

Table of Contents

Chapter 1 Introduction.....	6
Project introduction.....	6
Overview of the project.....	6
Aims and objectives.....	6
Chapter – 2 Analysis	7
Introduction to analysis.....	7
Why to perform analysis?.....	7
Analysis methodology.....	8
Feasibility study.....	10
Software Requirements Specification	11
Functional requirements	11
Non-functional requirements.....	13
Requirement Prioritization	14
Hardware software specification.....	15
Use case diagram	16
Initial class diagram.....	22
Brief description about project.....	22
Natural language analysis (NLA).....	22
Chapter – 4 Design	24
Structural model	24
Final class diagram	24
Behavioral model.....	27
Activity diagram.....	27
Database model.....	39
Entity-Relationship diagram	40
UI modeling	41
Chapter 5 Implementation.....	46
Chapter-6 Testing	47
Chapter 6 Other project issues.....	54
Limitation of the project	54
Future work	54

Risk management	54
Configuration management	56
User manual	61
Chapter 7 Conclusion.....	71
References and bibliography.....	73
Appendix.....	74

Table of Figures

Figure 1 Data flow diagram	9
Figure 2 Use case diagram for notes	17
Figure 3 Use case diagram for plan	20
Figure 4 Initial class diagram.....	23
Figure 5 final class diagram	25
Figure 6 flow chart of the application.....	26
Figure 7 Activity diagram of registration	28
Figure 8 Activity diagram of login system	29
Figure 9 Create note activity diagram.....	30
Figure 10 Edit note activity diagram	31
Figure 11 Delete note activity diagram	32
Figure 12 Registration sequence diagram	35
Figure 13 User login sequence diagram.....	36
Figure 14 add note sequence diagram.....	37
Figure 15 Edit note sequence diagram.....	37
Figure 16 Delete note sequence diagram.....	38
Figure 17 ER diagram for the purposed application	40
Figure 18 configuration part1	56
Figure 19 configuration part2	57
Figure 20 configuration part3	58
Figure 21 configuration part4	59
Figure 22 configuration part5	60
Figure 23 Registration of user	61
Figure 24 logging in	62
Figure 25 Adding notebook by user.....	63
Figure 26 Deleting notebook by user.....	63
Figure 27 adding note by user.....	64
Figure 28 viewing note by user	65
Figure 29 viewing note full content by user	66
Figure 30 editing note by user.....	67
Figure 31 recovering data/ deleting data from trash	68
Figure 32 Back up user data	69
Figure 33 adding task by user	69
Figure 34 Viewing task by user	70
Figure 35 all folders and files for the system	74
Figure 36 code of header part1	75
Figure 37 code of header part2	76
Figure 38 code of header part3	77
Figure 39 code of header part4	78
Figure 40 code of header part5	79
Figure 41 code of footer	80
Figure 42 code of notebook part1.....	81
Figure 43 code of notebook part2.....	82
Figure 44 code of add note part1	83
Figure 45 code of add note part2	84
Figure 46 code of edit note part1	85

Figure 47 code of edit note part2	86
Figure 48 code of index note part1.....	87
Figure 49 code of index note part2.....	88
Figure 50 code of trash part1	89
Figure 51 code of trash part2	90
Figure 52 code of view note part1	91
Figure 53 code of view note part2	92
Figure 54 code of add task part1.....	93
Figure 55 code of add task part2.....	94
Figure 56 code of index task part1	95
Figure 57 code of index task part2	96
Figure 58 code of index task part3	97
Figure 59 code of view task.....	98
Figure 60 code of change password part1	99
Figure 61 code of change password part2	100
Figure 62 code of edit profile part1.....	101
Figure 63 code of edit profile part2.....	102
Figure 64 code of login history par1	103
Figure 65 code of login history part2	104
Figure 66 code of login part1.....	105
Figure 67 code of login part2.....	106
Figure 68 code of profile part1	107
Figure 69 code of profile part2	108
Figure 70 code of profile part3	109
Figure 71 code of register part1	110
Figure 72 code of register part2	111
Figure 73 code of export data part1\	112
Figure 74 code of export data part2	113
Figure 75 code of displaying all data	114
Figure 76 code of export data controller part1	115
Figure 77 code of export data controller part2	116
Figure 78 code of notebook controller part1	117
Figure 79 code of notebook controller part2	118
Figure 80 code of notebook controller part3	119
Figure 81 code of note controller part1	120
Figure 82 code of note controller part2	121
Figure 83 code of note controller part3	122
Figure 84 code of note controller part4	123
Figure 85 code of note controller part4	124
Figure 86 code of profile controller part1	125
Figure 87 code of task controller part1	126
Figure 88 code of task controller part2	127
Figure 89 code of task controller part3	128
Figure 90 code of trash controller part1	129
Figure 91 code of trash controller part2	130
Figure 92 code of users controller part1	131
Figure 93 code of users controller part2	132

Figure 94 code of users controller part3	133
Figure 95 code of users controller part4	134
Figure 96 code of users controller part5	135
Figure 97 code of users controller part6	136
Figure 98 code of users controller part7	137
Figure 99 code of users controller part8	138
Figure 100 code of export model.....	139
Figure 101 code of note model part1.....	140
Figure 102 code of note model part2.....	141
Figure 103 code of note model part3.....	142
Figure 104 code of notebook model part1	143
Figure 105 code of notebook model part2	144
Figure 106 code of todo model part1.....	145
Figure 107 code of todo model part2.....	146
Figure 108 code of todo model part3.....	147
Figure 109 code of trash model.....	148
Figure 110 code of user model part1.....	149
Figure 111 code of user model part2.....	150
Figure 112 code of user model part3.....	151
Figure 113 code of user model part4.....	152
Figure 114 code of testing part1	153
Figure 115 code of testing part2.....	154
Figure 116 code of testing controller part1	155
Figure 117 code of testing controller part2	156
Figure 118 code of testing controller part3	157

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 for 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)

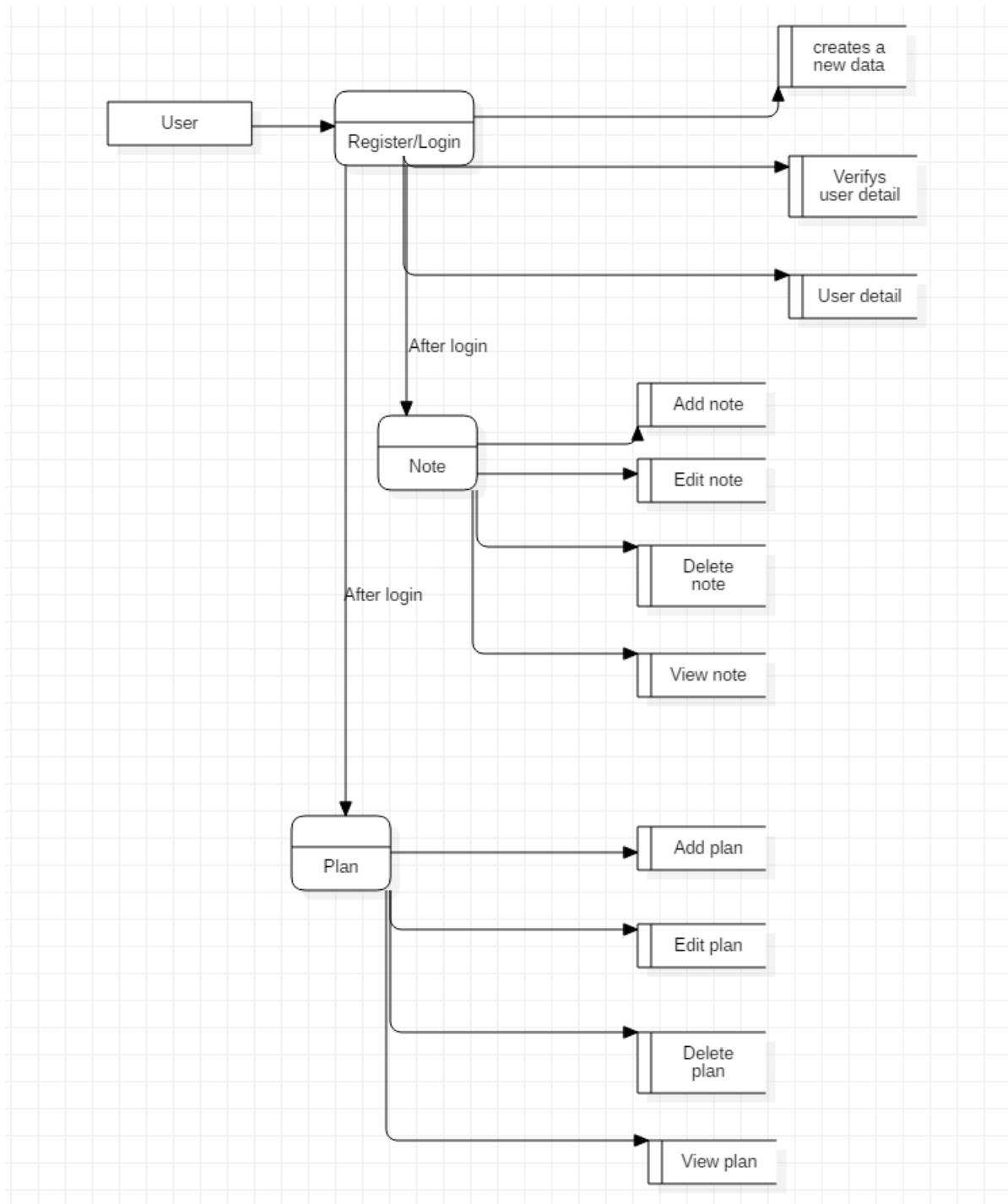


Figure 1 Data flow diagram

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

SOCIO-ECONOMIC 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, confidentiality 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

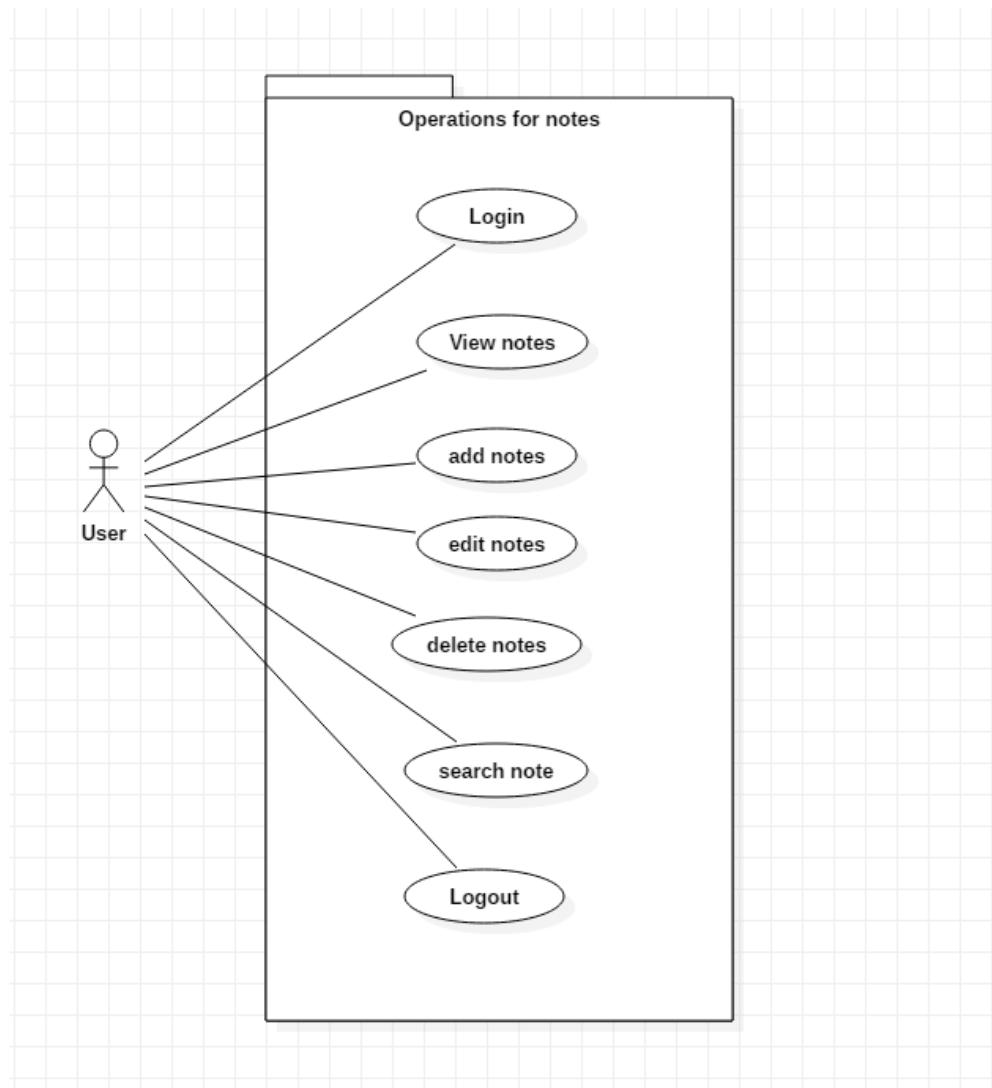


Figure 2 Use case diagram for notes

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	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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	<ol style="list-style-type: none"> 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

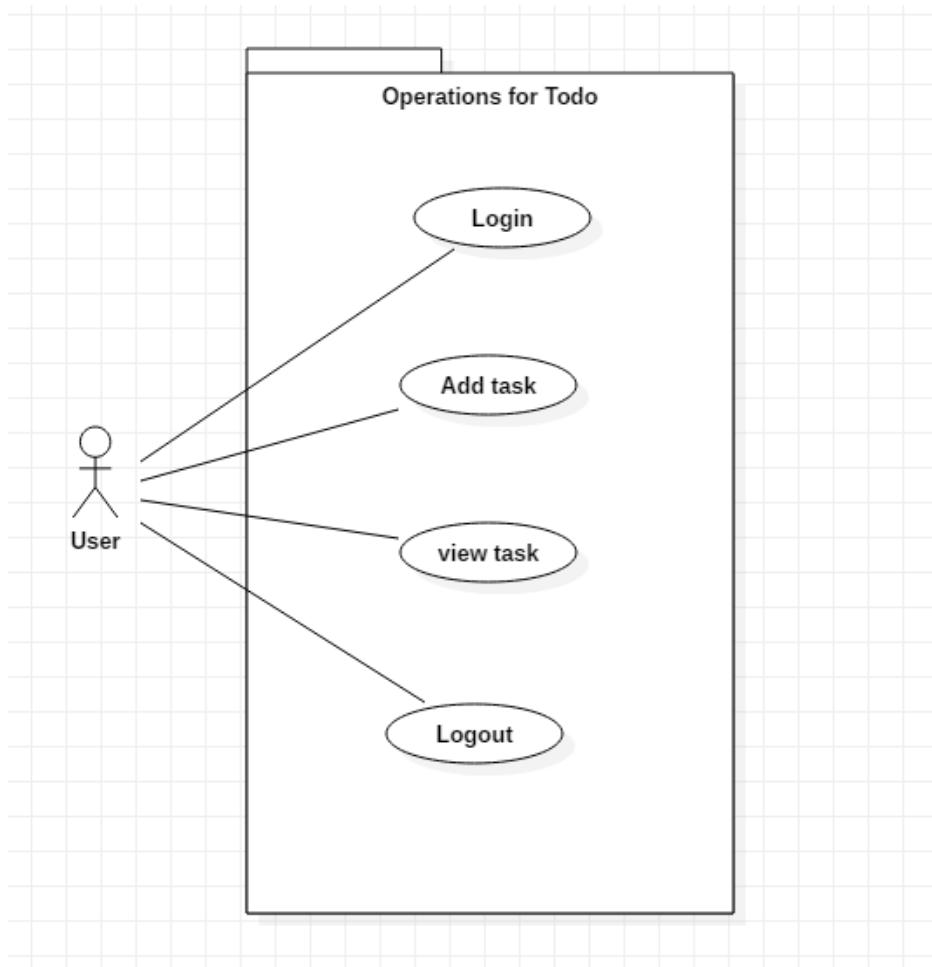


Figure 3 Use case diagram for plan

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	<ol style="list-style-type: none">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	<ol style="list-style-type: none">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)

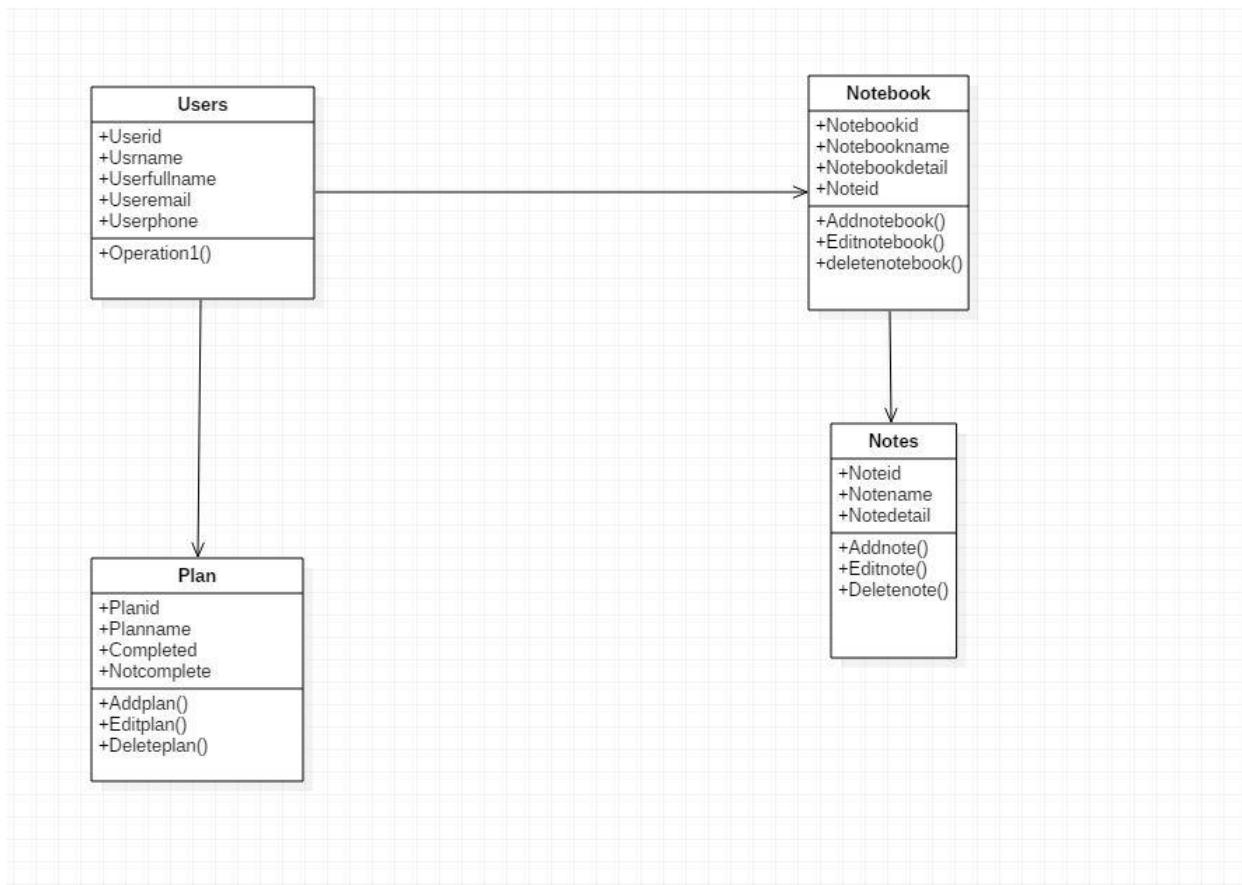


Figure 4 Initial class diagram

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
<pre> classDiagram class MyClass { +attribute1 : int -attribute2 : float #attribute3 : Circle +op1(in p1 : bool, in p2 : String) -op2(input p3 : int) : float #op3(out p6) : Class6* } </pre>	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.
<pre> classDiagram class Class1 class Class2 Class1 " " Class2 </pre>	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.
<pre> classDiagram class Class1 class Class2 Class1 "*" --> "1" Class2 </pre>	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.
<pre> classDiagram class Class1 class Class2 Class1 "*" --> "1" Class2 </pre>	The UML representation of a composition relationship shows composition as a <i>filled</i> diamond shape on the containing class end of the lines that connect contained class to the containing class.
<pre> classDiagram class Class1 class Class2 Class1 " " --> Class2 </pre>	A dependency is a semantic connection between dependent and independent model elements.

