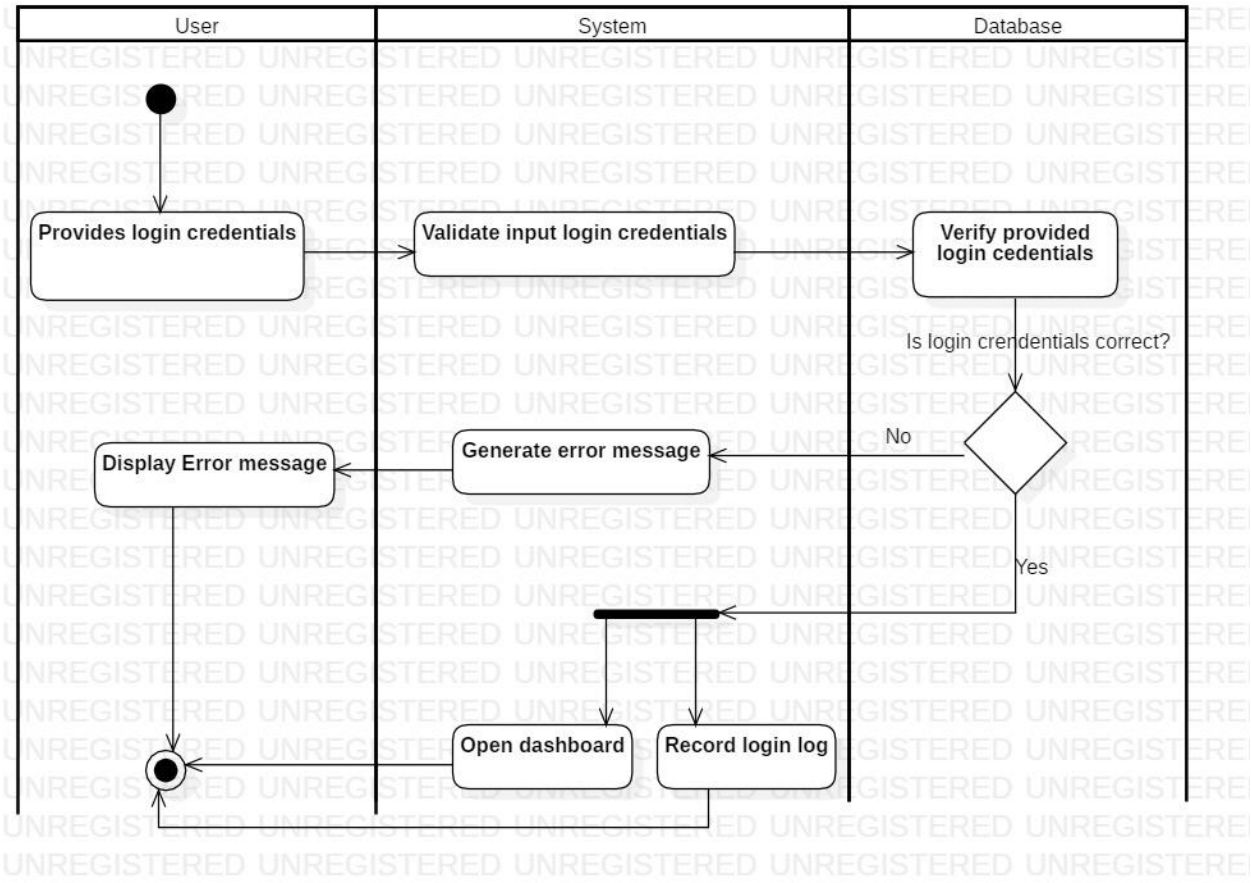


Figure 8Activity diagram of login system

Here user provides login (username and password) in the login page. The provided data is checked by system and transfer that data towards the database. In the database, the provided data is verified. If login information matches in the database, then the dashboard page is opened along with the login log is created. If provided data doesn't match with database then the message is shown 'Either username or password is incorrect'.

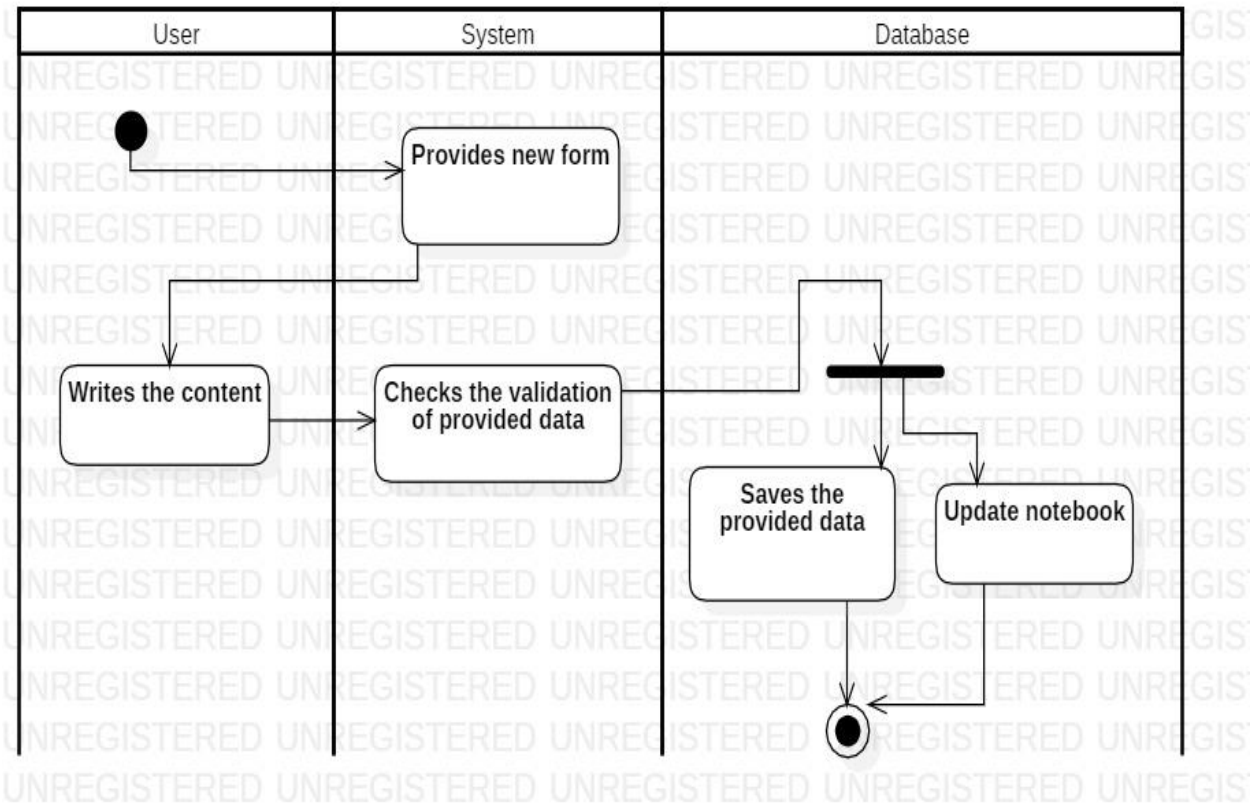


Figure 9Create note activity diagram

When clicking on add new note, the system provides a form where users write down their content. After clicking on save button, the data is sent to the database through an application where checks for any error. Then data is saved along with an update on a notebook where a note has been assigned.

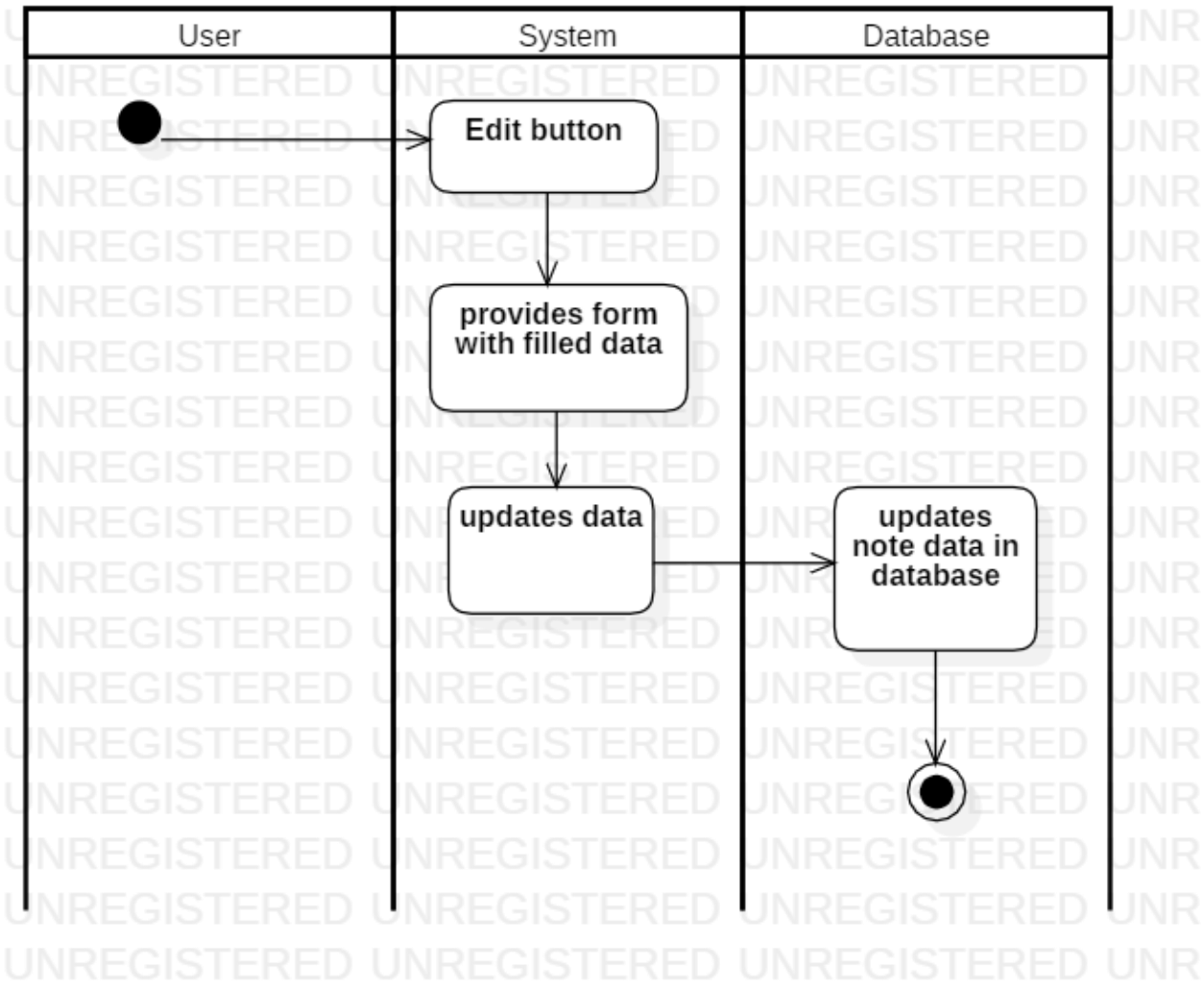


Figure 10 Edit note activity diagram

User clicks on the edit button of system, then a new window is open where note detail will be displayed for editing. There all detail of that particular note is displayed. Now the user makes the change and saves data. Here system sends what has been edited and runs update query in the database. After this note will be updated.

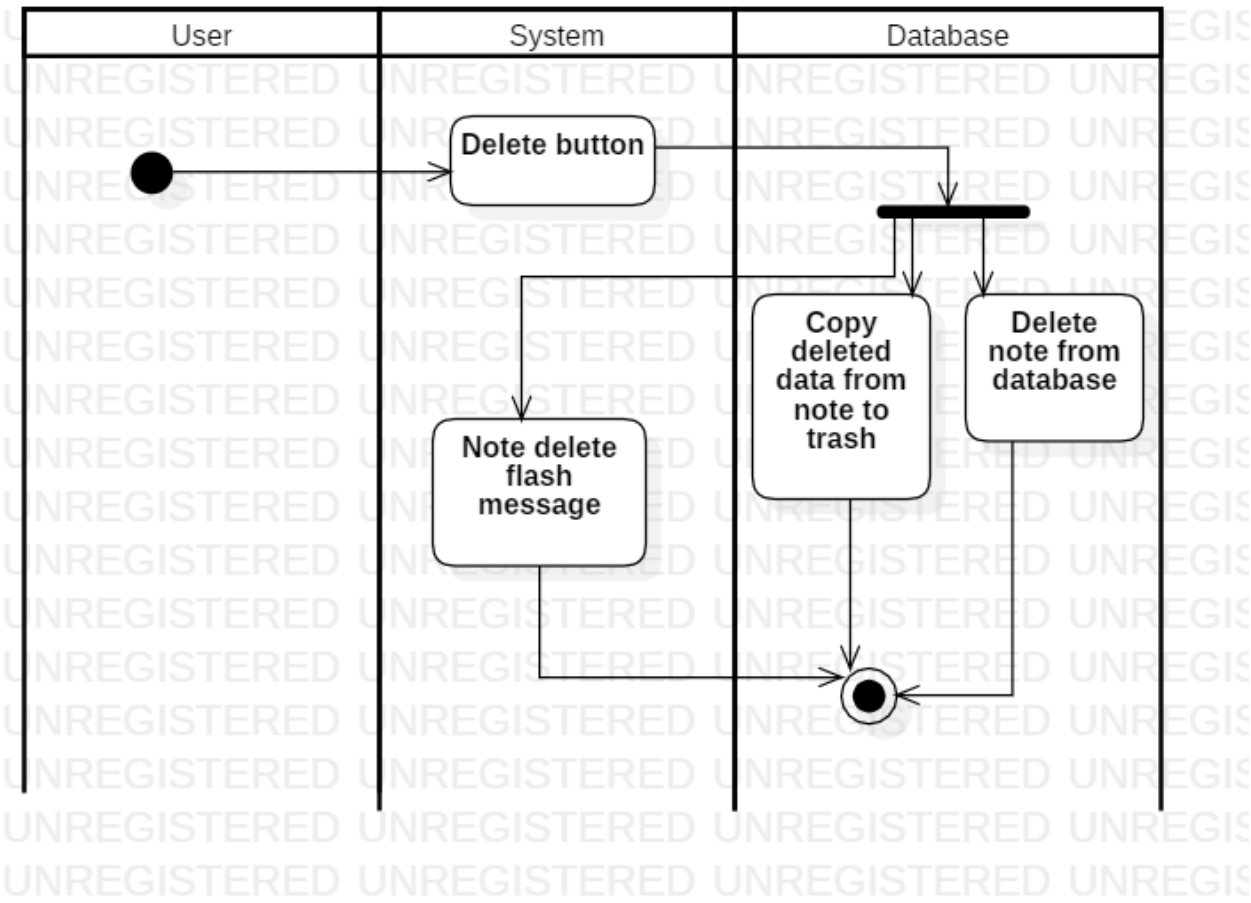


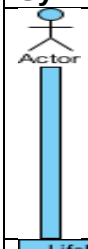
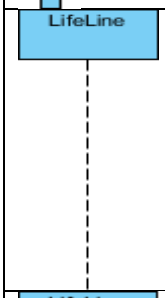
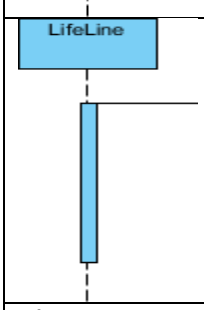
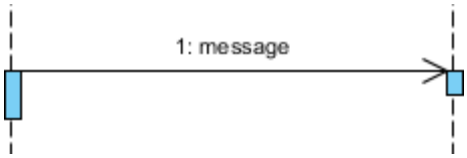
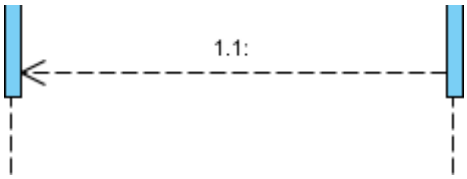
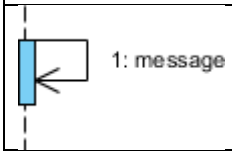
Figure 11 Delete note activity diagram

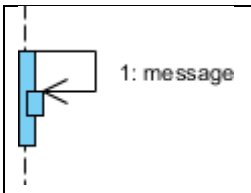
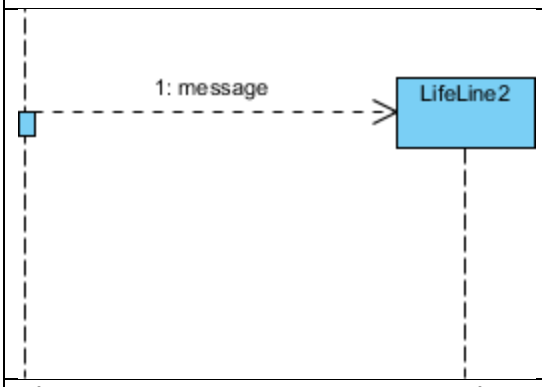
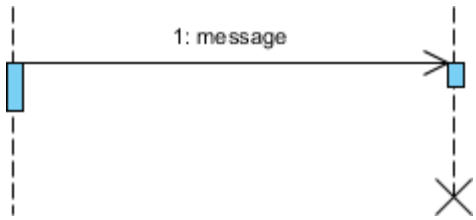
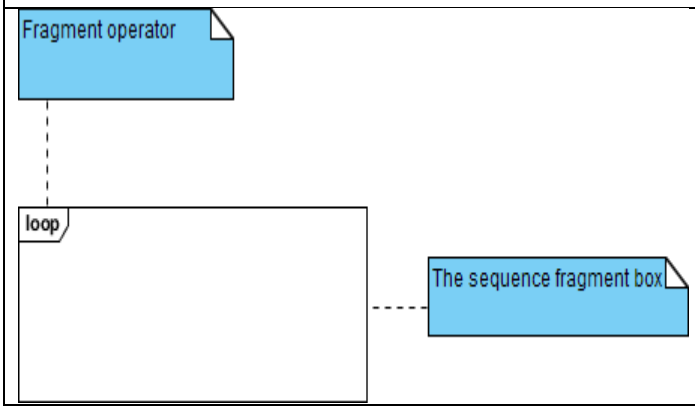
After user views note, users click on the delete button which runs multiple functions. In the database, insert query is done in which all data from the deleted note is copied and executed. At the same time delete query is executed which deletes that particular note data from the database. After that flash message is shown to users then delete activity ends.

Sequence diagram

Sequence diagram is a diagram that represents communications between two or more than two objects. Sequence diagram shows at what time object communicate with whom. It is basically record of flow of messages from object to objects in particular time.

Notation used

Symbol	Description
	Represent roles played by human users, external hardware, or other subjects.
	A lifeline represents an individual participant in the Interaction.
	A thin rectangle on a lifeline) represents the period during which an element is performing an operation.
	A message defines a particular communication between Lifelines of an Interaction. Call message is a kind of message that represents an invocation of operation of target lifeline
	Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.
	Self-message is a kind of message that represents the invocation of message of the same lifeline.

	Recursive message is a kind of message that represents the invocation of message of the same lifeline. It's target points to an activation on top of the activation where the message was invoked from.
	Create message is a kind of message that represents the instantiation of (target) lifeline.
	Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.
	A note (comment) gives the ability to attach various remarks to elements. A comment carries no semantic force, but may contain information that is useful to a modeler.
	A sequence fragment is represented as a box, called a combined fragment, which encloses a portion of the interactions within a sequence diagram. The fragment operator (in the top left cornet) indicates the type of fragment.;

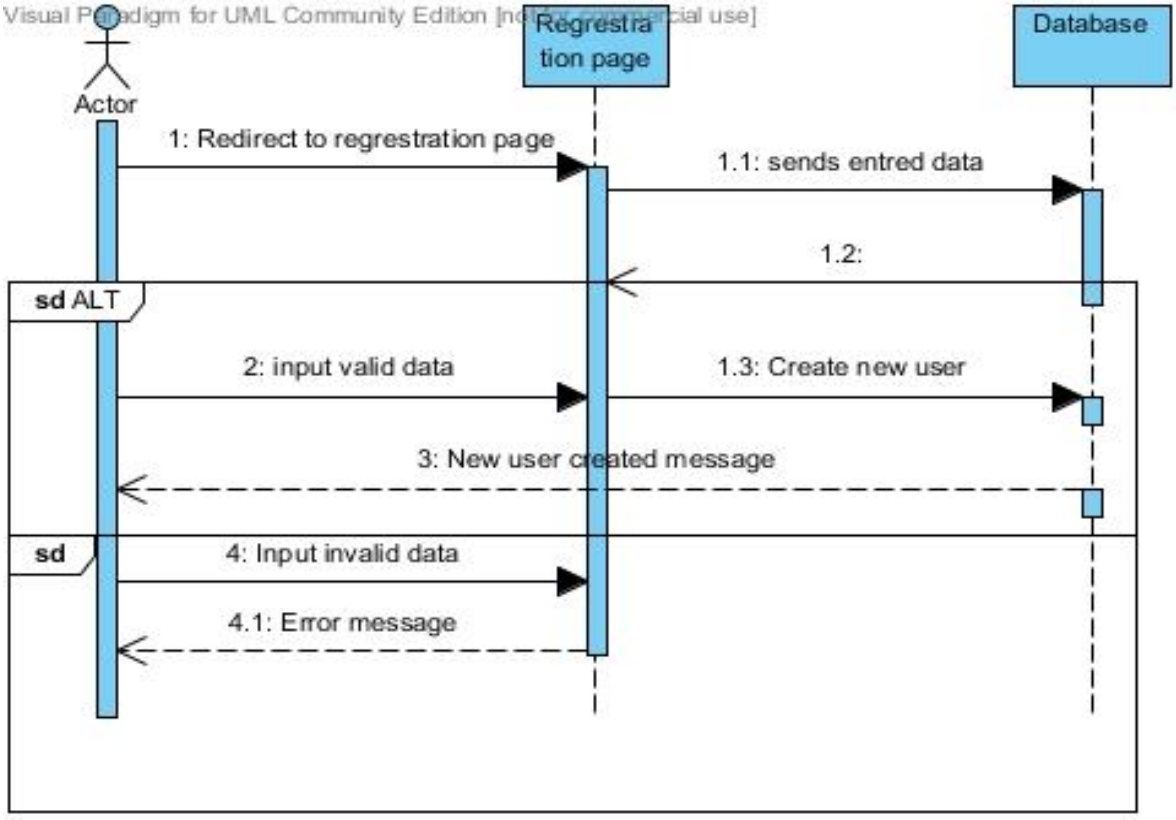


Figure 12Regrestration sequence diagram

When actor click on action button which redirects to login page and necessary data is provided from database. Now actor inputs valid data, those data are carried to database and there a new user account is created. Message is passed to user interface as “registration complete”. If data provided from actor is invalid, message is provided to actor as invalid data. this two phases are inside alternate frame.

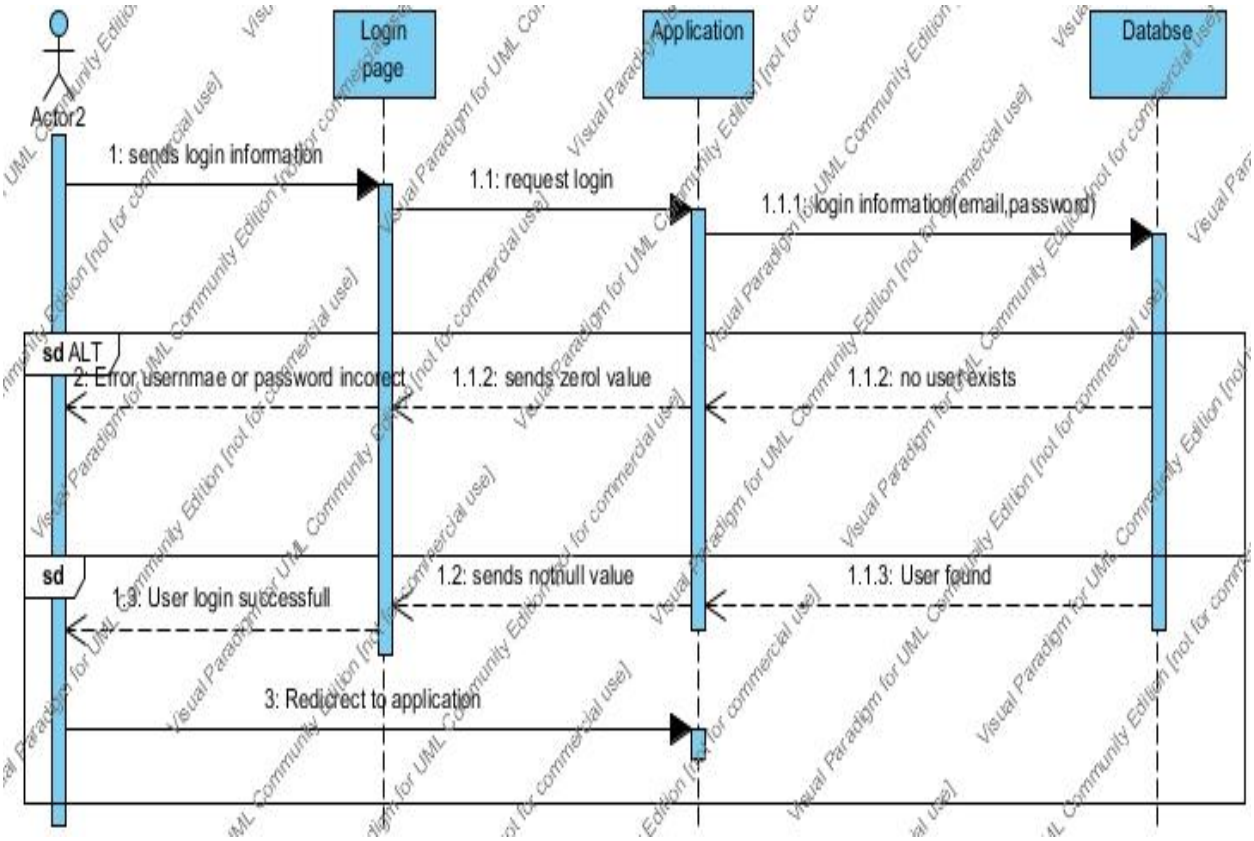


Figure 13User login sequence diagram

Now the user is in login page. Here, user provides login credentials and login request is passed into application. From respective functions, necessary data provided by user is passed into database and alternative frame will start. If the login credentials are incorrect then database fails to find that particular user account so sends to respective function as no user found. Then that function output value is evaluated then error message is generated and provided to the interface of user. If the login credentials match in the database returns value 1 to its respective function and successful message is displayed to user.

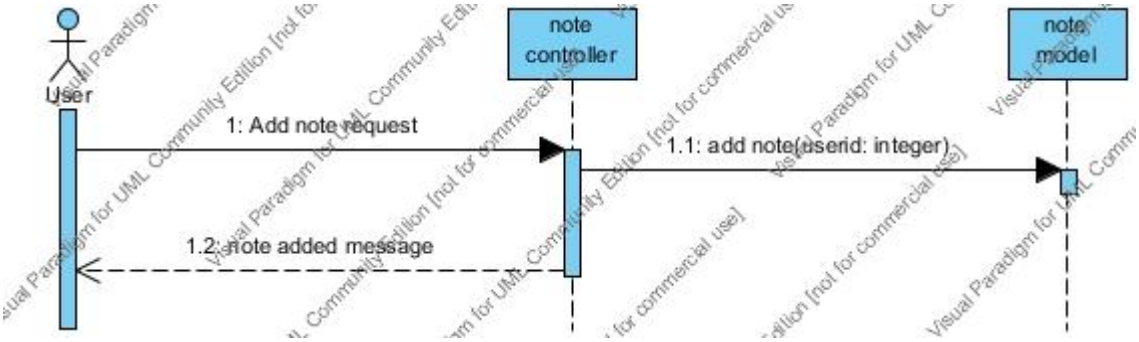


Figure 14 add note sequence diagram

When user click on add note button, the respective controller provides new page where user add their content. And when user hit save button current user id is provided to model. Now model runs add note query then message is displayed to user.