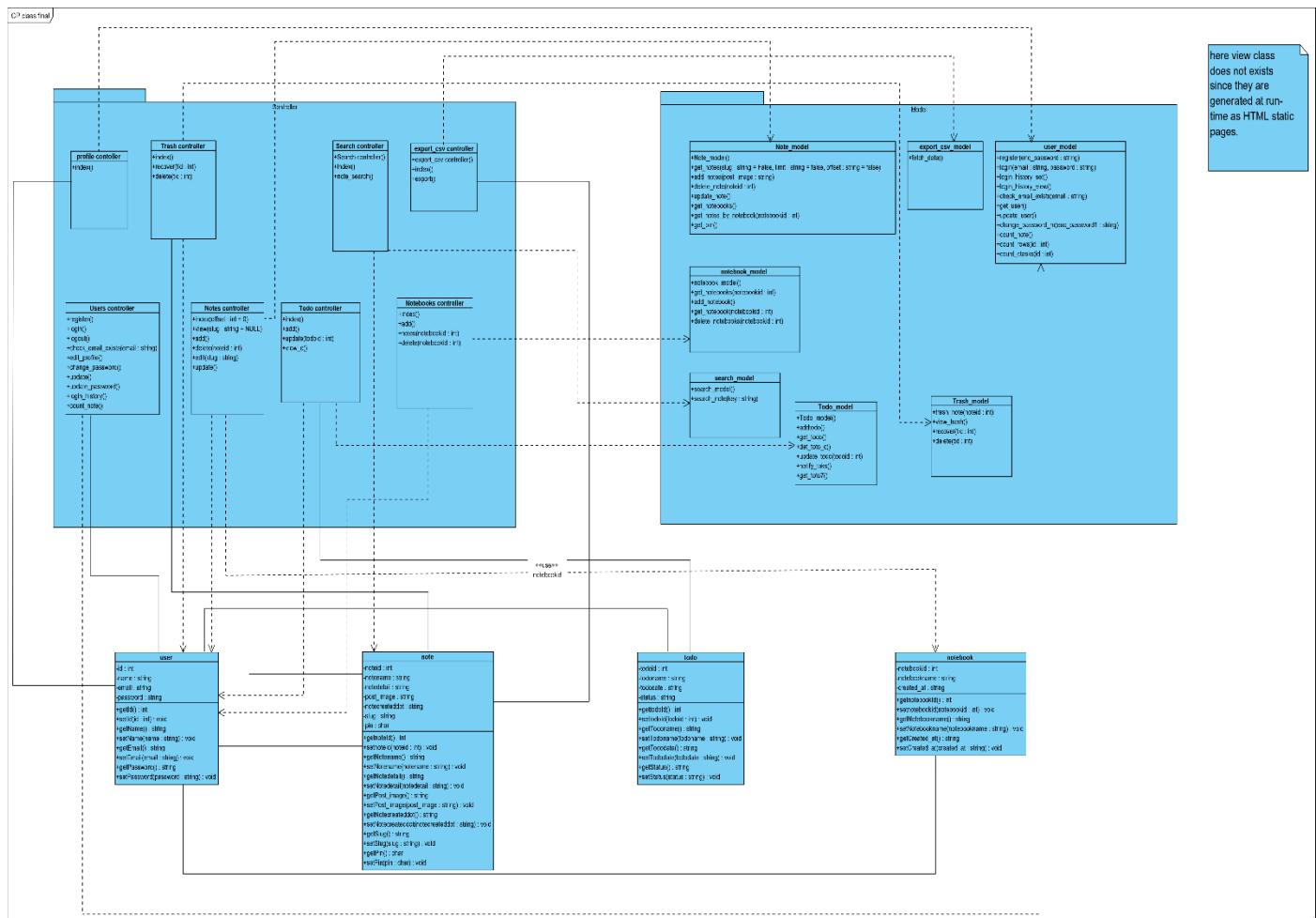


Figure 5 final class diagram

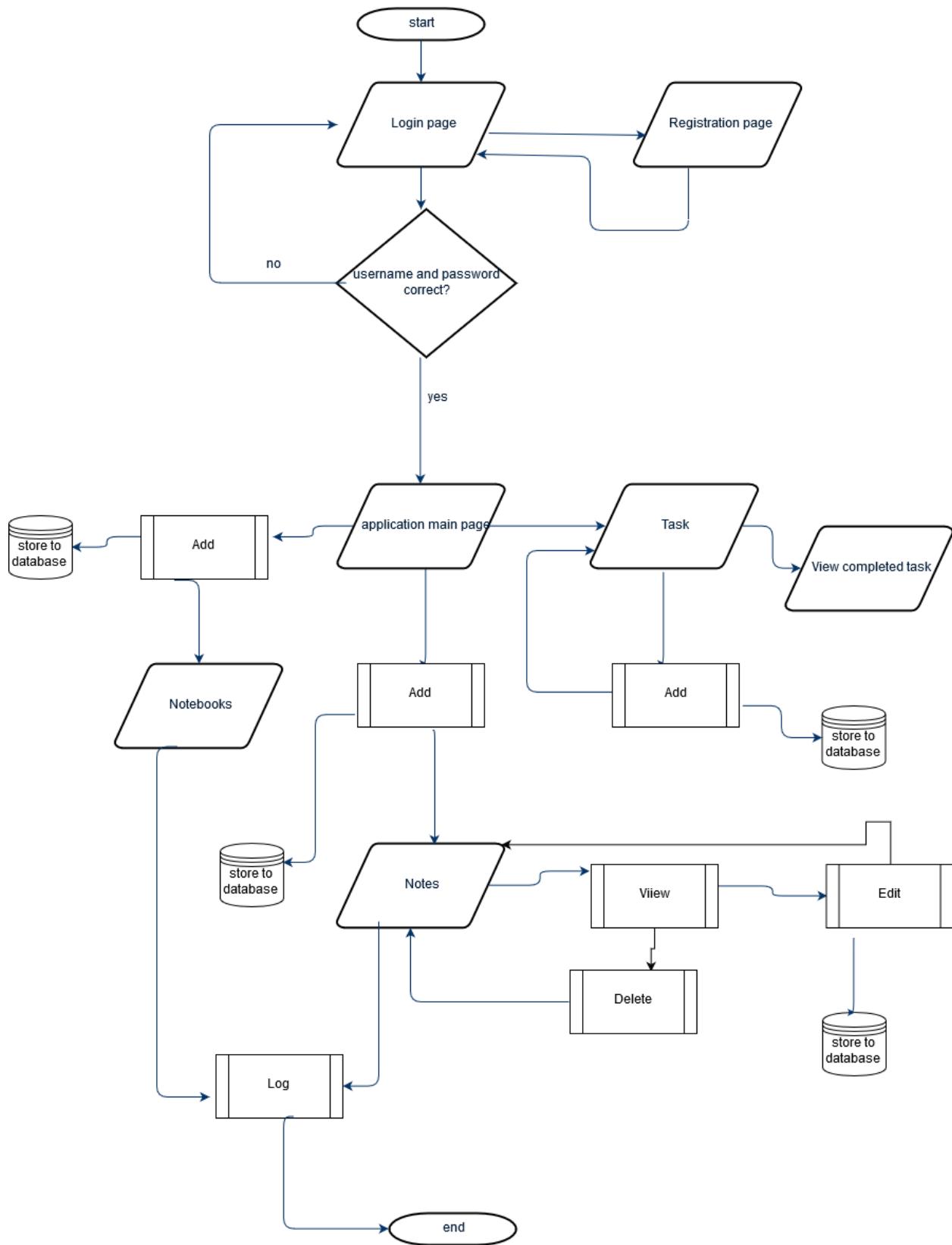


Figure 6 flow chart of the application

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

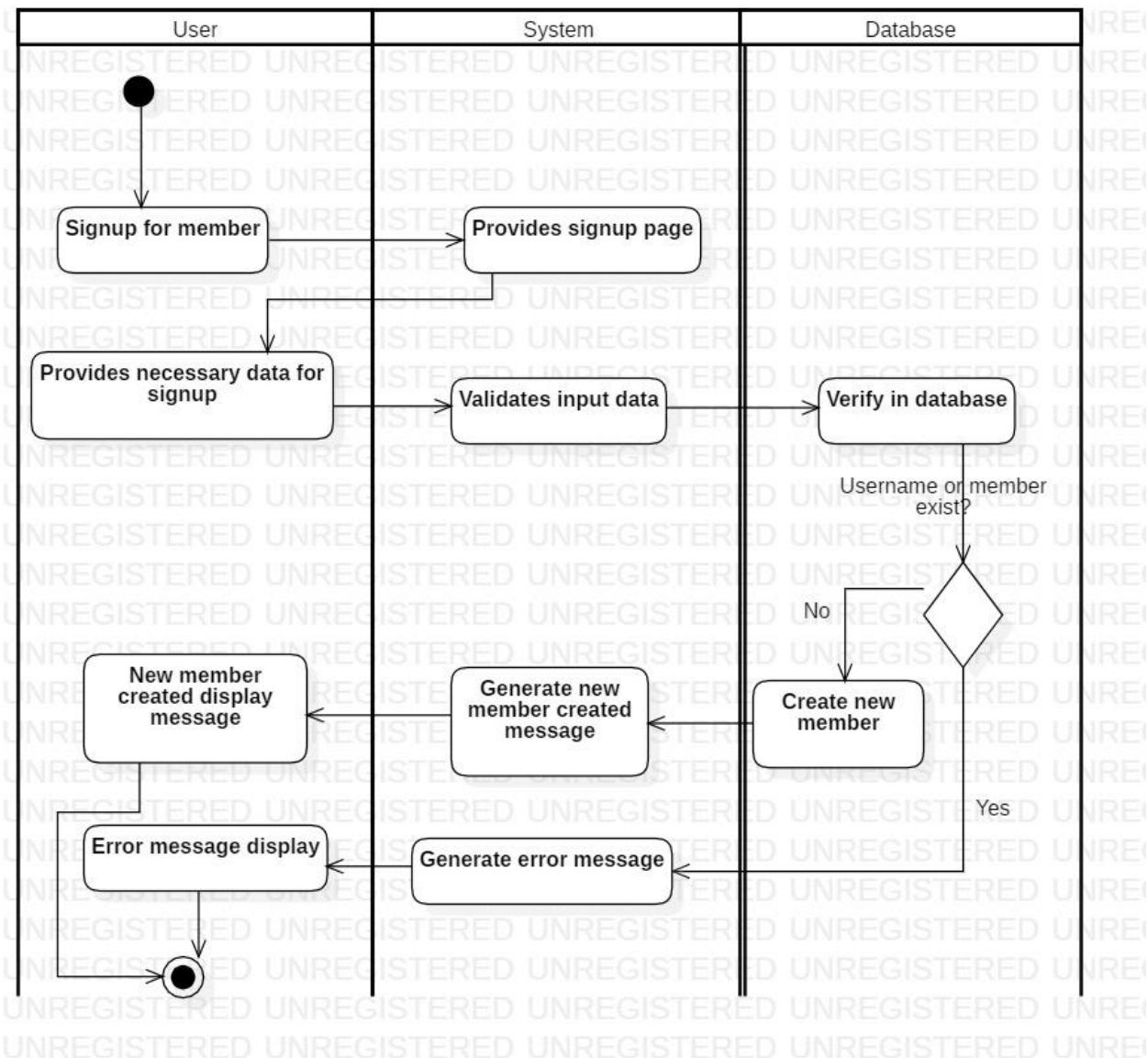


Figure 7Activity diagram of registration

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.

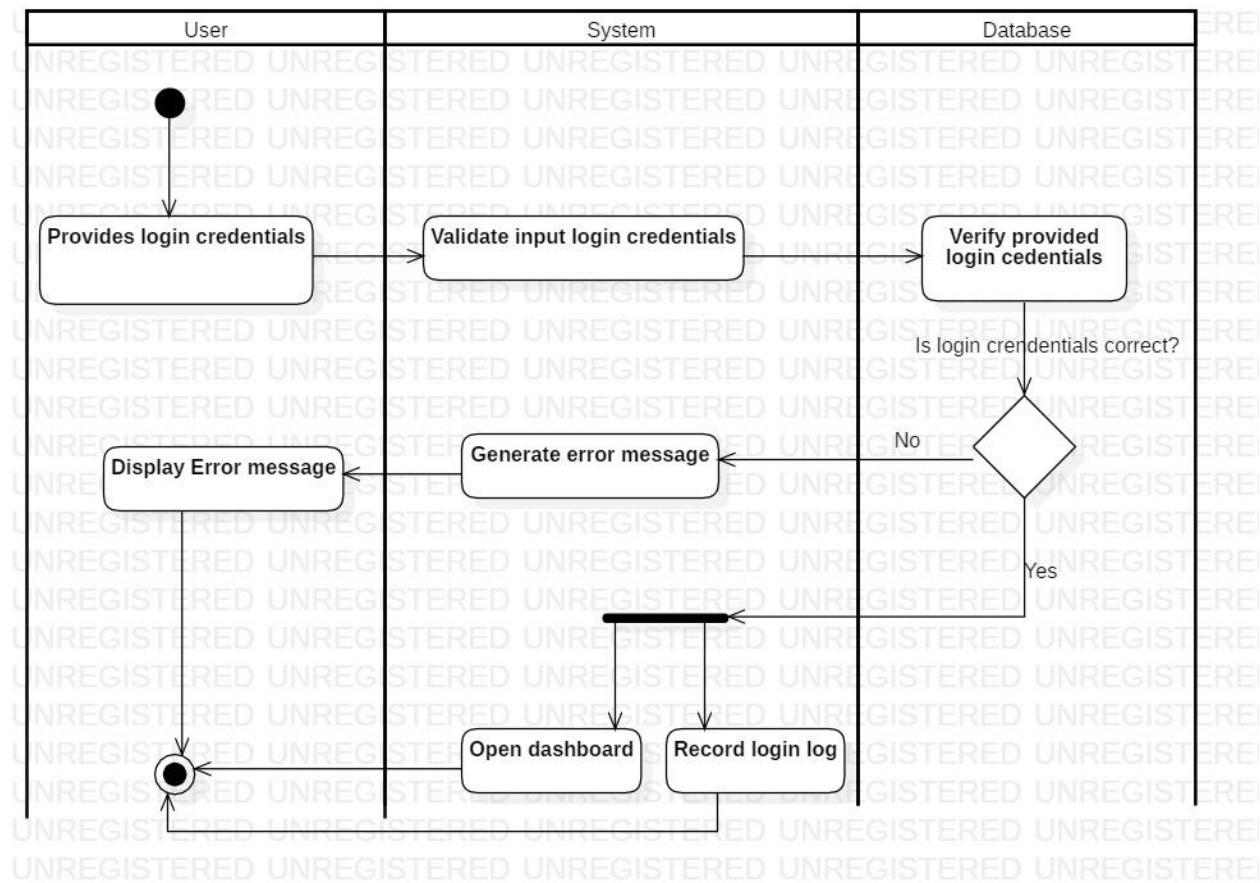


Figure 8Activity diagram of login system

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'.

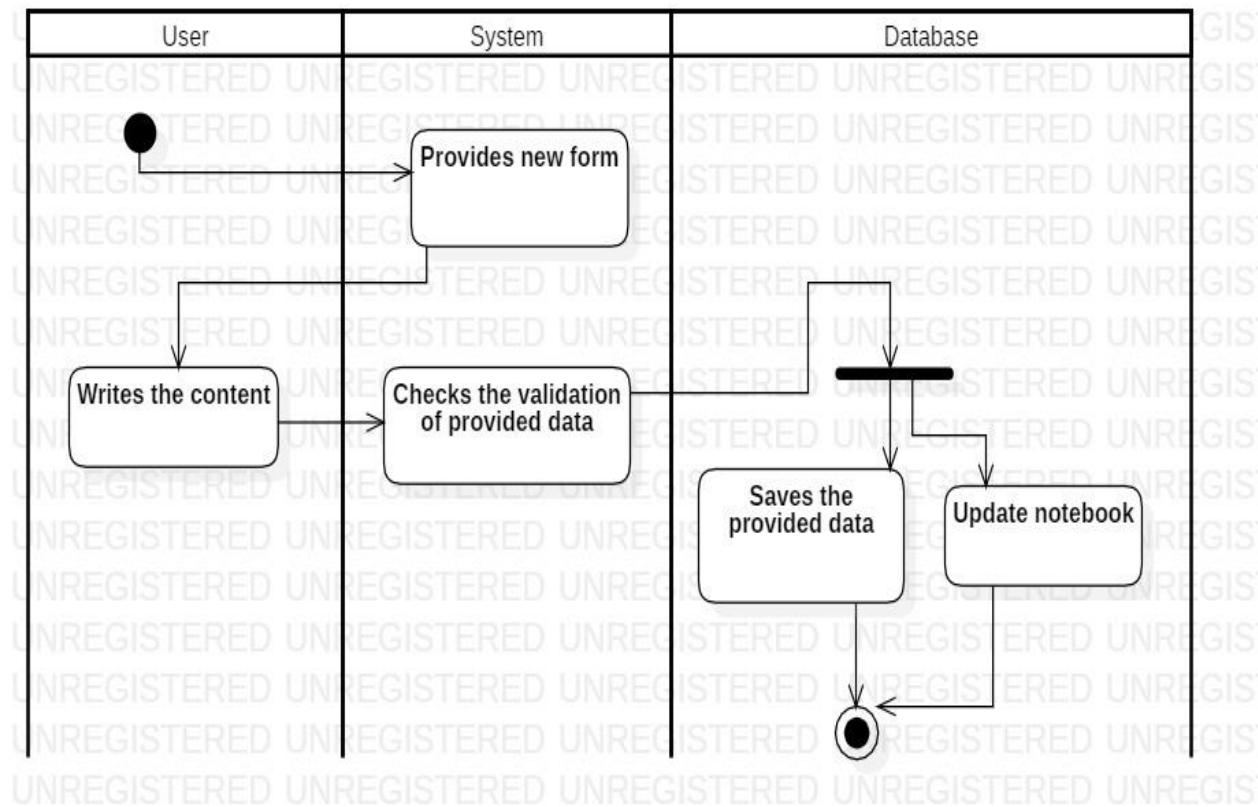


Figure 9Create note activity diagram

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.

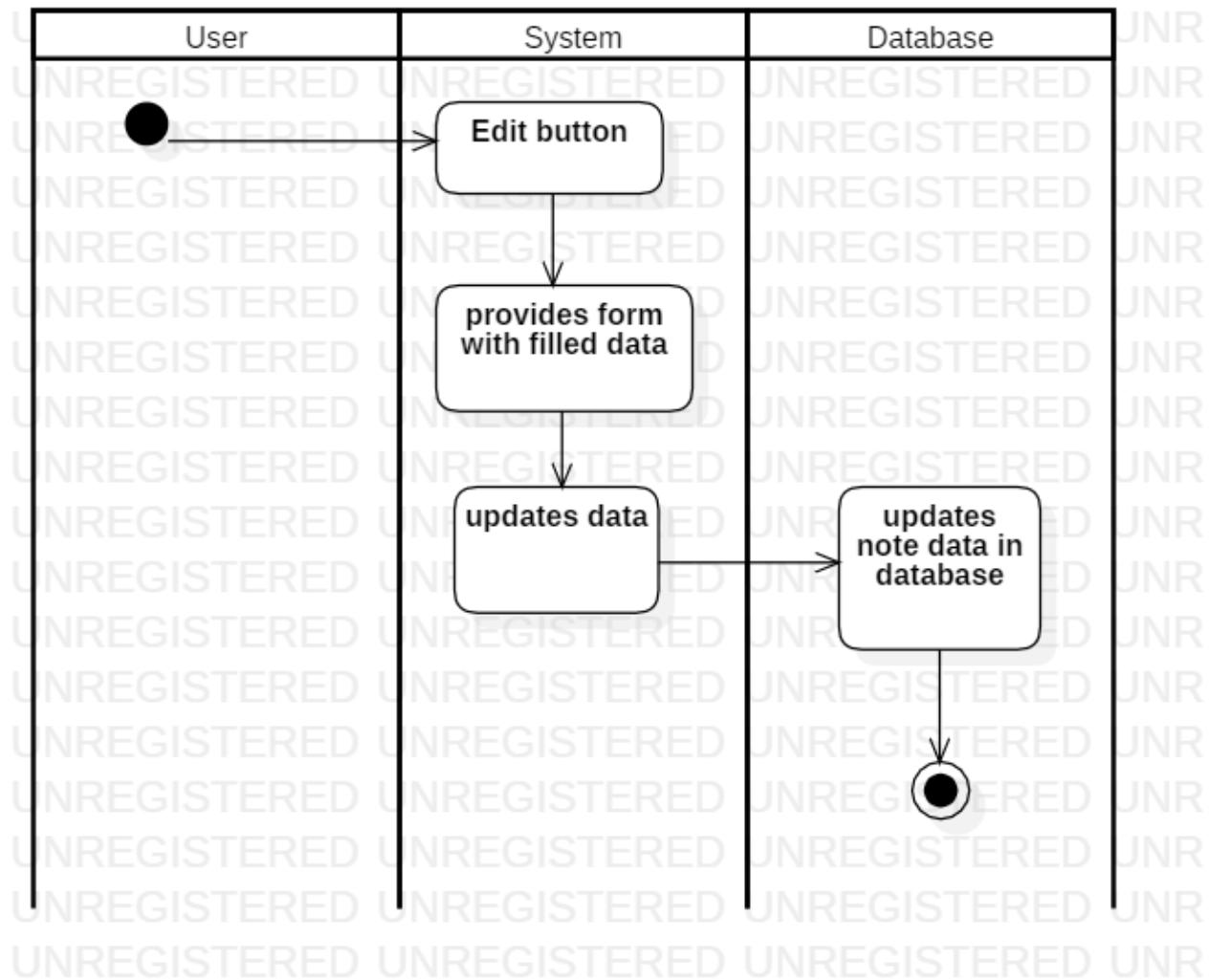


Figure 10 Edit note activity diagram

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.

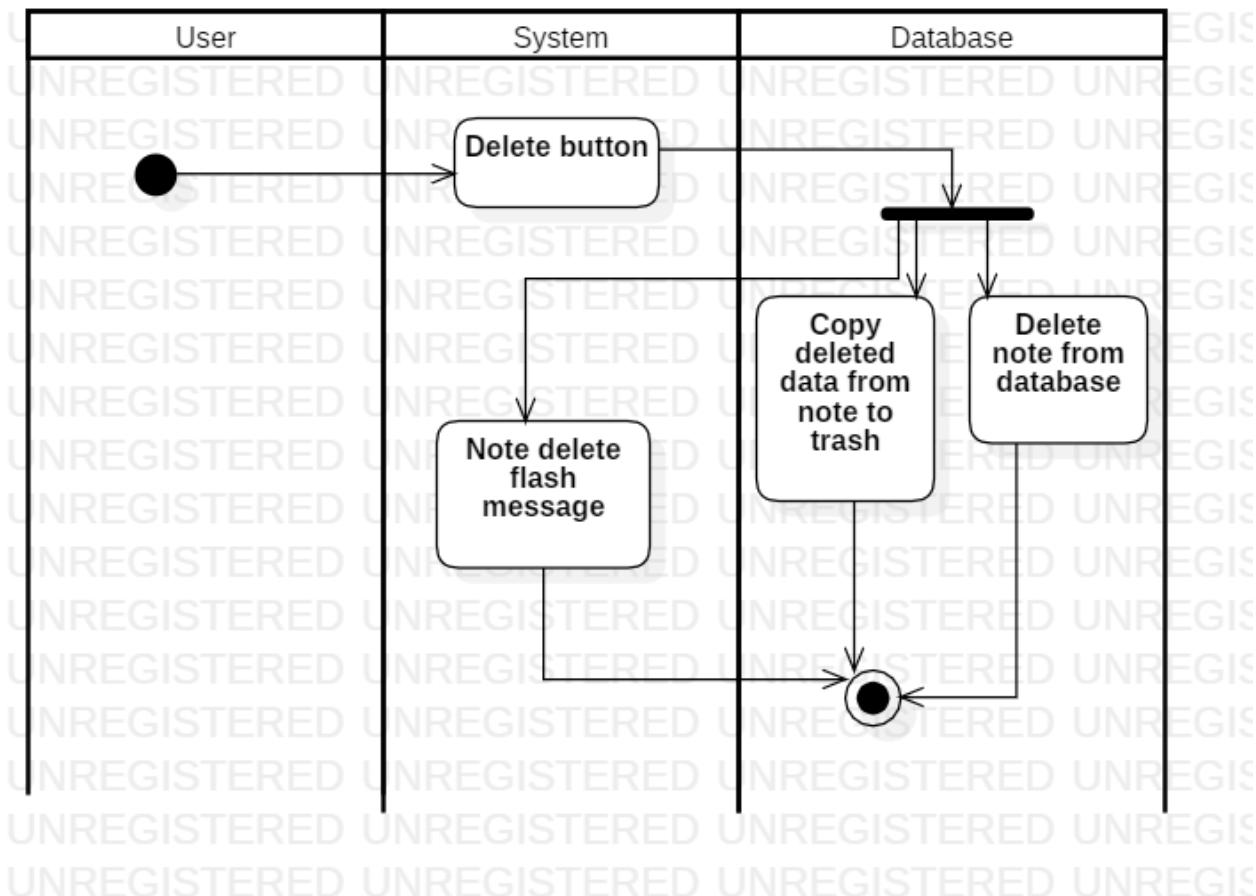


Figure 11 Delete note activity diagram

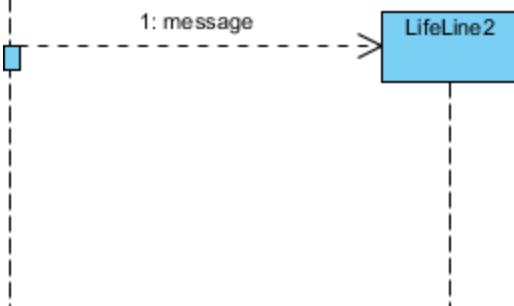
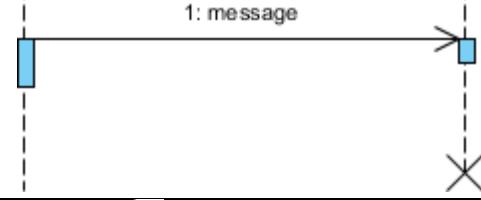
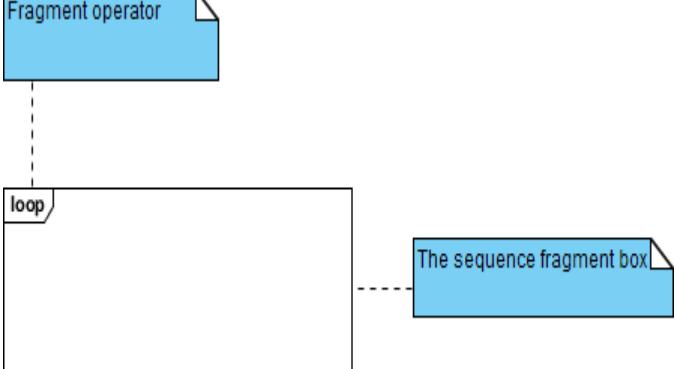
After user views note, users click on the delete button which runs multiple functions. In the database, insert query is done in which all data from the deleted note is copied and executed. At the same time delete query is executed which deletes that particular note data from the database. After that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

	<p>Recursive message is a kind of message that represents the invocation of message of the same lifeline. Its target points to an activation on top of the activation where the message was invoked from.</p>
	<p>Create message is a kind of message that represents the instantiation of (target) lifeline.</p>
	<p>Destroy message is a kind of message that represents the request of destroying the lifecycle of target lifeline.</p>
	<p>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.</p>
	<p>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 corner) indicates the type of fragment.;</p>

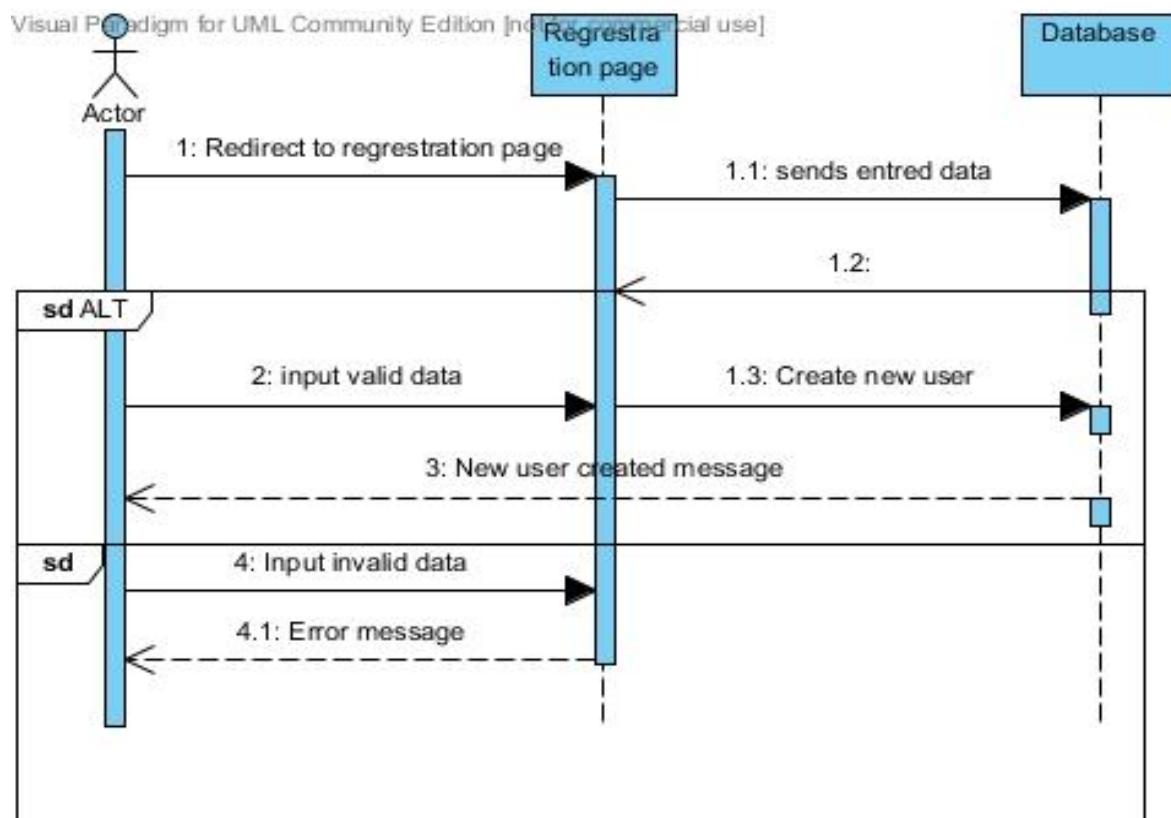


Figure 12 Registration sequence diagram

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.

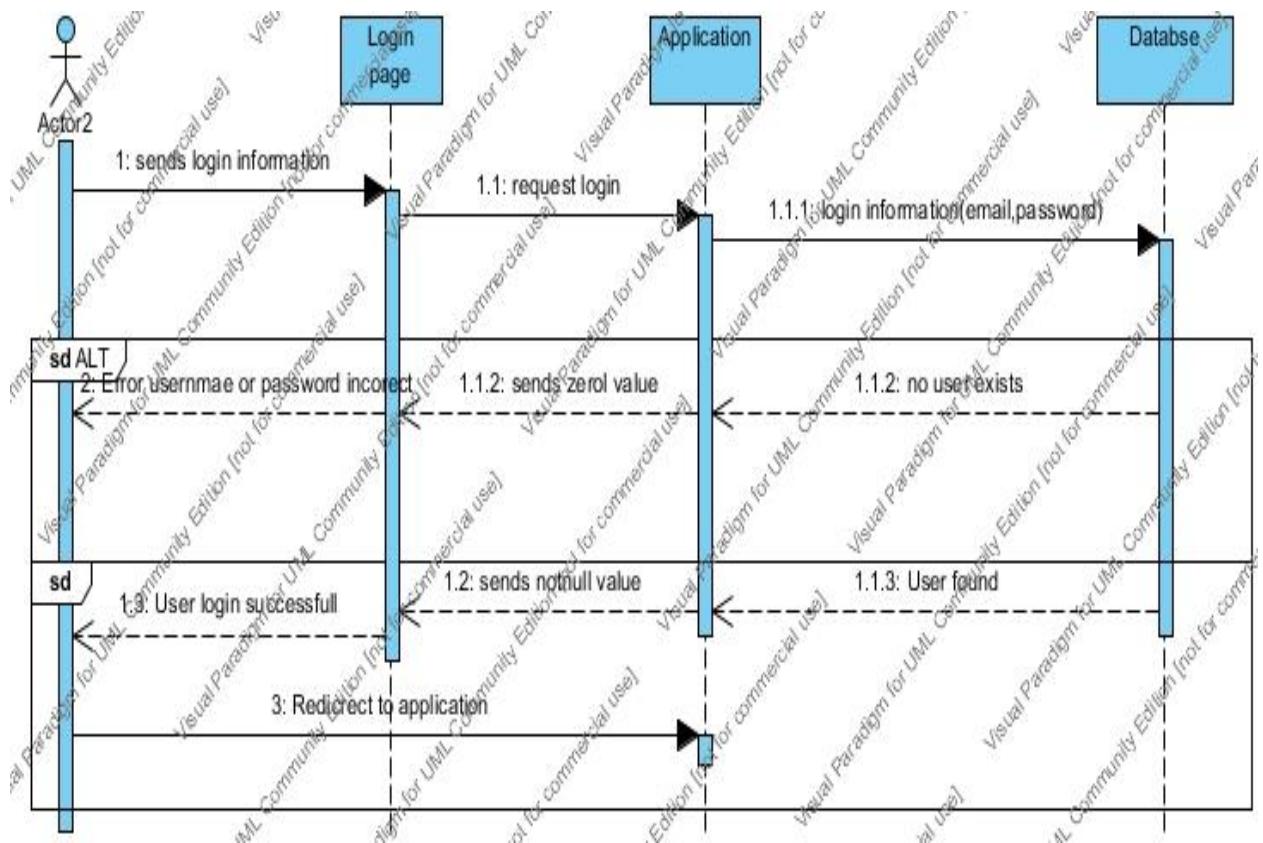


Figure 13 User login sequence diagram

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.

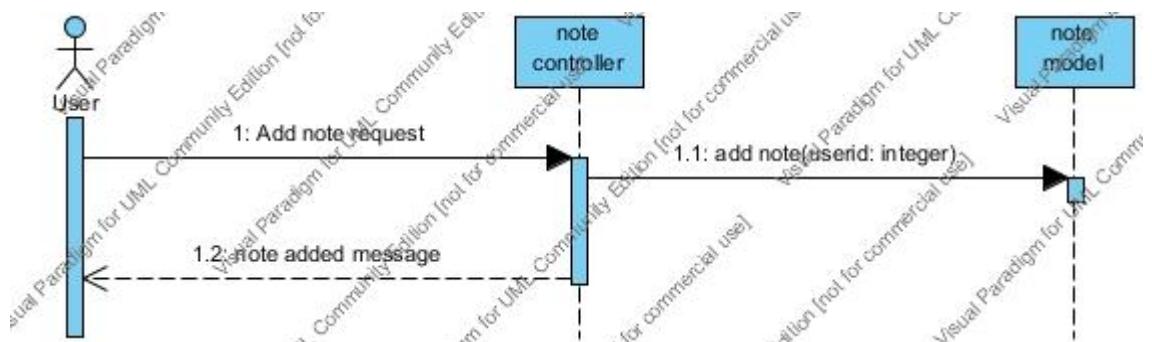


Figure 14 add note sequence diagram

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.

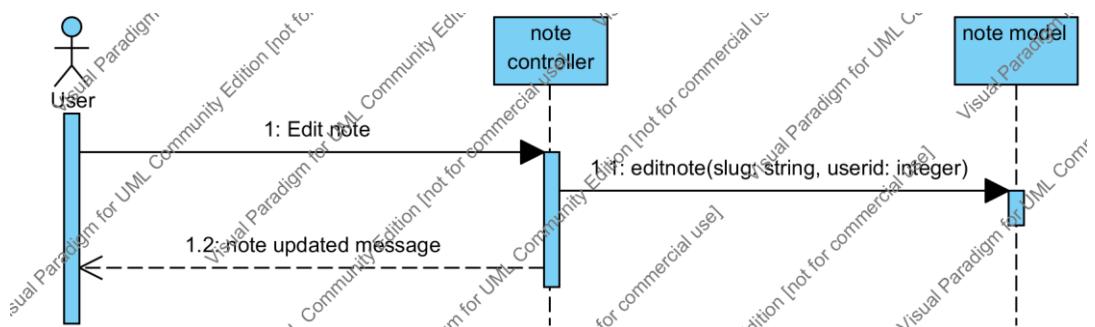


Figure 15 Edit note sequence diagram

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.

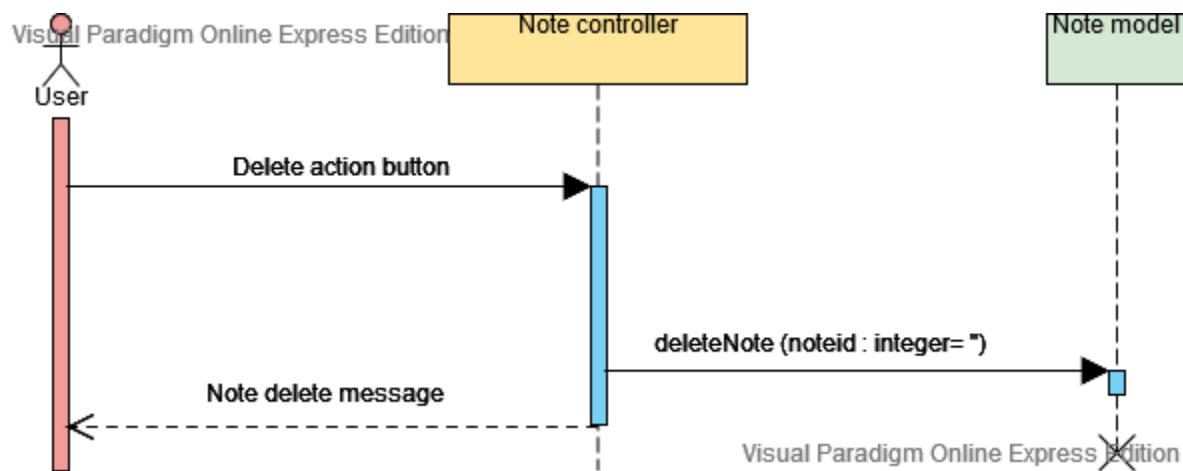


Figure 16 Delete note sequence diagram

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

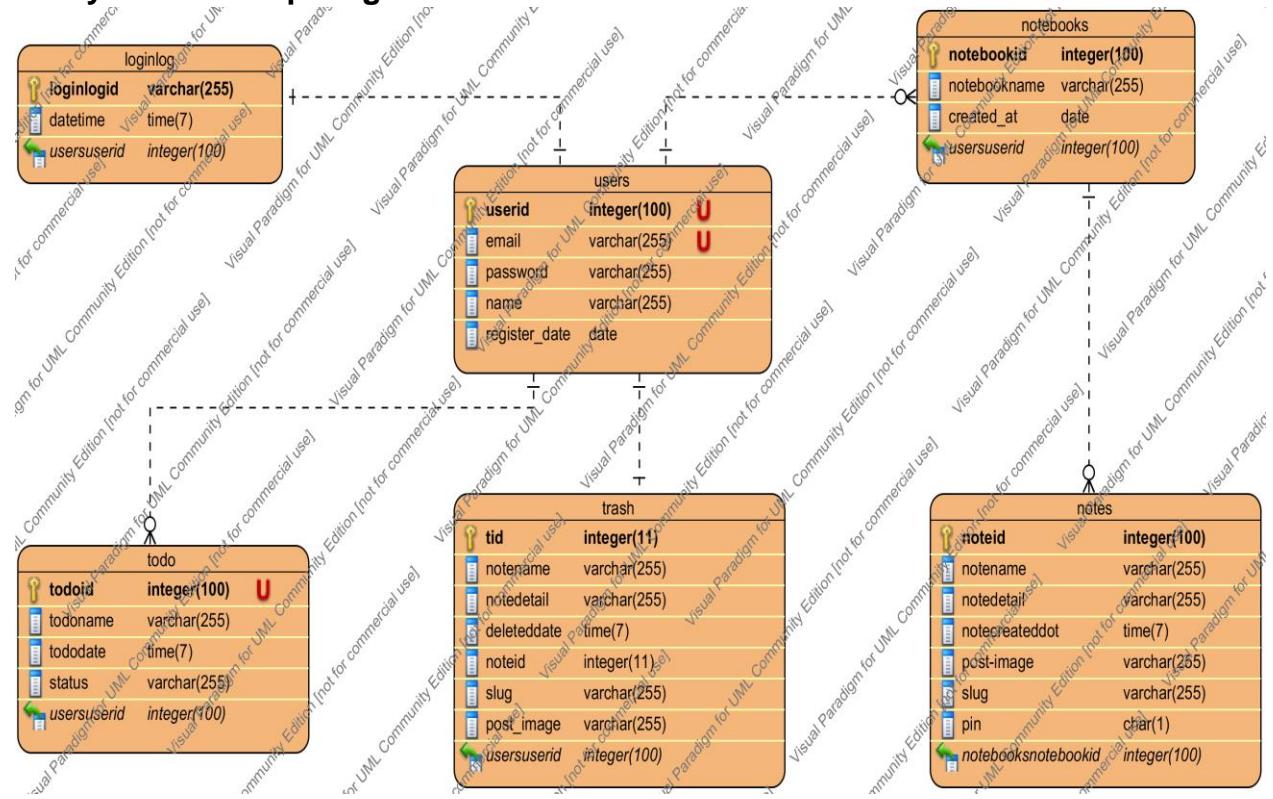
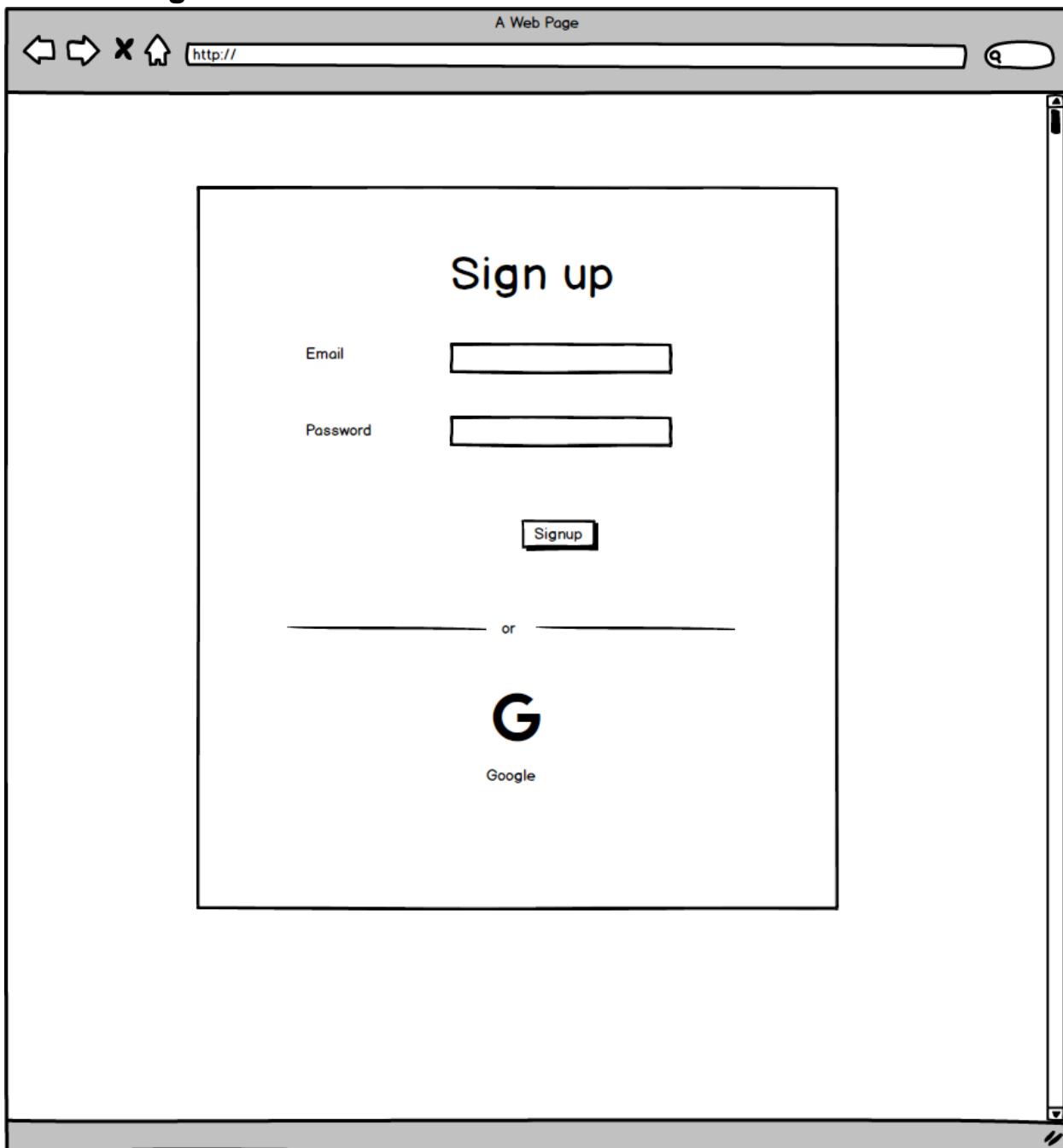
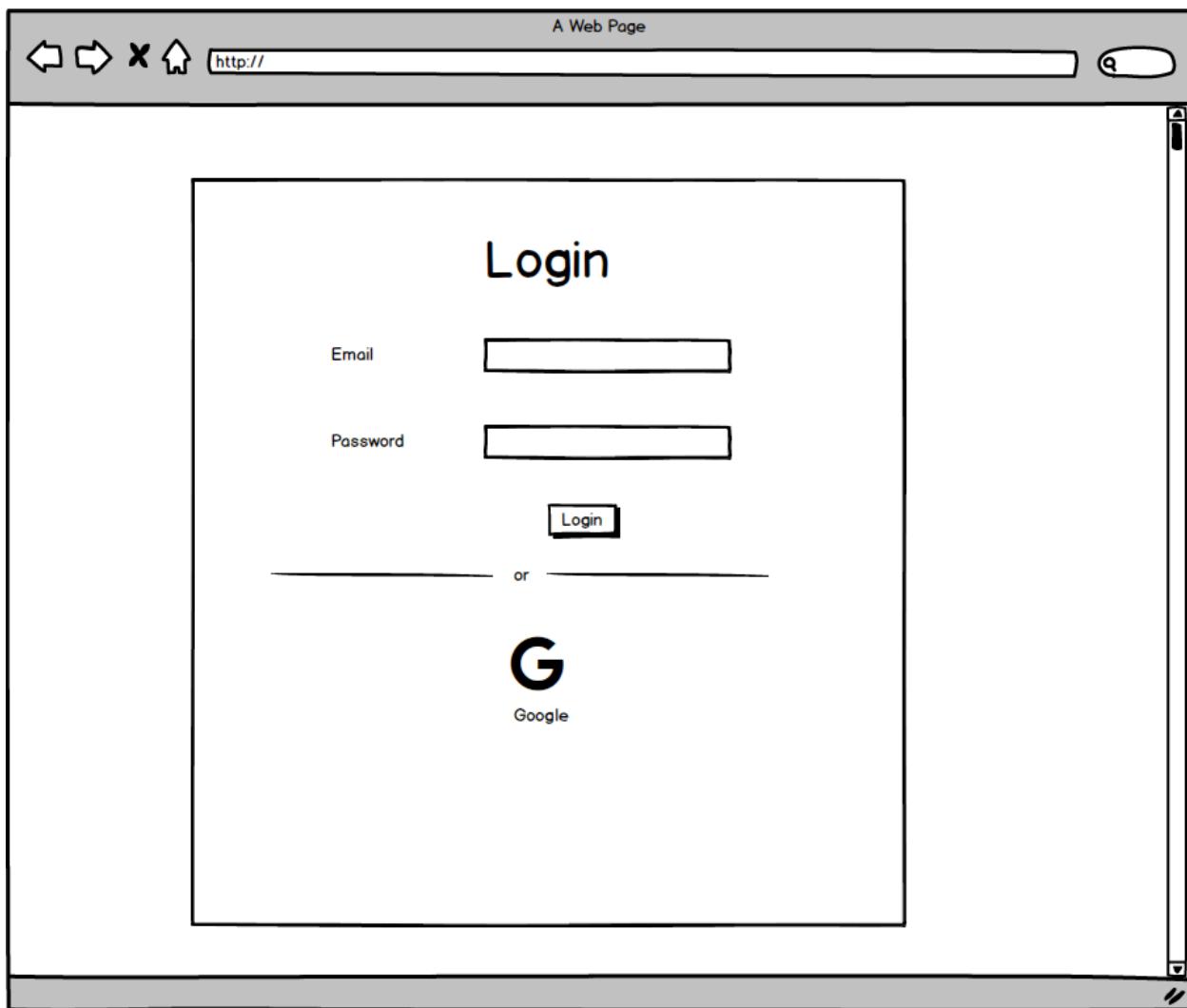
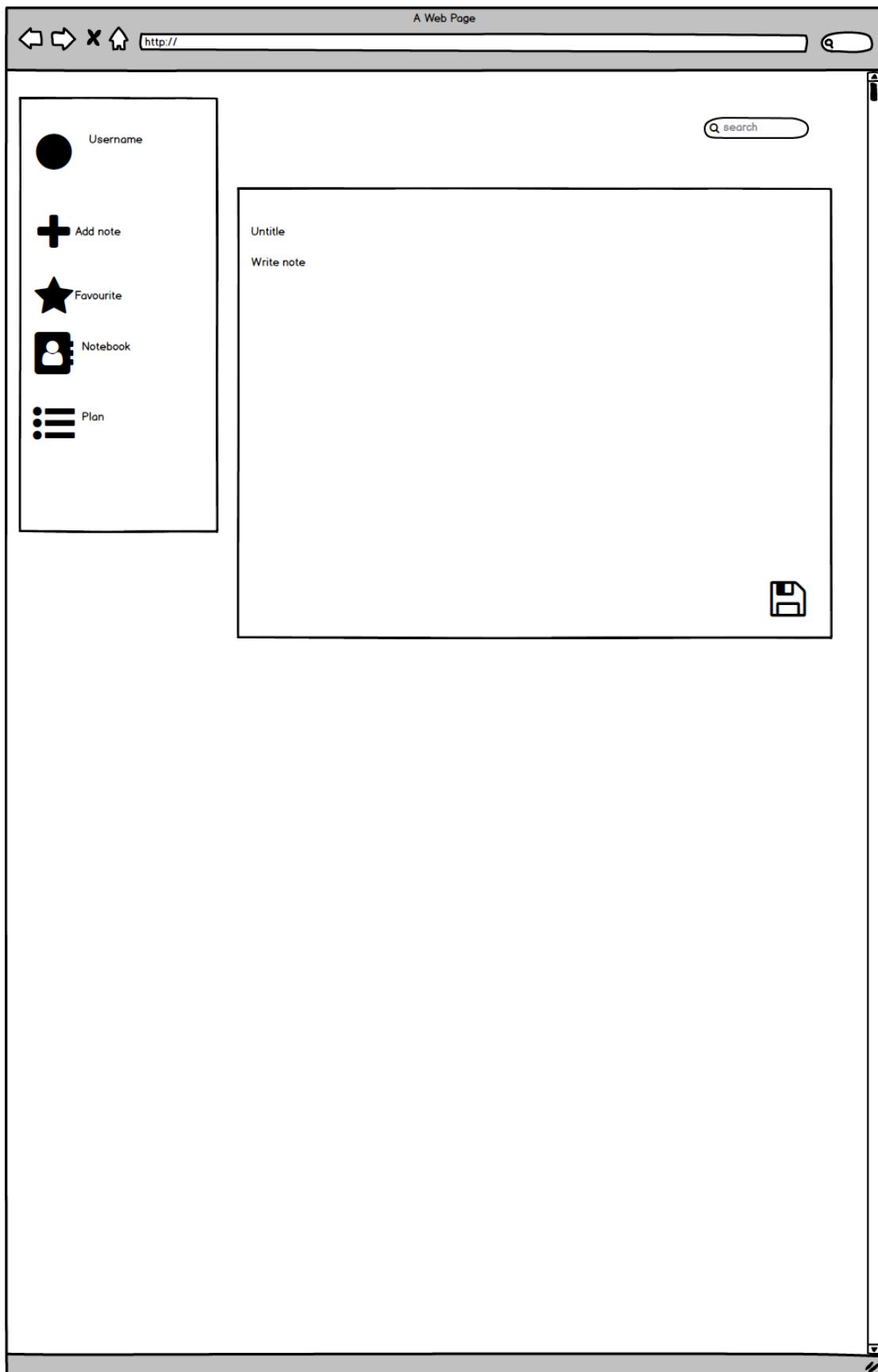


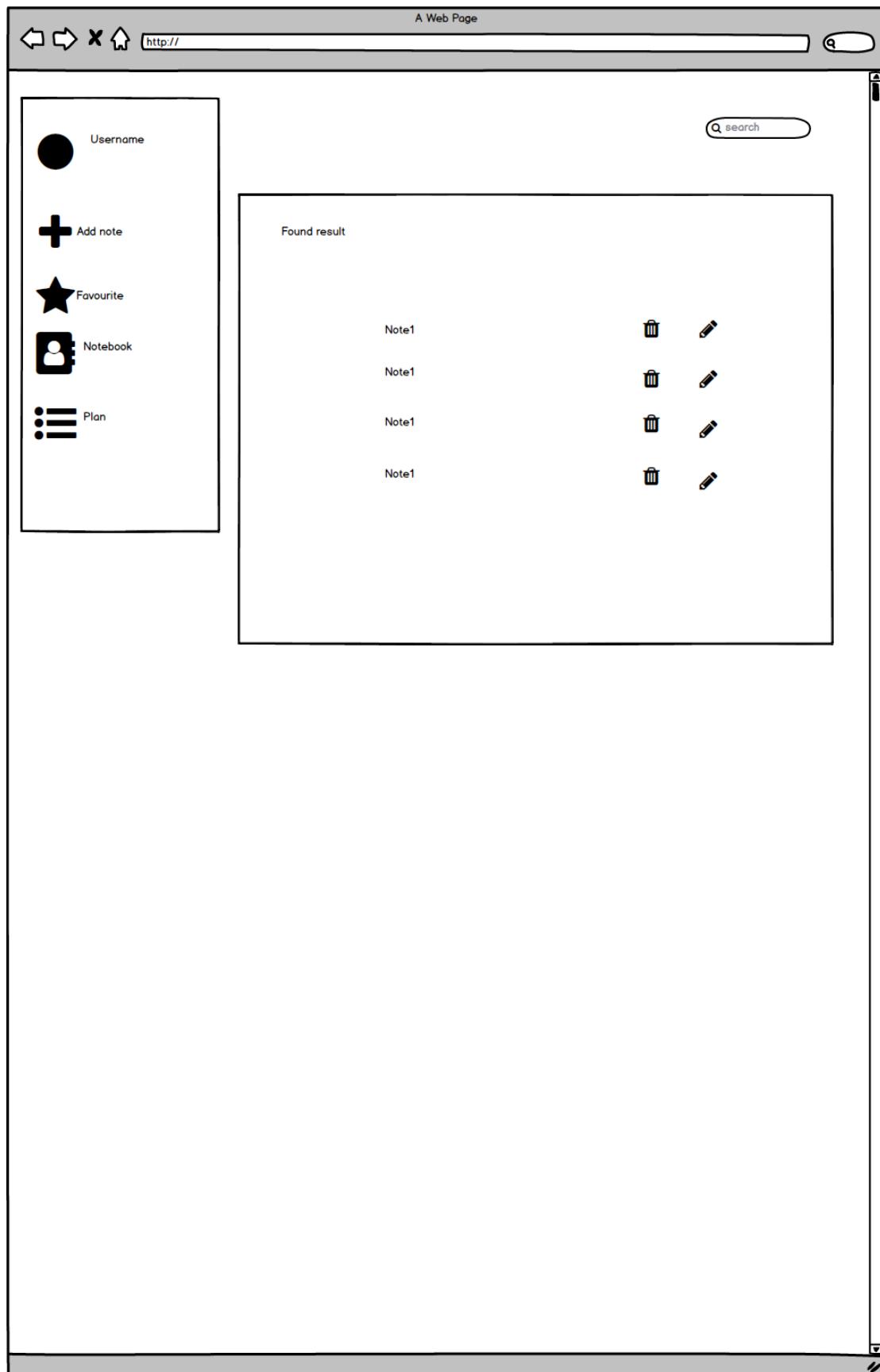
Figure 17 ER diagram for the purposed application