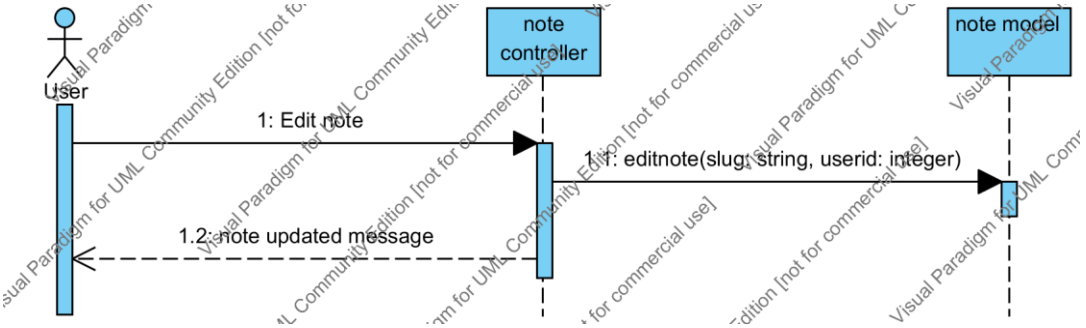


Figure 15 Edit note sequence diagram

Here user hit edit note and current user id and user note slug is passed into model where query is fired to save the edited content. Now new message is generated to ensure that any changes made is saved.

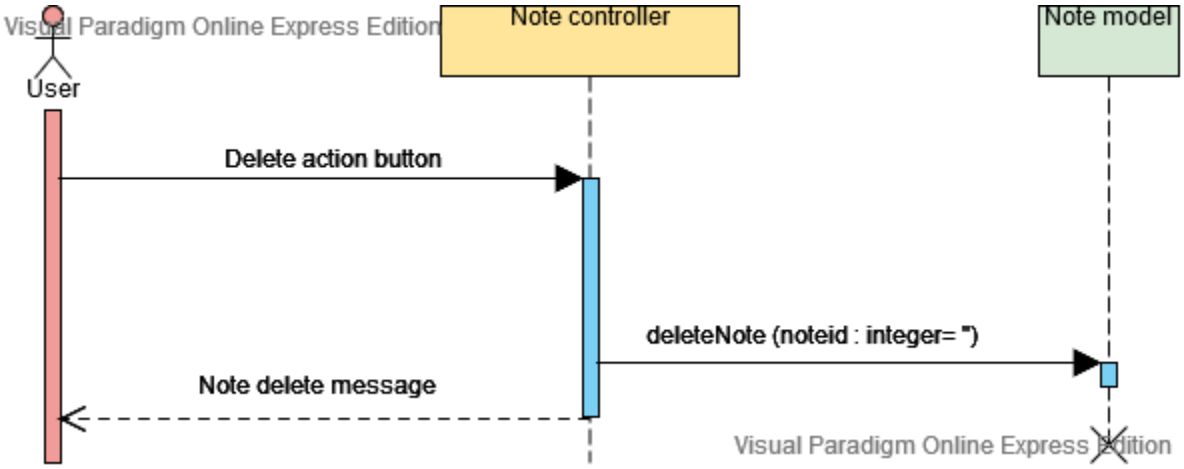


Figure 16 Delete note sequence diagram

When user hit on delete button, the controller grabs user id and note id. Now model fires delete query. After this deleted message is provided to user.

Database model

Data dictionary

Users entity

Data name	Data type	
Userid	integer(255)	Primary key
email	Varchar(255)	Unique key
password	Varchar(255)	Not null
name	Varchar(255)	Not null
Register_date	Date	Not null

Loginlog entity

Data name	Data type	
loginlogid	Varchar(255)	Primary key
Userid	Varchar(255)	Foreign key
Datetime	Datetime	Not null

Trash entity

Data name	Data type	
tid	integer(255)	Primary key
Userid	Varchar(255)	Foreign key
notename	Varchar(255)	Not null
notedetail	Varchar(255)	Not null
deleteddated	Datetime	Not null
Noteid	Integer9255)	Foreign key
Slug	Varchar(255)	Not null
Post_image	Varchar(255)	Not null

Todo entity

Data name	Data type	
todoid	Integer(255)	Primary key
todoname	Varchar(255)	Not null
tododate	Date	Not null
Userid	Varchar(255)	Foreign key
status	Varchar(255)	Not null

Notebooks entity

Data name	Data type	
notebookid	integer(255)	Primary key
notebookname	Varchar(255)	Not null
Created_at	Date	Not null
Userid	Varchar(255)	Foreign key

Note entity

Data name	Data type	
noteid	integer(255)	Primary key
notename	Varchar(255)	Not null
notedetail	Varchar(255)	Not null
notebookid	Varchar(255)	Foreign key
Post_image	Varchar(255)	Not null
Slug	Varchar(255)	Not null
Pin	Char(1)	Not null
Notecreateddot	Date	Not null

Entity-Relationship diagram

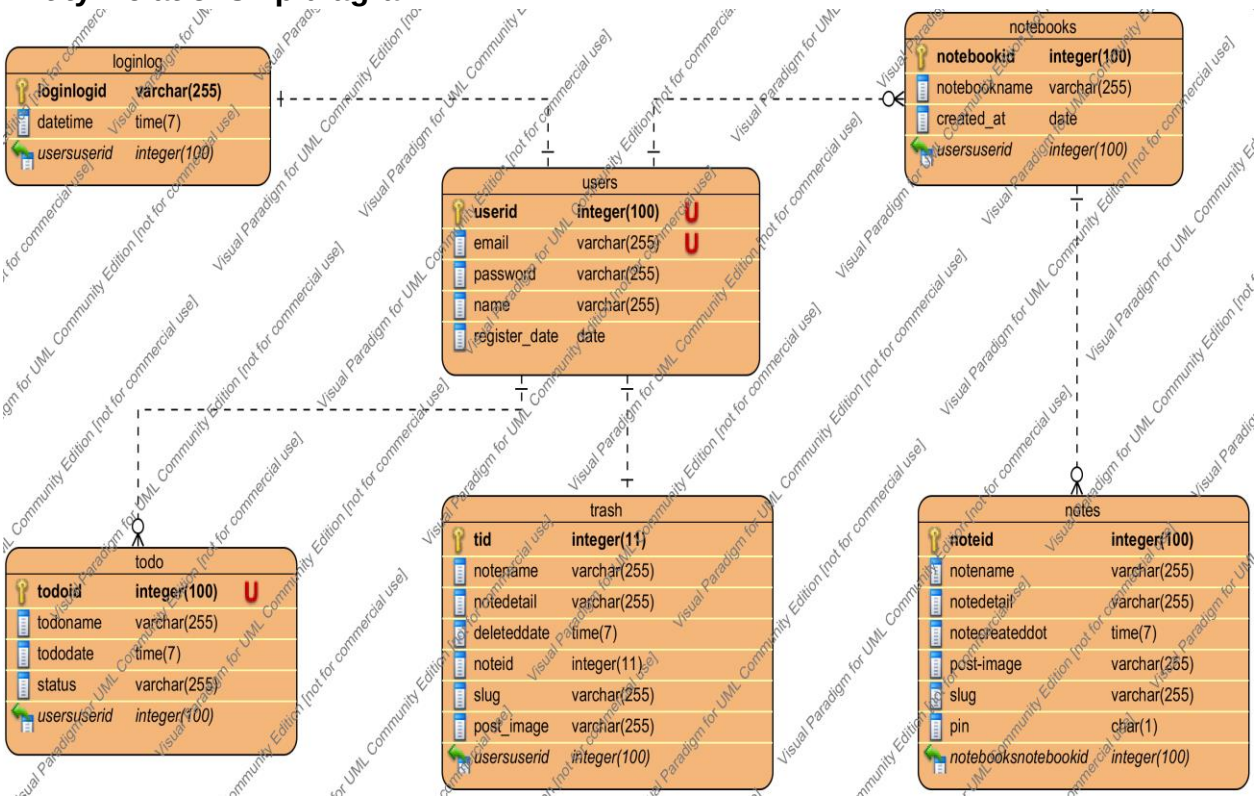
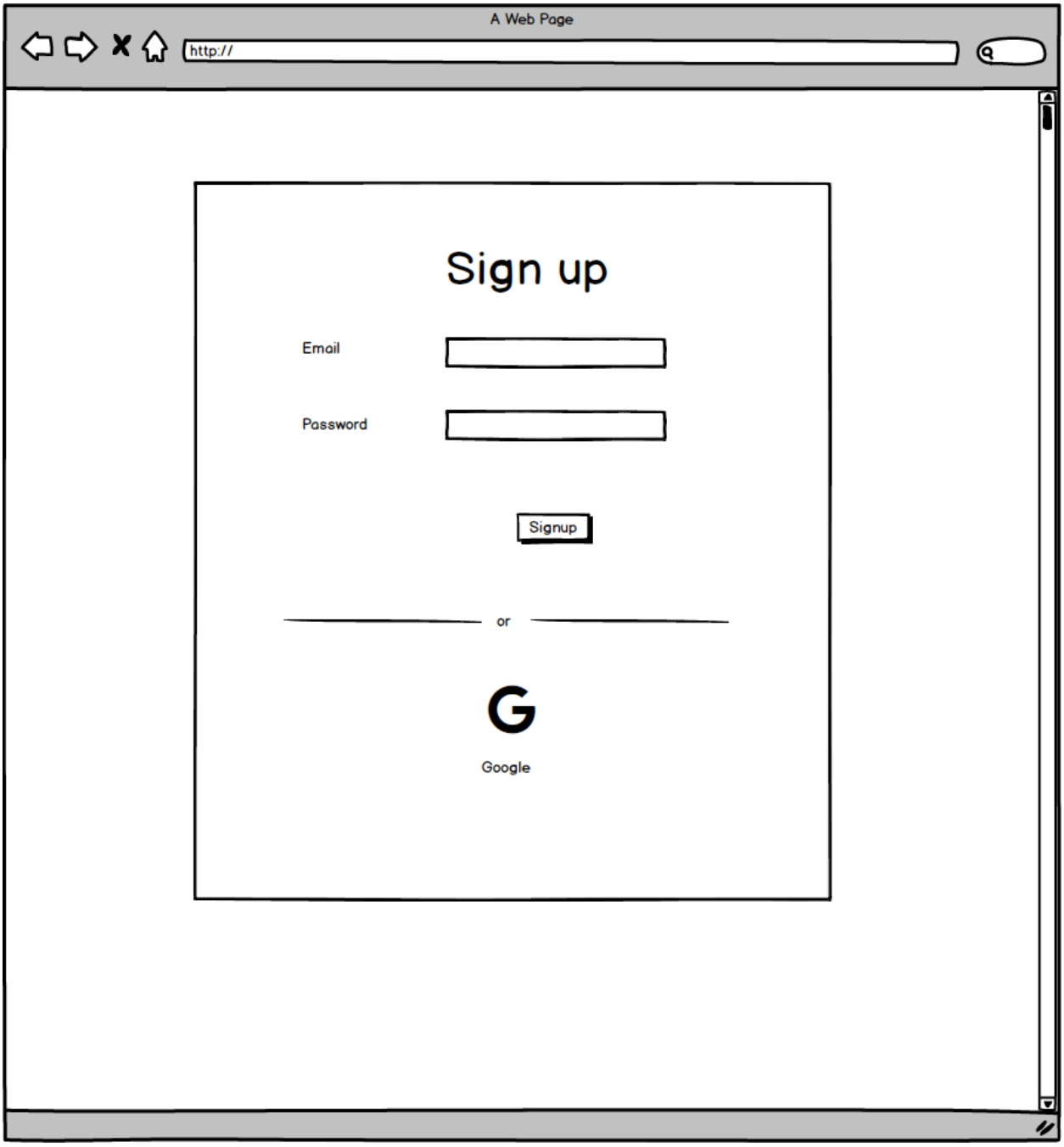
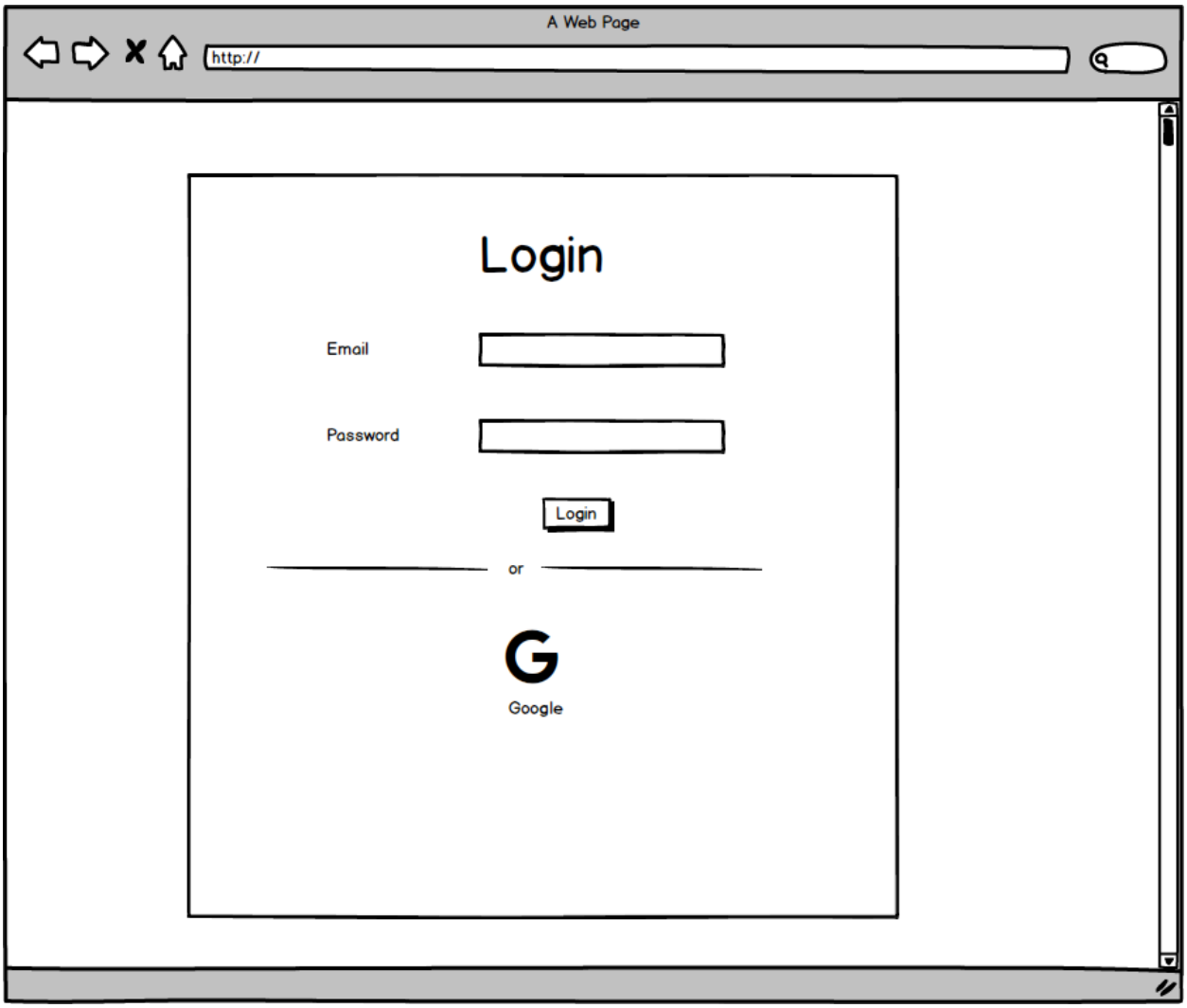
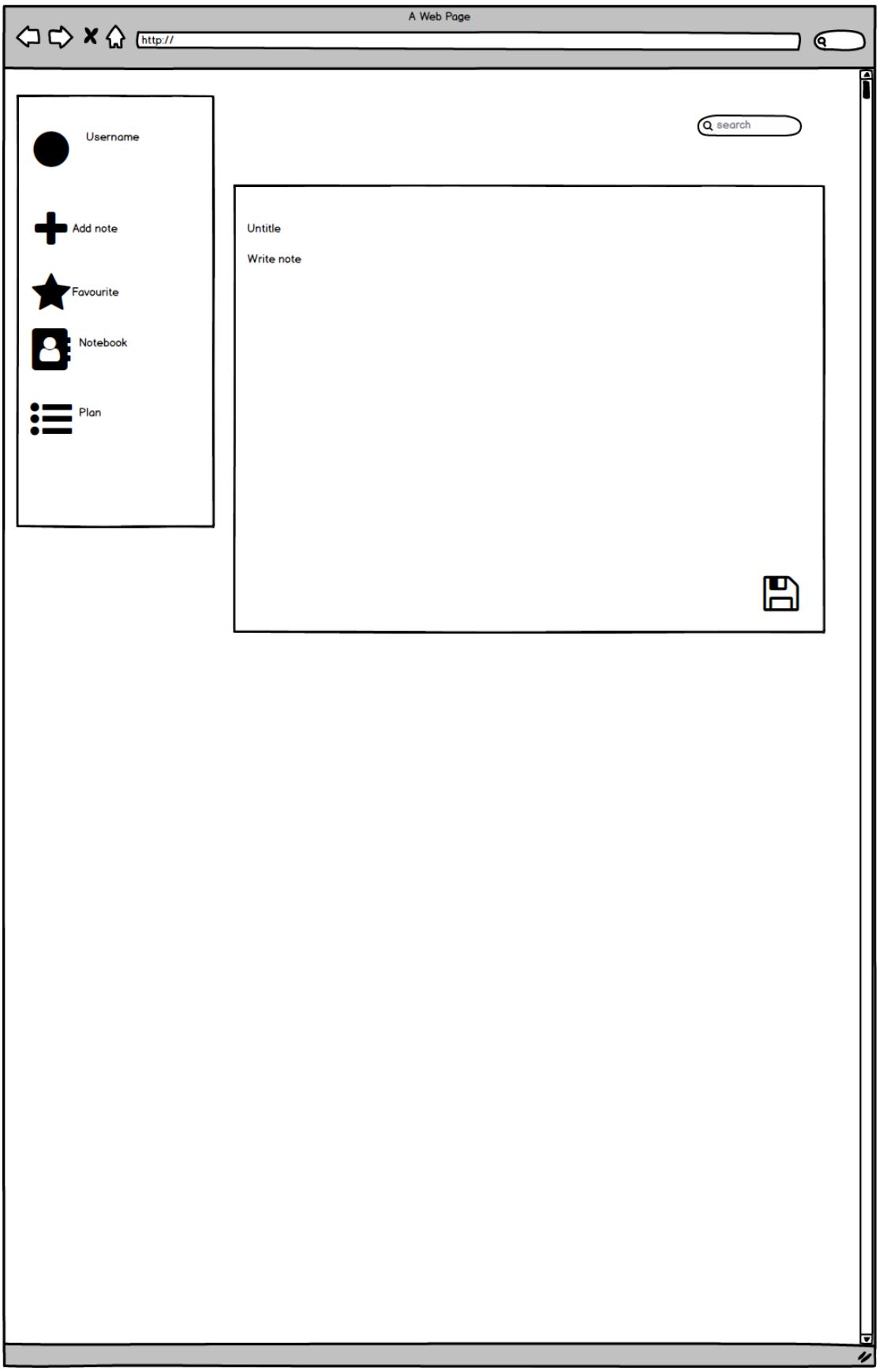


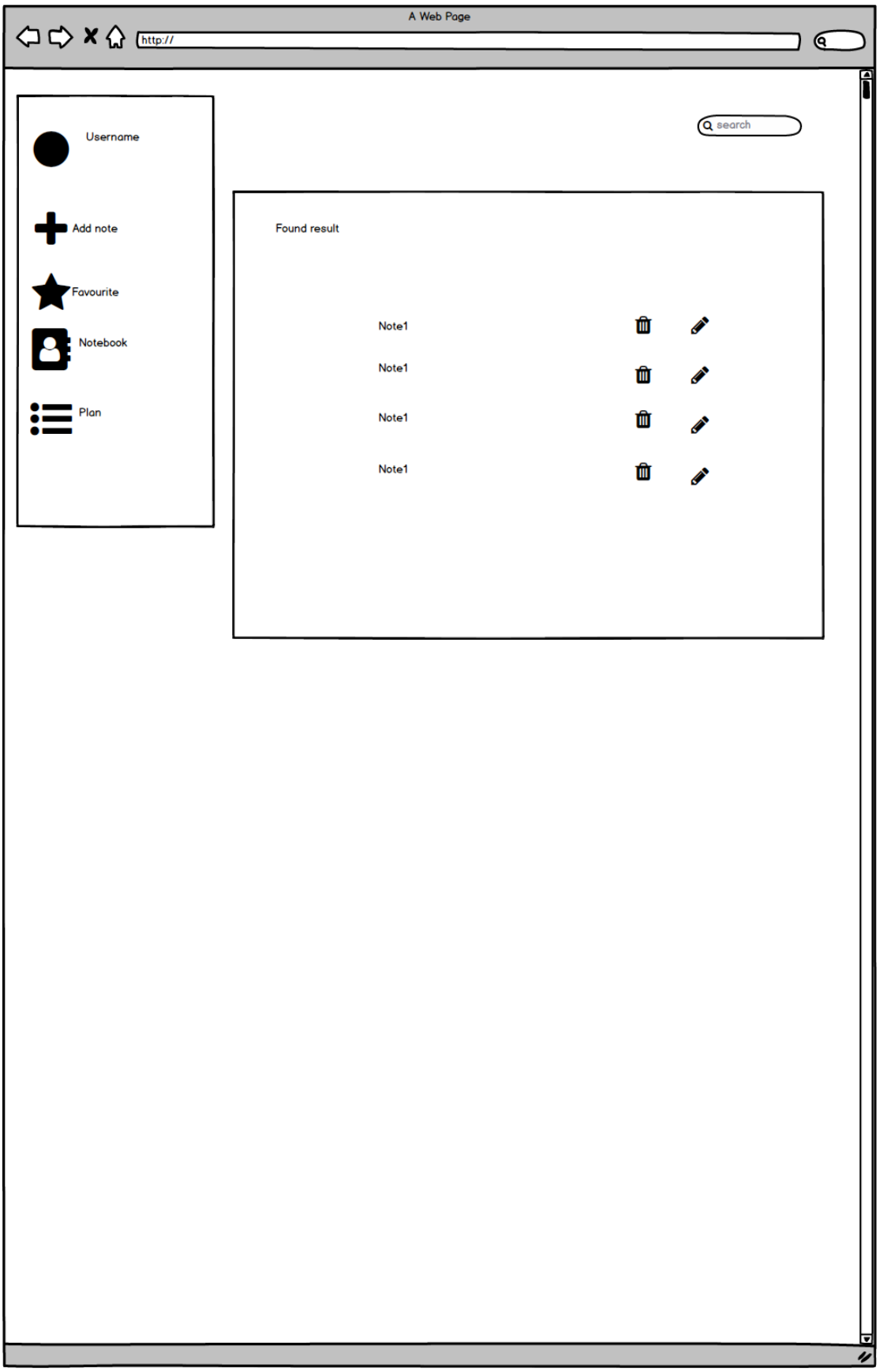
Figure 17 ER diagram for the purposed application

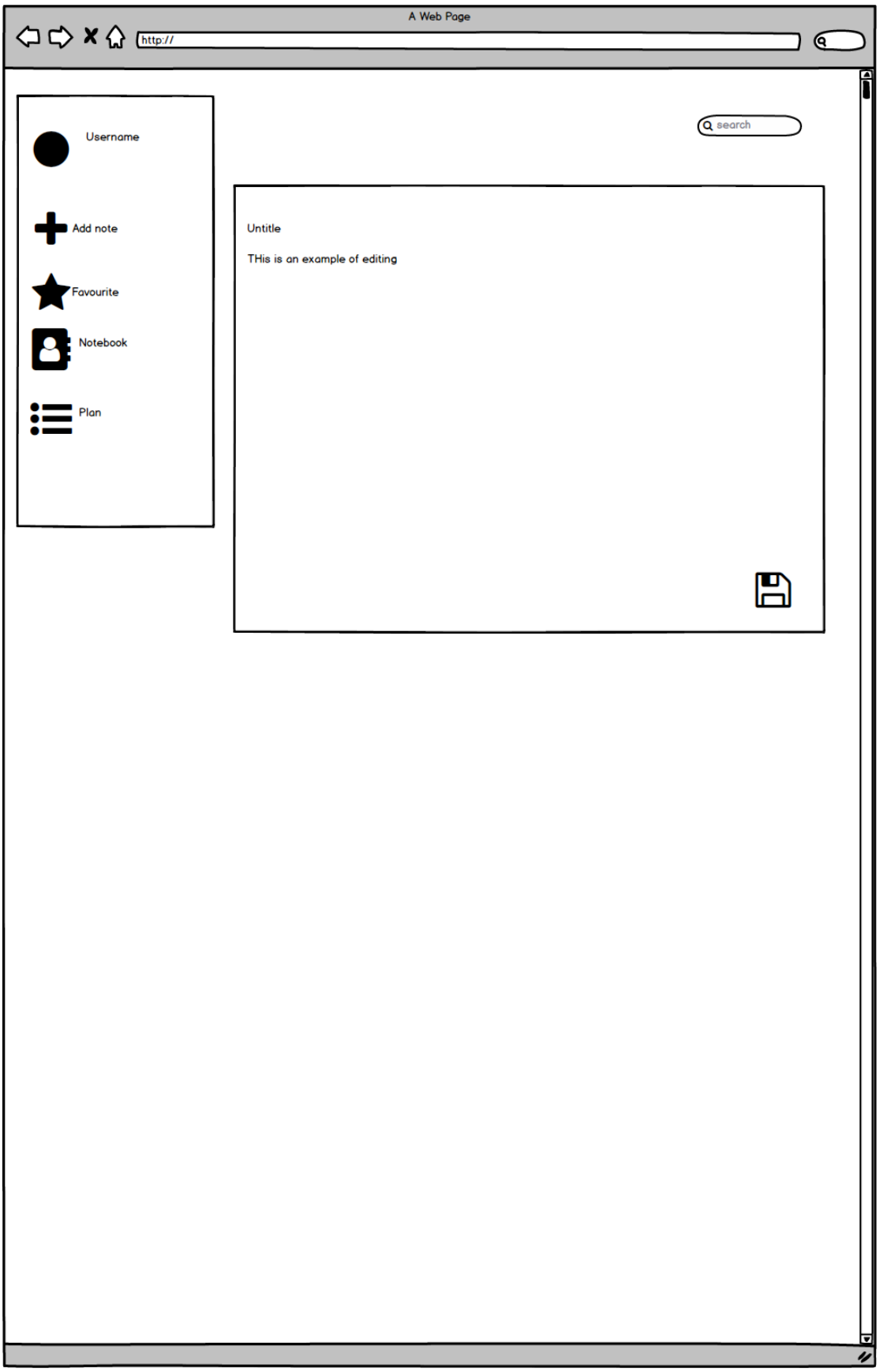
UI modeling




A Web Page
http://
Login
Email
Password
Login
or
Google


A Web Page
http://
Username
Add note
Favourite
Notebook
Plan
search
Untitle
Write note


A Web Page
http://
Username
Add note
Favourite
Notebook
Plan
search
Found result
Note1
Note1
Note1
Note1

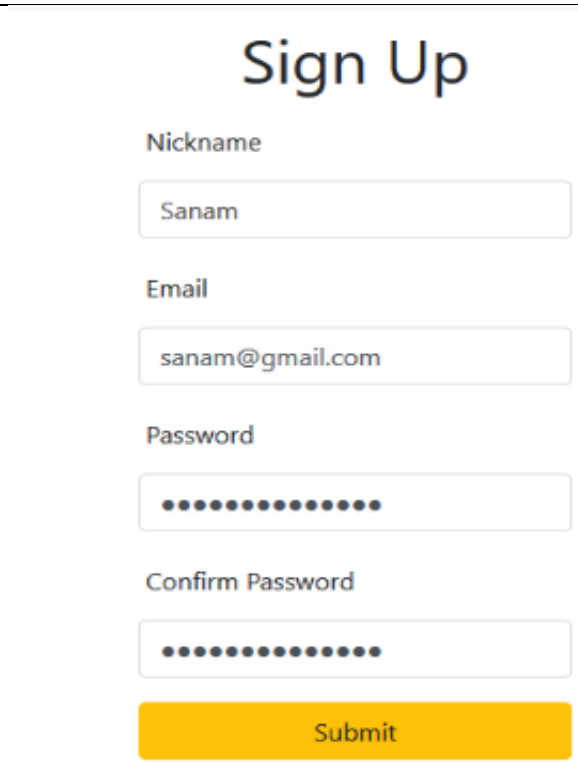
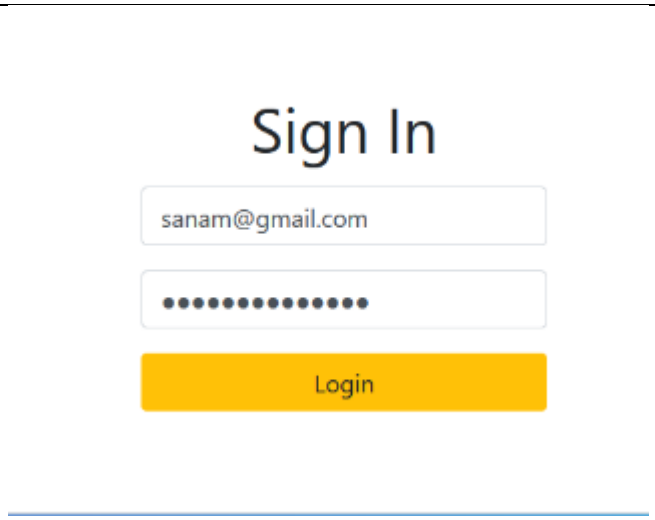