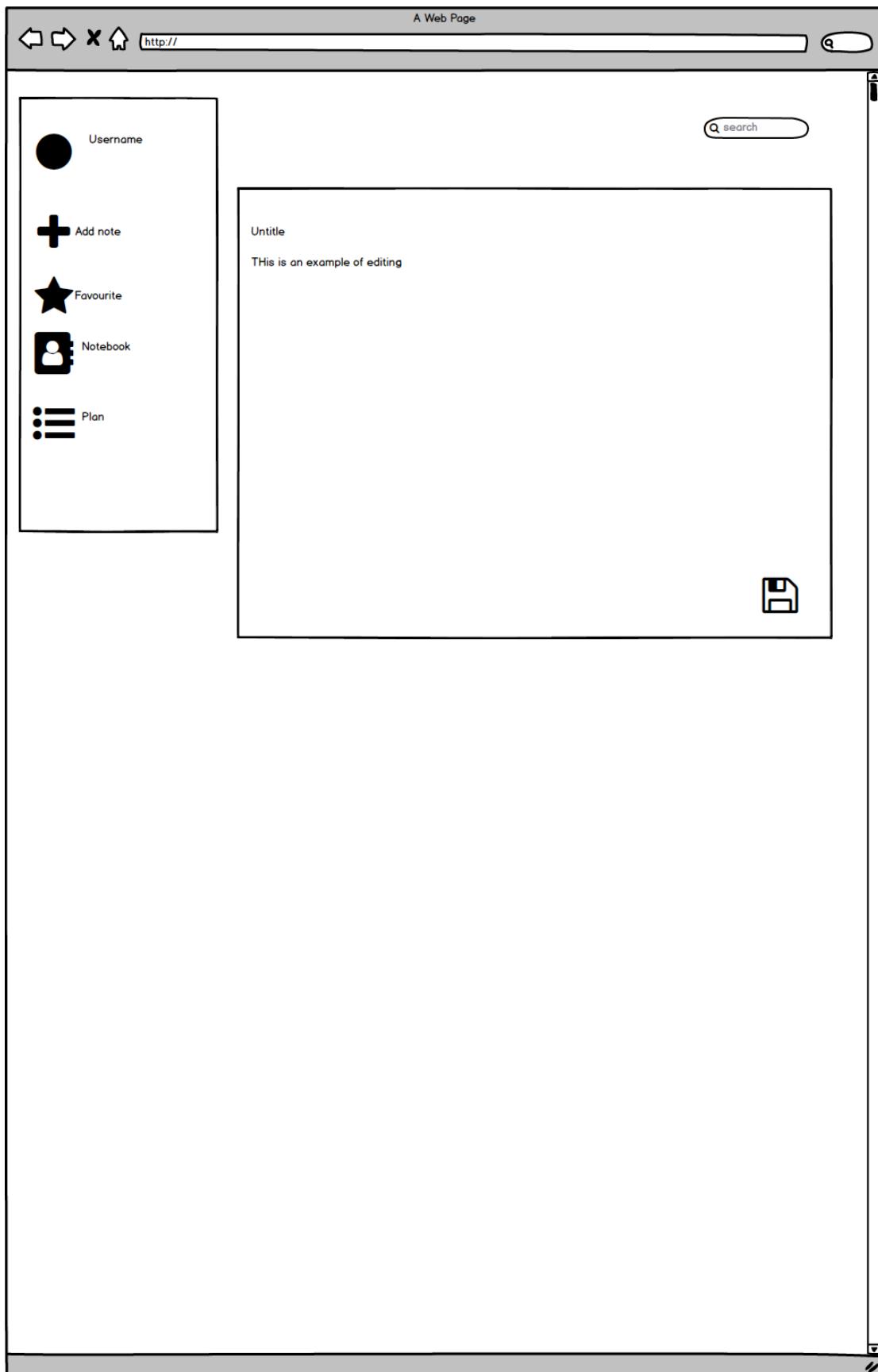
UI modeling









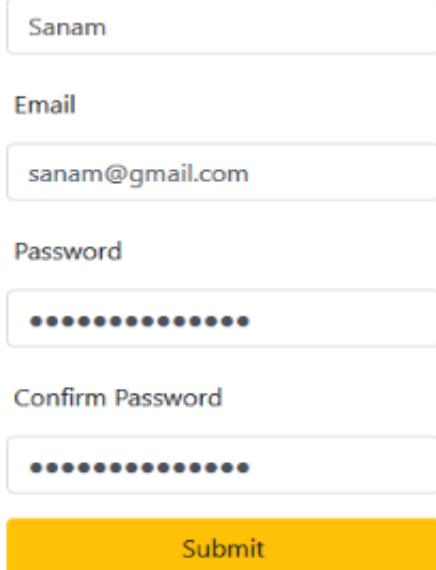
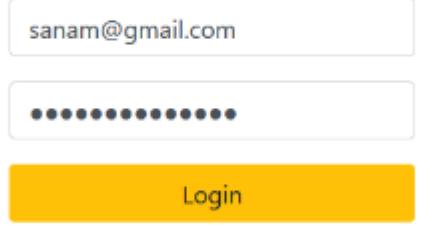


Chapter 5 Implementation

[Code's picture](#)

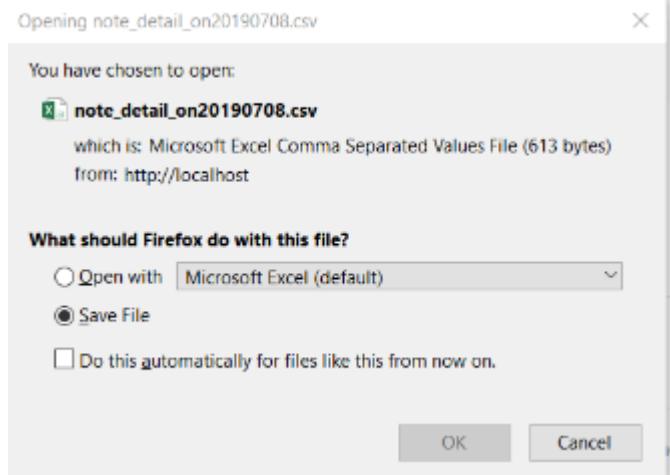
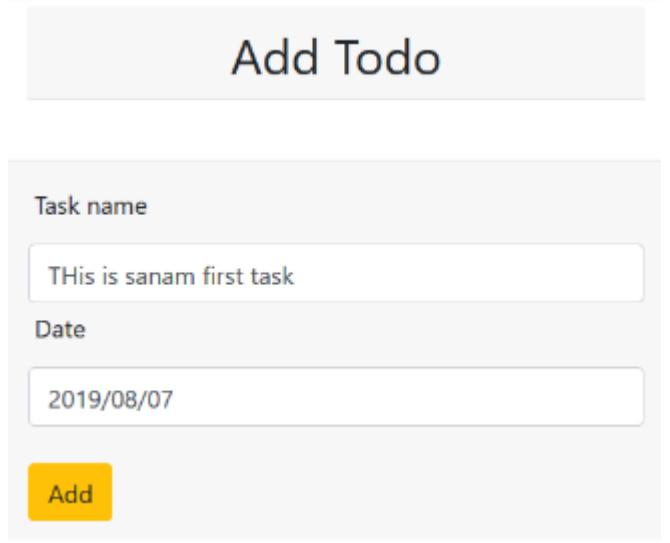
Chapter-6 Testing

Black box testing of my system are done as following

Functionalit y	View of testing as screenshot	Test result
Registration	<p style="text-align: center;">Sign Up</p> <p>Nickname</p>  <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="text" value="Sanam"/> </div> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="text" value="sanam@gmail.com"/> </div> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="password" value="*****"/> </div> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="password" value="*****"/> </div> <div style="background-color: #ffcc00; color: black; padding: 5px; border-radius: 5px; text-align: center;">Submit</div>	Registration test was successful
Login	<p style="text-align: center;">Sign In</p>  <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="text" value="sanam@gmail.com"/> </div> <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;"> <input type="password" value="*****"/> </div> <div style="background-color: #ffcc00; color: black; padding: 5px; border-radius: 5px; text-align: center;">Login</div>	Login was successful

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

Display added note	<h2>Latest notes</h2> <p>My first note</p> <p>Saved on: 2019-07-08 17:13:32</p>  <p>This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first...</p> <p>Read more</p>	All tasks are complete. Add new task now!	Note was displayed successful
--------------------	---	---	-------------------------------

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

Display tasks	<p>Your's today's perform list</p> <p>Uncomplete task(s) from past 1 day:</p> <p>Add task</p> <p>Task completed</p> <p>THis is sanam first task [Set task as completed]</p> <p>Uncomplete task(s) from past 7 days:</p> <p>THis is sanam first task [Set task as completed]</p>	Tasks was displayed successfully
Edit profile	<h1>Edit Profile</h1> <p>Nickname</p> <p>Sanam_shrestha</p> <p>Email</p> <p>sanam@gmail.com</p> <p>Update</p>	Profile edited successfully

Change password	<h1>Change password</h1> <p>Old password</p> <input type="password"/> <p>New password</p> <input type="password"/> <p>Confirm password</p> <input type="password"/> <p>Update password</p>	Password updated successfully
-----------------	---	-------------------------------

[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
- Treat 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.

```
C:.
  application
    cache
    config
    controllers
    core
    helpers
    hooks
    language
      english
    libraries
    logs
    models
    third_party
    views
      errors
        cli
        html
      includes
      notebooks
      notes
      pages
      search
      testing
      todo
      users
  assets
    css
    images
      notes
    pdf
  CP
    Analysis
    Backup
      backup_2019-07-12-20-33
        application
          cache
          config
          controllers
          core
```

Figure 18 configuration part1

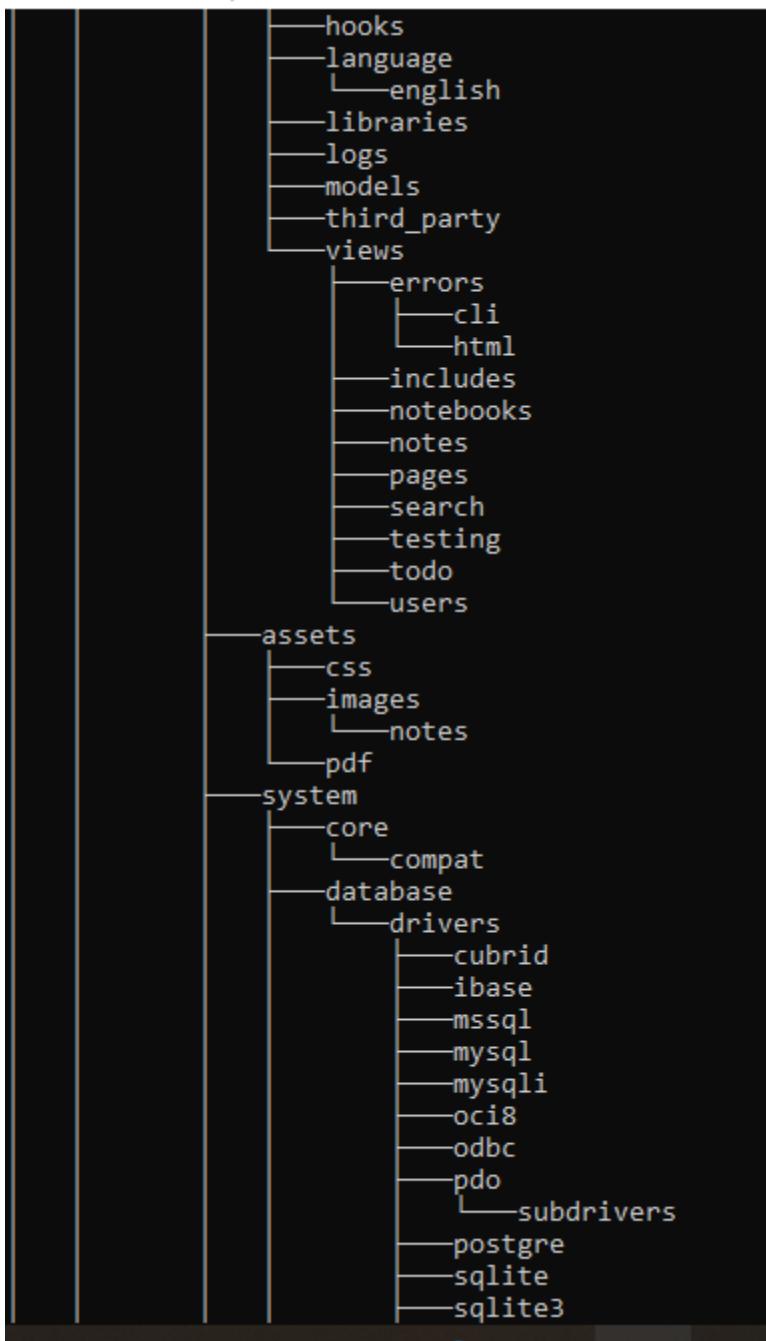


Figure 19 configuration part2

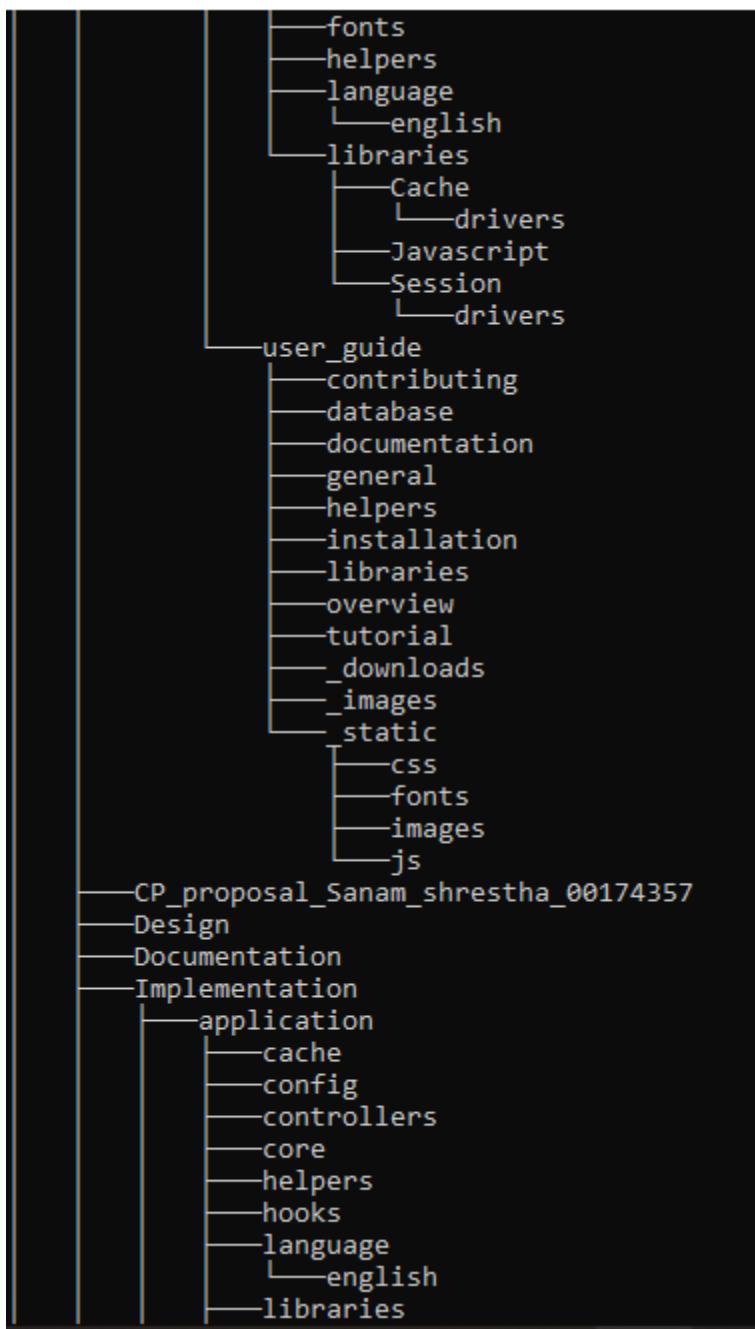


Figure 20 configuration part3

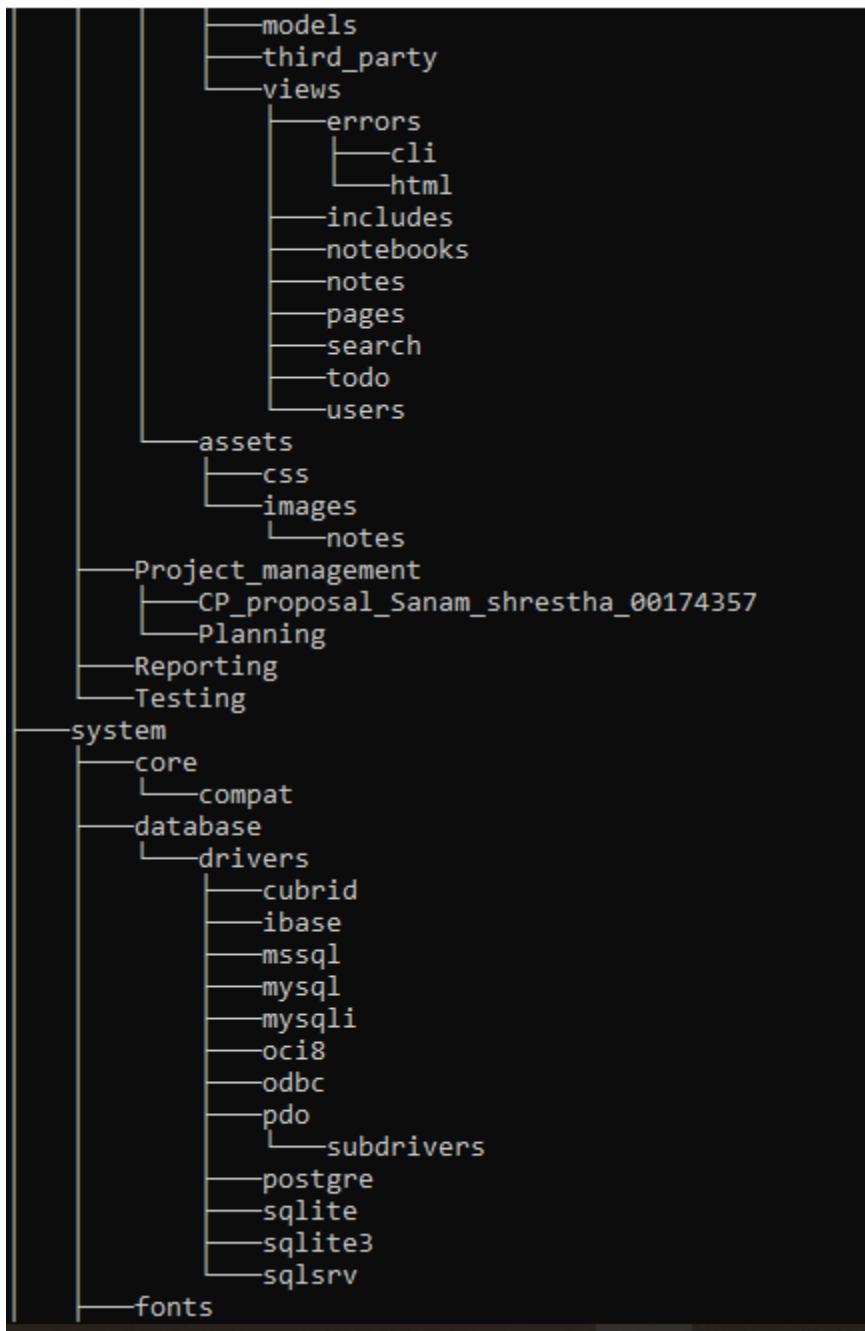


Figure 21 configuration part4

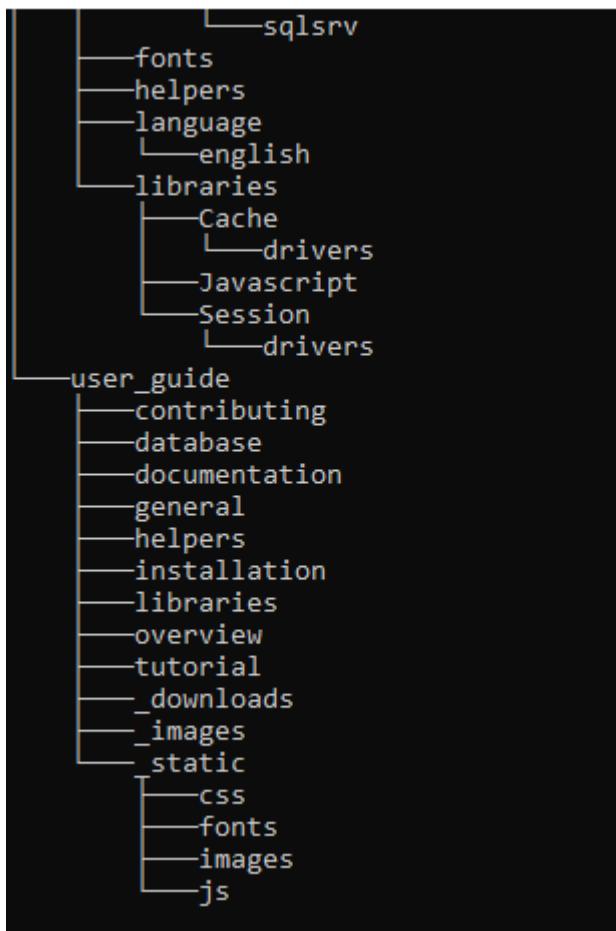


Figure 22 configuration part5

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**.

The image shows a registration form titled "Sign Up". It includes fields for Nickname (containing "Sanam"), Email (containing "sanam@gmail.com"), Password (represented by a series of dots), and Confirm Password (also represented by a series of dots). A large yellow "Submit" button is at the bottom.

Nickname	Sanam
Email	sanam@gmail.com
Password	*****
Confirm Password	*****
Submit	

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.

Sign In

sanam@gmail.com

Login

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.

N&T Tasks Notebooks Notes View All Notes Add Notebook Add Note Profile Logout Help

Login sucessfull.

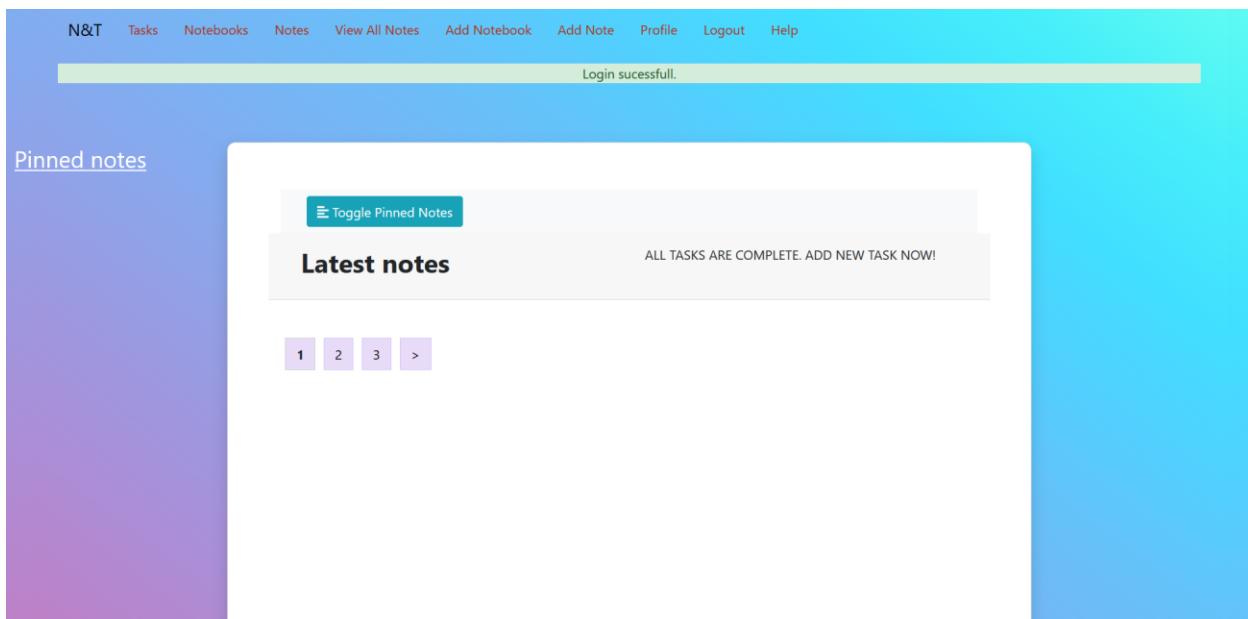
Pinned notes

Toggle Pinned Notes

Latest notes

ALL TASKS ARE COMPLETE. ADD NEW TASK NOW!

1 2 3 >



Adding notebook

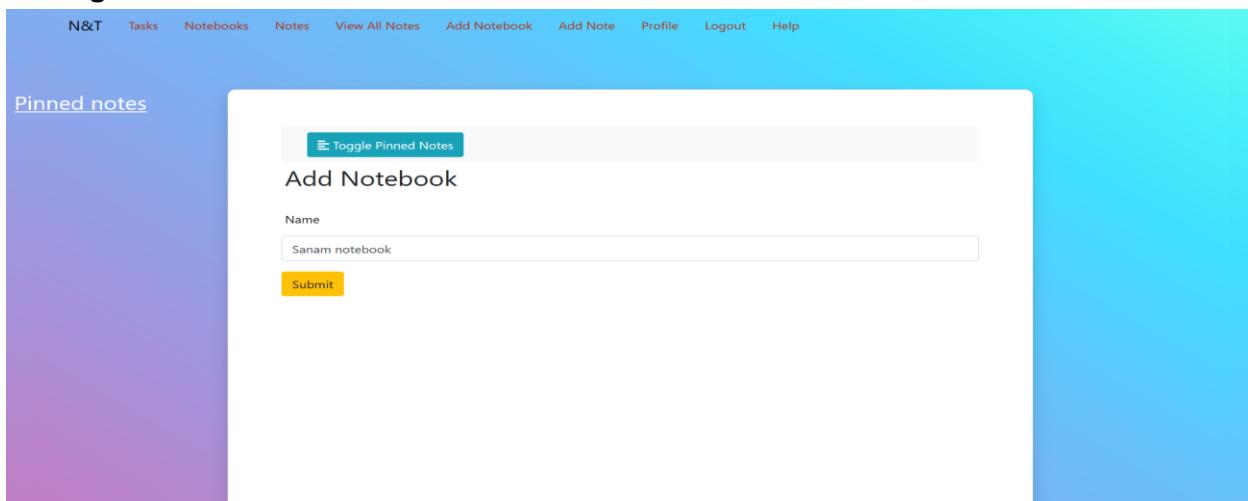


Figure 25 Adding notebook by user

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

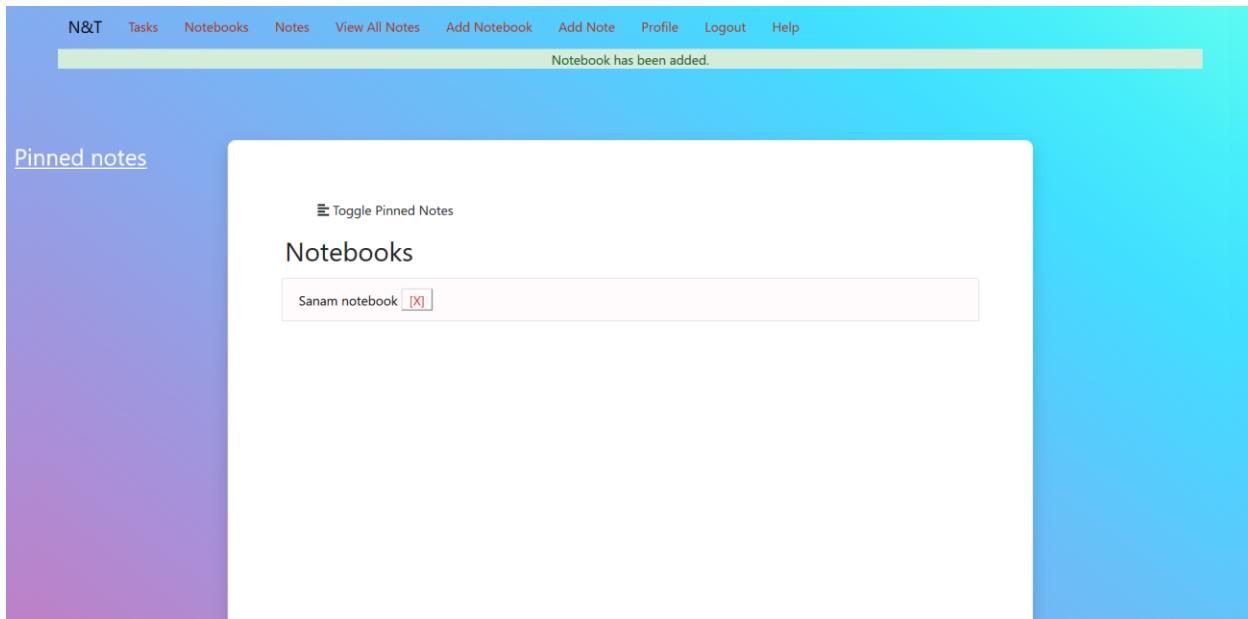


Figure 26 Deleting notebook by user

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

The screenshot shows a web application interface for adding a note. At the top, there is a navigation bar with links: N&T, Tasks, Notebooks, Notes, View All Notes, Add Notebook, Add Note, Profile, Logout, and Help. Below the navigation bar, the title "Pinned notes" is displayed. A button labeled "Toggle Pinned Notes" is visible. The main area is titled "Add note". It contains fields for "Note name" (with "My first note" entered) and "Pin note?" (with radio buttons for "Yes" and "No"). A rich text editor toolbar is present above a text area containing placeholder text. The text area has a character count of 42. Below the text area, there is a "Notebook" dropdown set to "Sanam notebook", an "Upload Image" button, and a "Browse..." link. A "Submit" button is located at the bottom of the form.

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

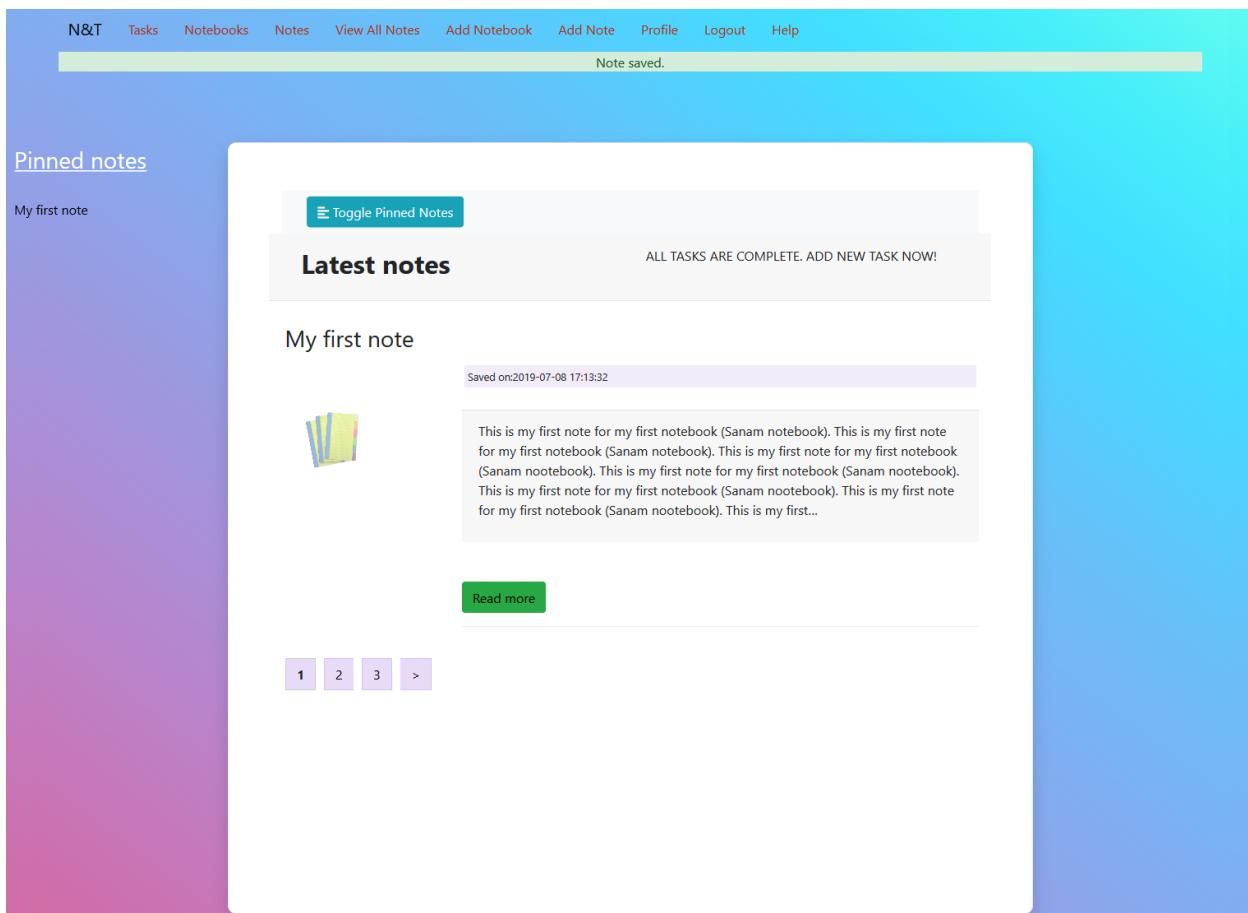


Figure 28 viewing note by user

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.

Editing a note

The screenshot shows the 'Edit Note' interface. At the top left is a sidebar titled 'Pinned notes' containing a single note titled 'My first note'. The main area has a title 'Edit Note'. A 'Note name' field contains 'My first note'. To its right is a 'Pin note?' section with a radio button for 'Yes' (unchecked) and one for 'No' (checked). Below these are sections for 'Note detail:' (containing a rich text editor toolbar), 'Notebook' (set to 'Sanam notebook'), and 'Upload Image' (with a 'Browse...' button and a message 'No file selected'). At the bottom right is a prominent yellow 'Update note' button.

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

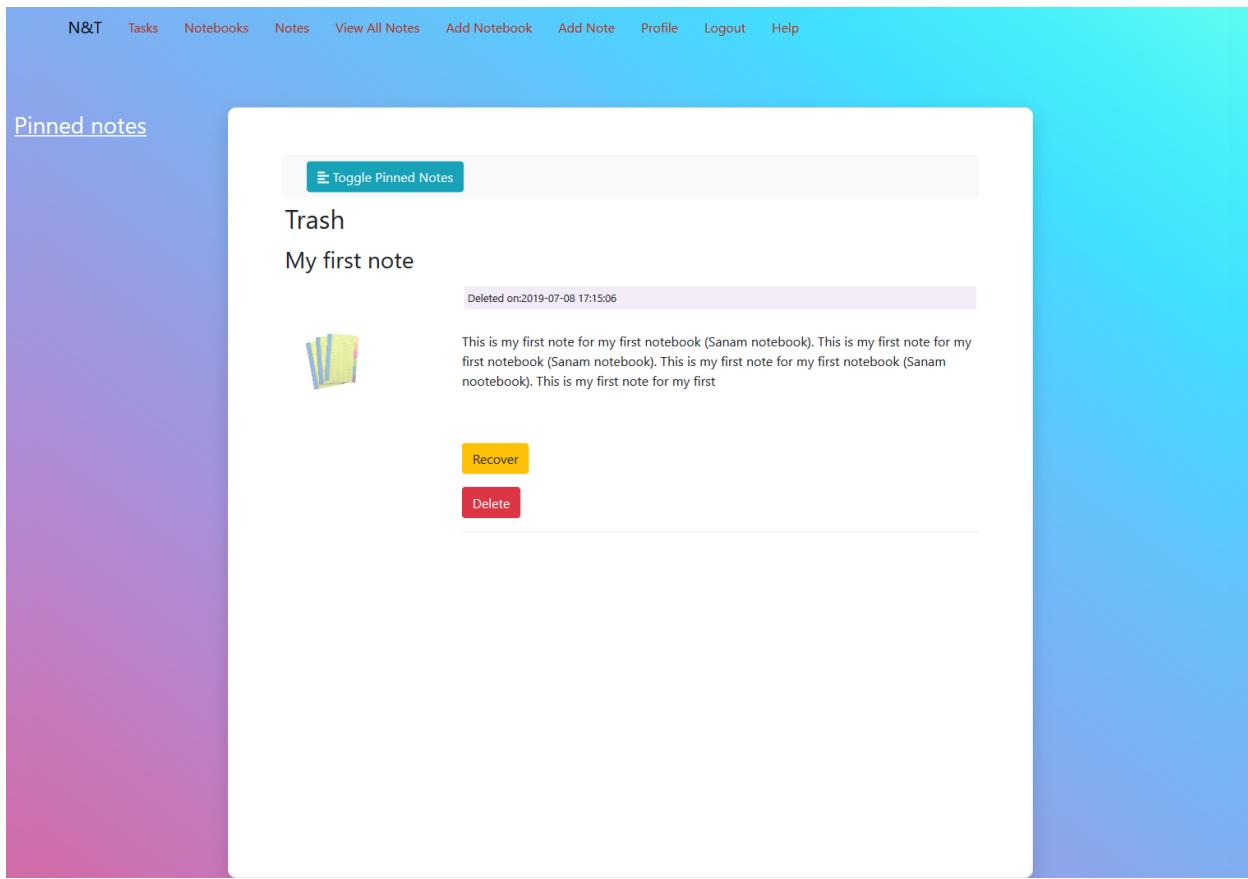


Figure 31 recovering data/ deleting data from trash

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

The screenshot shows a web application interface. At the top, there is a navigation bar with links: N&T, Tasks, Notebooks, Notes, View All Notes, Add Notebook, Add Note, Profile, Logout, and Help. Below the navigation bar, there is a sidebar on the left labeled "Pinned notes". The main content area has a title "Export data of notes". Under this title, there is a table titled "Notes data" with two columns: "Note name" and "Description". A single row is shown in the table:

Note name	Description
My first note	This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first note for my first notebook (Sanam notebook). This is my first note for my first

A green "Export" button is located at the top right of the table.

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

The screenshot shows a web application interface. At the top, there is a navigation bar with links: N&T, Tasks, Notebooks, Notes, View All Notes, Add Notebook, Add Note, Profile, Logout, and Help. Below the navigation bar, there is a sidebar on the left. The main content area has a title "Add Todo". There are two input fields: "Task name" containing "This is sanam first task" and "Date" containing "2019/08/07". At the bottom, there is a yellow "Add" button.

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

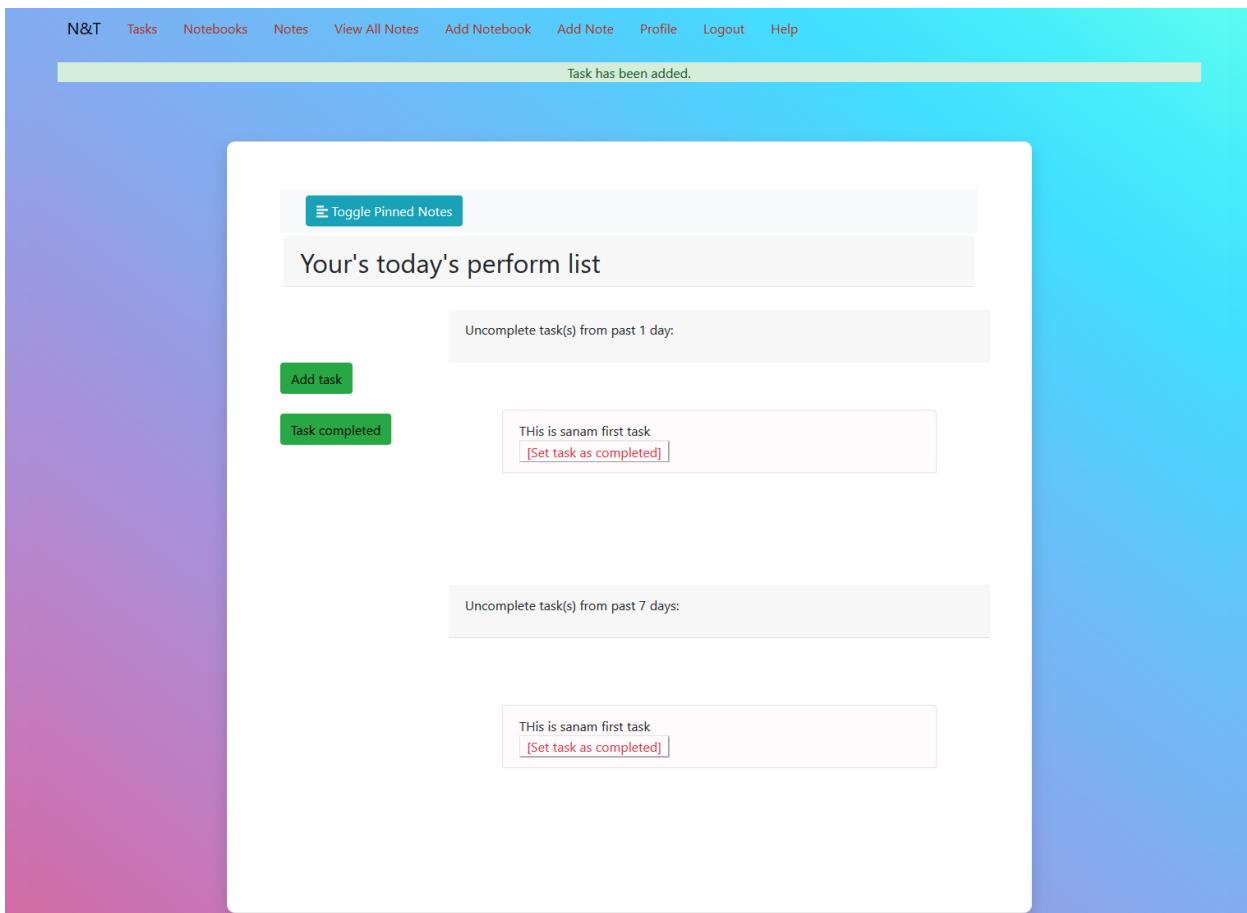


Figure 34 Viewing task by user

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 corrected.

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

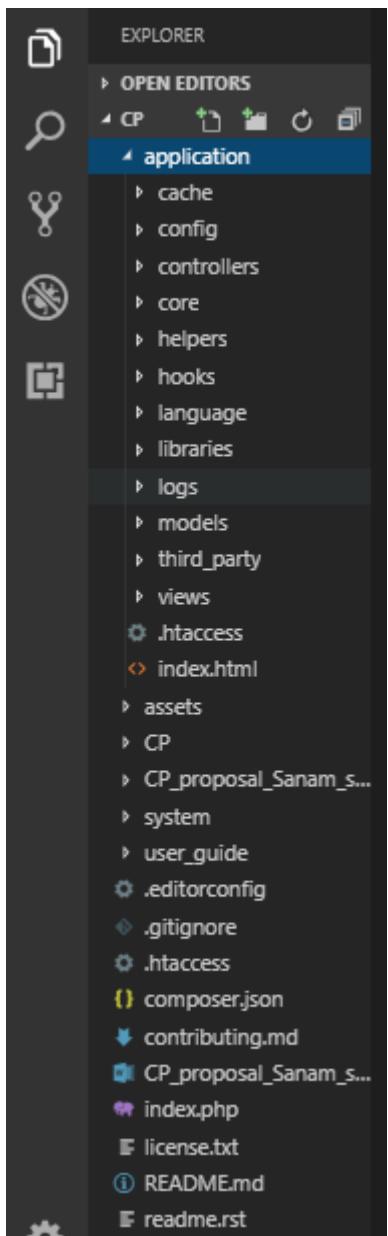


Figure 35 all folders and files for the system

```
header.php
application > views > includes > header.php
1  <!--defining this is html5-->
2  <!DOCTYPE html>
3  <html lang="en">
4
5  <!--start of head section-->
6  <head>
7      <meta charset="utf-8" />
8      <link rel="apple-touch-icon" sizes="76x76" href="../assets/img/apple-icon.png">
9      <link rel="icon" type="image/png" href="../assets/img/favicon.png">
10     <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
11     <meta content='width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0' name='viewport' />
12
13     <!--start of title section-->
14     <!-- Title Page-->
15     <title>
16         | Welcome to Note & Todolist
17     </title>
18     <!--end of title section-->
19     <!-- Scrollbar Custom CSS -->
20     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/malihu-custom-scrollbar-plugin/3.1.5/jquery.mCustomScrollbar.min.css">
21     <!-- Fonts and icons -->
22     <link href="assets/css/material-design-iconic-font.min.css" rel="stylesheet" media="all">
23     <link href="assets/font-awesome.min.css" rel="stylesheet" media="all">
24
25     <link rel="stylesheet" href="https://bootswatch.com/flatly/bootstrap.min.css">
26     <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css"
27     integrity="sha384-fmqOCqbTlWI1j8LyTjo7mOUStjsKC4pOpQbqi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
28     <script src="http://cdn.ckeditor.com/4.11.4/standard/ckeditor.js"></script>
29
30     <!-- CSS Files -->
31     <link href="assets/css/bootstrap.min.css" rel="stylesheet" />
32     <link href="assets/select2/select2.min.css" rel="stylesheet" media="all">
33     <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet"
34     integrity="sha384-gg0yR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
35     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
36     integrity="sha384-Jj5mVgyd0p3pxB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
37     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.bundle.min.js"
```

Figure 36 code of header part1

```
header.php x
application > views > includes > header.php
38 <script type="text/javascript" integrity="sha384-xrRyWqdh3PHs8keKZN+8zzc5TX0GRTLCmivcbNjWm2rs5C8PRhcEn3czEjhAO9o" 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="sha384-9gVQ4dYFvwW5jIDZnLElnxCjeSWFghJivGPXr1jddIhOegiu1Fw05qRGvFX0dJZ4" 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 <!-- F0nt Awsome JS-->
52 <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/solid.js"
53 integrity="sha384-tzzSw1/Vo+0NSUhStP3bvWlPq+uvzCMfrN1fEFe+xBmv1C/AtVX5K0uZtmcHitFZ" crossorigin="anonymous"></script>
54 <script defer src="https://use.fontawesome.com/releases/v5.0.13/js/fontawesome.js"
55 integrity="sha384-60Irr52G08Np0FSZdxz1xdnSnd1D4vdcf/q2myIU00VsqaGHJsB0RaBE01VTOY" 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(!$_SESSION->userdata('logged_in')) : ?>
```

Figure 37 code of header part2

```
header.php
application > views > includes > header.php
1   <a class="nav-link" href="=php echo base_url(); ?users/login">Login<span class="sr-only">(current)</span></a>
2   <a class="nav-link" href="=php echo base_url(); ?users/register">Register <span class="sr-only">(current)</span></a>
3   <?php endif; ?>
4   <!-- end of if condition-->
5
6   <!-- checks in session data for displaying these part-->
7   <!-- start of if condition-->
8   <?php if($this->session->userdata('logged_in')) : ?>
9     <a class="nav-link" href="=php echo base_url(); ?todo">Tasks<span class="sr-only">(current)</span></a>
10    <a class="nav-link" href="=php echo base_url(); ?notebooks">Notebooks<span class="sr-only">(current)</span></a>
11    <a class="nav-link" href="=php echo base_url(); ?notes">Notes<span class="sr-only">(current)</span></a>
12    <a class="nav-link" href="=php echo base_url(); ?notes/_search">View All Notes<span class="sr-only">(current)</span></a>
13    <a class="nav-link" href="=php echo base_url(); ?notebooks/add">Add Notebook<span class="sr-only">(current)</span></a>
14    <a class="nav-link" href="=php echo base_url(); ?notes/add">Add Note<span class="sr-only">(current)</span></a>
15    <a class="nav-link" href="=php echo base_url(); ?profile">Profile<span class="sr-only">(current)</span></a>
16    <a class="nav-link" href="=php echo base_url(); ?users/logout">Logout<span class="sr-only">(current)</span></a>
17
18  <?php endif; ?>
19  <!-- end of if condition-->
20  <a class="nav-link" href="=php echo base_url(); ?pages/help">Help<span class="sr-only">(current)</span></a>
21
22 </ul>
23 </div>
24 </nav>
25 <!--end of nav part-->
26 <div class = "text text-center">
27   <!-- flash message-->
28
29   <?php if($this->session->flashdata('user_registered')): ?>
30   | <?php echo '<p class="alert-success">' . $this->session->flashdata('user_registered') . '</p>'; ?>
31   <?php endif; ?>
32
33   <?php if($this->session->flashdata('note_added')): ?>
34   | <?php echo '<p class="alert-success">' . $this->session->flashdata('note_added') . '</p>'; ?>
35   <?php endif; ?>
36
37   <?php if($this->session->flashdata('note_updated')): ?>
38   | <?php echo '<p class="alert-success">' . $this->session->flashdata('note_updated') . '</p>'; ?>
```

Figure 38 code of header part3

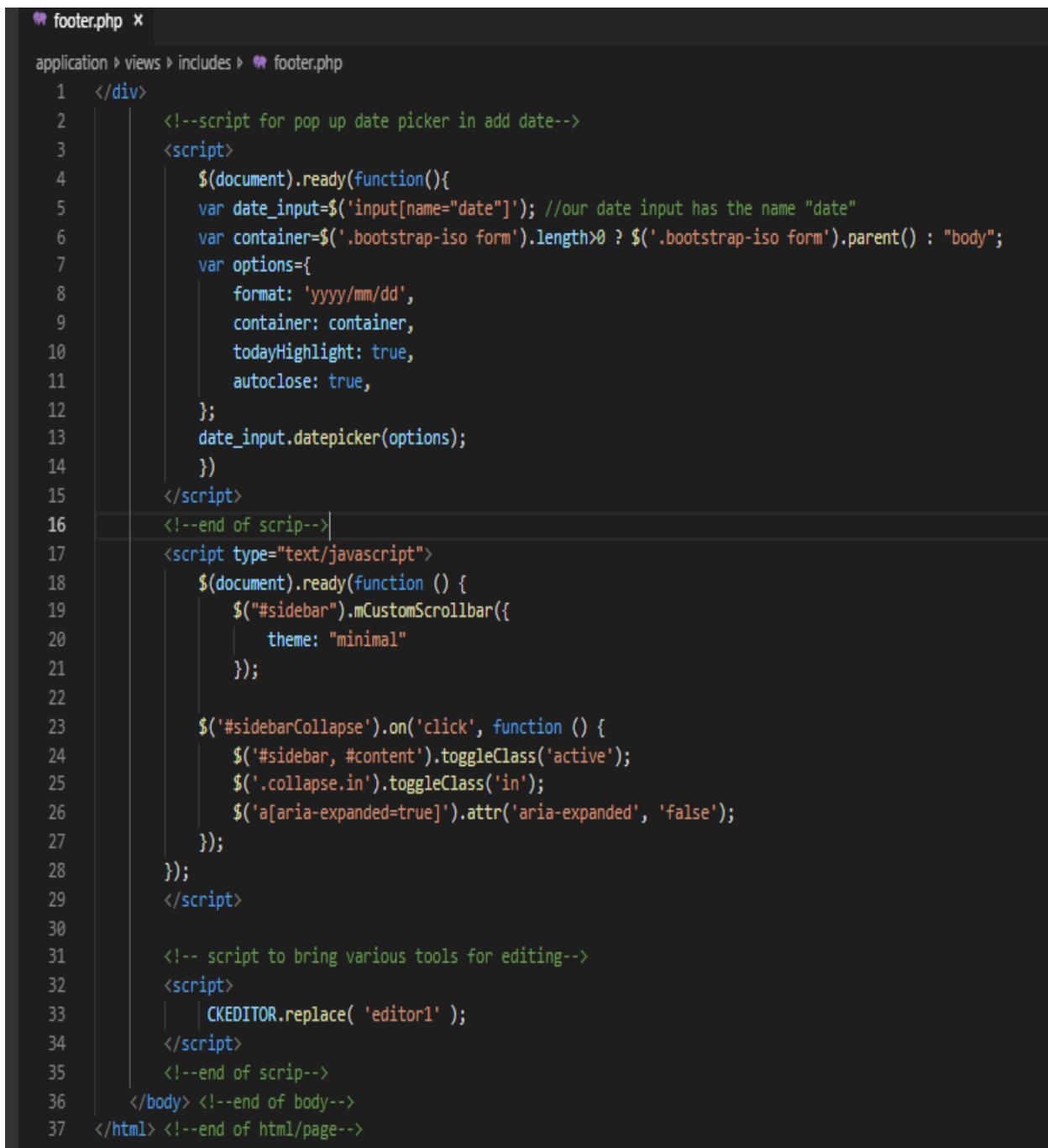
```
header.php
application > views > includes > header.php
111     <p class="alert-success">'. $this->session->flashdata('note_updated'). '</p> ?>
112     <?php endif; ?>
113
114     <?php if($this->session->flashdata('notebook_added')): ?>
115         <?php echo '<p class="alert-success">'. $this->session->flashdata('notebook_added'). '</p>'; ?>
116     <?php endif; ?>
117
118     <?php if($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 X
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



The screenshot shows a code editor window with the file 'footer.php' open. The code is written in HTML and includes several JavaScript snippets for date pickers, sidebar scrolling, and CKEDITOR initialization.