A Web Page
http://
Username
Add note
Favourite
Notebook
Plan
search
Untitle
THis is an example of editing

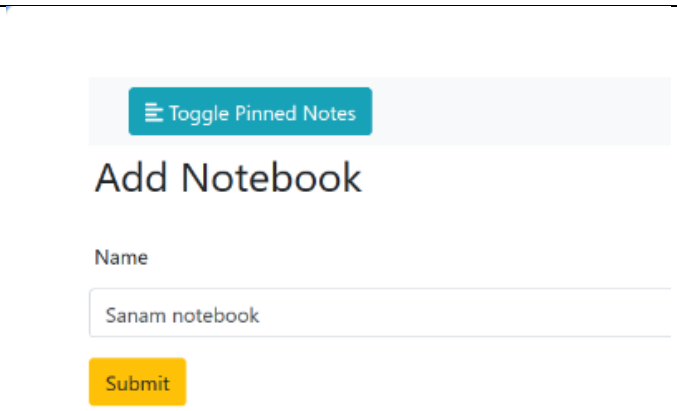
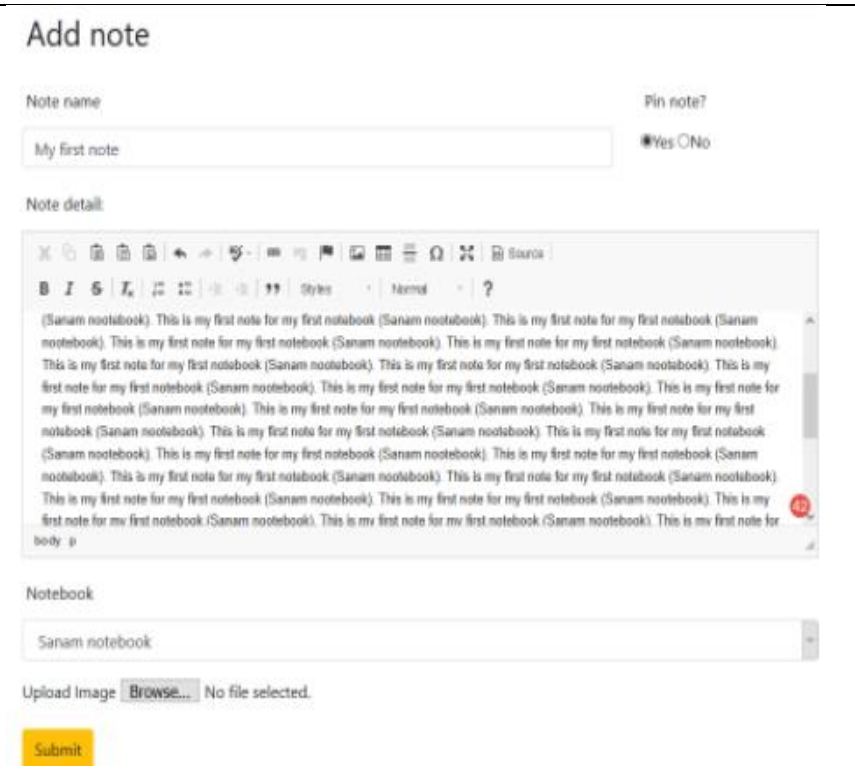
Chapter 5 Implementation

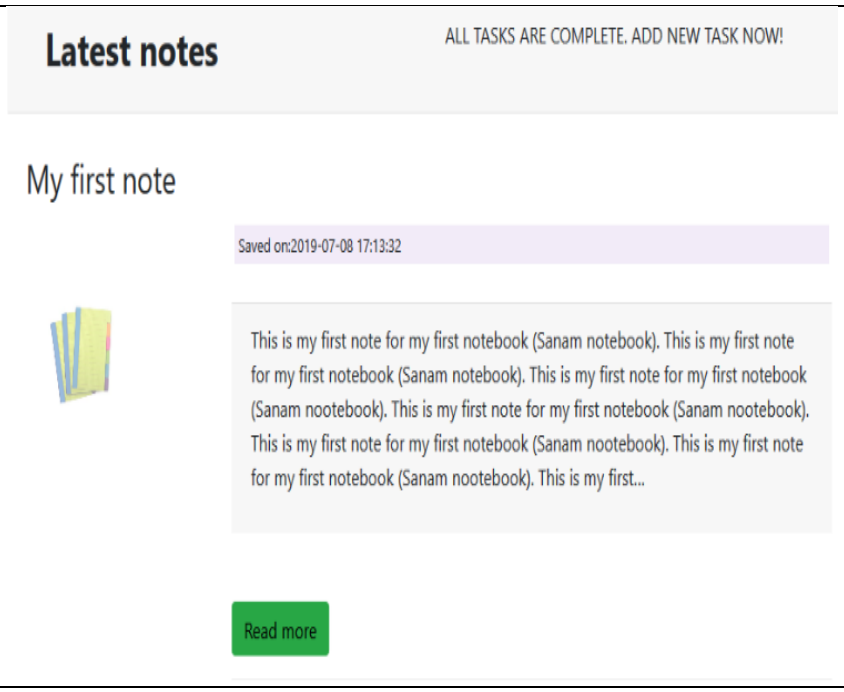
Code's picture

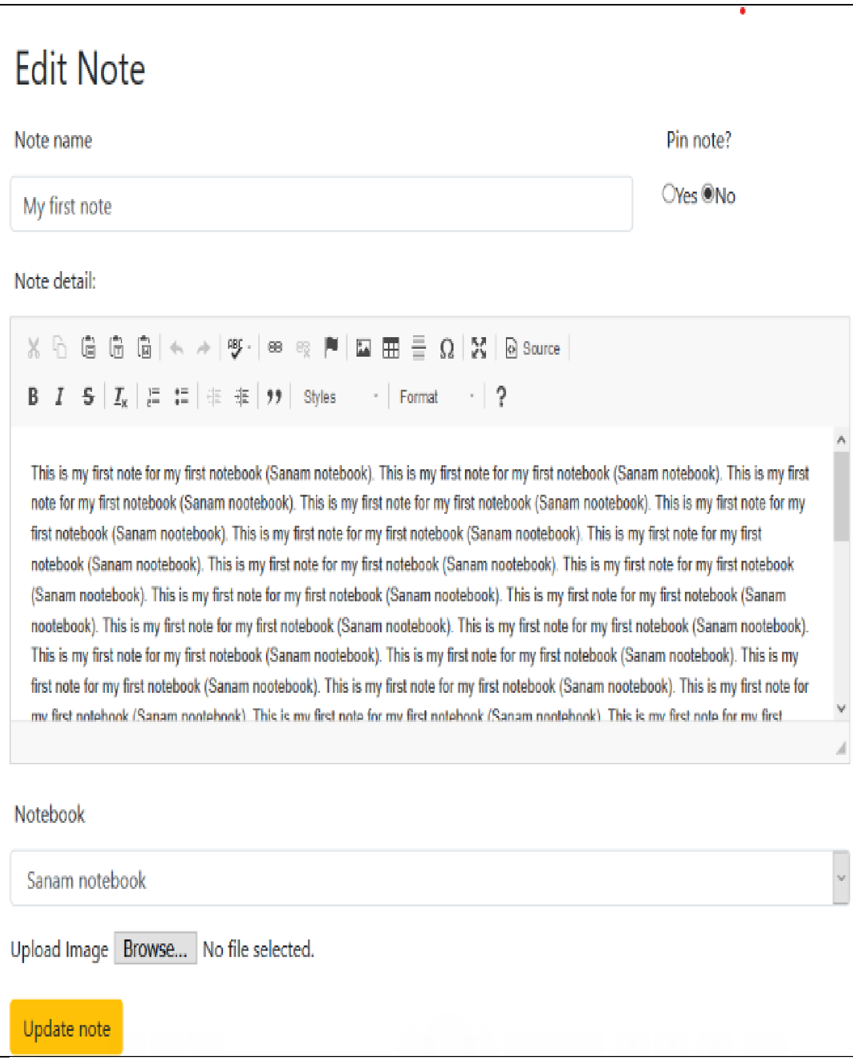
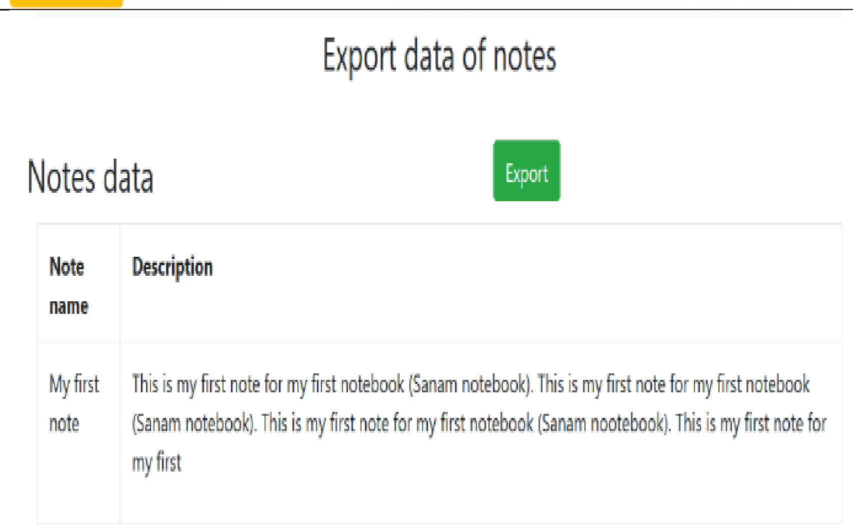
Chapter-6 Testing

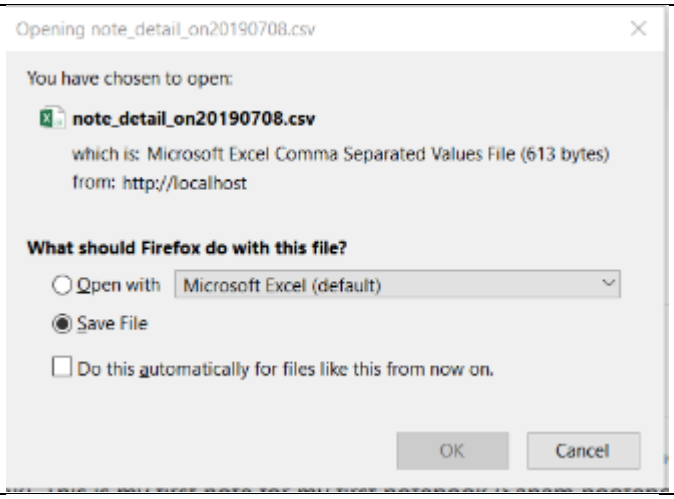
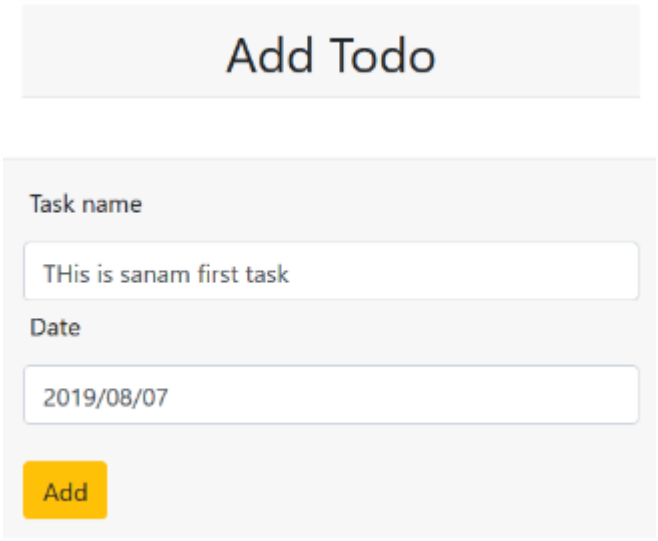
Black box testing of my system are done as following

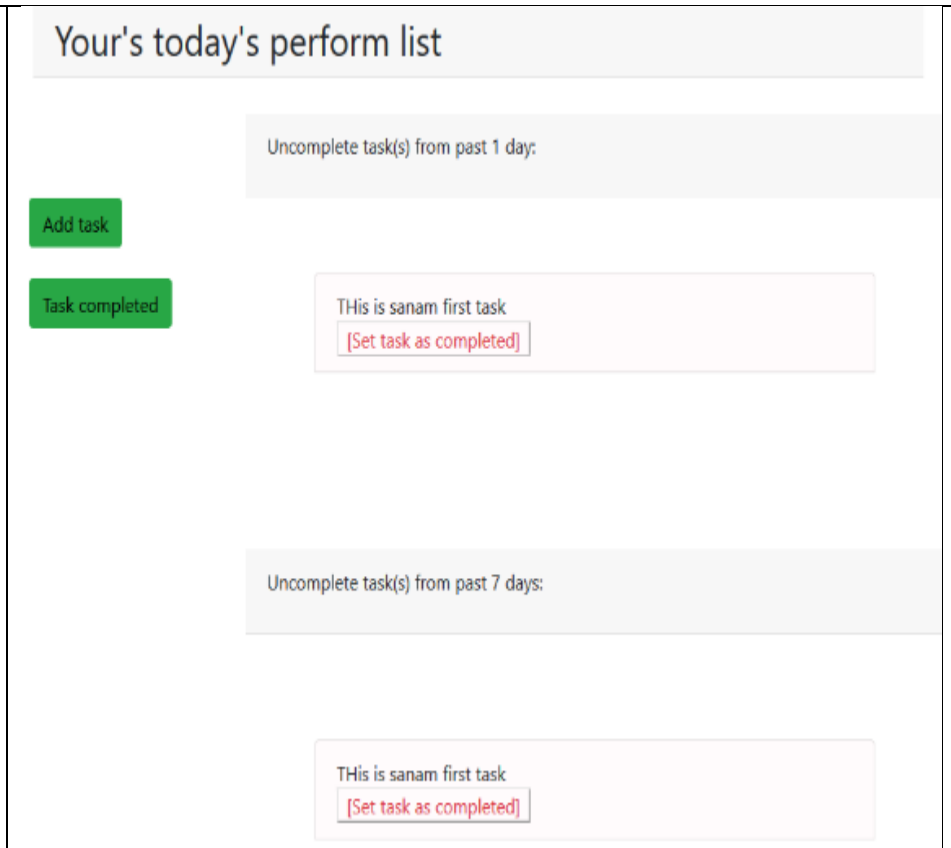
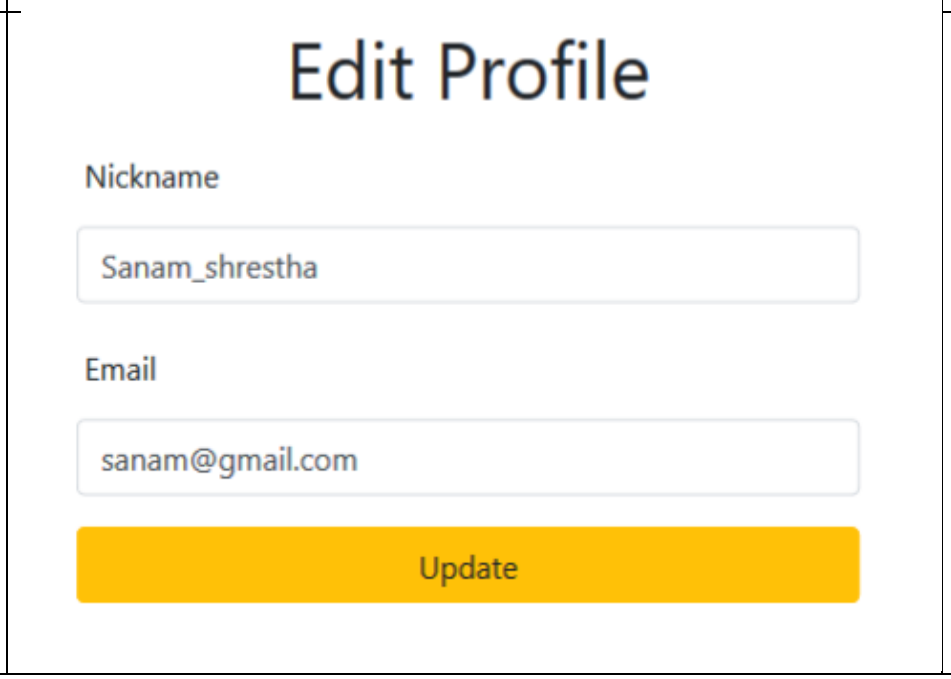
Functionality	View of testing as screenshot	Test result
Registration		Registration test was successful
Login		Login was successful

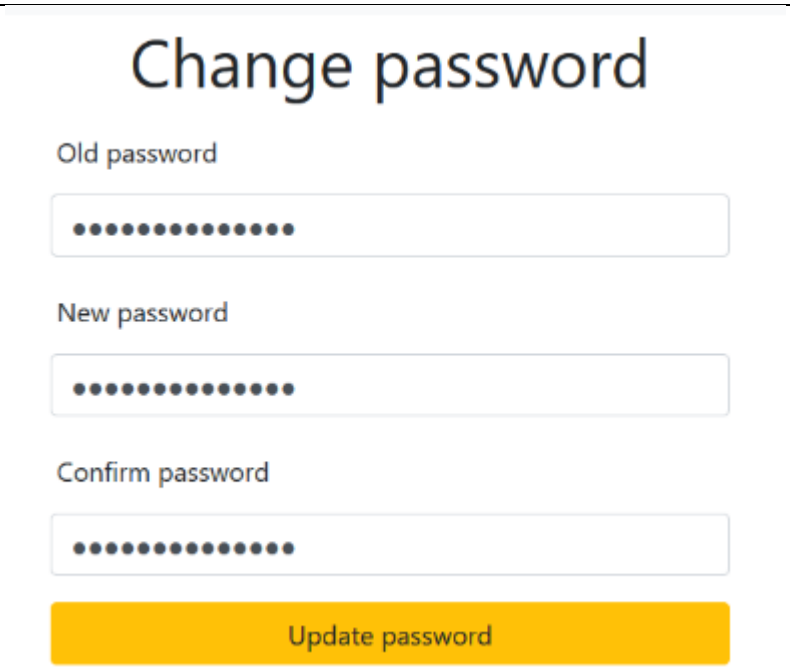
Add notebook		Notebook was added successful
Add note		Note was added successful

Display added note		Note was displayed successful

<table>
<tr>
<td>Edit note</td>
<td> 
</td>
<td>Note was edited successful</td>
</tr>
<tr>
<td>Export data</td>
<td> 
</td>
<td>Data was exported successful</td>
</tr>
</table>

Export data			Data was exported successful
Add task			Task was added successful

Display tasks		Tasks was displayed successful
Edit profile		Profile edited successful

Change password		Password updated successful

See remaining testing in appendix.

Chapter 6 Other project issues

Limitation of the project

Although todo have many functionality and features in it, this application does have some boundaries that this application can perform. First of all, this application is totally dependent on the internet so without internet it cannot perform its full system activities. Also, this application does not support image size greater than 2048 kb which makes it as major limitation.

Future work

The application which was purposed in very first has many functionalities. Major functional requirements are present in the application but some were left. Due to the project time constraint those missing functionalities are left over for future work or for coming next updates. Major future functionalities are

- Search notes through note name.
- Share note to other users.
- Pre-installed note template

Risk management

Risk implies future uncertainty about deviation from expected earnings or expected outcome. Risk measures the uncertainty that an investor is willing to take to realize a gain from an investment. (The Economic Times, 2019)

Risk Management is the process of identifying, analyzing and responding to risk factors throughout the life of a project and in the best interests of its objectives. Proper risk management implies control of possible future events and is proactive rather than reactive. (Stanleigh, 2019)

Steps for risk management are:

- Identify the risk
- Analyze the risk
- Rank the risk
- Teat the risk
- Monitor and review the risk (Development, 2019)

Risk type	Risk	Likelihood	Consequences	Impact	Action	Result
Time related	Miss project deadline	Medium	High	High	Analyze and Plan time estimation for project properly	Worked as per action suggestion
	Unexpected project scope expansions.	High	High	High	Understand project and prepare necessary	Worked as per action suggestion
Technical	Unable to gather functional requirements	High	Very high	Very high	Proper analysis during requirement gathering phase	Worked as per action suggestion
	Change in requirements	High	Very high	Very high	Proper analysis during requirement gathering phase	Worked as per action suggestion
	Difficult project modules integration.	High	Medium	High	Go for expert of that field	Worked as per action suggestion
Unavoidable	Natural disasters	Low	Very high	Very high	Regular backup of the system and data	Worked as per action suggestion

Configuration management

Configuration management (CM) is a governance and system engineering process for ensuring consistency among physical and logical assets in an operational environment. The configuration management process seeks to identify and track individual configuration items (CIs), documenting functional capabilities and interdependencies. Administrators, technicians and software developers can use configuration management tools to verify the effect a change to one configuration item has on other systems. (SearchITOperations, 2019)

My project documentation and necessary codes are located in my local computer and in GitHub repository (https://github.com/saanam/CP_00174357_sanam_shrestha.git). Under this codes and algorithms are reside and under subfolder name called CP all documentation is synced. The main reason of choosing GitHub as another primary storage is because it acts as version controller and is synced regularly through GitBash.

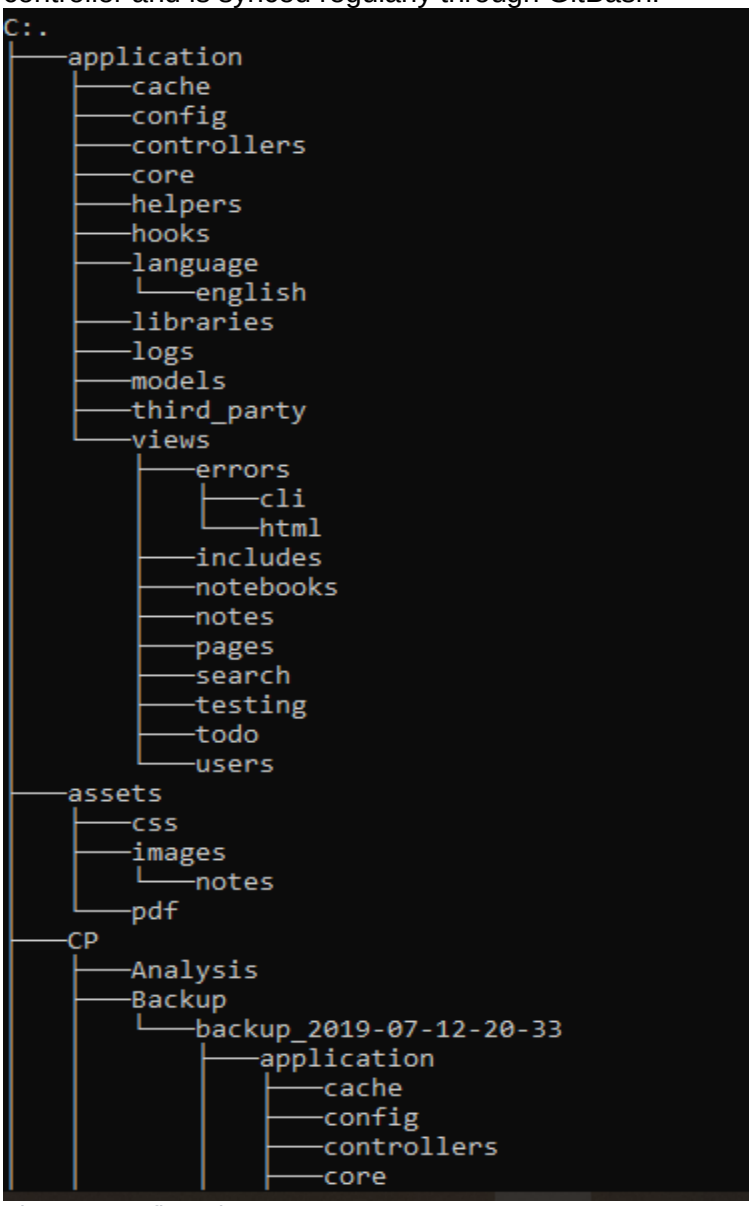


Figure 18 configuration part1

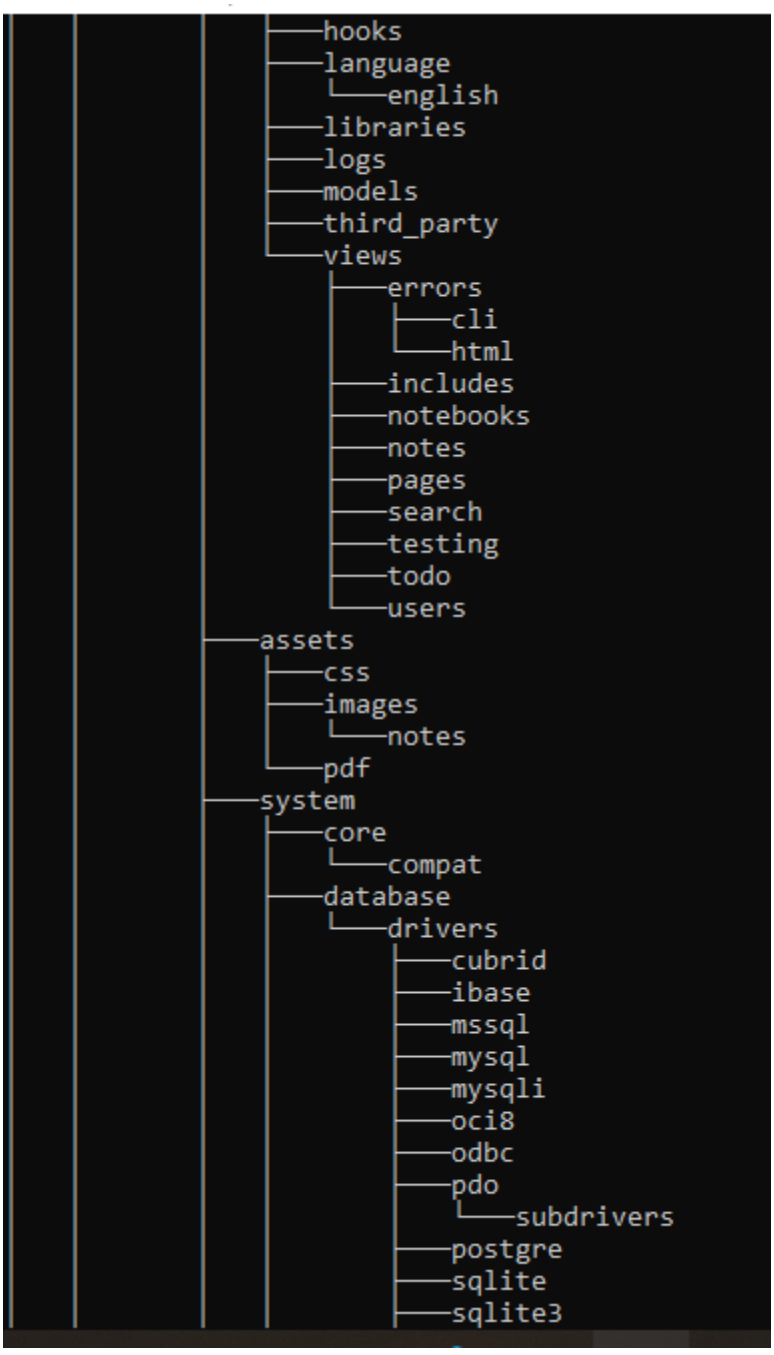


Figure 19 configuration part2

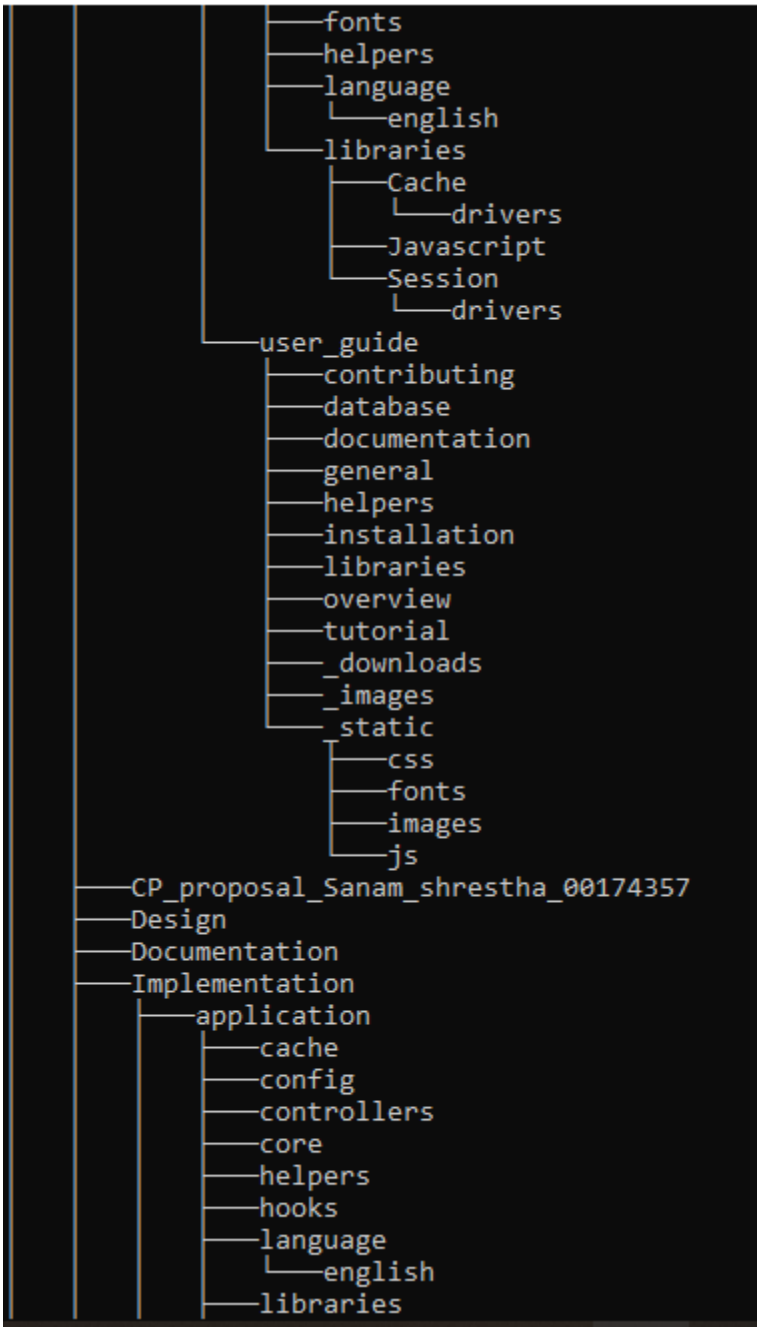


Figure 20 configuration part3

Figure 21 configuration part4

Figure 22 configuration part5

User manual

User manual to use Todo application is shown below:

Registration

To use the application, you must type **web address** in the **web bar**. Then go to **sign up**.