```
application > views > includes > footer.php
1  </div>
2  |    <!--script for pop up date picker in add date-->
3  |    <script>
4  |        $(document).ready(function(){
5  |            var date_input($('input[name="date"]'); //our date input has the name "date"
6  |            var container=$('.bootstrap-iso form').length>0 ? $('.bootstrap-iso form').parent() : "body";
7  |            var options={
8  |                format: 'yyyy/mm/dd',
9  |                container: container,
10 |                todayHighlight: true,
11 |                autoclose: true,
12 |            };
13 |            date_input.datepicker(options);
14 |        })
15 |    </script>
16 |    <!--end of script-->
17 |    <script type="text/javascript">
18 |        $(document).ready(function () {
19 |            $("#sidebar").mCustomScrollbar({
20 |                theme: "minimal"
21 |            });
22 |
23 |            $('#sidebarCollapse').on('click', function () {
24 |                $('#sidebar, #content').toggleClass('active');
25 |                $('.collapse.in').toggleClass('in');
26 |                $('a[aria-expanded=true]').attr('aria-expanded', 'false');
27 |            });
28 |        });
29 |    </script>
30 |
31 |    <!-- script to bring various tools for editing-->
32 |    <script>
33 |        CKEDITOR.replace( 'editor1' );
34 |    </script>
35 |    <!--end of script-->
36 |    </body> <!--end of body-->
37 |</html> <!--end of html/page-->
```

Figure 41 code of footer

```
add.php  x

application > views > notebooks > add.php
1  <!--this is add page of notebook where user can add new notebook -->
2  <!--this part presents cart section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">
6              <div class="container-fluid">
7
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">
9                      <i class="fas fa-align-left"></i>
10                     <span>Toggle Pinned Notes</span>
11                 </button>
12
13             </div>
14         </nav>
15         <!--This part shows title of page-->
16         <h2 class="text"><?= $title ;?></h2>
17         <!-- any errors encountered will be shown in validation_errors part-->
18         <div class="text text-danger text-center"><strong><?php echo validation_errors(); ?>
19             </strong></div>
20         <!--form_open_multipart is like <form> which supports image also -->
21         <?php echo form_open_multipart('notebooks/add'); ?>
22             <div class="form-group">
23                 <label class="text">Name</label>
24                 <input type="text" class="form-control" name="name" placeholder="Enter name">
25             </div>
26             <button type="submit" class="btn btn-warning">Submit</button>
27
28         </form>
29         <!--close form part-->
30     </div>
31 </div>
32 <!--end of card section-->
33 <div class="background">
34
35 </div>
36 <!-- end of add page odnotebook-->
```

Figure 42 code of notebook part1

```
index.php x

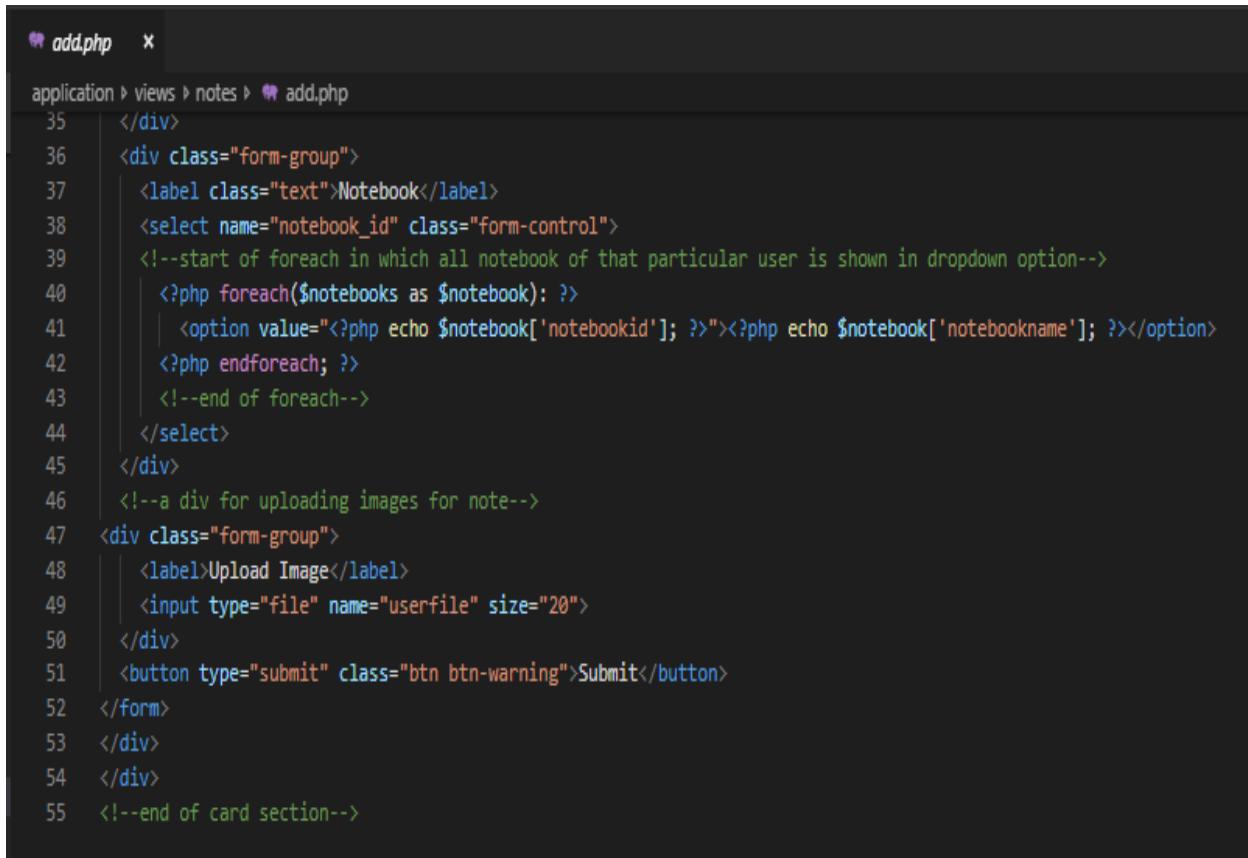
application > views > notebooks > index.php

1  <!-- this is index page of notebook where created notebooks are displayed-->
2  <!--this part presents card section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <nav class="navbar navbar-expand-lg">
6              <div class="container-fluid">
7
8                  <button type="button" id="sidebarCollapse" class="btn">
9                      <i class="fas fa-align-left"></i>
10                     <span>Toggle Pinned Notes</span>
11                 </button>
12
13             </div>
14         </nav>
15         <!--This part shows title of page-->
16         <h2 class="text"><?= $title; ?></h2>
17         <div class="group">
18             <?php foreach($notebooks as $notebook): ?>
19                 <div class="list-group-item notebook"><a href="<?php
20                     echo site_url('/notebooks/notes/'.$notebook['notebookid']); ?>"><?php
21                     echo $notebook['notebookname']; ?></a>
22                     <?php if($this->session->userdata('user_id') == $notebook['user_id']) :?>
23                     <form class="notebook-delete" action = "notebooks/delete/<?php
24                         echo $notebook["notebookid"]; ?>" method = "POST">
25                         <input type="submit" class="btn-link text-danger" value="[X]">
26
27                     </form>
28                     <?php endif; ?>
29                     </div><br />
30             <?php endforeach; ?>
31         </div>
32     </div>
33
34
35     <!--end of card section-->
36     <div class="background">
37         </div>
```

Figure 43 code of notebook part2

```
add.php *  
application > views > notes > add.php  
1  <!--this is add page of note where user can add notes-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4    <div class="card-body">  
5      <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6        <div class="container-fluid">  
7          <button type="button" id="sidebarCollapse" class="btn btn-info">  
8            <i class="fas fa-align-left"></i>  
9            <span>Toggle Pinned Notes</span>  
10           </button>  
11       </div>  
12     </div>  
13   </nav>  
14   <!--this part shown title of current page-->  
15   | <h2 class="text"><?= $title ?></h2>  
16   <!-- if any errors occurs, it is hown in this part-->  
17   <div class="text text-danger text-center"><strong><?php echo validation_errors(); ?>  
18     </strong></div>  
19   <!--opens form supporting images-->  
20   <?php echo form_open_multipart('notes/add'); ?>  
21   <div class="row">  
22     <div class="form-group col-md-9">  
23       <label class="text">Note name</label>  
24       <input type="text" class="form-control" name="notename" placeholder="Add Title">  
25     </div>  
26     <div class="form-group col-md-3 ">  
27       <label class="text">Pin note?</label><br />  
28       <input type="radio" name="pin" value="2" >Yes  
29       <input type="radio" name="pin" value="1" checked>No<br>  
30     </div></div>  
31     <div class="form-group">  
32       <label class="text">Note detail:</label>  
33       <textarea id="editor1" class="form-control" name="notedetail" placeholder="Add Body"></textarea>  
34     </div>  
35     <div class="form-group">  
36       <label class="text">Notebook</label>
```

Figure 44 code of add note part1



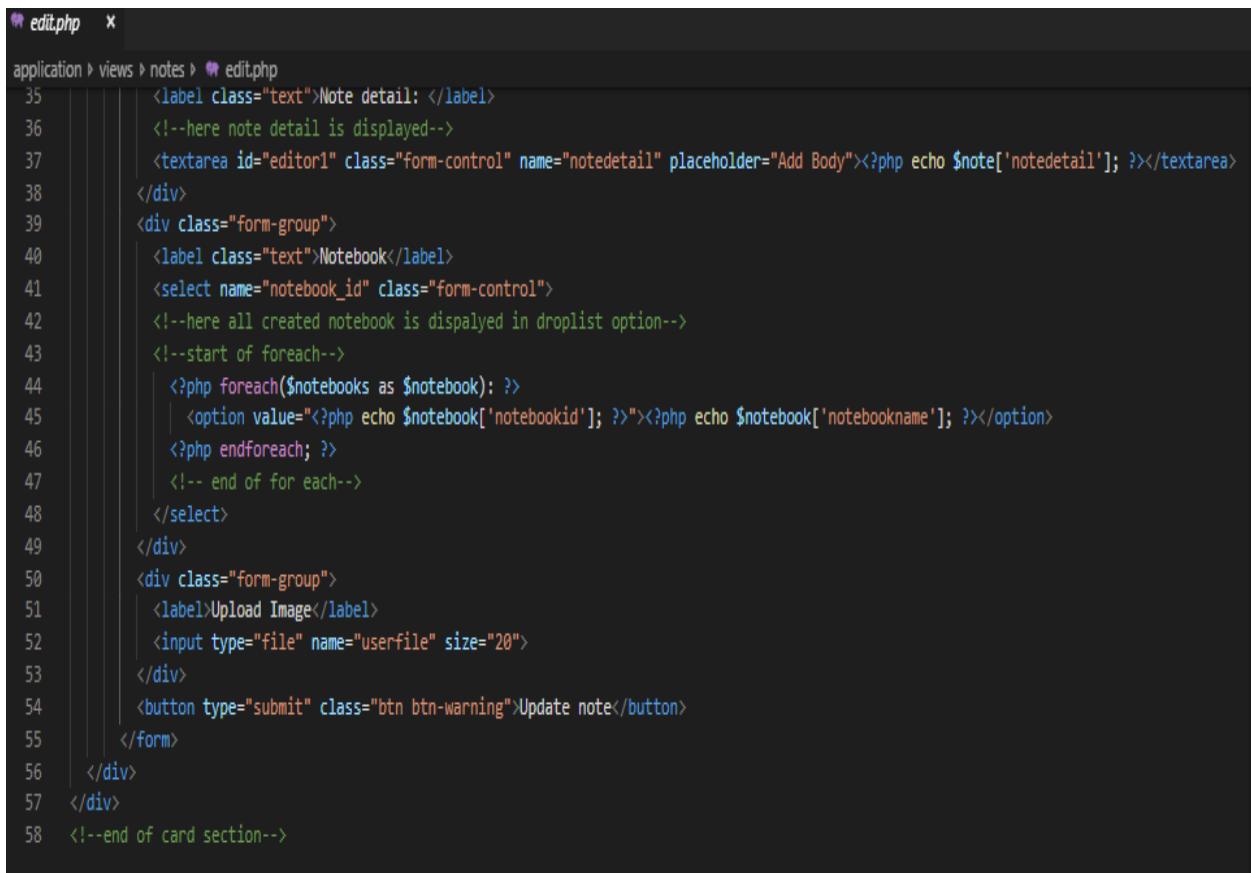
The screenshot shows a code editor window with the file name 'add.php' at the top. The code is a part of a PHP application for note addition. It includes HTML form elements like input fields, a select dropdown for choosing a notebook, and a file upload input. PHP code is used for generating the dropdown options based on an array of notebooks. The code is well-structured with comments explaining the logic.

```
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
1  <!--this is a page of edit note where user will edit their note-->
2  <!--start of card section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">
6              <div class="container-fluid">
7
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">
9                      <i class="fas fa-align-left"></i>
10                     <span>Toggle Pinned Notes</span>
11                 </button>
12
13             </div>
14         </nav>
15         <!--this shows title of this page-->
16         <h2 class="text"><?= $title ?></h2>
17         <!-- if any errors occurs, it is shown in this part-->
18         <?php echo validation_errors(); ?>
19
20         <!--opens form-->
21         <?php echo form_open('notes/update'); ?>
22         <input type="Hidden" name="id" value=<?php echo $note['noteid'];?>>
23         <div class="row">
24             <div class="form-group col-md-9">
25                 <label class="text">Note name</label>
26                 <!--here note name is displayed-->
27                 <input type="text" class="form-control" name="notename" placeholder="Add Title" value=<?php echo $note['notename']; ?>" >
28             </div>
29             <div class="form-group col-md-3 ">
30                 <label class="text">Pin note?</label><br />
31                 <input type="radio" name="pin" value="2" >Yes
32                 <input type="radio" name="pin" value="1" checked>No<br>
33             </div></div>
34             <div class="form-group">
35                 <label class="text">Note detail:</label>
36                 <!--here note detail is displayed-->
```

Figure 46 code of edit note part1

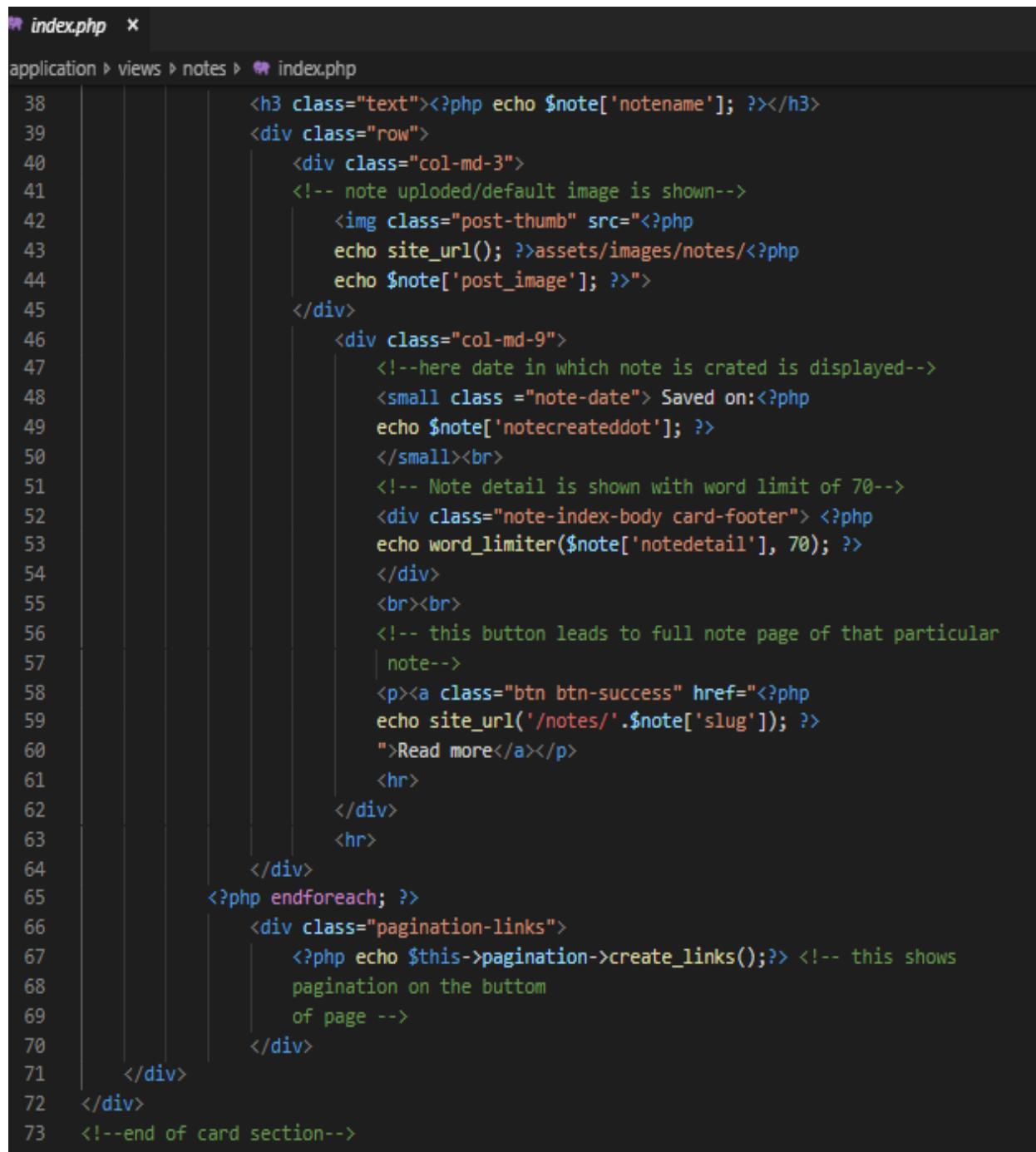


```
edit.php *  
application > views > notes > edit.php  
35     <label class="text">Note detail: </label>  
36     <!--here note detail is displayed-->  
37     <textarea id="editor1" class="form-control" name="notedetail" placeholder="Add Body"><?php echo $note['notedetail']; ?></textarea>  
38 </div>  
39 <div class="form-group">  
40     <label class="text">Notebook</label>  
41     <select name="notebook_id" class="form-control">  
42         <!--here all created notebook is dispalyed in dropdown option-->  
43         <!--start of foreach-->  
44         <?php foreach($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  
1  <!--this is a index page of note where added notes are shown-->  
2  <!--this part presents cart section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7  
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
9                      <i class="fas fa-align-left"></i>  
10                     <span>Toggle Pinned Notes</span>  
11                 </button>  
12  
13             </div>  
14         </nav>  
15         <!--this shows title of this page-->  
16         <div class="row card-header">  
17             <div class="col-md-6">  
18                 <h2 class="text"><strong><?= $title ?></strong></h2>  
19             </div>  
20             <!--div to show tasks notification-->  
21             <div class="col-md-6">  
22                 <?php if($notify > 0)>  
23                 {?>  
24                     <a href="<?php echo base_url(); ?>todo" > <h6  
25                         class=" notify text"><center><strong><?php  
26                         echo 'YOU GOT ' ;?><?php echo $notify; ?><?php  
27                         echo ' TASK(S) TO COMPLETE!' ; ?>  
28                     </a></strong></center></h6>  
29                 <?php }  
30                 else{?>  
31                     <h6 class=" notify text"><center><?php  
32                         echo 'ALL TASKS ARE COMPLETE. ADD NEW TASK NOW! ' ;?><?php  
33                     </strong></center></h6>  
34                 </div>  
35             </div><br />  
36             <?php foreach((array) $notes as $note) : ?>  
37                 <!-- here name of note is displayed-->
```

Figure 48 code of index note part1



The screenshot shows a code editor window with the file 'index.php' open. The file is located in the 'application/views/notes' directory. The code is a PHP template for displaying a list of notes. It uses Bootstrap's grid system with 'col-md-3' and 'col-md-9' classes. It includes logic to echo note names and images, and to display note details like creation date and a truncated note body. It also includes a button to read more about a note and a pagination section at the bottom.

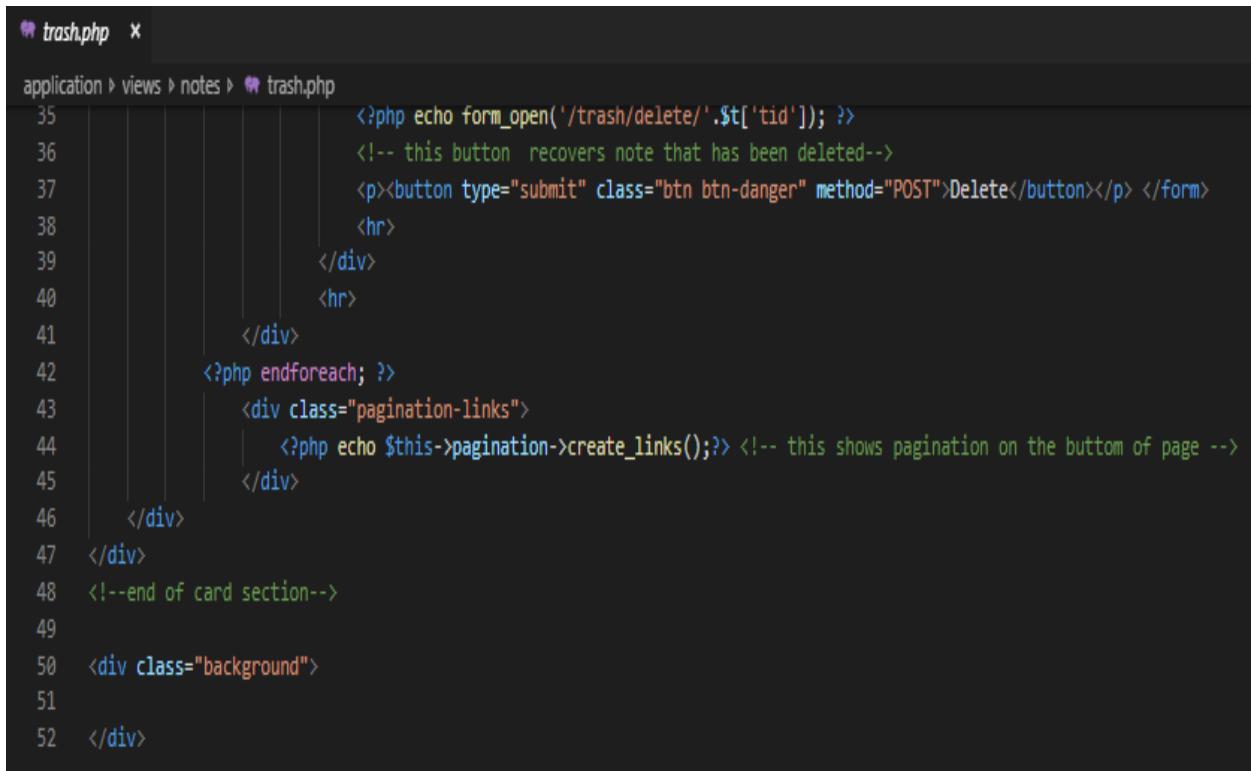
```
index.php
application > views > notes > index.php

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

Figure 49 code of index note part2

```
trash.php ×  
application › views › notes › trash.php  
1  <!--this is a index page of trash where moved notes are shown-->  
2  <!--this part presents cart section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7  
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
9                      <i class="fas fa-align-left" style="font-size: 1.2em; color: #007bff; vertical-align: middle; margin-right: 5px;"></i>  
10                     <span>Toggle Pinned Notes</span>  
11                  </button>  
12  
13          </div>  
14      </nav>  
15      <!--this shows title of this page-->  
16      <h2 class="text"><?= $title ?></h2>  
17      <?php foreach((array) $trash as $t) : ?>  
18          <!-- here name of note is displayed-->  
19          <h3 class="text"><?php echo $t['notename']; ?></h3>  
20          <div class="row">  
21              <div class="col-md-3">  
22                  <!-- note uploaded/default image is shown-->  
23                    
24              </div>  
25              <div class="col-md-9">  
26                  <!--here date in which note is created is displayed-->  
27                  <small class="note-date"> Deleted on:<?php echo $t['deleteddate']; ?><br>  
28                  <!-- Note detail is shown with word limit of 70-->  
29                  <div class="note-index-body"> <?php echo word_limiter($t['notedetail'], 70); ?>  
30                  </div>  
31                  <br><br>  
32                  <?php echo form_open('/trash/recover/' . $t['tid']); ?>  
33                  <!-- this button recovers note that has been deleted-->  
34                  <p><button type="submit" class="btn btn-warning" method="POST">Recover</button></p> </form>  
35                  <?php echo form_open('/trash/delete/' . $t['tid']); ?>  
36                  <!-- this button recovers note that has been deleted-->  
37                  <p><button type="submit" class="btn btn-danger" method="POST">Delete</button></p> </form>
```

Figure 50 code of trash part1



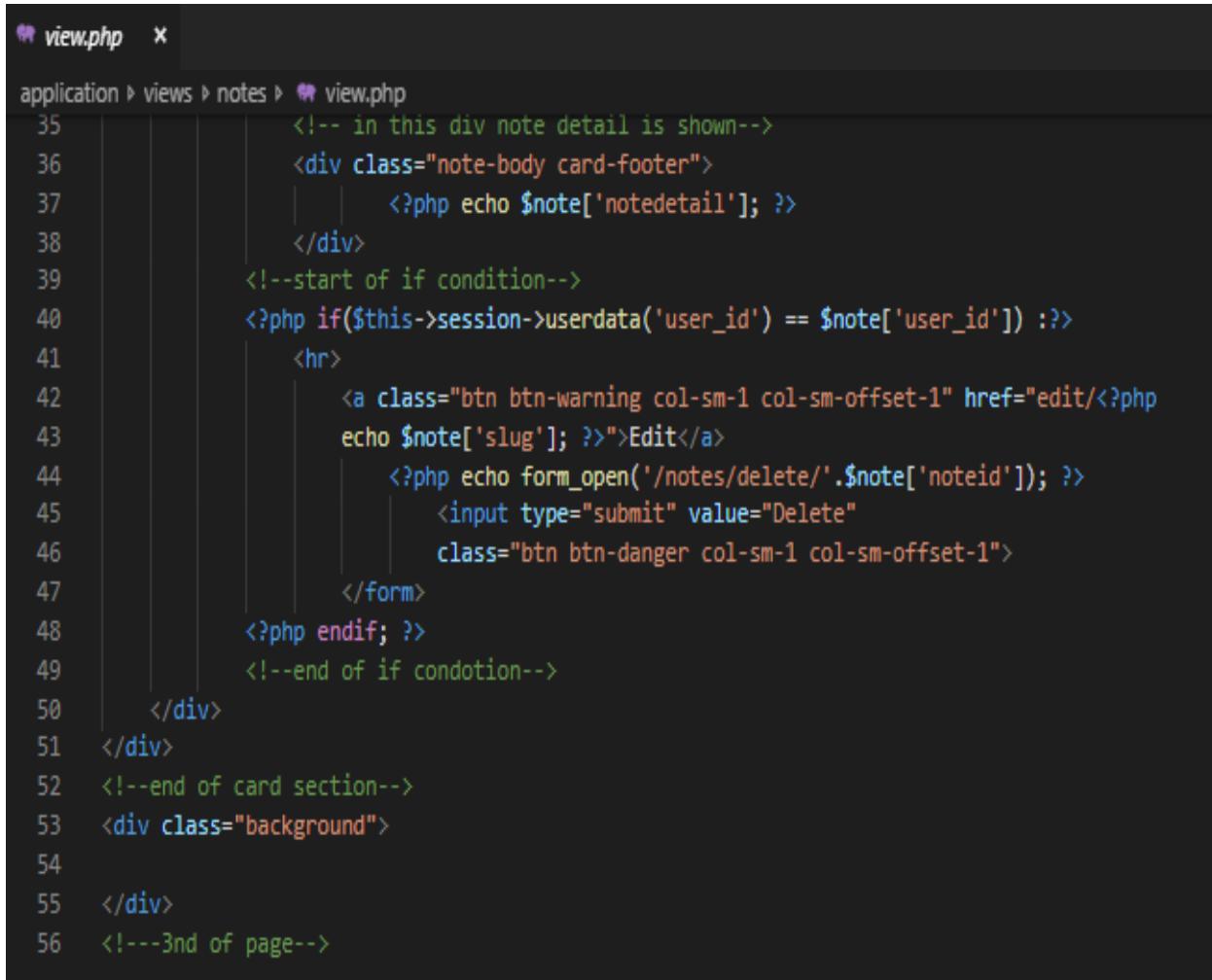
The screenshot shows a code editor window with the file 'trash.php' open. The code is a PHP script with embedded HTML and CSS. It includes a form for deleting notes, pagination links, and a background section. The code is numbered from 35 to 52.

```
35 <?php echo form_open('/trash/delete/'.$t['tid']); ?>
36     <!-- this button recovers note that has been deleted-->
37     <p><button type="submit" class="btn btn-danger" method="POST">Delete</button></p> </form>
38     <hr>
39     </div>
40     <hr>
41     </div>
42     <?php endforeach; ?>
43     <div class="pagination-links">
44         <?php echo $this->pagination->create_links();?> <!-- this shows pagination on the bottom of page -->
45     </div>
46 </div>
47 </div>
48 <!--end of card section-->
49
50 <div class="background">
51
52 </div>
```

Figure 51 code of trash part2

```
view.php  X
application > views > notes > view.php
1  <!-- this is a view page for full screen of a particular note-->
2  <!--start of card section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <div class="toggle">
6              <nav class="navbar navbar-expand-lg ">
7                  <div class="container-fluid">
8
9                      <button type="button" id="sidebarCollapse" class="btn btn-info">
10                         <i class="fas fa-align-left"></i>
11                         <span>Toggle Pinned Notes</span>
12                     </button>
13
14                 </div>
15             </nav>
16         </div>
17     <!--this is note name-->
18     <h2 class="text text-underline"><strong><?php echo $note['notename']; ?></strong></h2>
19     <small class = "note-date"> Saved on:<?php echo $note['notecreateddot']; ?></small><br>
20     <div class="row">
21         <div class="col-sm-4">
22
23             </div>
24             <!-- in this div picture is shown-->
25             <div class="col-sm-4">
26                 
29             </div>
30
31         <div class="col-sm-4">
32
33             </div>
34         </div>
35             <!-- in this div note detail is shown-->
36             <div class="note-body card-footer">
37                 <?php echo $note['notedetail']; ?>
```

Figure 52 code of view note part1



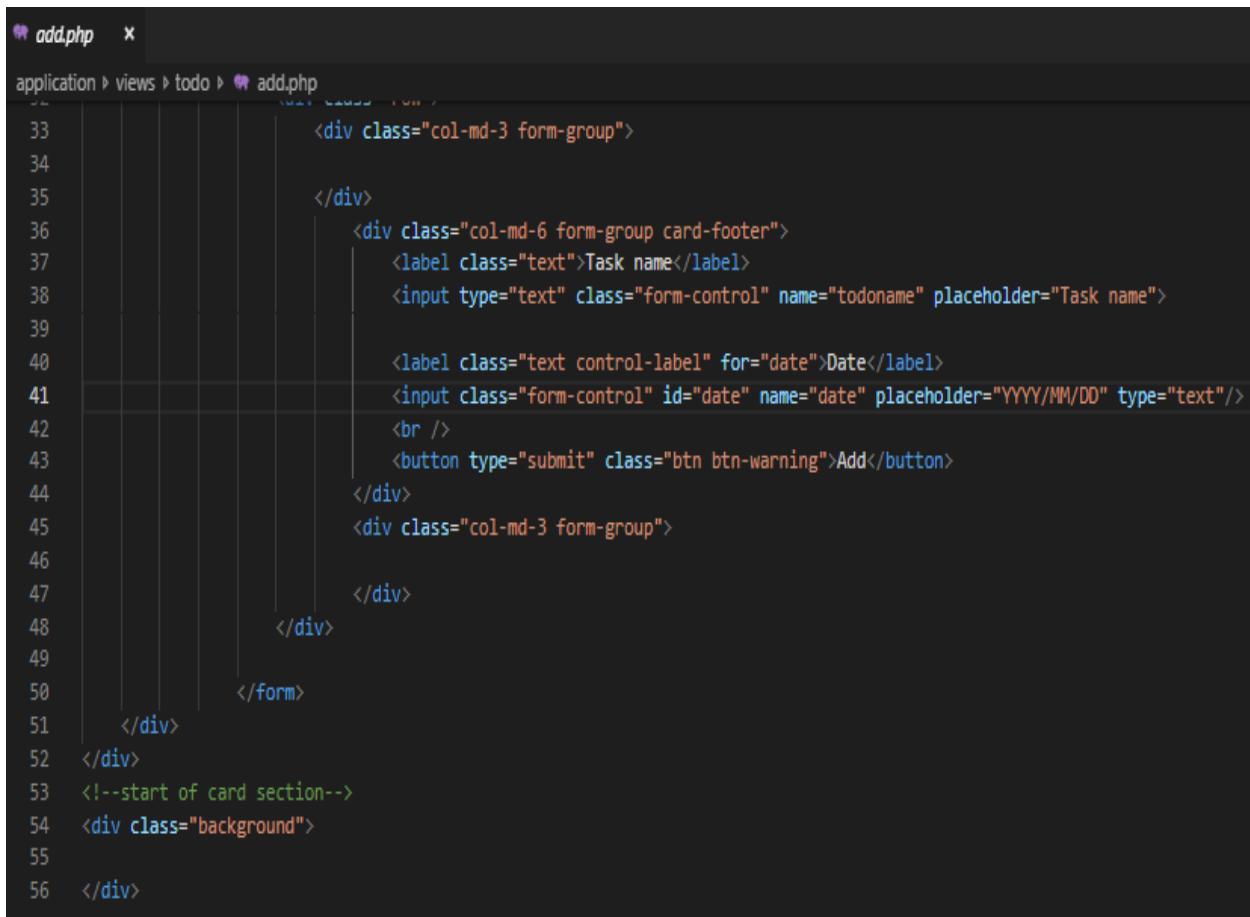
The screenshot shows a code editor window with the file 'view.php' open. The code is part of a PHP application's view layer, specifically for displaying a note detail. The code includes HTML and PHP logic for rendering a note body, handling user authentication via session, and providing edit and delete functionality through a form. It also includes comments indicating sections like the note detail div and the end of the card section.

```
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  
1  <!--this is a page of todo list where tasks can be added-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7  
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
9                      <i class="fas fa-align-left"></i>  
10                     <span>Toggle Pinned Notes</span>  
11                 </button>  
12  
13             </div>  
14         </nav>  
15         <div class="row">  
16             <div class="col-md-3 form-group">  
17  
18                 </div>  
19             <div class="col-md-6 form-group">  
20                 <!--this shows title of this page-->  
21                 <h2 class="text text-center card-header"><?= $title ;?></h2>  
22             </div>  
23             <div class="col-md-3 form-group">  
24  
25                 </div>  
26         </div>  
27         <!-- any errors encountered, it will be shown here-->  
28         <div class="text text-danger text-center"><strong><?php echo validation_errors(); ?>  
29             </strong></div>  
30         <!--opens form support images-->  
31         <?php echo form_open_multipart('todo/add'); ?>  
32             <div class="row">  
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>
```

Figure 54 code of add task part1



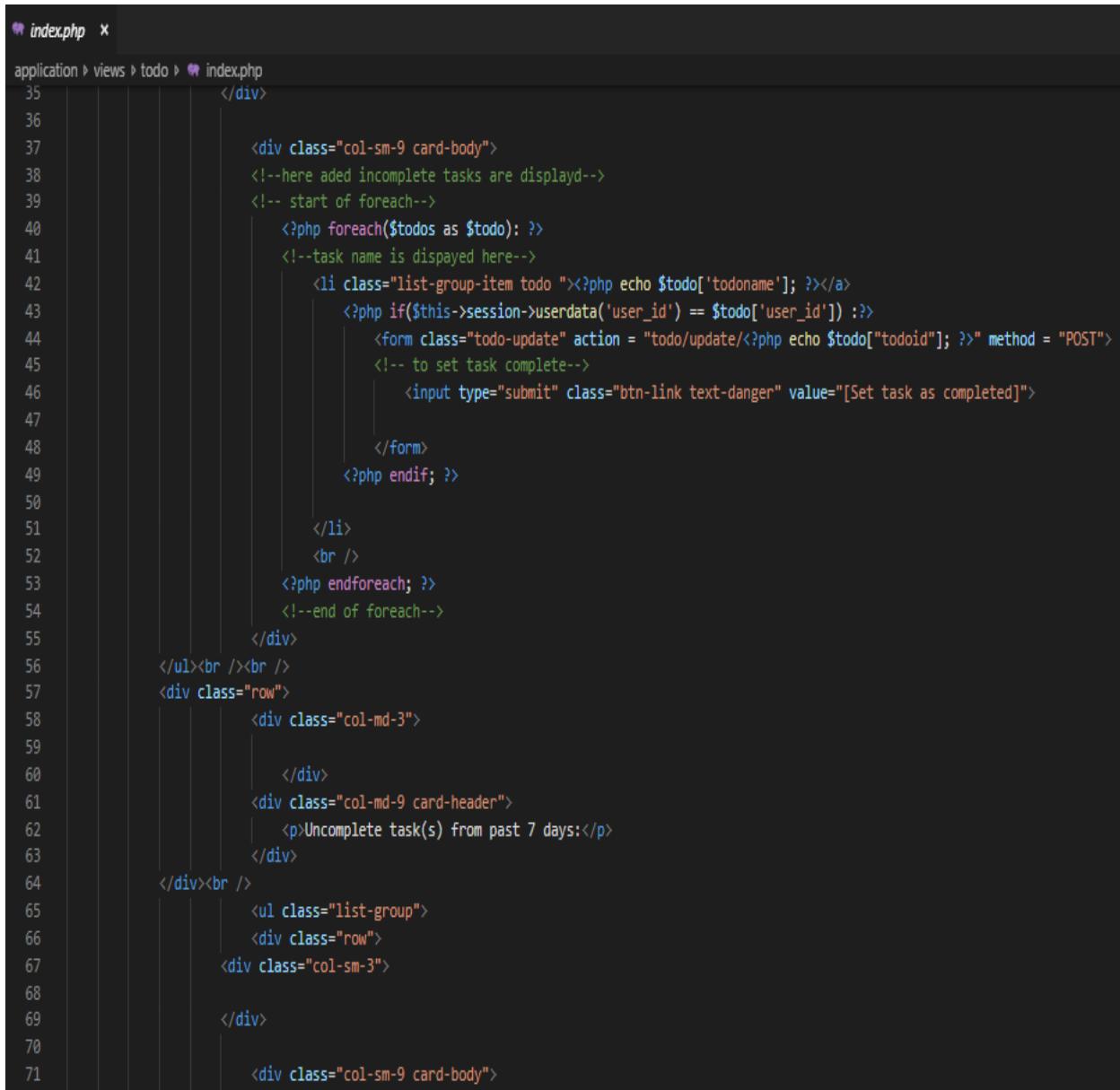
The screenshot shows a code editor window with the file name 'add.php' at the top. The code is a PHP script with embedded HTML and CSS. It includes a form for adding a task, with fields for task name and date, and a submit button. The code uses Bootstrap classes like 'col-md-3' and 'form-control'. The editor interface shows line numbers from 33 to 56.

```
33     <div class="col-md-3 form-group">
34         </div>
35     <div class="col-md-6 form-group card-footer">
36         <label class="text">Task name</label>
37         <input type="text" class="form-control" name="todoname" placeholder="Task name">
38
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  
1  <!--this is a page of todo list where incompleted tasks are shown and new tasks are added-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
8                      <i class="fas fa-align-left">/i>  
9                      <span>Toggle Pinned Notes</span>  
10                 </button>  
11             </div>  
12         </nav>  
13         <!--this shows title of this page-->  
14         <h2 class="text card-header"><?= $title; ?></h2><br />  
15  
16             <ul class="list-group">  
17                 <div class="row">  
18                     <div class="col-md-3">  
19                         </div>  
20                     <div class="col-md-9 card-header">  
21                         <p>Uncomplete task(s) from past 1 day:</p>  
22                     </div><br />  
23                 </div>  
24                 <div class="row">  
25                     <div class="col-sm-3">  
26                         <!--add tasks button-->  
27                         <a class="btn btn-success" href="<?php echo base_url(); ?>todo/add">Add task</a>  
28                         <br />  
29                         <br />  
30                         <!--view completed task button-->  
31                         <a class="btn btn-success" href="<?php echo base_url(); ?>todo/view">Task completed</a>  
32                     </div>  
33                 <div class="col-sm-9 card-body">
```

Figure 56 code of index task part1



The screenshot shows a code editor window with the file name "index.php" at the top. The code is a part of a PHP application for managing tasks. It includes HTML and PHP code for displaying incomplete tasks and tracking user activity over the past 7 days.

```
index.php
application > views > todo > index.php
35     </div>
36
37     <div class="col-sm-9 card-body">
38         <!--here added incomplete tasks are displayed-->
39         <!-- start of foreach-->
40         <?php foreach($todos as $todo): ?>
41             <!--task name is displayed 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             </div class="col-sm-9 card-body">
```

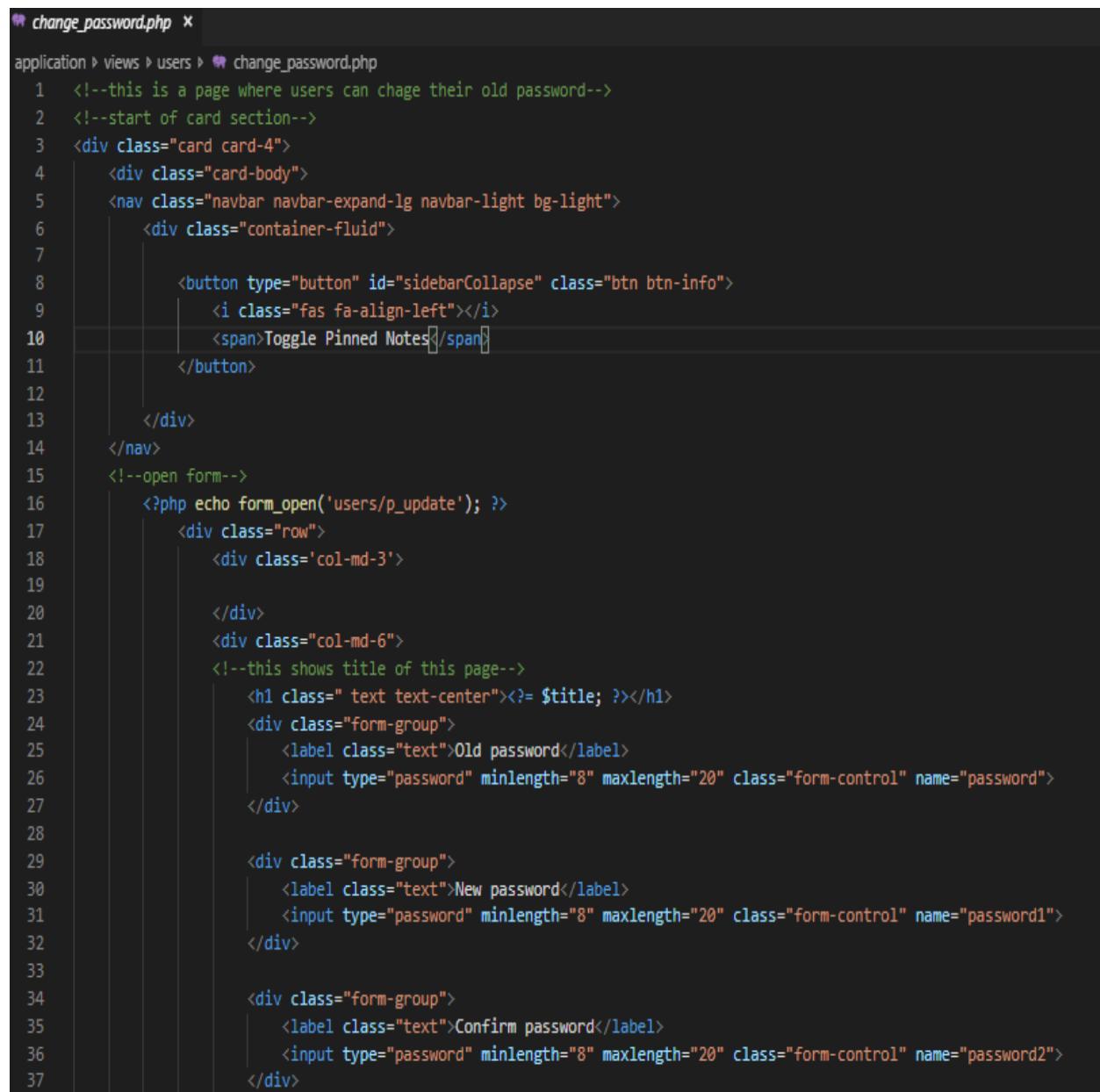
Figure 57 code of index task part2

```
index.php
application > views > todo > index.php
  6
  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                         </form>
  82                     <?php endif; ?>
  83
  84
  85                 </li>
  86                 <br />
  87             <?php endforeach; ?>
  88             <!--end of foreach-->
  89         </div>
  90     </div>
  91 </div>
  92 <!--start of card section-->
  93 <div class="background">
  94
  95 </div>
```

Figure 58 code of index task part3

```
view.php x
application > views > todo > view.php
1  <!--this is a page of todo list where completed tasks are shown-->
2  <!--start of card section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">
6              <div class="container-fluid">
7
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">
9                      <i class="fas fa-align-left"></i>
10                     <span>Toggle Pinned Notes</span>
11                 </button>
12
13             </div>
14         </nav>
15         <!--this shows title of this page-->
16         <h2 class="text card-header"><?= $title; ?></h2>
17         <ul class="list-group">
18             <!--start of foreach-->
19             <?php foreach($todos as $todo): ?>
20                 <!--all completed tasks are displayed-->
21                 <li class="list-group-item todo"><?php
22                     echo $todo['todoname']; ?></a>
23
24                 </li>
25             <?php endforeach; ?>
26             <!--end of foreach-->
27         </ul>
28     </div>
29 </div>
30 <!--end of card section-->
31     <div class="background">
32
33     </div>
```

Figure 59 code of view task



The screenshot shows a code editor with a dark theme. The file is named 'change_password.php' and is located in the 'views/users' directory. The code is a PHP script with embedded HTML and CSS. It includes a navigation bar with a sidebar collapse button and a title. Below the navigation, there are three form groups for 'Old password', 'New password', and 'Confirm password', each with a label and an input field. The code uses Bootstrap classes like 'form-control' and 'text-center'.

```
1 <!--this is a page where users can chage their old password-->
2 <!--start of card section-->
3 <div class="card card-4">
4   <div class="card-body">
5     <nav class="navbar navbar-expand-lg navbar-light bg-light">
6       <div class="container-fluid">
7
8         <button type="button" id="sidebarCollapse" class="btn btn-info">
9           <i class="fas fa-align-left"></i>
10          <span>Toggle Pinned Notes</span>
11        </button>
12
13      </div>
14    </nav>
15    <!--open form-->
16    <?php echo form_open('users/p_update'); ?>
17    <div class="row">
18      <div class='col-md-3'>
19
20        </div>
21        <div class="col-md-6">
22          <!--this shows title of this page-->
23          <h1 class=" text text-center"><?= $title; ?></h1>
24          <div class="form-group">
25            <label class="text">Old password</label>
26            <input type="password" minlength="8" maxlength="20" class="form-control" name="password">
27          </div>
28
29          <div class="form-group">
30            <label class="text">New password</label>
31            <input type="password" minlength="8" maxlength="20" class="form-control" name="password1">
32          </div>
33
34          <div class="form-group">
35            <label class="text">Confirm password</label>
36            <input type="password" minlength="8" maxlength="20" class="form-control" name="password2">
37          </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         </div>  
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  
1  <!--this is a page where users can edit their profile-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
8                      <i class="fas fa-align-left"></i>  
9                      <span>Toggle Pinned Notes</span>  
10                 </button>  
11             </div>  
12         </nav>  
13         <!--sends any error to validation_errors-->  
14         <?php echo validation_errors(); ?>  
15         <!--open form-->  
16         <?php echo form_open('users/update'); ?>  
17             <div class="row">  
18                 <div class='col-md-3'>  
19                     </div>  
20                 <div class="col-md-6">  
21                     <!--this shows title of this page-->  
22                     <h1 class="text text-center"><?= $title; ?></h1>  
23                     <div class="form-group">  
24                         <label class="text">Nickname</label>  
25                         <!-- here user's name is displayed-->  
26                         <input type="text" minlength="3" maxlength="50" class="form-control" name="name"  
27                             placeholder="Nickname" value="<?php echo $user['name']; ?>">  
28                     </div>  
29                     <div class="form-group">  
30                         <label class="text">Email</label>  
31                         <!-- here user's email is displayed-->  
32                         <input type="text" maxlength="100" class="form-control" name="email"  
33                             placeholder="Email" value="<?php echo $user['email']; ?>">
```