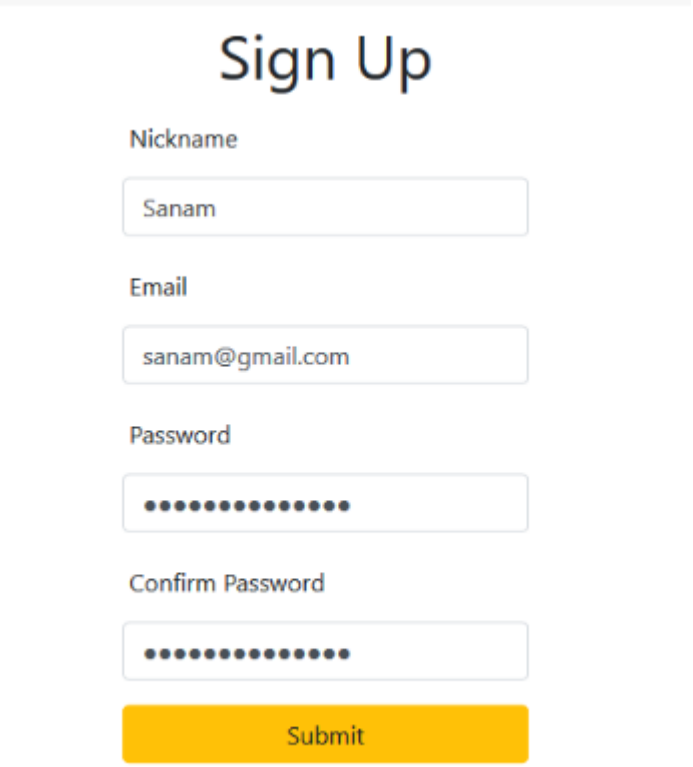


Figure 23 Registration of user

1. Fill all the field provided.
2. Password must contain at least 8 character, one uppercase letter, one number and one special character.
3. Then click to "**submit"** button. Now your account is created and ready for login.

Login

After registering account, page will redirect to login page.

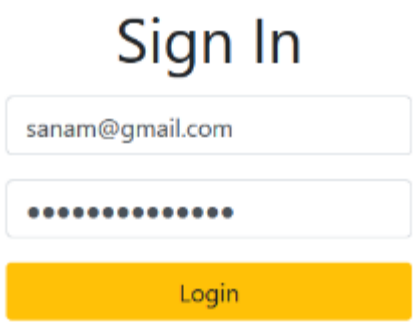
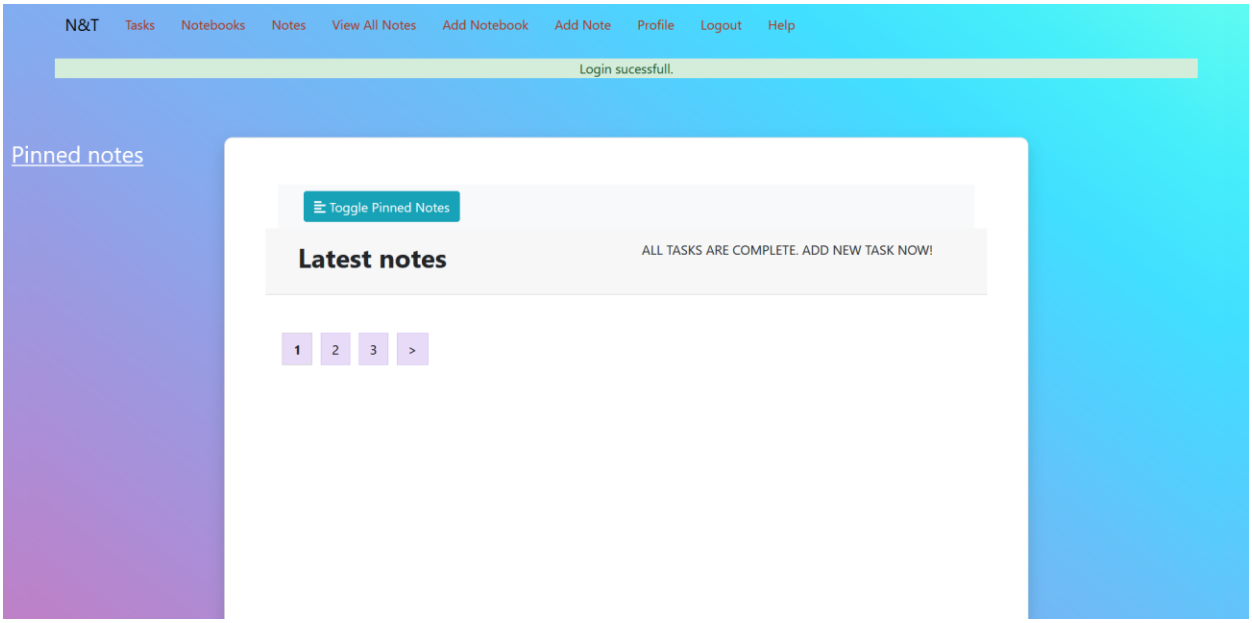


Figure 24 logging in

1. Now enter **email** in email field and **password** in password field.
2. If your login credentials are correct you will redirect to main page of application otherwise error message will be displayed.

Base page of application

This is a base page of Todo.



Adding notebook

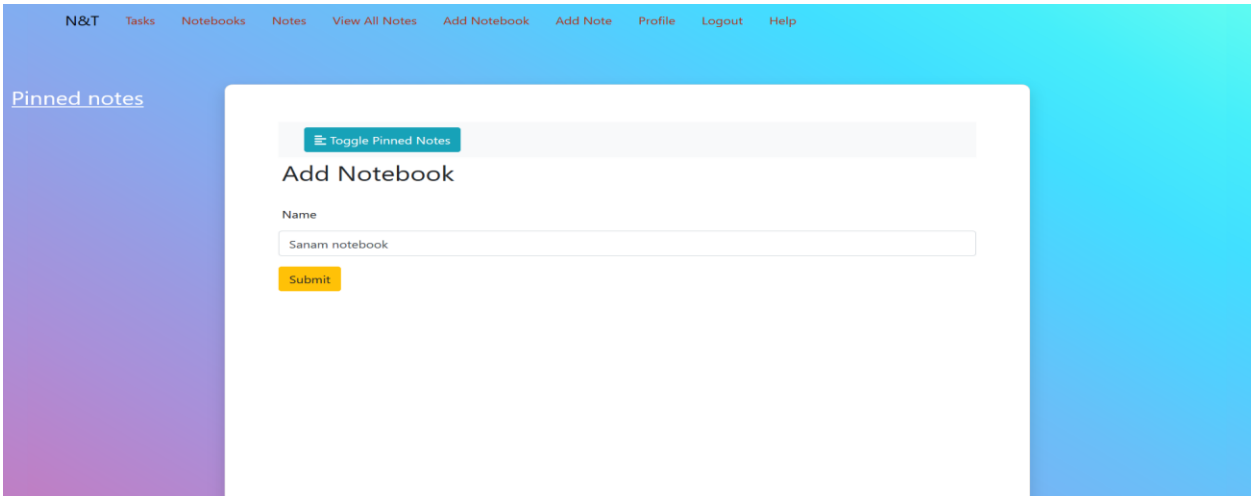


Figure 25 Adding notebook by user

After clicking on "**add notebook**", page is redirected to add notebook page where you can add notebook by typing title of notebook and clicking on "**Submit**" button.

Deleting notebook

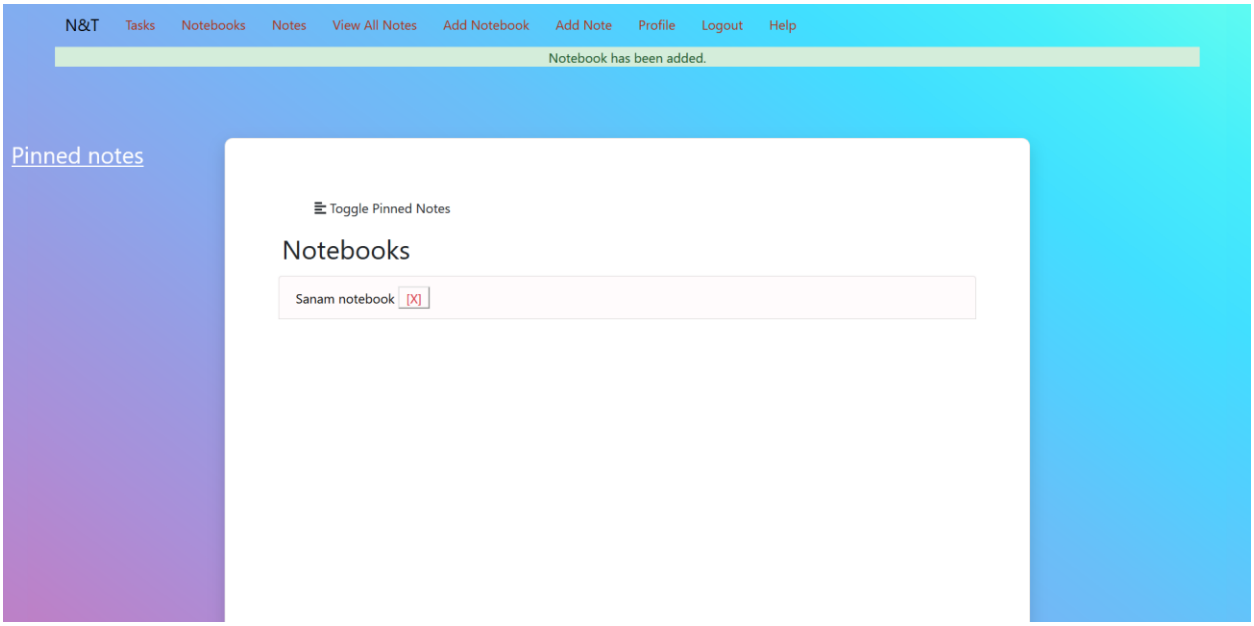


Figure 26 Deleting notebook by user

You can also delete notebook by visiting **"notebook"** section and all added notebook can be view there. Now user can delete any displayed notebook by clicking on **"[X]"** button**.**

Adding note

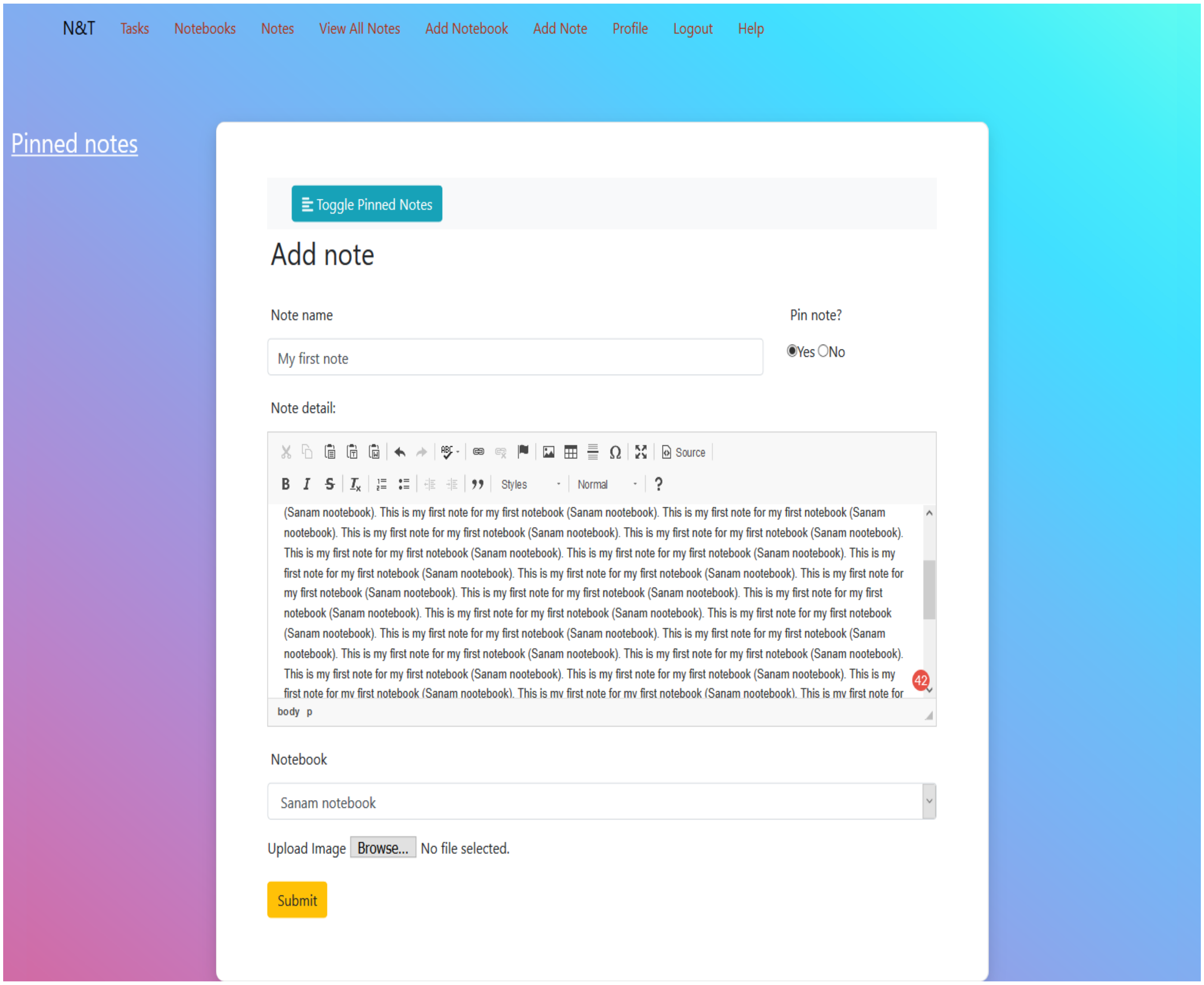


Figure 27 adding note by user

You can add note by clicking on **"add note"** section. After clicking on it page is redirected to add note page where user can add note. User can also select in which notebook this note will attach to. User can also attach image from their local pc (maximum picture size of 2048Kb). After clicking on **"Submit"** button note will be added.

Display all note

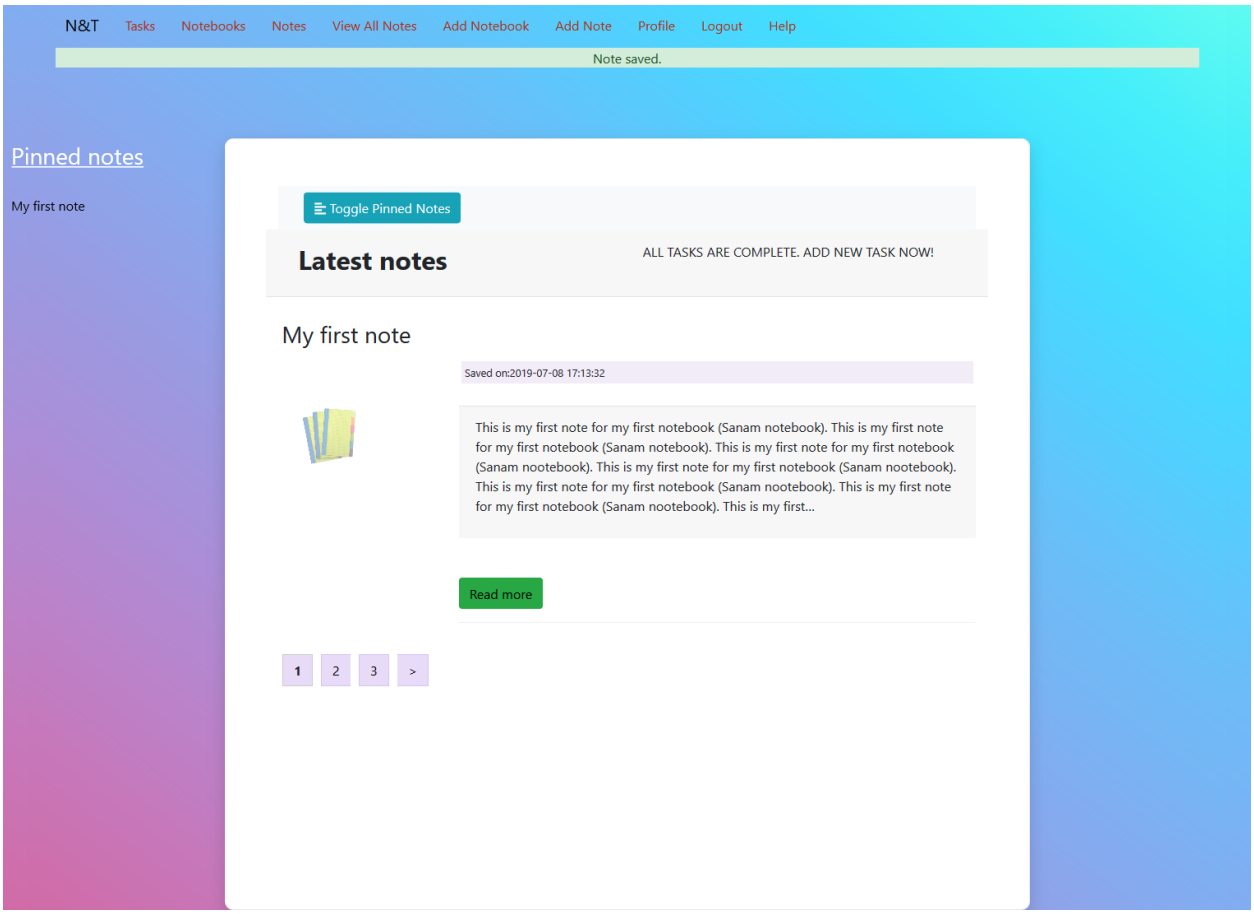


Figure 28 viewing note by user

Click on “Notes” to display all notes that has been added. Recent note that has been added will come at top. Click on “Read more” to view full note content.

View note full content

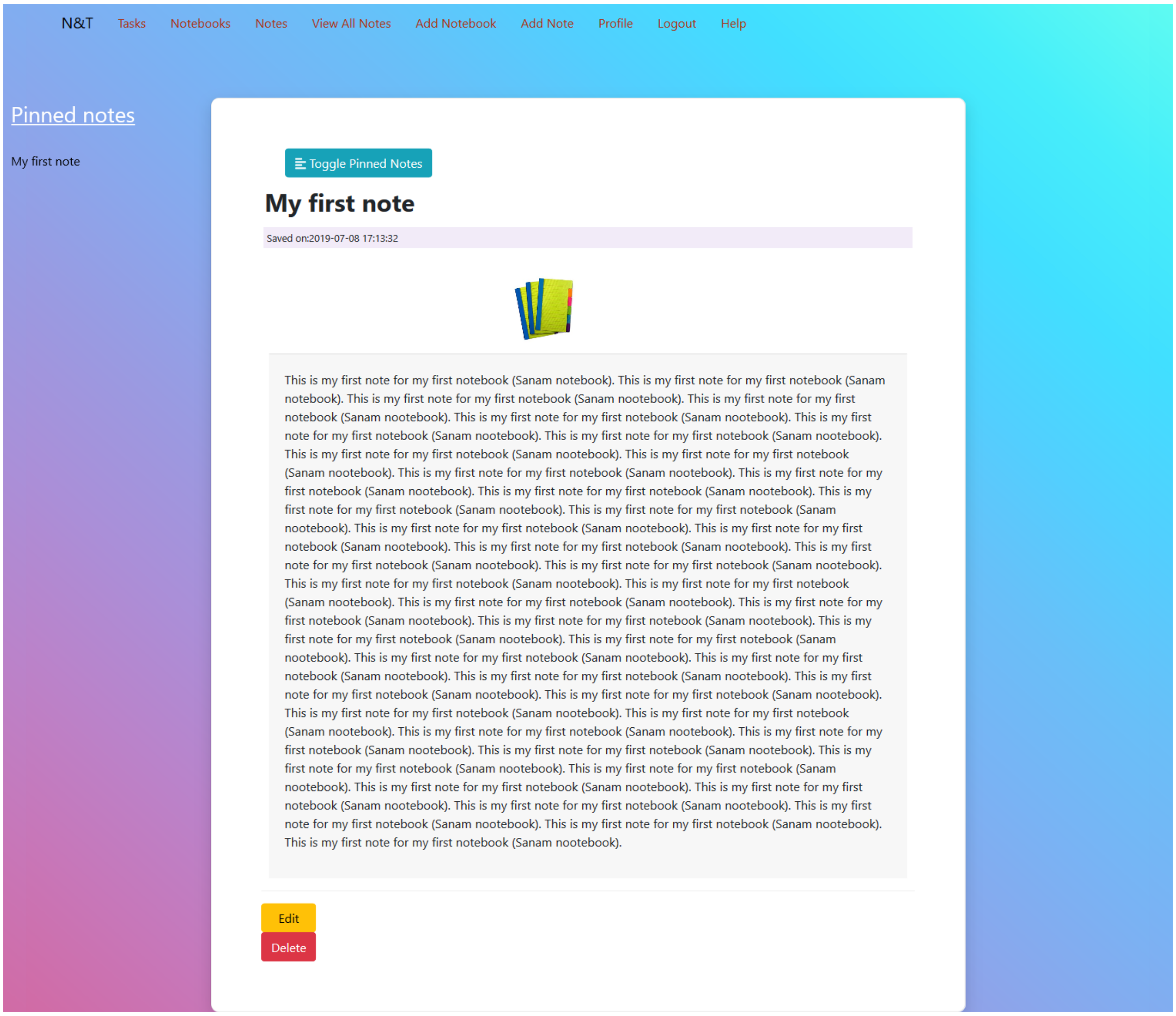


Figure 29 viewing note full content by user

Here you can view your full content of your note. Click on **"edit"** button to edit note and if you want to delete note click on **"delete"** button. Once note is deleted, deleted note will be store in "**trash"** section which can be accessed from **"profile"** section.

Editing a note

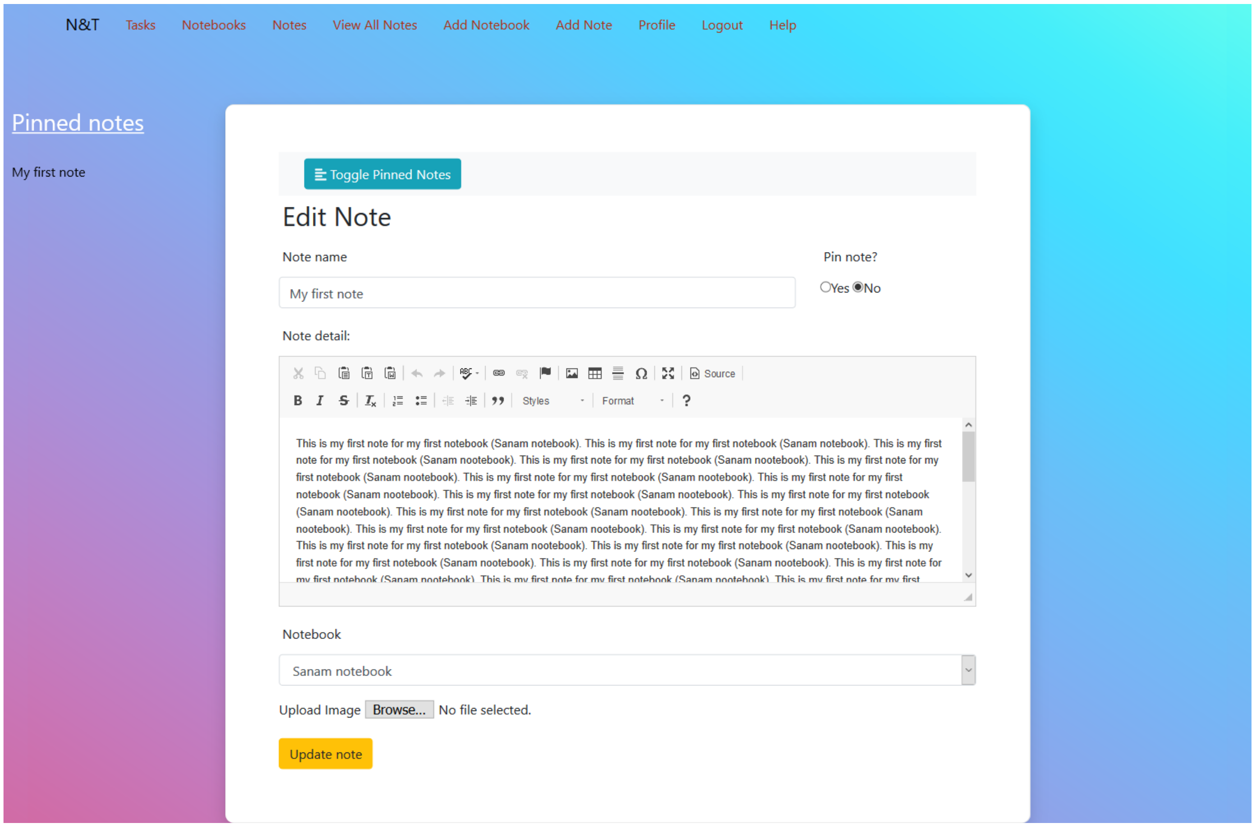


Figure 30 editing note by user

After clicking on edit button, a new page will load where note data will be loaded. Now it can be edited. After alteration complete clicking on **"update note"** will update the note.

Recovering data/deleting data from trash

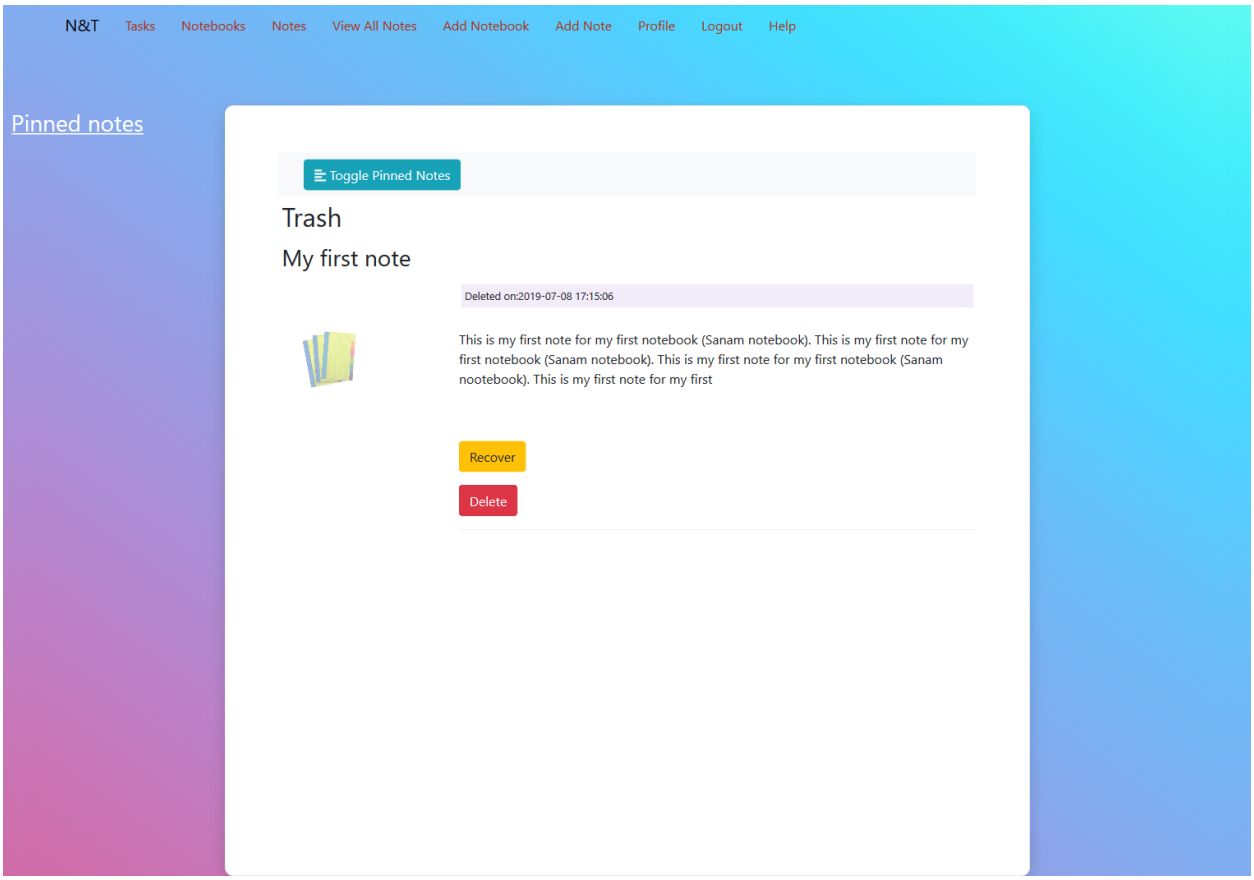


Figure 31 recovering data/ deleting data from trash

Any deleted note will be transferred into this section. Here user can recover deleted notes or delete permanently. To recover data then click on **recover** button. To delete data permanently click on **delete** button.

Baking up user data

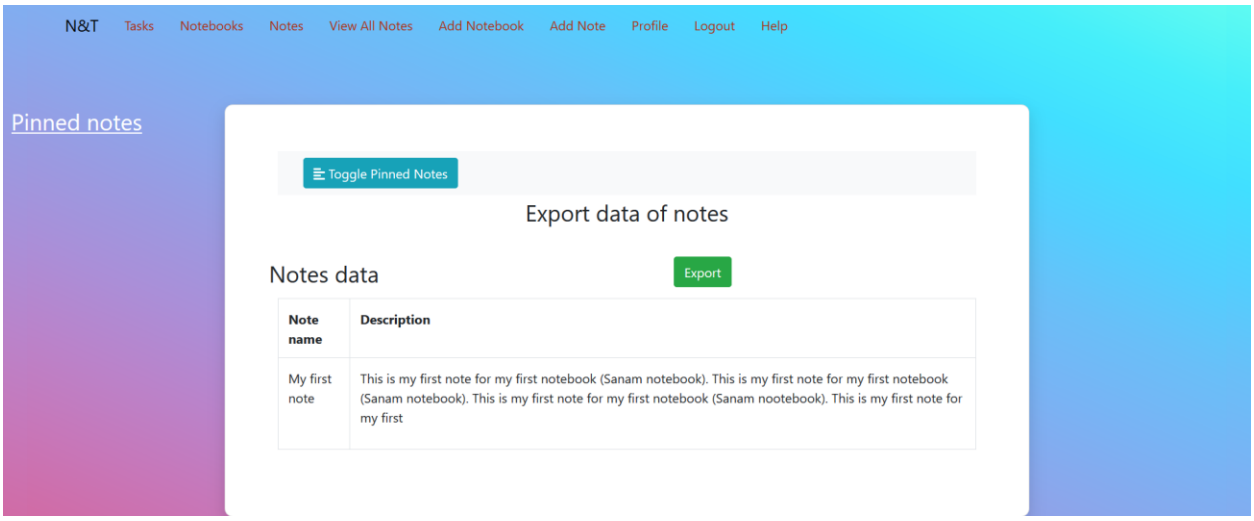


Figure 32 Back up user data.

To back up data, go to **"profile"** section and into it go to **"export data"**. now you can see what data you can back up. Once you hit **"export"** your data will be downloaded in .cvs format.

Add task

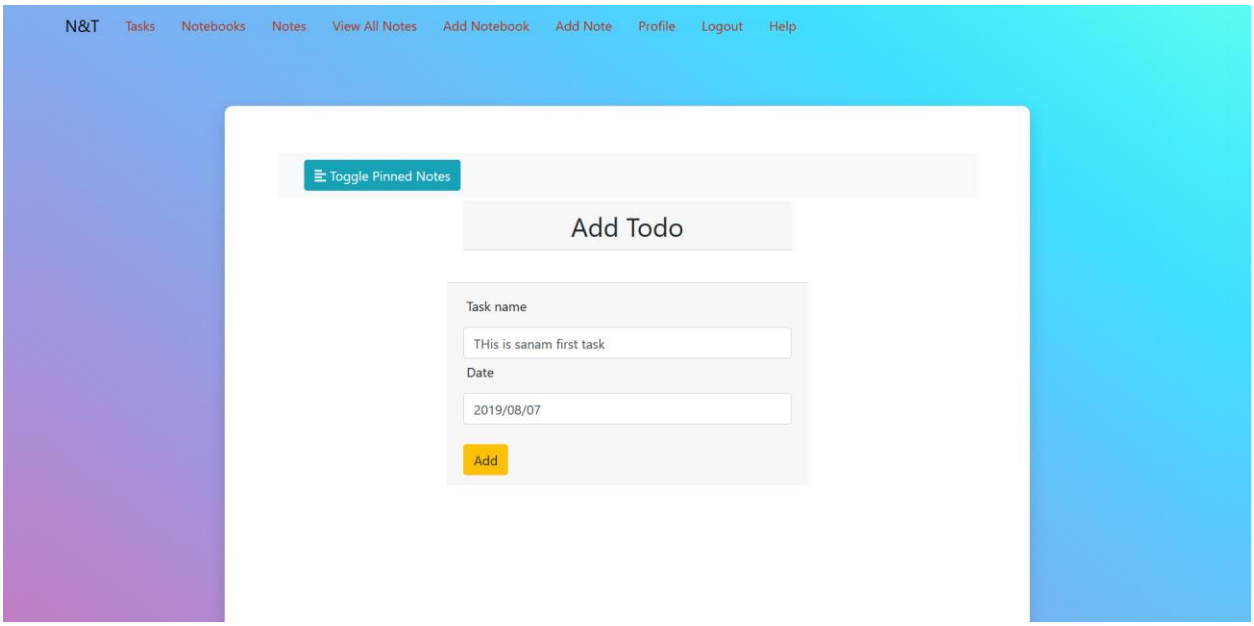


Figure 33 adding task by user

To add task, go to "**tasks**" section and inside it, go to add task. Once you are inside it you can add task by typing task name and task date. Once you hit add button your tasks will be added for that date.

View task

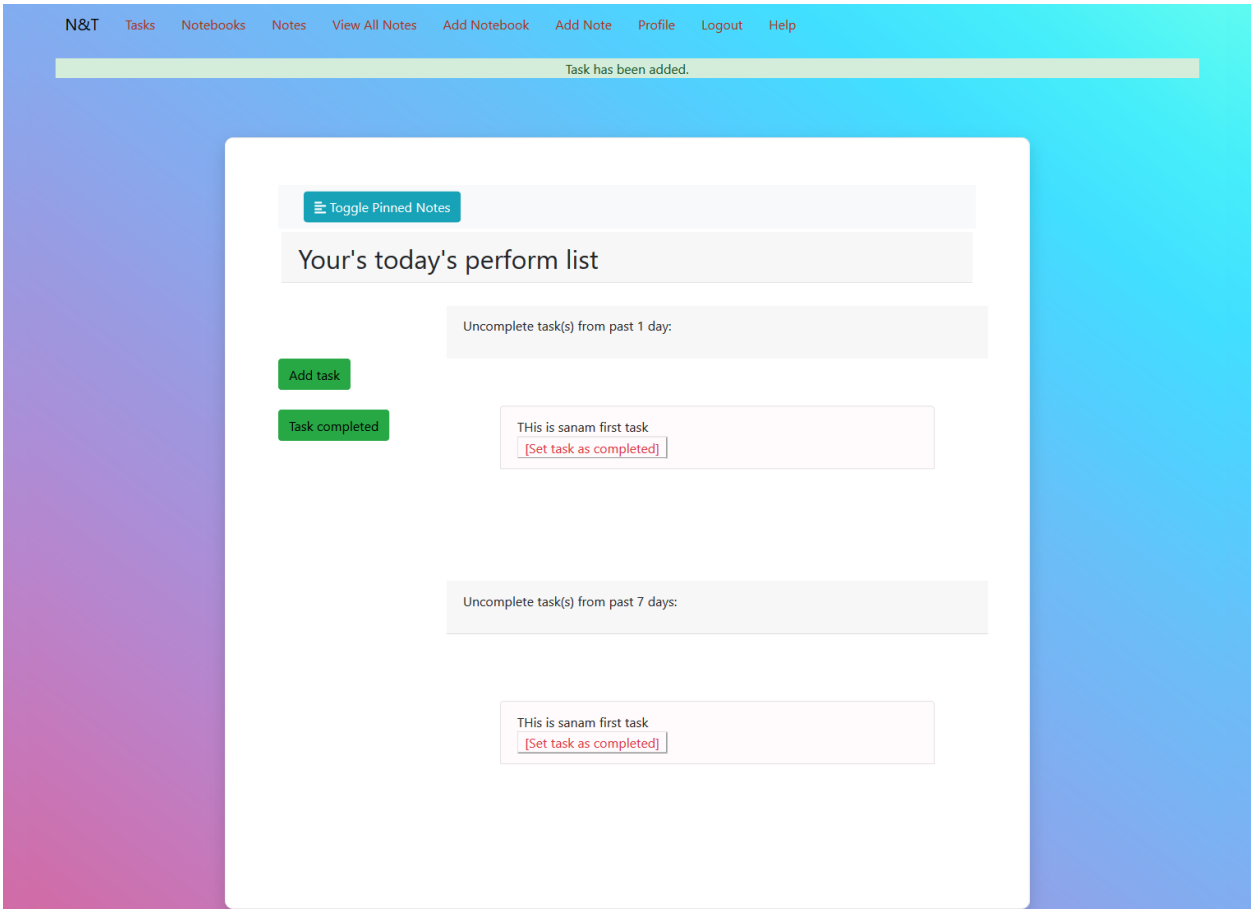


Figure 34 Viewing task by user

you can view task in tasks section. Here all your incomplete tasks from seven days from current date will be displayed. Once you completed your task, you can click on **"[Set task as complete]"** to set task as complete and it will vanish from the screen.

Chapter 7 Conclusion

This report contains every documentation of every phase. All the necessary activities are recorded and displayed as this report. First of all, the introduction of project is done in which about this application is described. Here general information about application is prepared where scope and domain is covered. After it, background of the project and justification is explained where previous state of the situation is shown. Justification of the situation is done providing proper reason about the project that could help to improve situation. Then overview of the project is described in which overall description of the project clarified. And at last of this chapter, aims and objectives of the purposed project is explained. The aim of this application is to provide a platform where users can save their day to day notes accordingly. To achieve this aim many objectives are set.

In chapter 2, analysis for the project to identify the requirements. A methodology is chosen to develop application. For this application Waterfall methodology is chosen and according to this method planning for the project is done. Functional and non-functional requirements are identified and MoSCoW prioritization is performed. After this overall system requirement for to run this application is identified. Use-case diagram and initial class diagram is analyzed and made.

In chapter 3, design for the application id done. Here final class diagram, activity diagram, sequence diagram, entity relationship diagram and user interface are developed. The developed diagram will help for implementation part.

In chapter 4, implementation of code is done. Here all the coding part is done and application is prepared for testing.

In chapter 5, testing is performed. Unit testing and black-box testing is performed to achieve the quality of the application. Any bug or error found in this stage will be immediately correctly.

In chapter 6, in this chapter remaining issue are identified. Issues like limitation of project, future work, risk management and configuration management is performed. What limitation of the project is identified. Any unfinished requirements of the project are listed and in upcoming future, those missing requirements are rolled out. Risk are identified and risk management planning is prepared. After this user manual is prepared to help user how to use the application.

References and bibliography

Stanleigh, M. (2019). *Risk Management…the What, Why, and How | | Business Improvement Architects*. [online] Bia.ca. Available at: https://bia.ca/risk-management-the-what-why-and-how/ [Accessed 12 Jul. 2019].

The Economic Times. (2019). *Definition of Risk | What is Risk ? Risk Meaning - The Economic Times*. [online] Available at: https://economictimes.indiatimes.com/definition/risk [Accessed 12 Jul. 2019].

Development, C. (2019). *What are the 5 Risk Management Process Steps?*. [online] Continuing Professional Development. Available at: https://continuingprofessionaldevelopment.org/risk-management-steps-in-risk-management-process/ [Accessed 12 Jul. 2019].

SearchITOperations. (2019). *What is configuration management (CM)? - Definition from WhatIs.com*. [online] Available at: https://searchitoperations.techtarget.com/definition/configuration-management-CM [Accessed 12 Jul. 2019].

Segue Technologies. (2019). *Waterfall vs. Agile: Which Methodology is Right for Your Project?*. [online] Available at: https://www.seguetech.com/waterfall-vs-agile-methodology/ [Accessed 11 Jul. 2019].

Yuwantoko, A., Daniel, S. and Ahmadiyah, A. (2017). Pembuatan Kakas Bantu untuk Mendeteksi Ketidaksesuaian Diagram Urutan (Sequence Diagram) dengan Diagram Kasus Penggunaan (Use Case Diagram). *Jurnal Teknik ITS*, 6(1).

Abdulkareem, S., Ali, N., Admodisastro, N. and Sultan, A. (2017). Class Diagram Critic: A Design Critic Tool for UML Class Diagram. *Advanced Science Letters*, 23(11), pp.11567-11571.

Chirkova, K. (2002). World Knowledge and Natural Language Analysis (review). *Language*, 78(1), pp.191-192.

Appendix

Code's picture

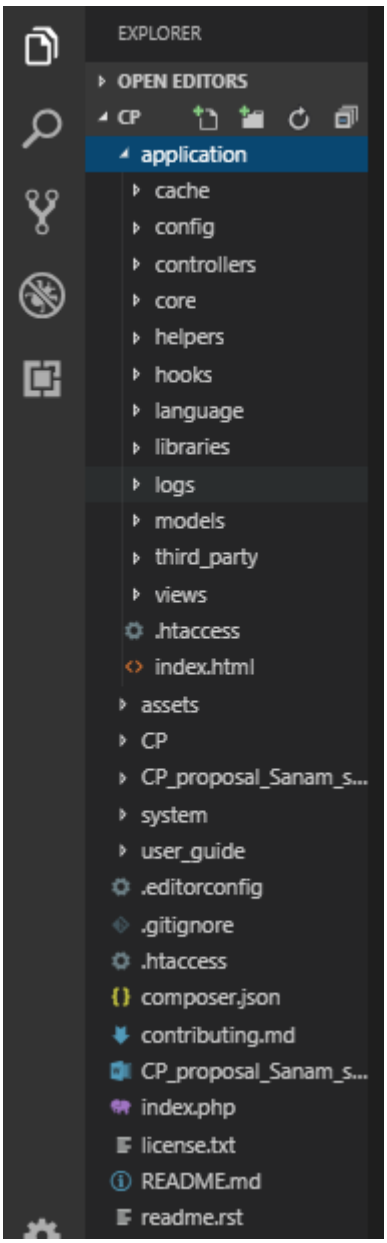


Figure 35 all folders and files for the system

header.php

application › views › includes › header.php

```
<!--defining this is html5-->
<!DOCTYPE html>
<html lang="en">

<!--start of head section-->
<head>
  <meta charset="utf-8" />
  <link rel="apple-touch-icon" sizes="76x76" href="../assets/img/apple-icon.png">
  <link rel="icon" type="image/png" href="../assets/img/favicon.png">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
  <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0, shrink-to-fit=no' name='viewport' />

  <!--start of title section-->
  <!-- Title Page-->
  <title>
    Welcome to Note & Todolist
  </title>
  <!--end of title section-->
<!-- Scrollbar Custom CSS -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/malihu-custom-scrollbar-plugin/3.1.5/jquery.mCustomScrollbar.min.css">
  <!--     Fonts and icons     -->
  <link href="assets/css/material-design-iconic-font.min.css" rel="stylesheet" media="all">
  <link href="assets/font-awesome.min.css" rel="stylesheet" media="all">

  <link rel="stylesheet" href="https://bootswatch.com/flatly/bootstrap.min.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css"
  integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
  <script src="http://cdn.ckeditor.com/4.11.4/standard/ckeditor.js"></script>

  <!-- CSS Files -->
  <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
  <link href="assets/select2/select2.min.css" rel="stylesheet" media="all">
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
  integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js"
```

Figure 36 code of header part1

header.php

application › views › includes › header.php

```
38    integrity="sha384-xrRywqdh3PHs8keKZN+8zzc5TX0GRTLCcmivcbNJWm2rs5C8PRhcEn3czEjhAO9o" crossorigin="anonymous"></script>
39
40    <!--  jQuery -->
41    <script src="https://cdn.jsdelivr.net/npm/pdfjs-dist@2.0.943/build/pdf.min.js"></script>
42    <script type="text/javascript" src="https://code.jquery.com/jquery-1.11.3.min.js"></script>
43
44    <!-- Bootstrap Date-Picker Plugin -->
45    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"
46    integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiwGPXr1jddIhOegiu1FwO5qRGvFXOdJZ4" crossorigin="anonymous">
47    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.4.1/js/bootstrap-datepicker.min.js"></script>
48    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.4.1/css/bootstrap-datepicker3.css"/>
49    <link rel="stylesheet" href="<?php echo base_url(); ?>/assets/css/style.css">
50
51    <!-- FOnt Awsome JS-->
52    <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/solid.js"
53    integrity="sha384-tzzSw1/Vo+0N5UhStP3bvwWPq+uvzCMfrN1fEFe+xBmv1C/AtVX5K0uZtmcHitFZ" crossorigin="anonymous"></script>
54    <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/fontawesome.js"
55    integrity="sha384-6OIrr52G08NpOFSZdxxz1xdNSndlD4vdcf/q2myIUVO0VsqaGHJsB0RaBE01VTOY" crossorigin="anonymous"></script>
56
57  </head>
58  <!--end of head section-->
59
60  <!--start of body section-->
61  <body>
62  <!--start of nav part-->
63  <nav class="navbar navbar-expand-lg navbar-custom py-0">
64    <a class="navbar-brand" href="<?php echo base_url(); ?>users/login">N&T</a> <!--echo base_url refers to localhost/cp-->
65      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
66      aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
67        <span class="navbar-toggler-icon"></span>
68      </button>
69
70        <div class="collapse navbar-collapse" id="navbarSupportedContent">
71          <ul class="navbar-nav mr-auto">
72
73              <!-- checks in session data for displaying these part -->
74              <?php if(!$this->session->userdata('logged_in')) : ?>
```

Figure 37 code of header part2

header.php

application ▸ views ▸ includes ▸ header.php

```
75              <a class="nav-link" href="<?php echo base_url(); ?>users/login">Login<span class="sr-only">(current)</span></a>
76              <a class="nav-link" href="<?php echo base_url(); ?>users/register">Register <span class="sr-only">(current)</span></a>
77            <?php endif; ?>
78            <!-- end of if condition-->
79
80            <!-- checks in session data for displaying these part-->
81            <!-- start of if condition-->
82            <?php if($this->session->userdata('logged_in')) : ?>
83                <a class="nav-link" href="<?php echo base_url(); ?>todo">Tasks<span class="sr-only">(current)</span></a>
84                <a class="nav-link" href="<?php echo base_url(); ?>notebooks">Notebooks<span class="sr-only">(current)</span></a>
85                <a class="nav-link" href="<?php echo base_url(); ?>notes">Notes<span class="sr-only">(current)</span></a>
86                <a class="nav-link" href="<?php echo base_url(); ?>notes/note_search">View All Notes<span class="sr-only">(current)</span></a>
87                <a class="nav-link" href="<?php echo base_url(); ?>notebooks/add">Add Notebook<span class="sr-only">(current)</span></a>
88                <a class="nav-link" href="<?php echo base_url(); ?>notes/add">Add Note<span class="sr-only">(current)</span></a>
89                <a class="nav-link" href="<?php echo base_url(); ?>profile">Profile<span class="sr-only">(current)</span></a>
90                <a class="nav-link" href="<?php echo base_url(); ?>users/logout">Logout<span class="sr-only">(current)</span></a>
91
92            <?php endif; ?>
93            <!-- end of if condition-->
94            <a class="nav-link" href="<?php echo base_url(); ?>pages/help">Help<span class="sr-only">(current)</span></a>
95        </ul>
96      </div>
97  </nav>
98  <!--end of nav part-->
99  <div class = "text text-center">
100   <!-- flash message-->
101
102     <?php if($this->session->flashdata('user_registered')): ?>
103       <?php echo '<p class="alert-success">'.$this->session->flashdata('user_registered').'</p>'; ?>
104     <?php endif; ?>
105
106     <?php if($this->session->flashdata('note_added')): ?>
107       <?php echo '<p class="alert-success">'.$this->session->flashdata('note_added').'</p>'; ?>
108     <?php endif; ?>
109
110     <?php if($this->session->flashdata('note_updated')): ?>
111       <?php echo '<p class="alert-success">'.$this->session->flashdata('note_updated').'</p>'; ?>
```

Figure 38code of header part3

header.php ×

application ▸ views ▸ includes ▸ header.php

```
112     <?php endif; ?>
113
114     <?php if($this->session->flashdata('notebook_added')): ?>
115       <?php echo '<p class="alert-success">'.$this->session->flashdata('notebook_added').'</p>'; ?>
116     <?php endif; ?>
117
118     <?php if($this->session->flashdata('note_deleted')): ?>
119       <?php echo '<p class="alert-warning">'.$this->session->flashdata('note_deleted').'</p>'; ?>
120     <?php endif; ?>
121
122     <p class="text"><strong><?php if($this->session->flashdata('login_failed')): ?>
123       <?php echo '<p class="alert-danger">'.$this->session->flashdata('login_failed').'</p>'; ?>
124     <?php endif; ?></strong></p>
125
126     <?php if($this->session->flashdata('user_loggedin')): ?>
127       <?php echo '<p class="alert-success">'.$this->session->flashdata('user_loggedin').'</p>'; ?>
128     <?php endif; ?>
129
130     <?php if($this->session->flashdata('user_logout')): ?>
131       <?php echo '<p class="alert-success">'.$this->session->flashdata('user_logout').'</p>'; ?>
132     <?php endif; ?>
133
134     <?php if($this->session->flashdata('notebook_deleted')): ?>
135       <?php echo '<p class="alert-warning">'.$this->session->flashdata('notebook_deleted').'</p>'; ?>
136     <?php endif; ?>
137
138     <?php if($this->session->flashdata('password_updated')): ?>
139       <?php echo '<p class="alert-primary">'.$this->session->flashdata('password_updated').'</p>'; ?>
140     <?php endif; ?>
141
142     <?php if($this->session->flashdata('profile_updated')): ?>
143       <?php echo '<p class="alert-primary">'.$this->session->flashdata('profile_updated').'</p>'; ?>
144     <?php endif; ?>
145
146     <?php if($this->session->flashdata('todo_alert')): ?>
147       <?php echo '<p class="alert-primary">'.$this->session->flashdata('todo_alert').'</p>'; ?>
148     <?php endif; ?>
```

Figure 39 code of header part4

header.php

application › views › includes › header.php

```
146     <?php if($this->session->flashdata('todo_alert')): ?>
147       <?php echo '<p class="alert-primary">'.$this->session->flashdata('todo_alert').'</p>'; ?>
148     <?php endif; ?>
149
150     <?php if($this->session->flashdata('old_passwordwrong')): ?>
151       <?php echo '<p class="alert-danger">'.$this->session->flashdata('old_passwordwrong').'</p>'; ?>
152     <?php endif; ?>
153
154     <?php if($this->session->flashdata('task_added')): ?>
155       <?php echo '<p class="alert-success">'.$this->session->flashdata('task_added').'</p>'; ?>
156     <?php endif; ?>
157
158   </div>
159   <!--start of container-fluid class-->
160   <div class = "wrapper">
161
162   <?php if($this->session->userdata('logged_in')): ?>
163     <nav id="sidebar">
164       <div class="sidebar-header">
165                 <h3><u>Pinned notes</u></h3><br />
166                 <?php foreach((array) $pinn as $pin): ?>
167                   <li><a href="<?php echo site_url('/notes/'.$pin['slug']); ?>"><?php echo $pin['notename']; ?></a><br /><br />
168                 <?php endforeach; ?>
169       </div>
170     </nav>
171     <?php endif; ?>
```

Figure 40 code of header part5

footer.php

application › views › includes › footer.php

```
</div>
        <!--script for pop up date picker in add date-->
        <script>
            $(document).ready(function(){
            var date_input=$('input[name="date"]'); //our date input has the name "date"
            var container=$('.bootstrap-iso form').length>0 ? $('.bootstrap-iso form').parent() : "body";
            var options={
                format: 'yyyy/mm/dd',
                container: container,
                todayHighlight: true,
                autoclose: true,
            };
            date_input.datepicker(options);
            })
        </script>
        <!--end of scrip-->
        <script type="text/javascript">
            $(document).ready(function () {
                $("#sidebar").mCustomScrollbar({
                    theme: "minimal"
                });

            $('#sidebarCollapse').on('click', function () {
                $('#sidebar, #content').toggleClass('active');
                $('.collapse.in').toggleClass('in');
                $('a[aria-expanded=true]').attr('aria-expanded', 'false');
            });
        });
        </script>

        <!-- script to bring various tools for editing-->
        <script>
            CKEDITOR.replace( 'editor1' );
        </script>
        <!--end of scrip-->
    </body> <!--end of body-->
</html> <!--end of html/page-->
```

Figure 41 code of footer

add.php

application ▸ views ▸ notebooks ▸ add.php

```
<!--this is add page of notebook where user can add new notebook -->
<!--this part presents cart section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
        <!--This part shows title of page-->
        <h2 class="text"><?= $title ;?></h2>
        <!-- any errors encountred will be shown in validation_errors part-->
        <div class="text text-danger text-center"><strong><?php echo validation_errors(); ?>
        </strong></div>
        <!--form_open_multipart is like <form> which supports image also -->
        <?php echo form_open_multipart('notebooks/add'); ?>
            <div class="form-group">
            <label class="text">Name</label>
            <input type="text" class="form-control" name="name" placeholder="Enter name">
            </div>
            <button type="submit" class="btn btn-warning">Submit</button>

        </form>
        <!--close form part-->
    </div>
</div>
<!--end of card section-->
<div class="background">

</div>
<!-- end of add page odnotebook-->
```

Figure 42 code of notebook part1

index.php

application ▸ views ▸ notebooks ▸ index.php

```
<!-- this is index page of notebook where created notebooks are displayed-->
<!--this part presents cart section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
        <!--This part shows title of page-->
        <h2 class="text"><?= $title; ?></h2>
        <div class="group">
            <?php foreach($notebooks as $notebook): ?>
                <div class="list-group-item notebook"><a href="<?php
                echo site_url('/notebooks/notes/'.$notebook['notebookid']); ?>"><?php
                echo $notebook['notebookname']; ?></a>
                <?php if($this->session->userdata('user_id') == $notebook['user_id']) :?>
                <form class="notebook-delete" action = "notebooks/delete/<?php
                echo $notebook["notebookid"]; ?>" method = "POST">
                    <input type="submit" class="btn-link text-danger" value="[X]">

                </form>
                <?php endif; ?>
                </div><br />
            <?php endforeach; ?>
        </div>
    </div>
</div>

<!--end of card section-->
<div class="background">
</div>
```

Figure 43 code of notebook part2

add.php

application ▸ views ▸ notes ▸ add.php

```
<!--this is add page of note where user can add notes-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this part shown title of current page-->
      <h2 class="text"><?= $title ?></h2>
 <!-- if any errors occurs, it is hown in this part-->
 <div class="text text-danger text-center"><strong><?php echo validation_errors(); ?>
  </strong></div>
<!--opens form supporting images-->
<?php echo form_open_multipart('notes/add'); ?>
<div class="row">
<div class="form-group col-md-9">
    <label class="text">Note name</label>
    <input type="text" class="form-control" name="notename" placeholder="Add Title">
  </div>
  <div class="form-group col-md-3 ">
  <label class="text">Pin note?</label><br />
<input type="radio" name="pin" value="2" >Yes
<input type="radio" name="pin" value="1"checked>No<br>
</div></div>
  <div class="form-group">
    <label class="text">Note detail: </label>
    <textarea id="editor1" class="form-control" name="notedetail" placeholder="Add Body"></textarea>
  </div>
  <div class="form-group">
    <label class="text">Notebook</label>
```

Figure 44 code of add note part1

add.php

application › views › notes › add.php

```
35      </div>
36      <div class="form-group">
37        <label class="text">Notebook</label>
38        <select name="notebook_id" class="form-control">
39        <!--start of foreach in which all notebook of that particular user is shown in dropdown option-->
40          <?php foreach($notebooks as $notebook): ?>
41            <option value="<?php echo $notebook['notebookid']; ?>"><?php echo $notebook['notebookname']; ?></option>
42          <?php endforeach; ?>
43          <!--end of foreach-->
44        </select>
45      </div>
46      <!--a div for uploading images for note-->
47    <div class="form-group">
48        <label>Upload Image</label>
49        <input type="file" name="userfile" size="20">
50      </div>
51      <button type="submit" class="btn btn-warning">Submit</button>
52    </form>
53    </div>
54    </div>
55    <!--end of card section-->
```

Figure 45 code of add note part2

edit.php

application › views › notes › edit.php

```
<!--this is a page of edit note where user will edit their note-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
      <h2 class="text"><?= $title ?></h2>
      <!-- if any errors occurs, it is hown in this part-->
      <?php echo validation_errors(); ?>

      <!--opens form-->
      <?php echo form_open('notes/update'); ?>
          <input type="Hidden" name="id" value="<?php echo $note['noteid'];?>">
      <div class="row">
        <div class="form-group col-md-9">
          <label class="text">Note name</label>
          <!--here note name is displayed-->
          <input type="text" class="form-control" name="notename" placeholder="Add Title" value="<?php echo $note['notename']; ?>" >
        </div>
        <div class="form-group col-md-3 ">
            <label class="text">Pin note?</label><br />
          <input type="radio" name="pin" value="2" >Yes
          <input type="radio" name="pin" value="1"checked>No<br>
        </div></div>
        <div class="form-group">
          <label class="text">Note detail: </label>
          <!--here note detail is displayed-->
```

Figure 46 code of edit note part1

edit.php

application › views › notes › edit.php

```
35            <label class="text">Note detail: </label>
36            <!--here note detail is displayed-->
37            <textarea id="editor1" class="form-control" name="notedetail" placeholder="Add Body"><?php echo $note['notedetail']; ?></textarea>
38          </div>
39          <div class="form-group">
40            <label class="text">Notebook</label>
41            <select name="notebook_id" class="form-control">
42            <!--here all created notebook is dispalyed in droplist option-->
43            <!--start of foreach-->
44              <?php foreach($notebooks as $notebook): ?>
45                <option value="<?php echo $notebook['notebookid']; ?>"><?php echo $notebook['notebookname']; ?></option>
46              <?php endforeach; ?>
47              <!-- end of for each-->
48            </select>
49          </div>
50          <div class="form-group">
51            <label>Upload Image</label>
52            <input type="file" name="userfile" size="20">
53          </div>
54          <button type="submit" class="btn btn-warning">Update note</button>
55        </form>
56      </div>
57    </div>
58    <!--end of card section-->
```

Figure 47 code of edit note part2

index.php

application ▸ views ▸ notes ▸ index.php

```
<!--this is a index page of note where added notes are shown-->
<!--this part presents cart section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
    <div class="row card-header">
        <div class="col-md-6">
            <h2 class="text"><strong><?= $title ?></strong></h2>
        </div>
        <!--div to show taks notification-->
        <div class="col-md-6">
            <?php if($notify > 0)
            {?>
               <a href="<?php echo base_url(); ?>todo"> <h6
               class=" notify text"<center><strong><?php
               echo 'YOU GOT ';?><?php echo $notify; ?><?php
               echo ' TASK(S) TO COMPLETE!'; ?>
               </a></strong></center></h6>
           <?php }
            else{?>
                <h6 class=" notify text"<center><?php
                echo 'ALL TASKS ARE COMPLETE. ADD NEW TASK NOW!' ;}?>
                </strong></center></h6>
        </div>
        </div><br />
            <?php foreach((array) $notes as $note) : ?>
            <!-- here name of note is displayed-->
```

Figure 48 code of index note part1

index.php

application ▸ views ▸ notes ▸ index.php

```
38                  <h3 class="text"><?php echo $note['notename']; ?></h3>
39                  <div class="row">
40                      <div class="col-md-3">
41                      <!-- note uploded/default image is shown-->
42                          <img class="post-thumb" src="<?php
43                          echo site_url(); ?>assets/images/notes/<?php
44                          echo $note['post_image']; ?>">
45                      </div>
46                          <div class="col-md-9">
47                              <!--here date in which note is crated is displayed-->
48                              <small class ="note-date"> Saved on:<?php
49                              echo $note['notecreateddot']; ?>
50                              </small><br>
51                              <!-- Note detail is shown with word limit of 70-->
52                              <div class="note-index-body card-footer"> <?php
53                              echo word_limiter($note['notedetail'], 70); ?>
54                              </div>
55                              <br><br>
56                              <!-- this button leads to full note page of that particular
57                               note-->
58                              <p><a class="btn btn-success" href="<?php
59                              echo site_url('/notes/'.$note['slug']); ?>
60                              ">Read more</a></p>
61                              <hr>
62                          </div>
63                          <hr>
64                  </div>
65              <?php endforeach; ?>
66                  <div class="pagination-links">
67                      <?php echo $this->pagination->create_links();?> <!-- this shows
68                      pagination on the buttom
69                      of page -->
70                  </div>
71      </div>
72  </div>
73  <!--end of card section-->
```

Figure 49 code of index note part2

trash.php

application ▸ views ▸ notes ▸ trash.php

```
<!--this is a index page of trash where moved notes are shown-->
<!--this part presents cart section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
        <h2 class="text"><?= $title ?></h2>
            <?php foreach((array) $trash as $t) : ?>
            <!-- here name of note is displayed-->
                <h3 class="text"><?php echo $t['notename']; ?></h3>
                <div class="row">
                    <div class="col-md-3">
                    <!-- note uploded/default image is shown-->
                        <img class="post-thumb" src="<?php echo site_url(); ?>assets/images/notes/<?php echo $t['post_image']; ?>">
                    </div>
                        <div class="col-md-9">
                            <!--here date in which note is crated is displayed-->
                            <small class ="note-date"> Deleted on:<?php echo $t['deleteddate']; ?></small><br>
                            <!-- Note detail is shown with word limit of 70-->
                            <div class="note-index-body"> <?php echo word_limiter($t['notedetail'], 70); ?>
                            </div>
                            <br><br>
                            <?php echo form_open('/trash/recover/'.$t['tid']); ?>
                            <!-- this button  recovers note that has been deleted-->
                            <p><button type="submit" class="btn btn-warning" method="POST">Recover</button></p> </form>
                            <?php echo form_open('/trash/delete/'.$t['tid']); ?>
                            <!-- this button  recovers note that has been deleted-->
                            <p><button type="submit" class="btn btn-danger" method="POST">Delete</button></p> </form>
```

Figure 50 code of trash part1

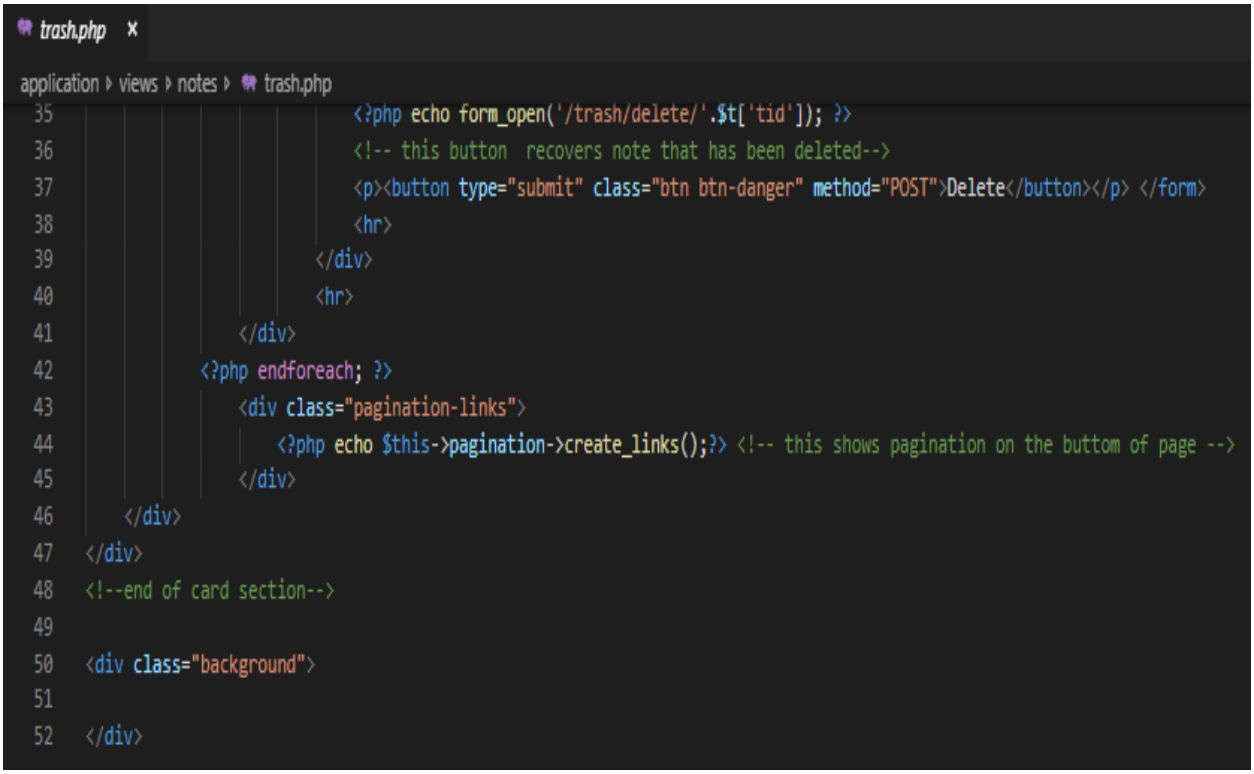


Figure 51 code of trash part2

view.php

application ▸ views ▸ notes ▸ view.php

```
<!-- this is a view page for full screen of a particular note-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
        <div class="toggel">
            <nav class="navbar navbar-expand-lg ">
                <div class="container-fluid">

                    <button type="button" id="sidebarCollapse" class="btn btn-info">
                        <i class="fas fa-align-left"></i>
                        <span>Toggle Pinned Notes</span>
                    </button>

                </div>
            </nav>
        </div>
    <!--this is note name-->
        <h2 class="text text-underline"><strong><?php echo $note['notename']; ?></strong></h2>
            <small class ="note-date"> Saved on:<?php echo $note['notecreateddot']; ?></small><br>
            <div class="row">
                <div class="col-sm-4">

                </div>
                <!-- in this div picture is shown-->
                <div class="col-sm-4">
                <img class="note-image"
                src="<?php echo site_url(); ?>assets/images/notes/<?php
                echo $note['post_image']; ?>">
                </div>

                <div class="col-sm-4">

                </div>
            </div>
                <!-- in this div note detail is shown-->
                <div class="note-body card-footer">
                        <?php echo $note['notedetail']; ?>
```

Figure 52 code of view note part1

view.php

application ▸ views ▸ notes ▸ view.php

```
35                  <!-- in this div note detail is shown-->
36                  <div class="note-body card-footer">
37                      <?php echo $note['notedetail']; ?>
38                  </div>
39              <!--start of if condition-->
40              <?php if($this->session->userdata('user_id') == $note['user_id']) :?>
41                  <hr>
42                      <a class="btn btn-warning col-sm-1 col-sm-offset-1" href="edit/<?php
43                      echo $note['slug']; ?>">Edit</a>
44                          <?php echo form_open('/notes/delete/'.$note['noteid']); ?>
45                              <input type="submit" value="Delete"
46                              class="btn btn-danger col-sm-1 col-sm-offset-1">
47                      </form>
48              <?php endif; ?>
49              <!--end of if condotion-->
50      </div>
51  </div>
52  <!--end of card section-->
53  <div class="background">
54
55  </div>
56  <!---3nd of page-->
```

Figure 53 code of view note part2

add.php

application ▸ views ▸ todo ▸ add.php

```
<!--this is a page of todo list where tasks can be added-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
        <div class="row">
            <div class="col-md-3 form-group">

            </div>
            <div class="col-md-6 form-group">
            <!--this shows title of this page-->
            <h2 class="text text-center card-header"><?= $title ;?></h2>
            </div>
            <div class="col-md-3 form-group">

            </div>
        </div>
        <!-- any errors encounted, it will be shown here-->
        <div class="text text-danger text-center"><strong><?php echo validation_errors(); ?>
        </strong></div>
        <!--opens form support images-->
            <?php echo form_open_multipart('todo/add'); ?>
                <div class="row">
                    <div class="col-md-3 form-group">

                    </div>
                        <div class="col-md-6 form-group card-footer">
                            <label class="text">Task name</label>
```

Figure 54 code of add task part1

add.php

application › views › todo › add.php

```
33                      <div class="col-md-3 form-group">
34
35                      </div>
36                          <div class="col-md-6 form-group card-footer">
37                              <label class="text">Task name</label>
38                              <input type="text" class="form-control" name="todoname" placeholder="Task name">
39
40                              <label class="text control-label" for="date">Date</label>
41                              <input class="form-control" id="date" name="date" placeholder="YYYY/MM/DD" type="text"/>
42                              <br />
43                              <button type="submit" class="btn btn-warning">Add</button>
44                          </div>
45                          <div class="col-md-3 form-group">
46
47                          </div>
48                      </div>
49
50                  </form>
51      </div>
52  </div>
53  <!--start of card section-->
54  <div class="background">
55
56  </div>
```

Figure 55 code of add task part2

index.php ×

application ▸ views ▸ todo ▸ index.php

```
<!--this is a page of todo list where incompleted tasks are shown and new tasks are added-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
        <h2 class="text card-header"><?= $title; ?></h2><br />

            <ul class="list-group">
                <div class="row">
                        <div class="col-md-3">

                            </div>
                        <div class="col-md-9 card-header">
                            <p>Uncomplete task(s) from past 1 day:</p>
                        </div><br />
                </div>
                <div class="row">
                    <div class="col-sm-3">
                    <!--add tasks button-->
                        <a class="btn btn-success" href="<?php echo base_url(); ?>todo/add">Add task</a>
                        <br />
                        <br />
                        <!--ciew completed task button-->
                        <a class="btn btn-success" href="<?php echo base_url(); ?>todo/view">Task completed</a>
                    </div>

                        <div class="col-sm-9 card-body">
```

Figure 56 code of index task part1

index.php

application ▸ views ▸ todo ▸ index.php

```
35                  </div>
36
37                  <div class="col-sm-9 card-body">
38                  <!--here aded incomplete tasks are displayd-->
39                  <!-- start of foreach-->
40                      <?php foreach($todos as $todo): ?>
41                      <!--task name is dispayed here-->
42                          <li class="list-group-item todo "><?php echo $todo['todoname']; ?></a>
43                              <?php if($this->session->userdata('user_id') == $todo['user_id']) :?>
44                                  <form class="todo-update" action = "todo/update/<?php echo $todo["todoid"]; ?>" method = "POST">
45                                  <!-- to set task complete-->
46                                      <input type="submit" class="btn-link text-danger" value="[Set task as completed]">
47
48                                  </form>
49                              <?php endif; ?>
50
51                          </li>
52                          <br />
53                      <?php endforeach; ?>
54                      <!--end of foreach-->
55                  </div>
56              </ul><br /><br />
57              <div class="row">
58                  <div class="col-md-3">
59
60                      </div>
61                  <div class="col-md-9 card-header">
62                      <p>Uncomplete task(s) from past 7 days:</p>
63                  </div>
64              </div><br />
65                  <ul class="list-group">
66                  <div class="row">
67                 <div class="col-sm-3">
68
69                  </div>
70
71                  <div class="col-sm-9 card-body">
```

Figure 57 code of index task part2

index.php

application ▸ views ▸ todo ▸ index.php

```
70
71                      <div class="col-sm-9 card-body">
72                      <!--here aded incomplete tasks are displayd-->
73                      <!-- start of foreach-->
74                          <?php foreach($tos as $to): ?>
75                          <!--task name is dispayed here-->
76                              <li class="list-group-item todo "><?php echo $to['todoname']; ?></a>
77                                  <?php if($this->session->userdata('user_id') == $to['user_id']) :?>
78                                      <form class="todo-update" action = "todo/update/<?php echo $to["todoid"]; ?>" method = "POST">
79                                      <!-- to set task complete-->
80                                          <input type="submit" class="btn-link text-danger" value="[Set task as completed]">
81
82                                      </form>
83                                  <?php endif; ?>
84
85                              </li>
86                              <br />
87                          <?php endforeach; ?>
88                          <!--end of foreach-->
89                      </div>
90                  </div>
91              </ul>
92      </div>
93  </div>
94  <!--start of card section-->
95  <div class="background">
96
97  </div>
```

Figure 58 code of index task part3

view.php

application ▸ views ▸ todo ▸ view.php

```
<!--this is a page of todo list where completed tasks are shown-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
        <h2 class="text card-header"><?= $title; ?></h2>
            <ul class="list-group">
                <!--start of foreach-->
                <?php foreach($todos as $todo): ?>
                <!--all completed tasks are displayed-->
                    <li class="list-group-item todo"><?php
                    echo $todo['todoname']; ?></a>

                    </li>
                <?php endforeach; ?>
                <!--end of foreach-->
                </ul>
    </div>
</div>
<!--end of card section-->
    <div class="background">

    </div>
```

Figure 59 code of view task

change_password.php

application ▸ views ▸ users ▸ change_password.php

```
<!--this is a page where users can chage their old password-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--open form-->
        <?php echo form_open('users/p_update'); ?>
            <div class="row">
                <div class='col-md-3'>

                </div>
                <div class="col-md-6">
                <!--this shows title of this page-->
                    <h1 class=" text text-center"><?= $title; ?></h1>
                    <div class="form-group">
                        <label class="text">Old password</label>
                        <input type="password" minlength="8" maxlength="20" class="form-control" name="password">
                    </div>

                    <div class="form-group">
                        <label class="text">New password</label>
                        <input type="password" minlength="8" maxlength="20" class="form-control" name="password1">
                    </div>

                    <div class="form-group">
                        <label class="text">Confirm password</label>
                        <input type="password" minlength="8" maxlength="20" class="form-control" name="password2">
                    </div>
```

Figure 60 code of change password part1

change_password.php

application › views › users › change_password.php

```
35                          <label class="text">Confirm password</label>
36                          <input type="password" minlength="8" maxlength="20" class="form-control" name="password2">
37                      </div>
38
39                  <button type="Submit" class="btn btn-warning btn-block">Update password</button>
40              </div>
41              <div class='col-md-3'>
42
43              </div>
44          </div>
45          </form>
46      </div>
47  </div>
48  <!--end of card section-->
49  <div class="background">
50
51  </div>
```

Figure 61 code of change password part2

edit_profile.php ×

application ▸ views ▸ users ▸ edit_profile.php

```
<!--this is a page where users can edit their profile-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
            <!--sends any error to validation_errors-->
            <?php echo validation_errors(); ?>
            <!--open form-->
            <?php echo form_open('users/update'); ?>
                <div class="row">
                    <div class='col-md-3'>

                    </div>
                    <div class="col-md-6">
                    <!--this shows title of this page-->
                        <h1 class=" text text-center"><?= $title; ?></h1>
                        <div class="form-group">
                            <label class="text">Nickname</label>
                            <!-- here user's name is displayed-->
                            <input type="text" minlength="3" maxlength="50" class="form-control" name="name"
                            placeholder="Nickname" value="<?php echo $user['name']; ?>">
                        </div>

                        <div class="form-group">
                            <label class="text">Email</label>
                            <!-- here user's email is displayed-->
                            <input type="text" maxlength="100" class="form-control" name="email"
                            placeholder="Email" value="<?php echo $user['email']; ?>">
```

Figure 62 code of edit profile part1

edit_profile.php

application › views › users › edit_profile.php

```
35                                  <!-- here user's email is displayed-->
36                                  <input type="text" maxlength="100" class="form-control" name="email"
37                                  placeholder="Email" value="<?php echo $user['email']; ?>">
38                              </div>
39
40                              <button type="Submit" class="btn btn-warning btn-block">Update</button>
41                          </div>
42                              <div class='col-md-3'>
43
44                              </div>
45                      </div>
46                  </form>
47          </div>
48      </div>
49      <!--end of card section-->
50      <div class="background">
51
52      </div>
```

Figure 63 code of edit profile part2

history.php

application ▸ views ▸ users ▸ history.php

```
<!--this is a page where login history is displayed of user-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
        <h2 class="text"> <?= $title; ?> </h2>
        <br />

        <div class="panel panel-default">
            <div class="panel-heading">
            </div>
            <div class="panel-body">
                <div class="table-responsive">
                    <table class="table table-bordered talble-striped">
                        <tr>
                            <th class="text">Email</th>
                            <th class="text">Logged time</th>
                        </tr>
                        <?php
                            foreach($login_history as $row): ?>

                                <tr>
                                    <td><?php echo $row->email; ?></td>
                                    <td><?php echo $row->logged_time; ?></td>
                                </tr><?php endforeach; ?>
                    </div>
                </div>
```

Figure 64 code of login history par1

history.php

application › views › users › history.php

```
33                                  <td><?php echo $row->email; ?></td>
34                                  <td><?php echo $row->logged_time; ?></td>
35                                </tr><?php endforeach; ?>
36                        </div>
37                    </div>
38                </div>
39            </div>
40        </div>
41    </div>
42    <!--end of card section-->
43
```

Figure 65 code of login history part2

login.php

application › views › users › login.php

```
<!--this is a page where users can log in-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
<!--form tag openning-->
<?php echo form_open('users/login'); ?>
<?php if(!$this->session->userdata('logged_in')) : ?>
    <div class="row">
        <div class='col-md-4'>

        </div>
       <div class='col-md-4'>
       <!--this shows title of this page-->
            <h1 class=" text text-center"><?php echo $title; ?></h1>
            <div class="form-group">
                <input type="text" maxlength="100" name="email" class="form-control"
                placeholder="Enter email" required autofocous>
            </div>
            <div class="form-group">
                <input type="password" minlength="8" maxlength="20" name="password" class="form-control"
                placeholder="Enter password" required autofocous>
            </div>
                <button type="submit" class="btn btn-warning btn-block">Login</button>

       </div>
       <div class='col-md-4'>

       </div>
    </div>
<?php echo form_close(); ?>
</div>
</div><?php endif;?>
<?php if($this->session->userdata('logged_in')) : ?>
    <div class="show-info">
        <div class="row">
            <div class="col-md-4">

            ...
```

Figure 66 code of login part1

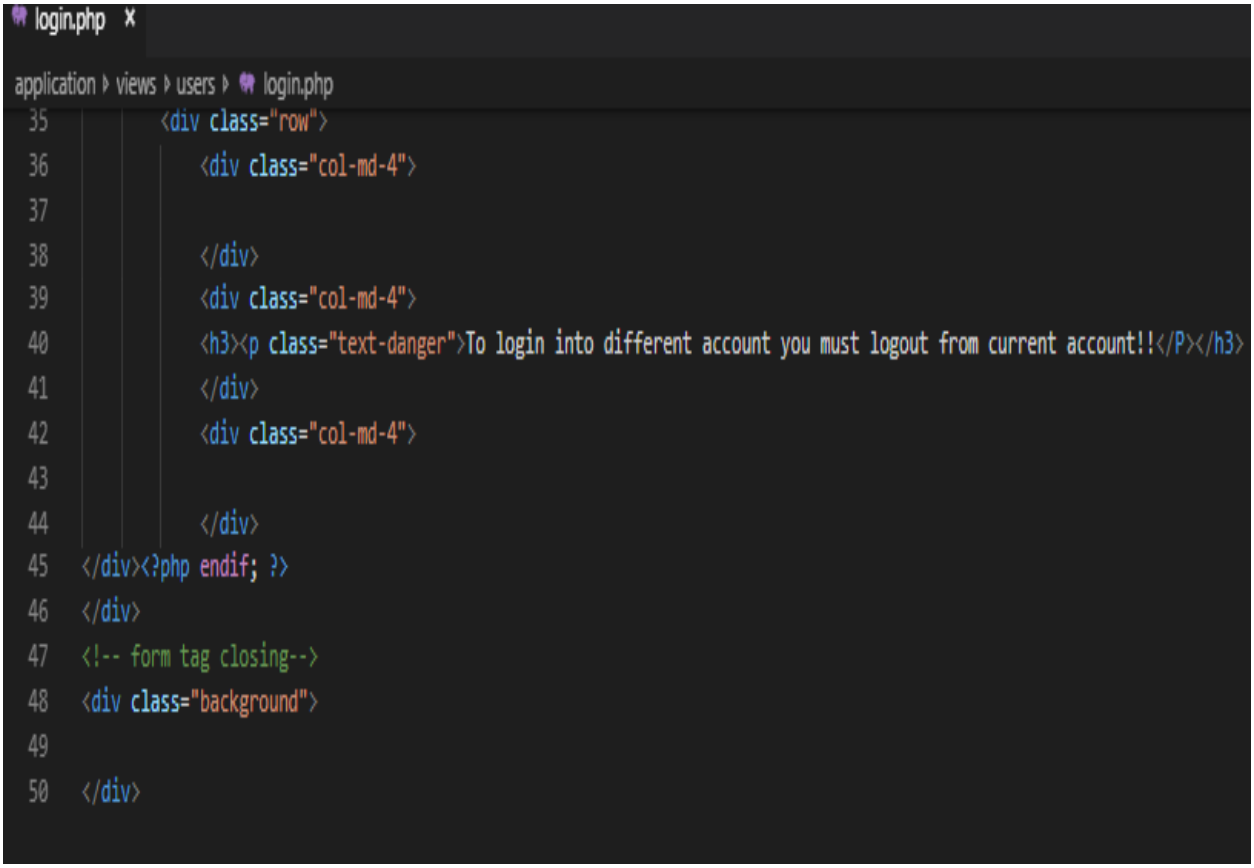


Figure 67 code of login part2

profile.php

application ▸ views ▸ users ▸ profile.php

```
<!--this is a page where user get option-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
    <!--this shows title of this page-->
        <h2 class="text card-header"><center><?= $title; ?></h2>
        <!--here user email is shown-->
            <div class="text text-center"><strong> Welcome <?php echo $this->session->userdata('email') ;?>
            </strong>
            </div></center></span></header>
            <div class="row">
                <div class="col-sm-4 col-md-4 card-footer"><center>
                    <a class="btn btn-success" href="<?php echo base_url(); ?>users/edit_profile">Edit
                    profile</a><br /><br />
                    <a class="btn btn-success" href="<?php echo base_url(); ?>users/change_password">Change
                    password</a><br /><br />
                    <a class="btn btn-success" href="<?php echo base_url(); ?>users/login_history">Login
                    log</a><br /><br />
                    <a class="btn btn-success" href="<?php echo base_url(); ?>export_csv/index">Export
                    data</a><br /><br />
                    <a class="btn btn-success" href="<?php echo base_url(); ?>profile/trash">
                    Trash</a><br /><br />
                    </center>
                </div>
                        <!--Total number of notes saved are shown-->
                        <div class="profile col-sm-4 col-md-4 text-danger card-footer">
                            <tr>
```

Figure 68 code of profile part1

profile.php

application › views › users › profile.php

```
35                          <!--Total number of notes saved are shown-->
36                          <div class="profile col-sm-4 col-md-4 text-danger card-footer">
37                              <tr>
38                                  <th class="text"><center><strong><u>Total number of
39                                  notes stored:</u></strong><center></th><br /> <br />
40                              </tr>
41                              <?php if( $row > 0)
42                                  {
43                                      ?><tr>
44                                      <td><center><strong><?php echo $row; ?></strong></center></td>
45                                      </tr>
46                                  <?php }
47                                  else{
48                                      echo 'There are no notes in your account.
49                                      Visit addnote to add notes. ';
50                                  }
51                              ?>
52                          </div>
53                              <!--Total number of tasks savcompleted are shown-->
54                              <div class="col-sm-4 col-md-4 text-danger card-footer">
55                              <tr>
56                                  <th class="text"><center><strong><u>Total number of
57                                  tasks added:</u></strong><center></th><br /> <br />
58                              </tr>
59                              <?php if( $task > 0)
60                                  {
61                                      ?><tr>
62                                      <td><center><strong><?php echo $task; ?></strong></center></td>
63                                      </tr>
64                                  <?php }
65                                  else{
66                                      echo 'You have not completed any tasks.
67                                      Visit todo, add task and complete your remaning tasks. ';
68                                  }
69                              ?>
70                              </div>
71              </div>
```

Figure 69 code of profile part2

Figure 70 code of profile part3

register.php

application ▸ views ▸ users ▸ register.php

```
<!--this is a page where can register -->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
        <!--sends any error to validation_errors-->
        <?php if(!$this->session->userdata('logged_in')) : ?>
        <div class="text text-center text-danger card-footer"><?php echo validation_errors(); ?>
            <?php echo 'Note: Password should be at least 8
                    characters in length and should include
                    at least one upper case letter, one
                    number, and one special character.'
                    ?> <br /> <br />
        </div>
        <!--form tag openning-->
        <?php echo form_open('users/register'); ?>
        <div class="row">
            <div class='col-md-4'>

            </div>
            <div class="col-md-4">
            <!--this shows title of this page-->
                <h1 class=" text text-center"><?= $title; ?></h1>
                <div class="form-group">
                    <label class="text">Nickname</label>
                    <input type="text" minlength="3" maxlength="50" class="form-control" name="name"
                    placeholder="Nickname" required>
                </div>

                <div class="form-group">
                    <label class="text">Email</label>
                    <input type="text" maxlength="100" class="form-control" name="email"
                    placeholder="Email" required>
                </div>

                <div class="form-group">
                    <label class="text">Password</label>
                    <input type="password" minlength="8" maxlength="20"class="form-control" name="password"
```

Figure 71 code of register part1

register.php

application ▸ views ▸ users ▸ register.php

```
38                      placeholder="Password" required>
39                  </div>
40
41                  <div class="form-group">
42                      <label class="text">Confirm Password</label>
43                      <input type="password" minlength="8" maxlength="20" class="form-control" name="password2"
44                      placeholder="Confirm password" required>
45                  </div>
46
47                  <button type="Submit" class="btn btn-warning btn-block">Submit</button>
48              </div>
49          <div class='col-md-4'>
50
51              </div>
52          </div>
53  </div><?php endif; ?>
54      <?php if($this->session->userdata('logged_in')) : ?>
55      <div class="show-info">
56          <div class="row">
57              <div class="col-md-4">
58
59              </div>
60              <div class="col-md-4">
61              <h3><p class="text-danger">To create new account you must logout from current account!!</P></h3>
62              </div>
63              <div class="col-md-4">
64
65              </div>
66  </div><?php endif; ?>
67  </div>
68  <?php echo form_close();?>
69  <!--form tag closing-->
70  <!--end of card section-->
71  <div class="background">
72
73  </div>
```

Figure 72 code of register part2

export_csv.php

application › views › export_csv.php

```
<!--this is a page where user can export data-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">

            <button type="button" id="sidebarCollapse" class="btn btn-info">
                <i class="fas fa-align-left"></i>
                <span>Toggle Pinned Notes</span>
            </button>

        </div>
    </nav>
        <!-- this is a title of current page-->
        <h3 class="text" align="center"> Export data of notes</h3>
            <br />
            <form method="POST" action="<?php echo base_url(); ?>export_csv/export">
                <div class="panel panel-default">
                    <div class="panel-heading">
                        <div class="row">
                            <div clss="col-md-6">
                                <h3 class="panel-title text"> Notes data</h3>
                            </div>
                            <div class="col-md-6" align="right">
                                <input type="submit" name="export" class="btn btn-success btn-xs" value="Export"/>

                            </div>
                        </div>
                    </div>
                    <div class="panel-body">
                        <div class="table-responsive">
                            <table class="table table-bordered talble-striped">
                            <!-- here all notes aree displayed of current user-->
                                <tr>
                                    <th class="text">Note name</th>
                                    <th class="text">Description</th>
```

Figure 73 code of export data part1

export_csv.php

application ▸ views ▸ export_csv.php

```
35                                  <tr>
36                                      <th class="text">Note name</th>
37                                      <th class="text">Description</th>
38                                  </tr>
39                                  <?php
40                                      foreach($note_data->result_array() as $row)
41                                      {
42                                          echo '
43                                              <tr>
44                                                  <td>'.$row['notename'].' </td>
45                                                  <td>'.$row['notedetail'].'</td>
46                                              </tr>
47                                          ';
48                                      }
49                                  ?>
50                              </div>
51                          </div>
52                      </div>
53                  </form>
54              </div>
55          </div>
56          <!--end of card section-->
57
```

Figure 74 code of export data part2

view.php

application › views › search › view.php

```
<!--this is a page of searchnote where all user noted are displayed-->
<!--start of card section-->
<div class="card card-4">
    <div class="card-body">
    <!--this shows title of this page-->
        <h2 class="text"> <?= $title; ?> </h2>
        <br />

            <div class="panel panel-default">
                <div class="panel-heading">
                </div>
                <div class="panel-body">
                    <div class="table-responsive">
                        <table class="table table-bordered talble-striped">
                            <tr>
                                <th class="text">Note Name</th>

                            </tr>
                            <!--start of foreach-->
                            <?php
                                foreach($notes as $row): ?>

                                    <!--here note name is displayed-->
                                    <tr>
                                        <td><a href="<?php echo site_url('/notes/'.$row->slug); ?>"><?php echo $row->notename; ?></td>

                                    </tr>
                        </div><?php endforeach; ?>
                    </div>
                </div>
            </form>
        </div>
    </div>
</div>
<!--end of card section-->
```

Figure 75 code of displaying all data

Export_csv.php ×

application ▸ controllers ▸ Export_csv.php

```php
<!-- this is a controller class of extport_csv
<?php
    class Export_csv extends CI_Controller
    {
        //This methode will be exectuded when new object from this class has been created
        public function __construct()
        {
            //when new object from this class has been created __construct methode will be executed
            parent::__construct();
            $this->load->model('export_csv_model');//this methodwill load export_csv_model
        }// end of function

        // this is root methode of this class. when we type base url/export_csv then this method will execute
        function index()
        {
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['pinn'] = $this->note_model->get_pin();
            $this->load->view('includes/header', $data);
            //this methode will return data which have been stored under $data variable
            $data['note_data'] = $this->export_csv_model->fetch_data();
            $this->load->view('export_csv', $data);
            $this->load->view('includes/footer');
        }// end of function

        //function to export data and to call respetive model
        function export()
        {
            $file_name = 'note_detail_on'.date('Ymd').'.csv';
            header("Content-Description: File Transfer");
            header("Content-Disposition: attachment; filename=$file_name");
            header("Content-Type: application/csv;");
            $note_data = $this->export_csv_model->fetch_data();

```

Figure 76 code of export data controller part1

Export_csv.php ×

application ▸ controllers ▸ Export_csv.php

```
33              header("Content-Description: File Transfer");
34              header("Content-Disposition: attachment; filename=$file_name");
35              header("Content-Type: application/csv;");
36              $note_data = $this->export_csv_model->fetch_data();
37
38              $file = fopen('php://output', 'w');
39              $header = array("notename", "notedetail");
40
41              fputcsv($file, $header);
42
43              foreach($note_data->result_array() as $key => $value)
44              {
45                  fputcsv($file, $value);
46              }
47
48              fclose($file);
49
50              exit;
51          } // end of function
52      }
```

Figure 77 code of export data controller part2

notebooks.php ×

application ▸ controllers ▸ notebooks.php

```php
<!--this is a controller class of notebook-->
<?php
    class Notebooks extends CI_Controller
    {
        //function which loads folowing data when called
        public function index()
        {
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['title'] = "Notebooks";
            $data['pinn'] = $this->note_model->get_pin();
            $data['notebooks'] = $this->notebook_model->get_notebooks();
            //load following assets
            $this->load->view('includes/header', $data);
            $this->load->view('notebooks/index', $data);
            $this->load->view('includes/footer');
        } // end of function

        //function to add new notebook
        public function add()
        {
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['title'] = "Add Notebook";
            $this->form_validation->set_rules('name','Name', 'required');
            if($this->form_validation->run() === FALSE)
            {
                $data['pinn'] = $this->note_model->get_pin();
                $this->load->view('includes/header', $data);
                $this->load->view('notebooks/add', $data);
                $this->load->view('includes/footer');
```

Figure 78 code of notebook controller part1

notebooks.php

application ▸ controllers ▸ notebooks.php

```
35                  $this->load->view('includes/header', $data);
36                  $this->load->view('notebooks/add', $data);
37                  $this->load->view('includes/footer');
38              }
39              else{
40                  $this->notebook_model->add_notebook();
41
42                  //set message
43                  $this->session->set_flashdata('notebook_added', 'Notebook has been added.');
44
45                  redirect('notebooks');
46              }
47          }// end of function
48
49          //function that calls model function when called
50          public function notes($notebookid)
51          {
52              $data['title']= $this->notebook_model->get_notebook($notebookid)->notebookname;
53              //load following assets
54              $data['notify'] = $this->todo_model->notify_tasks();
55              $data['notes'] = $this->note_model->get_notes_by_notebook($notebookid);
56              $data['pinn'] = $this->note_model->get_pin();
57              $this->load->view('includes/header', $data);
58              $this->load->view('notes/index', $data);
59              $this->load->view('includes/footer');
60          } // end of function
61
62          //function that calls model function when called
63          public function delete($notebookid)
64          {
65              //check login
66              if(!$this->session->userdata('logged_in'))
67              {
68                  //redirect page to users/login
69                  redirect('users/login');
70              }
71             $this->notebook_model->delete_notebook($notebookid);
```

Figure 79 code of notebook controller part2

notebooks.php ×

application ▸ controllers ▸ notebooks.php

```
64          {
65              //check login
66              if(!$this->session->userdata('logged_in'))
67              {
68                  //redirect page to users/login
69                  redirect('users/login');
70              }
71              $this->notebook_model->delete_notebook($notebookid);
72              //set message
73              $this->session->set_flashdata('notebook_deleted', 'Notebook deleted.');
74              //redirect page to notes
75              redirect('notebooks');
76          }
77      } // end of notebooks class
```

Figure 80 code of notebook controller part3

Notes.php ×

application ▸ controllers ▸ Notes.php

```php
<!-- thisi a controller class ofnotes-->
<?php
    class Notes extends CI_Controller
    {
        //function that calls model function when called
        public function index($offset = 0)
        {
            //pagination config
            $config['base_url'] = base_url() . 'notes/index/';
            $config['total_rows'] = $this->db->count_all('notes');
            $config['per_page'] = 3;
            $config['uri_segment'] = 3;
            // Produces: class="pagination-link"
            $config['attributes'] = array('class' => 'pagination-link');

            //initate pagination
            $this->pagination->initialize($config);

            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['title'] = 'Latest notes';
            $data['notify'] = $this->todo_model->notify_tasks();
            $data['notes'] = $this->note_model->get_notes(FALSE,
             $config['per_page'], $offset);
             $data['pinn'] = $this->note_model->get_pin();
            $this->load->view('includes/header', $data);
            $this->load->view('notes/index', $data);
            $this->load->view('includes/footer');

        }

        //function that calls model function when called
        public function view($slug = NULL)
        {
```

Figure 81 code of note controller part1

Notes.php ×

application ▸ controllers ▸ Notes.php

```
38          //check login
39          if(!$this->session->userdata('logged_in'))
40          {
41              redirect('users/login');
42          }
43          $data['note']= $this->note_model->get_notes($slug);
44
45          if (empty($data['note']))
46          {
47              show_404();
48          }
49          $data['notename'] = $data['note']['notename'];
50          $data['pinn'] = $this->note_model->get_pin();
51          $this->load->view('includes/header', $data);
52          $this->load->view('notes/view', $data);
53          $this->load->view('includes/footer');
54      }
55
56      //function that calls model function when called
57      public function add()
58      {
59          //check login
60          if(!$this->session->userdata('logged_in'))
61          {
62              redirect('users/login');
63          }
64          $data['title'] = 'Add note';
65          $data['notebooks'] = $this->note_model->get_notebooks();
66          $this->form_validation->set_rules
67          ('notename', 'Notename', 'required');
68          $this->form_validation->set_rules
69          ('notedetail', 'Notedetail', 'required');
70
71          if($this->form_validation->run() === FALSE)
72          {
73              $data['pinn'] = $this->note_model->get_pin();
74              $this->load->view('includes/header', $data);
```

Figure 82 code of note controller part2

Notes.php ×

application ▸ controllers ▸ Notes.php

```
75                  $this->load->view('notes/add', $data);
76                  $this->load->view('includes/footer');
77              }
78              else
79              {
80                  //upload image
81                  $config['upload_path'] = './assets/images
82                  /notes';
83                  $config['allowed_types'] = 'jpg|png|gif';
84                  $config['max_size'] = '2048';
85                  $config['max_width'] = '1500';
86                  $config['max_height'] = '1500';
87
88                  $this->load->library('upload', $config);
89                  if(!$this->upload->do_upload())
90                  {
91                      $errors = array('error' =>$this->
92                      upload->display_errors());
93                      $post_image = 'noimage.jpg';
94                  }
95                  else
96                  {
97                      $data = array('upload_data' =>$this->
98                      upload->data());
99                      $post_image = $_FILES['userfile']['name'];
100                 }
101                 $this->note_model->add_note($post_image);
102                 //set message
103                 $this->session->set_flashdata('note_added',
104                 'Note saved.');
105                 //redirect page to notes
106                 redirect('notes');
107
108             }
109         } //end of the function
110
111         public function delete($noteid)
```

Figure 83 code of note controller part3

Notes.php

application ▸ controllers ▸ Notes.php

```
112         {
113             //check login
114             if(!$this->session->userdata('logged_in'))
115             {
116                 //redirect page to users/login
117                 redirect('users/login');
118             }
119             //insert into trash table
120             $this->trash_model->trash_note($noteid);
121             //delete selected note
122             $this->note_model->delete_note($noteid);
123             //set message
124             $this->session->set_flashdata('note_deleted',
125             'Note deleted.');
126             //redirect page to notes
127             redirect('notes');
128         } //end of the function
129
130         //function that calls model function when called
131         public function edit($slug)
132         {
133             //check login
134             if(!$this->session->userdata('logged_in'))
135             {
136                 //redirect page to users/login
137                 redirect('users/login');
138             }
139             $data['note']= $this->note_model->get_notes($slug);
140             //check user
141             if($this->session->userdata('user_id') !=
142             $this->note_model->get_notes($slug)['user_id'])
143             {
144                 //redirect page to users/login
145                 redirect('users/login');
146             }
147             $data['notebooks'] = $this->note_model->
148             get_notebooks();
```

Figure 84 code of note controller part4

Notes.php

application › controllers › Notes.php

```
148             get_notebooks();
149             //show page 404 error if empty
150             if (empty($data['note']))
151             {
152                 show_404();
153             }
154             $data['title'] = 'Edit Note';
155             $data['pinn'] = $this->note_model->get_pin();
156             $this->load->view('includes/header', $data);
157             $this->load->view('notes/edit', $data);
158             $this->load->view('includes/footer');
159         } // end of function
160
161         //function that calls model function when called
162         public function update()
163             {
164                 //check login
165                 if(!$this->session->userdata('logged_in'))
166                 {
167                     redirect('users/login');
168                 }
169                 $this->note_model->update_note();
170                 //set message
171                 $this->session->set_flashdata('note_updated',
172                 'Note updated.');
173                 redirect('notes');
174             }
175     } // end of note class
```

Figure 85 code of note controller part4

Profile.php ×

application ▸ controllers ▸ Profile.php

```php
<!-- this is a controlelr class of profile-->
<?php
    class Profile extends CI_Controller
    {
        //function that calls model function when called
        public function index()
        {
            $this->load->model('user_model');
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['pinn'] = $this->note_model->get_pin();
            $this->load->view('includes/header', $data);
            $data['title'] = 'Profile';
            $id = $this->session->userdata('user_id');
            $data['row'] = $this->user_model->count_rows($id);
            $data['task'] = $this->user_model->count_ctasks($id);
            $this->load->view('users/profile', $data);
            $this->load->view('includes/footer');
        }
    } //end of function
```

Figure 86 code of profile controller part1

Todo.php

application ▸ controllers ▸ Todo.php

```php
<!-- this is a controller class of todo-->
<?php
    class Todo extends CI_Controller
    {
        //function that calls model function when called
        public function index()
        {
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['title'] = "Your's today's perform list ";
            $data['todos'] = $this->todo_model->get_todo();
            $data['tos'] = $this->todo_model->get_todo7();
            $data['pinn'] = $this->note_model->get_pin();
            //load following assets
            $this->load->view('includes/header', $data);
            $this->load->view('todo/index', $data);
            $this->load->view('includes/footer');
        } // end of function

        //function that calls model function when called
        public function add()
        {
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['title'] = 'Add Todo';
            $this->form_validation->set_rules('todoname','Todoname', 'required');
            $this->form_validation->set_rules('date','Tododate', 'required');
            if($this->form_validation->run() === FALSE)
            {
                $data['pinn'] = $this->note_model->get_pin();
                $this->load->view('includes/header',$data);
```

Figure 87 code of task controller part1

Todo.php

application ▸ controllers ▸ Todo.php

```
35              {
36                  $data['pinn'] = $this->note_model->get_pin();
37                  $this->load->view('includes/header',$data);
38                  $this->load->view('todo/add', $data);
39                  $this->load->view('includes/footer');
40              }
41              else{
42                  $this->todo_model->add_todo();
43
44                  //set message
45                  $this->session->set_flashdata('task_added', 'Task has been added.');
46
47                  redirect('todo');
48              }
49          } // end of function
50
51          //function that calls model function when called
52          public function update($todoid)
53          {
54              //check login
55              if(!$this->session->userdata('logged_in'))
56              {
57                  //redirect page to users/login
58                  redirect('users/login');
59              }
60              $this->todo_model->update_todo($todoid);
61
62              //redirect page to notes
63              redirect('todo');
64          } //end of function
65
66          //function that calls model function when called
67          public function view_c()
68          {
69              //check login
70              if(!$this->session->userdata('logged_in'))
71              {
```

Figure 88 code of task controller part2

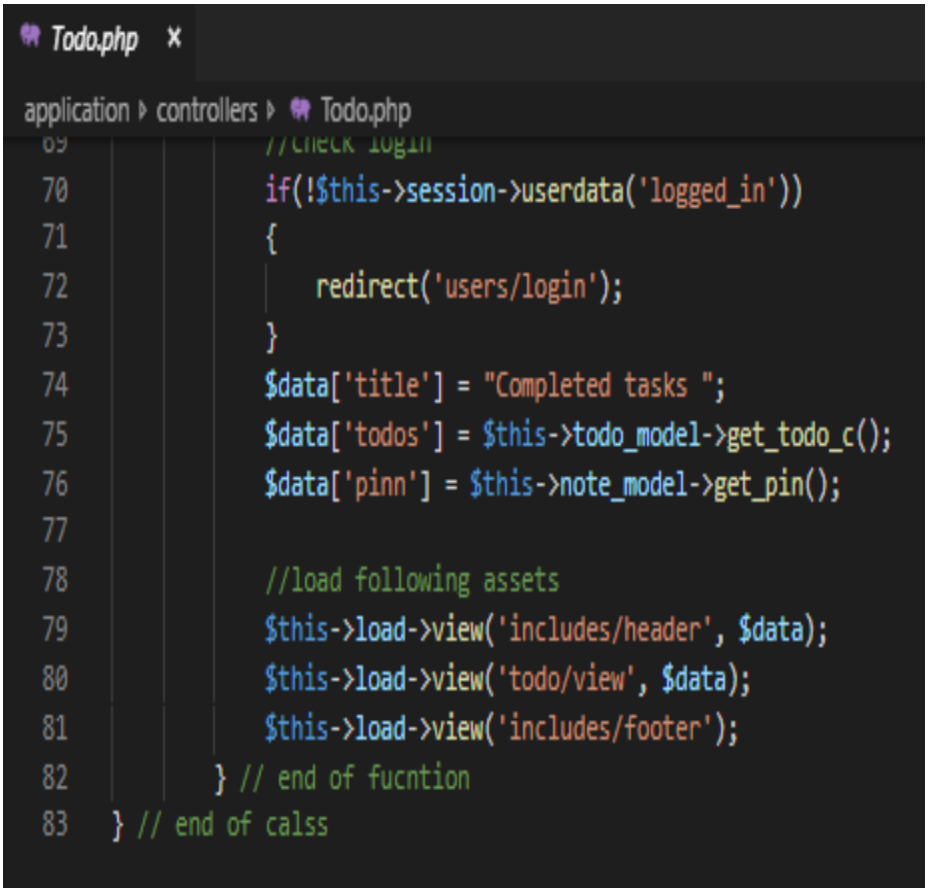


Figure 89 code of task controller part3

Trash.php

application ▸ controllers ▸ Trash.php

```php
<!-- this is a controller of trash page-->
<?php
    class Trash extends CI_Controller
    {
        //this function controls data from from trash view to model
        function index()
        {
            //check login
            if(!$this->session->userdata('logged_in'))
            {
                redirect('users/login');
            }
            $data['title'] = 'Trash';
            $data['trash'] = $this->trash_model->view_trash();
            $data['pinn'] = $this->note_model->get_pin();
            $this->load->view('includes/header', $data);
            $this->load->view('notes/trash', $data);
            $this->load->view('includes/footer');

        }

        //this function receivs $id from view and sends that data to model
        function recover($tid)
        {
            if(!$this->session->userdata('logged_in'))
            {
                //redirect page to users/login
                redirect('users/login');
            }
            $this->trash_model->recover($tid);
            $this->trash_model->delete($tid);

            redirect('profile/trash');
        }

        //this function receivs $id from view and sends that data to model
        function delete($tid)
```

Figure 90 code of trash controller part1

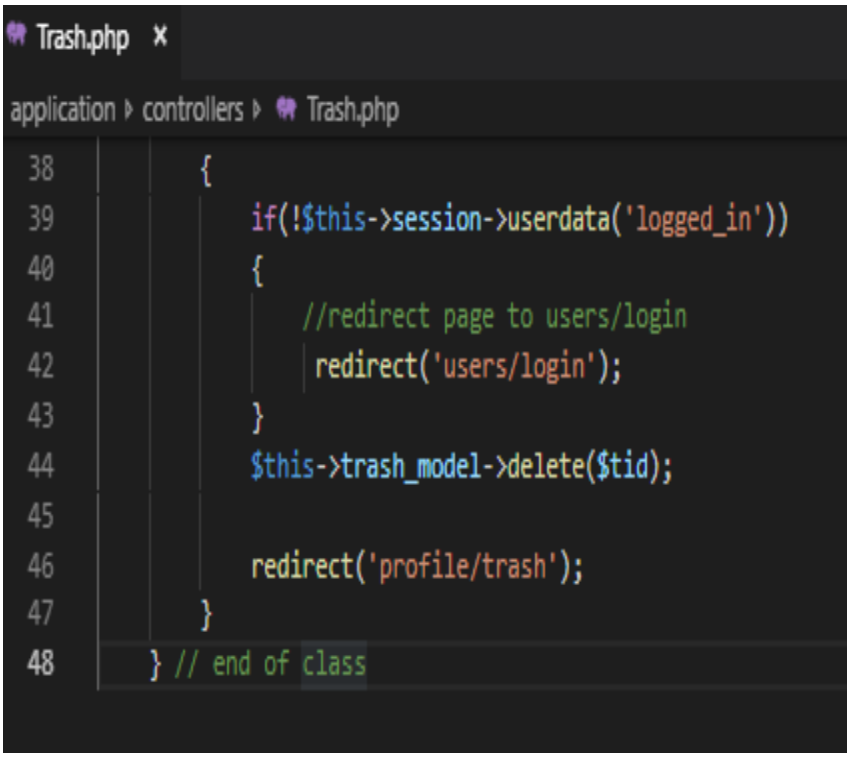


Figure 91 code of trash controller part2

users.php

application ▸ controllers ▸ users.php

```php
<!-- this is a controller class of user-->
<?php
    class Users extends CI_Controller
    {
        //register user
        public function register()
        {
            $data['title'] =  'Sign Up';
             // setting validation on name, email, password, and password2
            $this->form_validation->set_rules('name', 'Name', 'required');
            $this->form_validation->set_rules('email', 'Email', 'required|callback_check_email_exists');
            $this->form_validation->set_rules('password', 'Password', 'required');
            $this->form_validation->set_rules('password2', 'Confirm Password', 'matches[password]');
             $password = $this->input->post('password');
             // Validate password strength
             $uppercase = preg_match('@[A-Z]@', $password);
             $lowercase = preg_match('@[a-z]@', $password);
             $number    = preg_match('@[0-9]@', $password);
             $specialChars = preg_match('@[^\w]@', $password);
            //start of if condition
             if($this->form_validation->run() === FALSE ||!$uppercase || !$lowercase
             || !$number || !$specialChars || strlen($password) < 8)
             {
                 //load following assets if above condition match
                 $this->load->view('includes/header');
                 $this->load->view('users/register', $data);
                 $this->load->view('includes/footer');
             }
             else
             {
                 //encrypt password
                 $enc_password = md5($password);
                 //sends encrypted password in user_model register function
                 $this->user_model->register($enc_password);
                 //set message
                 $this->session->set_flashdata('user_registered', 'Registretion complete.
                  Please proceed to login page for login.');
```

Figure 92 code of users controller part1

users.php

application ▸ controllers ▸ users.php

```
38                  //redirect page to notes
39                  redirect('notes');
40                }
41                // end of if condition
42              } // end od function
43
44              //login user
45              public function login()
46              {
47                 //setting page title
48                $data['title'] =  'Sign In';
49
50                // setting validation on email and password
51                $this->form_validation->set_rules('email', 'Email', 'required');
52                $this->form_validation->set_rules('password', 'Password', 'required');
53
54                // start of if condition
55                 if($this->form_validation->run() === FALSE)
56                 {
57                     $this->load->view('includes/header', $data);
58                     $this->load->view('users/login', $data);
59                     $this->load->view('includes/footer');
60                 }
61                 else
62                 {
63                     //get email
64                     $email = $this->input->post('email');
65
66                     //get and encrypt password
67                     $password= md5($this->input->post('password'));
68
69                     //login user
70                     $user_id = $this->user_model->login($email, $password);
71
72                     //start of nested if condion
73                     if($user_id)
74                     {
```

Figure 93 code of users controller part2

users.php

application ▸ controllers ▸ users.php

```php
                //create session
                $user_data= array(
                    'user_id'=> $user_id,
                    'email' => $email,
                    'password' => $password,
                    'logged_in' => true
                );
                //stores above array data so it can be access anytime we need
                $this->session->set_userdata($user_data);
                $this->user_model->login_history_set();
                //set message
                $this->session->set_flashdata('user_loggedin', 'Login sucessfull.');
                //redirecr page to notes
                redirect('notes');

            }
            else
            {
             //set message
             $this->session->set_flashdata('login_failed', 'Username or password incorect');

             //redurect page to users/login
             redirect('users/login');
            }
            // end of nested if condition
            // end of if condition
        }
    } // end of function

    //Log user out
    public function logout()
    {
        //unset userdata(session)
        $this->session->unset_userdata('logged_in');
        $this->session->unset_userdata('user_id');
        $this->session->unset_userdata('email');
        $this->session->unset_userdata('password');
```

Figure 94 code of users controller part3

users.php

application > controllers > users.php

```
112
113             //set message
114             $this->session->set_flashdata('user_logout', 'logout complete');
115             //redurect page to users/login
116             redirect('users/login');
117         } // end of fucniton
118
119         //check email exists or not
120         public function check_email_exists($email)
121         {
122             $this->form_validation->set_message('check_email_exists', 'Email is taken');
123             if($this->user_model->check_email_exists($email))
124             {
125                 return TRUE;
126             }
127             else
128             {
129                 return FALSE;
130             }
131         } // end of fucntion
132
133         //function that calls model function when called
134         public function edit_profile()
135         {
136             //check login
137             if(!$this->session->userdata('logged_in'))
138             {
139                 //redirect page to users/login
140                 redirect('users/login');
141             }
142             $data['user']= $this->user_model->get_user();
143             $data['title'] = 'Edit Profile';
144             $data['pinn'] = $this->note_model->get_pin();
145             $this->load->view('includes/header', $data);
146             $this->load->view('users/edit_profile', $data);
147             $this->load->view('includes/footer');
148         } // end of fucnton
```

Figure 95 code of users controller part4

users.php

application > controllers > users.php

```
148         } // end of fucnton
149
150         //function that calls model function when called
151         public function change_password()
152         {
153             //check login
154             if(!$this->session->userdata('logged_in'))
155             {
156                 redirect('users/login');
157             }
158             $data['user']= $this->user_model->get_user();
159             $data['title'] = 'Change password';
160             $data['pinn'] = $this->note_model->get_pin();
161             $this->load->view('includes/header', $data);
162             $this->load->view('users/change_password', $data);
163             $this->load->view('includes/footer');
164         } // end of fucntion
165
166         //function that calls model function when called
167         public function update()
168         {
169             //check login
170             if(!$this->session->userdata('logged_in'))
171             {
172                 redirect('users/login');
173             }
174             $this->user_model->update_user();
175
176             //set message
177             $this->session->set_flashdata('profile_updated', 'Profile updated.');
178             //redirect page to profile
179             redirect('profile');
180         } // end of function
181
182         //function that calls model function when called
183         public function update_password()
184         {
185             //check login
```

Figure 96 code of users controller part5

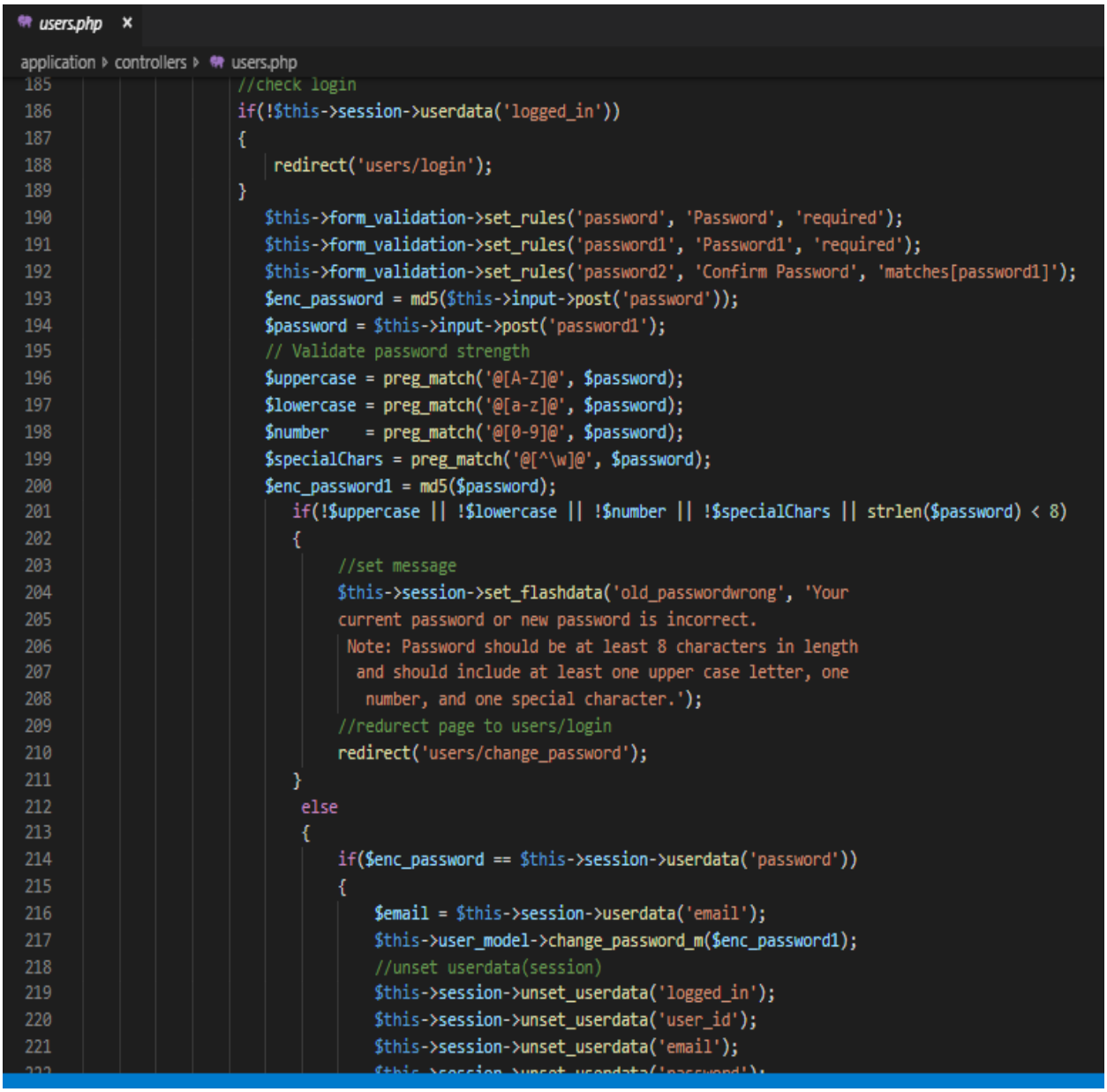


Figure 97 code of users controller part6

users.php

application ▸ controllers ▸ users.php

```
222                                 $this->session->unset_userdata('password');
223                                 //set message
224                                 $this->session->set_flashdata('password_updated', 'Your
225                                 has been updated, Login to verify password.');
226                                 //redurect page to users/login
227                                 redirect('users/login');
228                             }
229                             else
230                             {
231                                 //set message
232                                 $this->session->set_flashdata('old_passwordwrong', 'Your
233                                 current password is incorrect');
234                                 //redurect page to users/login
235                                 redirect('users/change_password');
236 
237                             }
238                         }
239 
240         } // end of function
241 
242         //function that calls model function when called
243         public function login_history()
244         {
245             //check login
246             if(!$this->session->userdata('logged_in'))
247             {
248                 redirect('users/login');
249             }
250 
251             //setting page title
252            $data['title'] =  'Loggin history';
253            $data['login_history'] = $this->user_model->login_history_view();
254            $data['pinn'] = $this->note_model->get_pin();
255            $this->load->view('includes/header', $data);
256            $this->load->view('users/history', $data);
257            $this->load->view('includes/footer');
258         } // end of fucntion
```

Figure 98 code of users controller part7

users.php

application › controllers › users.php

```
259
260         //function that calls model function when called
261         public function count_note()
262         {
263             //check login
264             if(!$this->session->userdata('logged_in'))
265             {
266                 redirect('users/login');
267             }
268             $data['count'] = $this->note_model->count_note();
269             $this->load->view('users/profile', $data);
270         } //end of function
271     } // end of user class
```

Figure 99 code of users controller part8

Export_csv_model.php ×

application ▸ models ▸ Export_csv_model.php

```php
<!--this is a model of export_csv form-->
<?php
    class Export_csv_model extends CI_Model
    {
        function fetch_data() //this function will fetch data
        {
            //here user id is extracted from session
            $id= array(
                'user_id' => $this->session->userdata('user_id'));
                //here user notes data are extracted from database of current logged in user
                $query = $this->db->get('notes');
                $this->db->order_by('noteid', 'DESC');
                $query = $this->db->get('notes');
                return $this->db->get_where('notes', $id);
        }
    }
```

Figure 100 code of export model

Note_model.php

application ▸ models ▸ Note_model.php

```php
<!--this is a model of note-->
<?php
    class Note_model extends CI_Model
    {
        //defulat constructor which load data from database
        public function __constructor()
        {
            $this->load->database();
        }

        //function to get note from database
        public function get_notes($slug = FALSE, $limit = FALSE, $offset = FALSE)
        {
            if($limit)
            {
                $this->db->limit($limit, $offset);
            }
            //user id is taken from session
            $user_id = $this->session->userdata('user_id');
            if($slug === FALSE)
            {
                $id= array(
                    'user_id' => $this->session->userdata('user_id'));
                //all notes are selected from database of current logged in user
                $this->db->order_by('noteid', 'DESC');
                $query = $this->db->get_where('notes', $id);
                return $query->result_array(); //returns $query in result_array()
            }
            $query = $this->db->get_where('notes', array('slug'=> $slug)); //starts query
            return $query->row_array();
        }

        //create new note function where $post_image as parameter
        public function add_note($post_image)
        {
            $slug = url_title($this->input->post('notename'));
            //array containing many data
```

Figure 101 code of note model part1

Note_model.php ×

application › models › Note_model.php

```
38              $data = array(
39                  'notename' => $this->input->post('notename'),
40                  'pin' => $this->input->post('pin'),
41                  'slug' => $slug,
42                  'notedetail' => $this->input->post('notedetail'),
43                  'notebook_id' => $this->input->post('notebook_id'),
44                  'user_id' => $this->session->userdata('user_id'),
45                  'post_image' => $post_image
46              );
47              return $this->db->insert('notes', $data); //runs insert query
48          }
49
50          //delete note function where $noteid as parameter
51          public function delete_note($noteid)
52              {
53                  //delete data query and returns true
54                  $this->db->where('noteid',$noteid);
55                  $this->db->delete('notes');
56                  return True;
57              }
58
59              //update note function
60              public function update_note()
61              {
62                  $slug = url_title($this->input->post('notename'));
63                  $data = array(
64                      'notename' => $this->input->post('notename'),
65                      'pin' => $this->input->post('pin'),
66                      'slug' => $slug,
67                      'notedetail' => $this->input->post('notedetail'),
68                      'notebook_id' => $this->input->post('notebook_id')
69                  );
70                      $this->db->where('noteid', $this->input->post('id'));
71                      return $this->db->update('notes', $data);
72              }
73
74              //function to get notebook
```

Figure 102 code of note model part2

Note_model.php

application › models › Note_model.php

```
75          public function get_notebooks()
76          {
77              //get user id from session of current logged in user
78              $id= array(
79                  'user_id' => $this->session->userdata('user_id'));
80              $this->db->order_by('notebookname');
81              $query = $this->db->get('notebooks');
82              $query = $this->db->get_where('notebooks', $id);
83              return $query->result_array();
84          }
85
86          //function to get notebook from passed notebook id from controller
87          public function get_notes_by_notebook($notebook_id)
88          {
89              $this->db->order_by('notes.noteid', 'DESC');
90              $this->db->join('notebooks', 'notebooks.notebookid = notes.notebook_id');
91              $query = $this->db->get_where('notes', array('notebook_id'=> $notebook_id));
92              return $query->result_array();
93          }
94
95          function get_pin()
96          {
97              $query = $this->db->query('SELECT * FROM notes WHERE user_id = '.$this->session->userdata('user_id').' AND pin = 2');
98              return $query->result_array();
99          }
100     }
```

Figure 103 code of note model part3

Notebook_model.php ×

application ▸ models ▸ Notebook_model.php

```php
<!--this is a model of notebook-->
<?php
    class Notebook_model extends CI_Model
    {
        //defulat constructor which load data from database
        public function __construct()
        {
            $this->load->database();
        }

         //function to get notebook from database
        public function get_notebooks()
            {
                $id= array(
                    'user_id' => $this->session->userdata('user_id'));
                $this->db->order_by('notebookname');
                $query = $this->db->get('notebooks');
                $query = $this->db->get_where('notebooks', $id);
                return $query->result_array();
            }

        //function to add new notebook
        public function add_notebook()
        {
            $data = array(
                'notebookname'=> $this->input->post('name'),
                'user_id' => $this->session->userdata('user_id')
            );

            return $this->db->insert('notebooks', $data);
        }

        //function to get notebook from databasse and get notebookid from controller
        public function get_notebook($notebook_id)
        {
            $query = $this->db->get_where('notebooks', array('notebookid'=>$notebook_id));
            return $query->row();
```

Figure 104 code of notebook model part1

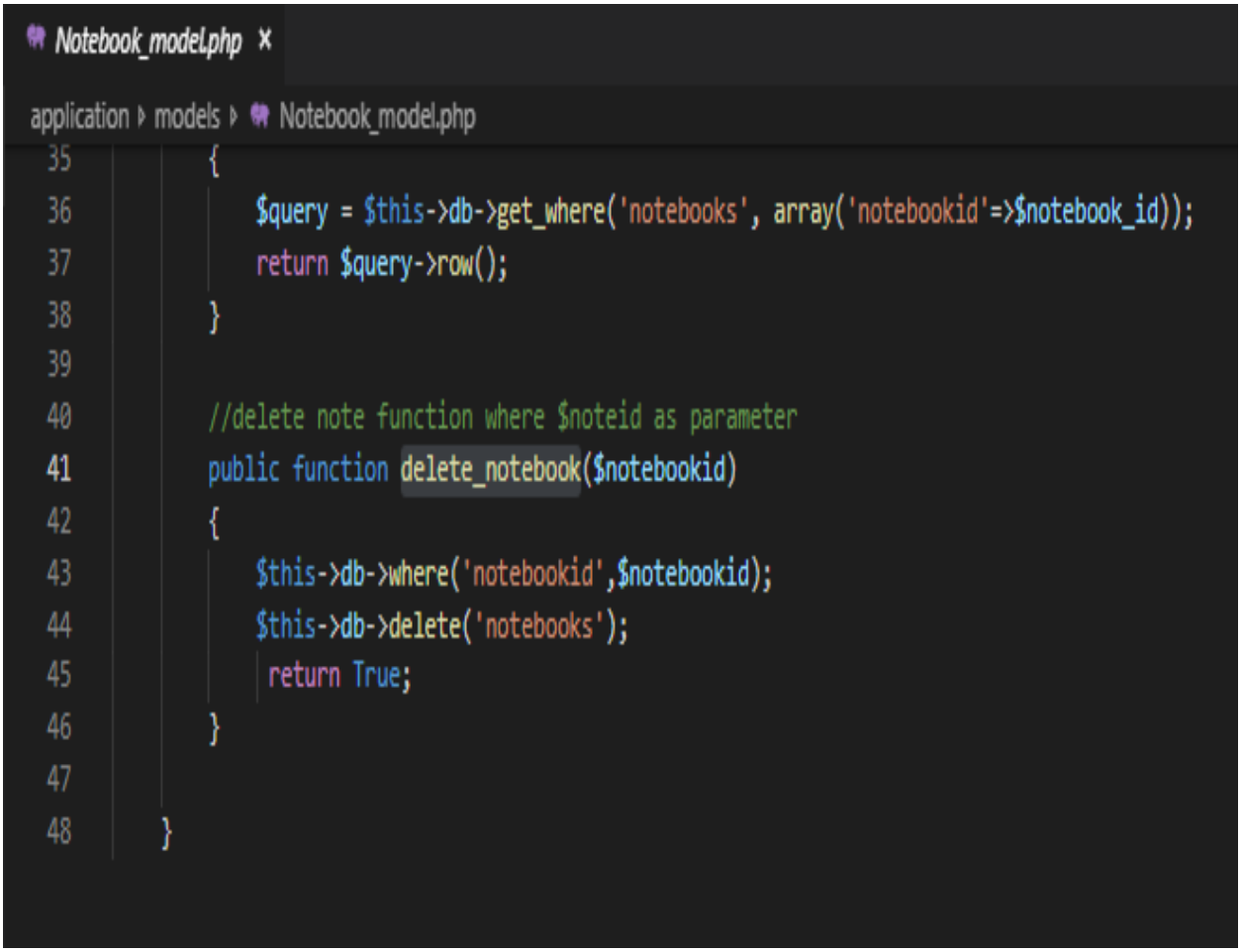


Figure 105 code of notebook model part2

Todo_model.php

application ▸ models ▸ Todo_model.php

```php
<!--this is a model of todo-->
<?php
    class Todo_model extends CI_Model
    {
        public function __construct()
        {
            $this->load->database();
        }

        //function to add tasks
        public function add_todo()
        {
            $data = array(
                'todoname'=> $this->input->post('todoname'),
                'user_id' => $this->session->userdata('user_id'),
                'tododate' => $this->input->post('date')
            );
            return $this->db->insert('todo', $data);
        }

        //function to set tasks  from database
        public function get_todo()
        {
            $id= array(
                'user_id' => $this->session->userdata('user_id'),
                'status' => 'incompleted',
                'tododate' => date('y-m-d')
            );
                $this->db->order_by('todoid', 'DESC');
                $query = $this->db->get_where('todo', $id);
                return $query->result_array();
        }

        //function to get tasks
        public function get_todo_c()
        {
            $id= array(
```

Figure 106 code of todo model part1

Todo_model.php ×

application ▸ models ▸ Todo_model.php

```php
35          public function get_todo_c()
36          {
37              $id= array(
38                  'user_id' => $this->session->userdata('user_id'),
39                  'status' => 'completed'
40              );
41                  $this->db->order_by('todoid', 'DESC');
42                  $query = $this->db->get_where('todo', $id);
43                  return $query->result_array();
44          }
45
46          //update todo function
47          public function update_todo($todoid)
48          {
49              $id = $this->session->userdata('user_id');
50              $data = array(
51                  'status' => 'completed'
52                  );
53                  $this->db->where('todoid', $todoid);
54                  return $this->db->update('todo', $data);
55          }
56
57          //function to count incomplete tasks
58          function notify_tasks()
59          {
60              $id= array(
61                  'user_id' => $this->session->userdata('user_id'));
62                  $this->db->count_all_results('todo');  // Produces an integer
63                  $this->db->where('user_id', $this->session->userdata('user_id'));
64                  $this->db->where('status', 'incompleted');
65                  $this->db->where('tododate', date('Y-m-d'));
66                  $this->db->from('todo');
67                  return $this->db->count_all_results();
68          }
69
70          function get_todo7()
71          {
```

Figure 107 code of todo model part2

Todo_model.php

application › models › Todo_model.php

```
72              $id= array(
73                  'user_id' => $this->session->userdata('user_id'),
74                  'status' => 'incompleted',
75                  'tododate' => date('y-m-d')
76              );
77              $this->db->select('*');
78              $this->db->where('tododate BETWEEN DATE_ADD(CURDATE(), INTERVAL -7 DAY) AND CURDATE()');
79              $this->db->where('user_id', $this->session->userdata('user_id'));
80              $this->db->where('status', 'incompleted');
81              $result = $this->db->get('todo');
82                  return $result->result_array();
83          }
84      } //end of class
```

Figure 108 code of todo model part3

Trash_model.php ×

application ▸ models ▸ Trash_model.php

```php
<!-- this is a model of trash-->
<?php
    class Trash_model extends CI_Model
    {
        //this function copys data from note table to trash table
        function trash_note($noteid)
        {
            $this->db->query('INSERT INTO trash (noteid, notename, notedetail, user_id, notebook_id, slug, post_image)
            SELECT noteid, notename, notedetail, user_id, notebook_id, slug, post_image
            FROM notes WHERE noteid = '.$noteid.'');
        }

        //this function shows all data from trash table
        function view_trash()
        {
            $id= array(
                'user_id' => $this->session->userdata('user_id'));
            $query = $this->db->get_where('trash', $id);
            return $query->result_array();
        }

        //this function copy data from trsh table to note table
        function recover($tid)
        {
            $this->db->query('INSERT INTO notes (notename, notedetail, user_id, notebook_id, slug, post_image)
            SELECT notename, notedetail, user_id, notebook_id, slug, post_image
            FROM trash WHERE tid = '.$tid.'');
        }

        //this function delete data from trsh table
        function delete($tid)
        {
            //delete data query and returns true
            $this->db->where('tid',$tid);
            $this->db->delete('trash');
            return True;
        }
```

Figure 109 code of trash model

user_model.php ×

application ▸ models ▸ user_model.php

```php
<!--this is a model of user-->
<?php
    class User_model extends CI_Model
    {
        public function register($enc_password)
        {
            //user data array
            $data = array(
                'name'=> $this->input->post('name'),
                'email'=> $this->input->post('email'),
                'password'=> $enc_password
            );
            //insert user
            return $this->db->insert('users', $data);
        }

        //LOG USER IN
        public function login($email, $password)
        {
            //validate
            $this->db->where('email', $email);
            $this->db->where('password', $password);

            $result= $this->db->get('users');
            if($result->num_rows() == 1)
            {

                return $result->row(0)->userid;
            }
            else
            {
                return FALSE;
            }
        }

        //function to get login log from database
        public function login_history_set()
```

Figure 110 code of user model part1

user_model.php ×

application › models › user_model.php

```
38          {
39              $data = array(
40                  'user_id' => $this->session->userdata('user_id'),
41                  'email' => $this->session->userdata('email')
42              );
43              $this->db->insert('login_history', $data);
44          }
45
46          //function to view login log
47          public function login_history_view()
48          {
49              $data = array(
50                  'user_id' => $this->session->userdata('user_id'),
51                  'email' => $this->session->userdata('email')
52              );
53
54              $query = $this->db->get('login_history');
55              $query = $this->db->get_where('login_history', $data);
56              return $query->result();
57          }
58
59          //check email exists
60          public function check_email_exists($email)
61          {
62              $query = $this->db->get_where('users', array('email'=> $email));
63              if(empty($query->row_array()))
64              {
65                  return TRUE;
66              }
67              else
68              {
69                 return FALSE;
70              }
71          }
72
73          //function to get users
74          public function get_user()
```

Figure 111 code of user model part2

user_model.php

application ▸ models ▸ user_model.php

```php

    //function to get users
    public function get_user()
    {
        $id= array(
            'userid' => $this->session->userdata('user_id'));
        $query = $this->db->get('users');
        $query = $this->db->get_where('users', $id);
        return $query->row_array();; //returns $query in result_array()
    }

    // function to update user
    public function update_user()
    {
        $id = $this->session->userdata('user_id');

        $data = array(
            'name' => $this->input->post('name'),
            'email' => $this->input->post('email')
        );

            $this->db->where('userid', $id);
            return $this->db->update('users', $data);
    }

    //function to change password of user
    public function change_password_m($enc_password1)
    {
        $id = $this->session->userdata('user_id');
        $data = array(
            'password'=> $enc_password1
        );
            $this->db->where('userid', $id);
            return $this->db->update('users', $data);
    }

    //function to count notes of that particular user
```

Figure 112 code of user model part3

user_model.php ×

application ▸ models ▸ user_model.php

```
109         public function count_note()
110         {
111             $id= array(
112                 'user_id' => $this->session->userdata('user_id'));
113             $query = $this->db->query('SELECT * FROM notes where user_id ='.$id);
114             return $query->num_rows();
115         }
116 
117         //function to count rows and return integer value
118         public function count_rows($id)
119         {
120             $this->db->count_all_results('notes');  // Produces an integer
121             $this->db->where('user_id', ($id));
122             $this->db->from('notes');
123             return $this->db->count_all_results();;
124         }
125 
126         //function to count tasks and reutrn integer value
127         public function count_ctasks($id)
128         {
129             $this->db->count_all_results('todo');  // Produces an integer
130             $this->db->where('user_id', ($id));
131             $this->db->from('todo');
132             return $this->db->count_all_results();
133         }
134     } // end of user_model class
```

Figure 113 code of user model part4

Testing

index.php

application ▸ views ▸ testing ▸ index.php

```
<!--this is a index page of note where added notes are shown-->
<!--this part presents cart section-->
<div class="card card-4">
    <div class="card-body">
    <!--this shows title of this page-->
    <div class="row card-header">
        <div class="col-md-6">
            <h2 class="text"><strong>Testing</strong></h2>
        </div>
    </div>
    <div class="container">
        <?php if($try):
        {
            echo $try;
        } endif; ?>

        <?php if($try1):
        {
            echo $try1;
        } endif; ?>

        <?php if($try2):
        {
            echo $try2;
        } endif; ?>

        <?php if($try3):
        {
            echo $try3;
        } endif; ?>

        <?php if($try4):
        {
            echo $try4;
        } endif; ?>
```

Figure 114 code of testing part1

```
36
37          <?php if($try5):
38          {
39              echo $try5;
40          } endif; ?>
41
42          <?php if($try6):
43          {
44              echo $try6;
45          } endif; ?>
46
47          <?php if($try7):
48          {
49              echo $try7;
50          } endif; ?>
51      </div>
52
```

Figure 115 code of testing part2

Testing.php

application ▸ controllers ▸ Testing.php

```php
<?php
    defined('BASEPATH') OR exit('No direct script access allowed');
    class Testing extends CI_Controller
    {
        public function _construct()
        {
            parent::_construct();
            $this->load->library('unit_test');
        }

        public function index()
        {
            $test = $this->check_array();
            $expected_result = 'is_array';
            $test_name = 'Array check';
            $test_note = 'This is a test for checking array input and output';

            $test1 = $this->pass_login('asd@asd', 'Sanamasd@12345');
            $expected_result1 = 1;
            $test_name1 = 'Login Test';
            $test_note1 = 'This is a test for pass login';

            $test2 = $this->fail_login('asdasd@asd', 'Sanamasd@12345');
            $expected_result2 = True;
            $test_name2 = 'Login Test';
            $test_note2 = 'This is a test for fail login';

            $test3 = $this->chk_email('sanam@gmail.com');
            $expected_result3 = False;
            $test_name3 = 'Check email';
            $test_note3 = 'This is a test for checking email exists or not';

            $test4 = $this->chk_email('test@gmail.com');
            $expected_result4 = True;
            $test_name4 = 'Check email';
            $test_note4 = 'This is a test for checking email exists or not';

```

Figure 116 code of testing controller part1

Testing.php ×

application › controllers › Testing.php

```
38              $test5 = $this->count_notes('1');
39              $expected_result5 = 6;
40              $test_name5 = 'count all notes';
41              $test_note5 = 'This is a test for counting notes';
42
43              $test6 = $this->count_tasks('1');
44              $expected_result6 = 21;
45              $test_name6 = 'count all tasks';
46              $test_note6 = 'This is a test for counting tasks';
47
48              $test7 = $this->delete_notebook('1');
49              $expected_result7 = True;
50              $test_name7 = 'Delete notebook';
51              $test_note7 = 'This is a pass test for deleting notebooks';
52
53              $try['try'] = $this->unit->run($test, $expected_result, $test_name, $test_note);
54              $try['try1'] = $this->unit->run($test1, $expected_result1, $test_name1, $test_note1);
55              $try['try2'] = $this->unit->run($test2, $expected_result2, $test_name2, $test_note2);
56              $try['try3'] = $this->unit->run($test3, $expected_result3, $test_name3, $test_note3);
57              $try['try4'] = $this->unit->run($test4, $expected_result4, $test_name4, $test_note4);
58              $try['try5'] = $this->unit->run($test5, $expected_result5, $test_name5, $test_note5);
59              $try['try6'] = $this->unit->run($test6, $expected_result6, $test_name6, $test_note6);
60              $try['try7'] = $this->unit->run($test7, $expected_result7, $test_name7, $test_note7);
61              $this->load->view('testing/index',$try);
62              $this->load->view('includes/footer');
63          }
64
65          function check_array()
66          {
67              $ut = array(
68                  'name' => 'test',
69                  'email' => 'test@test.com',
70                  'password' => 'password');
71                  return $ut;
72          }
73
74          function pass_login($email, $password)
```

Figure 117 code of testing controller part2

Testing.php ×

application ▸ controllers ▸ Testing.php

```
75          {
76              return $this->user_model->login($email, md5($password));
77          }
78
79          function fail_login($email, $password)
80          {
81              return $this->user_model->login($email, md5($password));
82          }
83
84          function chk_email($email)
85          {
86              return $this->user_model->check_email_exists($email);
87          }
88
89          function count_notes($id)
90          {
91              return $this->user_model->count_rows($id);
92          }
93
94          function count_tasks($id)
95          {
96              return $this->user_model->count_ctasks($id);
97          }
98
99          function delete_notebook($notebook_id)
100         {
101             return $this->notebook_model->delete_notebook($notebook_id);
102         }
103     }
104 ?>
```

Figure 118 code of testing controller part3