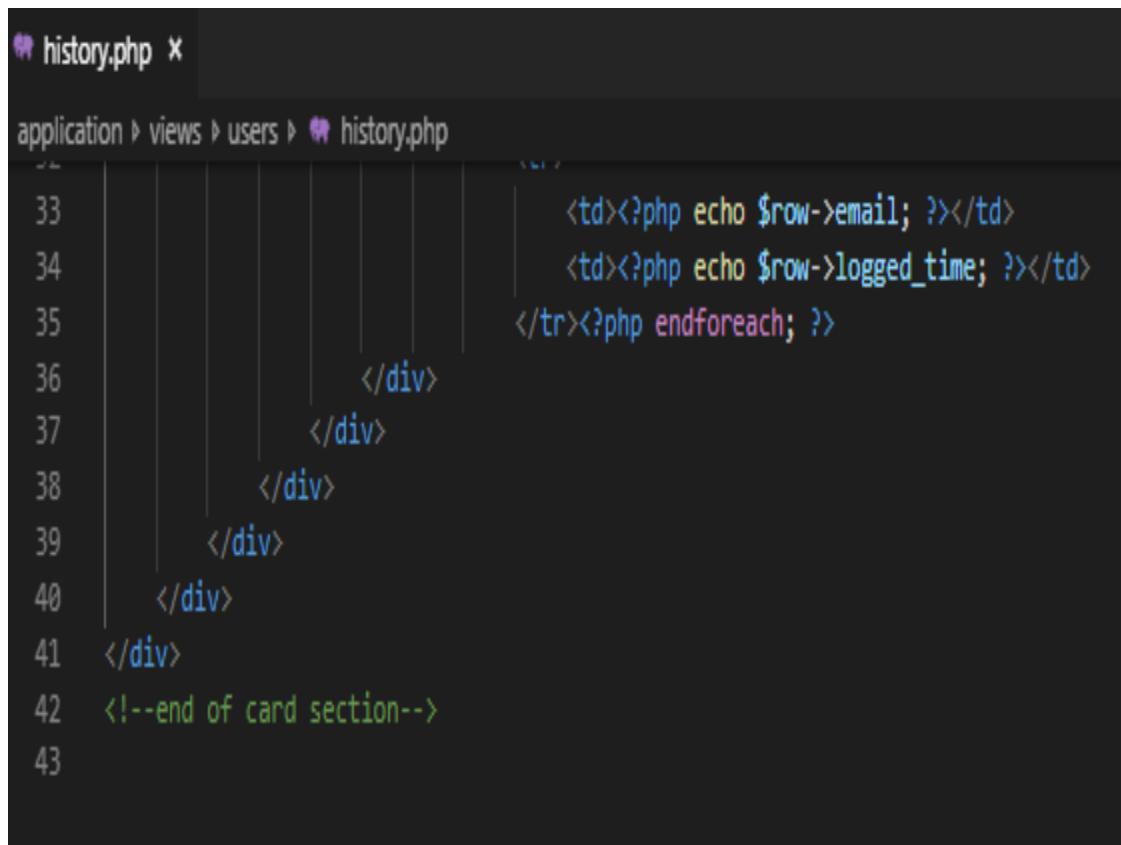
Figure 62 code of edit profile part1

```
edit_profile.php ×
application > views > users > edit_profile.php
35 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
36 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
37 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
38 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
39 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
40 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
41 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
42 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
43 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
44 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
45 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
46 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
47 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
48 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
49 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
50 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
51 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
52 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
```

Figure 63 code of edit profile part2

```
history.php *  
application > views > users > history.php  
1  <!--this is a page where login history is displayed of user-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7  
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
9                      <i class="fas fa-align-left"></i>  
10                     <span>Toggle Pinned Notes</span>  
11                 </button>  
12  
13             </div>  
14         </nav>  
15         <!--this shows title of this page-->  
16         <h2 class="text"> <?= $title; ?> </h2>  
17         <br />  
18  
19         <div class="panel panel-default">  
20             <div class="panel-heading">  
21             </div>  
22             <div class="panel-body">  
23                 <div class="table-responsive">  
24                     <table class="table table-bordered table-striped">  
25                         <tr>  
26                             <th class="text">Email</th>  
27                             <th class="text">Logged time</th>  
28                         </tr>  
29                         <?php  
30                             foreach($login_history as $row): ?>  
31  
32                             <tr>  
33                                 <td><?php echo $row->email; ?></td>  
34                                 <td><?php echo $row->logged_time; ?></td>  
35                             </tr><?php endforeach; ?>  
36  
37             </div>  
        </div>
```

Figure 64 code of login history part1



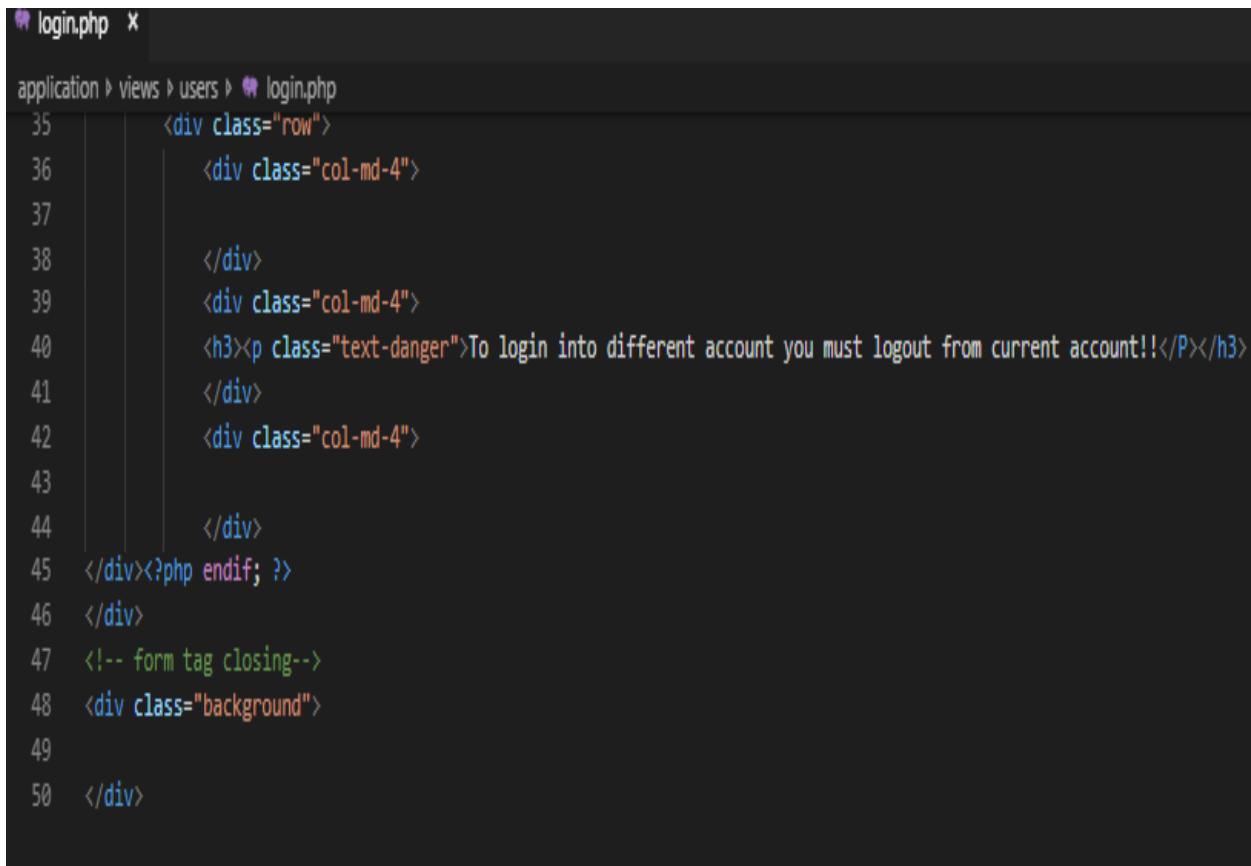
The screenshot shows a code editor window with the file 'history.php' open. The file is located in the 'application/views/users' directory. The code is part of a card section and includes PHP logic for displaying user email and logged-in time.

```
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 x
application > views > users > login.php
1  <!--this is a page where users can log in-->
2  <!--start of card section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <!--form tag openning-->
6          <?php echo form_open('users/login'); ?>
7          <?php if(!$this->session->userdata('logged_in')) : ?>
8              <div class="row">
9                  <div class='col-md-4'>
10
11                  </div>
12                  <div class='col-md-4'>
13                      <!--this shows title of this page-->
14                      <h1 class="text text-center"><?php echo $title; ?></h1>
15                      <div class="form-group">
16                          <input type="text" maxlength="100" name="email" class="form-control"
17                          placeholder="Enter email" required autofocus>
18                      </div>
19                      <div class="form-group">
20                          <input type="password" minlength="8" maxlength="20" name="password" class="form-control"
21                          placeholder="Enter password" required autofocus>
22                      </div>
23                          <button type="submit" class="btn btn-warning btn-block">Login</button>
24
25                  </div>
26                  <div class='col-md-4'>
27
28                  </div>
29              </div>
30          <?php echo form_close(); ?>
31      </div>
32      </div><?php endif;?>
33      <?php if($this->session->userdata('logged_in')) : ?>
34          <div class="show-info">
35              <div class="row">
36                  <div class="col-md-4">
37                      ...
```

Figure 66 code of login part1



The screenshot shows a code editor window with the file 'login.php' open. The code is a PHP script with embedded HTML and CSS. It uses Bootstrap's grid system with rows and columns. A specific line of code, line 40, contains a warning message: '

<p class="text-danger">To login into different account you must logout from current account!!</p></h3>'. The code is numbered from 35 to 50.

```
login.php X

application > views > users > login.php
35     <div class="row">
36         <div class="col-md-4">
37
38             </div>
39             <div class="col-md-4">
40                 <h3><p class="text-danger">To login into different account you must logout from current account!!</p></h3>
41             </div>
42             <div class="col-md-4">
43
44             </div>
45     </div><?php endif; ?>
46     </div>
47     <!-- form tag closing-->
48     <div class="background">
49
50     </div>
```

Figure 67 code of login part2

```
profile.php ×  
application > views > users > profile.php  
1  <!--this is a page where user get option-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7  
8                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
9                      <i class="fas fa-align-left"></i>  
10                     <span>Toggle Pinned Notes</span>  
11                 </button>  
12  
13             </div>  
14         </nav>  
15         <!--this shows title of this page-->  
16         <h2 class="text card-header"><center><?= $title; ?></h2>  
17         <!--here user email is shown-->  
18         <div class="text text-center"><strong> Welcome <?php echo $this->session->userdata('email') ;?>  
19             </strong>  
20         </div></center></span></header>  
21         <div class="row">  
22             <div class="col-sm-4 col-md-4 card-footer"><center>  
23                 <a class="btn btn-success" href="<?php echo base_url(); ?>users/edit_profile">Edit  
24                     profile</a><br /><br />  
25                 <a class="btn btn-success" href="<?php echo base_url(); ?>users/change_password">Change  
26                     password</a><br /><br />  
27                 <a class="btn btn-success" href="<?php echo base_url(); ?>users/login_history">Login  
28                     log</a><br /><br />  
29                 <a class="btn btn-success" href="<?php echo base_url(); ?>export_csv/index">Export  
30                     data</a><br /><br />  
31                 <a class="btn btn-success" href="<?php echo base_url(); ?>profile/trash">  
32                     Trash</a><br /><br />  
33             </center>  
34         </div>  
35         <!--Total number of notes saved are shown-->  
36         <div class="profile col-sm-4 col-md-4 text-danger card-footer">  
37             <tr>
```

Figure 68 code of profile part1



The screenshot shows a code editor window with the file 'profile.php' open. The code is written in PHP and displays user statistics. It includes sections for notes and tasks, each with conditional logic to show either a count or a message if no items exist.

```
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 remaining tasks. ';
68             }
69         ?>
70     </div>
71     </div>
72 </div>
```

Figure 69 code of profile part2



The screenshot shows a code editor window with the file 'profile.php' open. The code is a PHP script with some HTML and CSS. It includes a loop that outputs rows of data from a database, a conditional statement for users who have not completed any tasks, and a comment indicating the end of a section. The code is color-coded for syntax highlighting.

```
profile.php X
application > views > users > profile.php
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 remaining tasks. ';
68     }
69     ?>
70     </div>
71     </div>
72 </div>
73 </div>
74 <!--end of card section-->
75 <div class="background">
76
77 </div>
```

Figure 70 code of profile part3

```
register.php ×  
application › views › users › register.php  
1  <!--this is a page where can register -->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <!--sends any error to validation_errors-->  
6          <?php if(!$this->session->userdata('logged_in')) : ?>  
7              <div class="text text-center text-danger card-footer"><?php echo validation_errors(); ?>  
8                  <?php echo 'Note: Password should be at least 8  
9                      characters in length and should include  
10                     at least one upper case letter, one  
11                     number, and one special character.'  
12                  ?> <br /> <br />  
13          </div>  
14          <!--form tag opening-->  
15          <?php echo form_open('users/register'); ?>  
16          <div class="row">  
17              <div class="col-md-4">  
18  
19                  </div>  
20                  <div class="col-md-4">  
21                      <!--this shows title of this page-->  
22                      <h1 class="text text-center"><?= $title; ?></h1>  
23                      <div class="form-group">  
24                          <label class="text">Nickname</label>  
25                          <input type="text" minlength="3" maxlength="50" class="form-control" name="name"  
26                          placeholder="Nickname" required>  
27                      </div>  
28  
29                      <div class="form-group">  
30                          <label class="text">Email</label>  
31                          <input type="text" maxlength="100" class="form-control" name="email"  
32                          placeholder="Email" required>  
33                      </div>  
34  
35                      <div class="form-group">  
36                          <label class="text">Password</label>  
37                          <input type="password" minlength="8" maxlength="20" class="form-control" name="password"
```

Figure 71 code of register part1

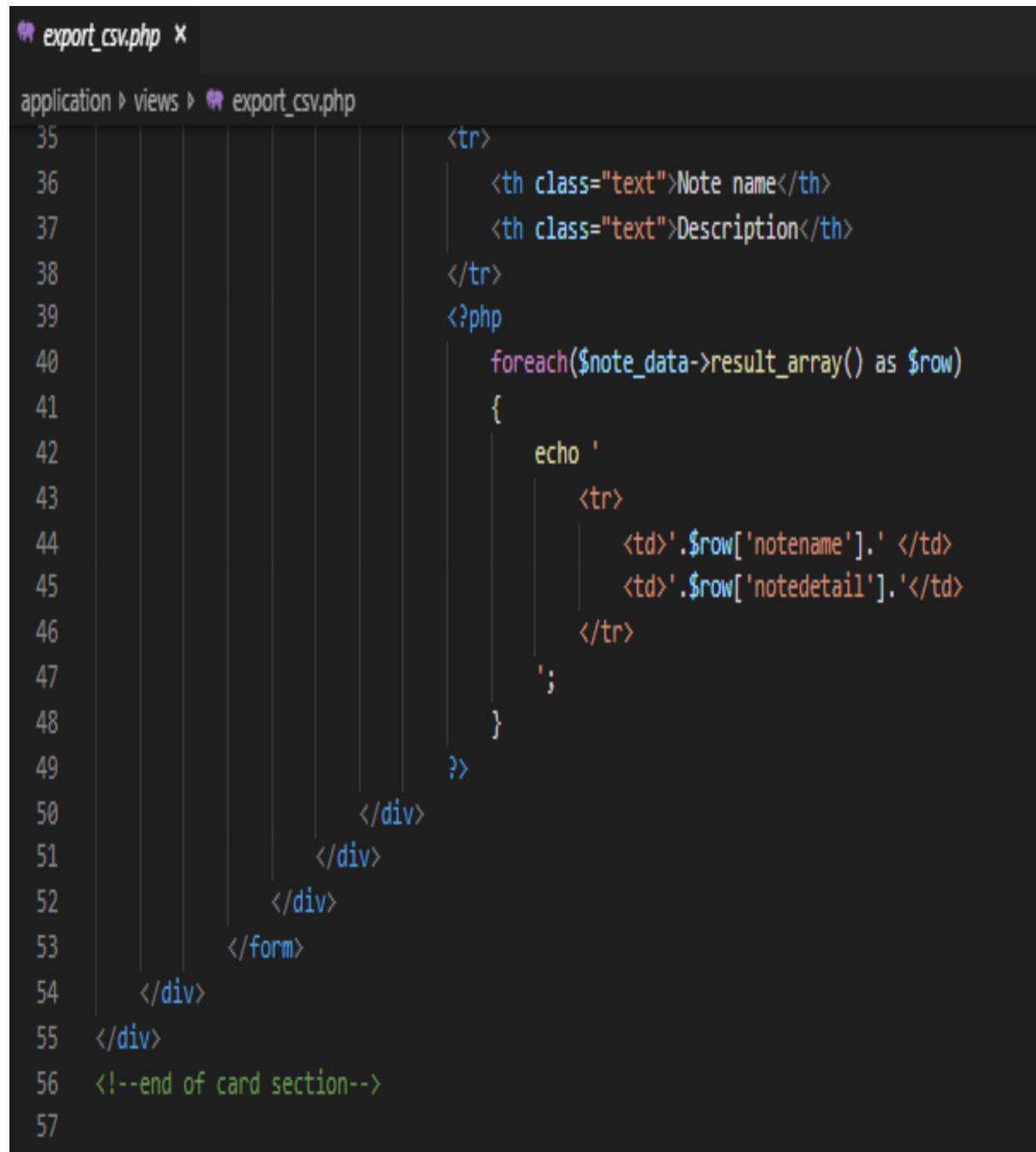
```
register.php
application > views > users > register.php

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  
1  <!--this is a page where user can export data-->  
2  <!--start of card section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <nav class="navbar navbar-expand-lg navbar-light bg-light">  
6              <div class="container-fluid">  
7                  <button type="button" id="sidebarCollapse" class="btn btn-info">  
8                      <i class="fas fa-align-left"></i>  
9                      <span>Toggle Pinned Notes</span>  
10                 </button>  
11             </div>  
12         </div>  
13     </nav>  
14     <!-- this is a title of current page-->  
15     <h3 class="text align="center"> Export data of notes</h3>  
16     <br />  
17     <form method="POST" action="<?php echo base_url(); ?>export_csv/export">  
18         <div class="panel panel-default">  
19             <div class="panel-heading">  
20                 <div class="row">  
21                     <div class="col-md-6">  
22                         <h3 class="panel-title text"> Notes data</h3>  
23                     </div>  
24                     <div class="col-md-6" align="right">  
25                         <input type="submit" name="export" class="btn btn-success btn-xs" value="Export"/>  
26                     </div>  
27                 </div>  
28             </div>  
29         </div>  
30         <div class="panel-body">  
31             <div class="table-responsive">  
32                 <table class="table table-bordered talble-striped">  
33                     <!-- here all notes aree displayed of current user-->  
34                     <tr>  
35                         <th class="text">Note name</th>  
36                         <th class="text">Description</th>  
37                     ..
```

Figure 73 code of export data part1\



The screenshot shows a code editor window with the file name "export_csv.php" at the top. The code is a PHP script that generates an HTML table for a CSV export. It includes a header row with columns for Note name and Description, followed by a loop that iterates over an array of rows from \$note_data. Each row is output as a table row containing two cells with the values from the 'notename' and 'notedetail' keys respectively. The code uses echo statements to build the HTML string, which is then output via a print statement. The entire output is wrapped in a large div element.

```
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
```

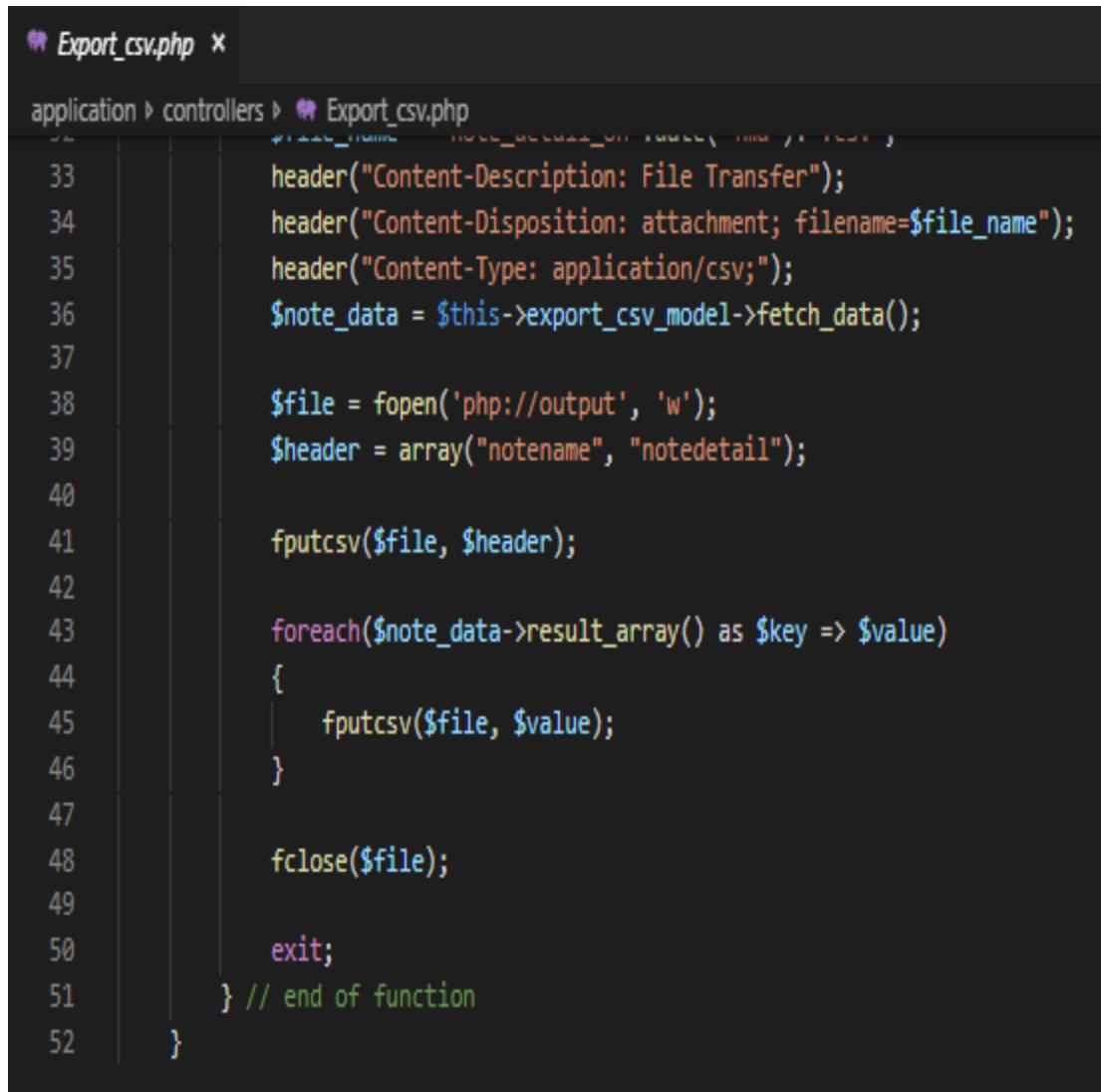
Figure 74 code of export data part2

```
view.php X
application > views > search > view.php
1  <!--this is a page of searchnote where all user noted are displayed-->
2  <!--start of card section-->
3  <div class="card card-4">
4      <div class="card-body">
5          <!--this shows title of this page-->
6          <h2 class="text">?= $title; ?> </h2>
7          <br />
8
9          <div class="panel panel-default">
10         <div class="panel-heading">
11             </div>
12             <div class="panel-body">
13                 <div class="table-responsive">
14                     <table class="table table-bordered table-striped">
15                         <tr>
16                             <th class="text">Note Name</th>
17
18                         </tr>
19                         <!--start of foreach-->
20                         <?php
21                         foreach($notes as $row): ?>
22
23                         <!--here note name is displayed-->
24                         <tr>
25                             <td><a href="<?php echo site_url('/notes/'.$row->slug); ?>"><?php echo $row->notename; ?></td>
26
27                         </tr>
28                     </div><?php endforeach; ?>
29
30                 </div>
31             </form>
32         </div>
33     </div>
34 </div>
35 <!--end of card section-->
```

Figure 75 code of displaying all data

```
Export_csv.php x
application > controllers > Export_csv.php
1  <!-- this is a controller class of extport_csv
2  <?php
3  class Export_csv extends CI_Controller
4  {
5      //This methode will be exectuded when new object from this class has been created
6      public function __construct()
7      {
8          //when new object from this class has been created __construct methode will be executed
9          parent::__construct();
10         $this->load->model('export_csv_model');//this methodwill load export_csv_model
11     }// end of function
12
13     // this is root methode of this class. when we type base url/export_csv then this method will execute
14     function index()
15     {
16         //check login
17         if(!$this->session->userdata('logged_in'))
18         [
19             redirect('users/login');
20         ]
21         $data['pinn'] = $this->note_model->get_pin();
22         $this->load->view('includes/header', $data);
23         //this methode will return data which have been stored under $data variable
24         $data['note_data'] = $this->export_csv_model->fetch_data();
25         $this->load->view('export_csv', $data);
26         $this->load->view('includes/footer');
27     }// end of function
28
29     //function to export data and to call respetive model
30     function export()
31     {
32         $file_name = 'note_detail_on'.date('Ymd').'.csv';
33         header("Content-Description: File Transfer");
34         header("Content-Disposition: attachment; filename=$file_name");
35         header("Content-Type: application/csv;");
36         $note_data = $this->export_csv_model->fetch_data();
37     }
}
```

Figure 76 code of export data controller part1



The screenshot shows a code editor window with the title "Export_csv.php". The file path is "application > controllers > Export_csv.php". The code is a PHP script for exporting data to CSV. It includes headers for Content-Type, Content-Disposition, and Content-Description. It uses the `fopen`, `fputcsv`, and `fclose` functions to create a CSV file. It also uses `exit;` at the end of the function. The code is numbered from 33 to 52.

```
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

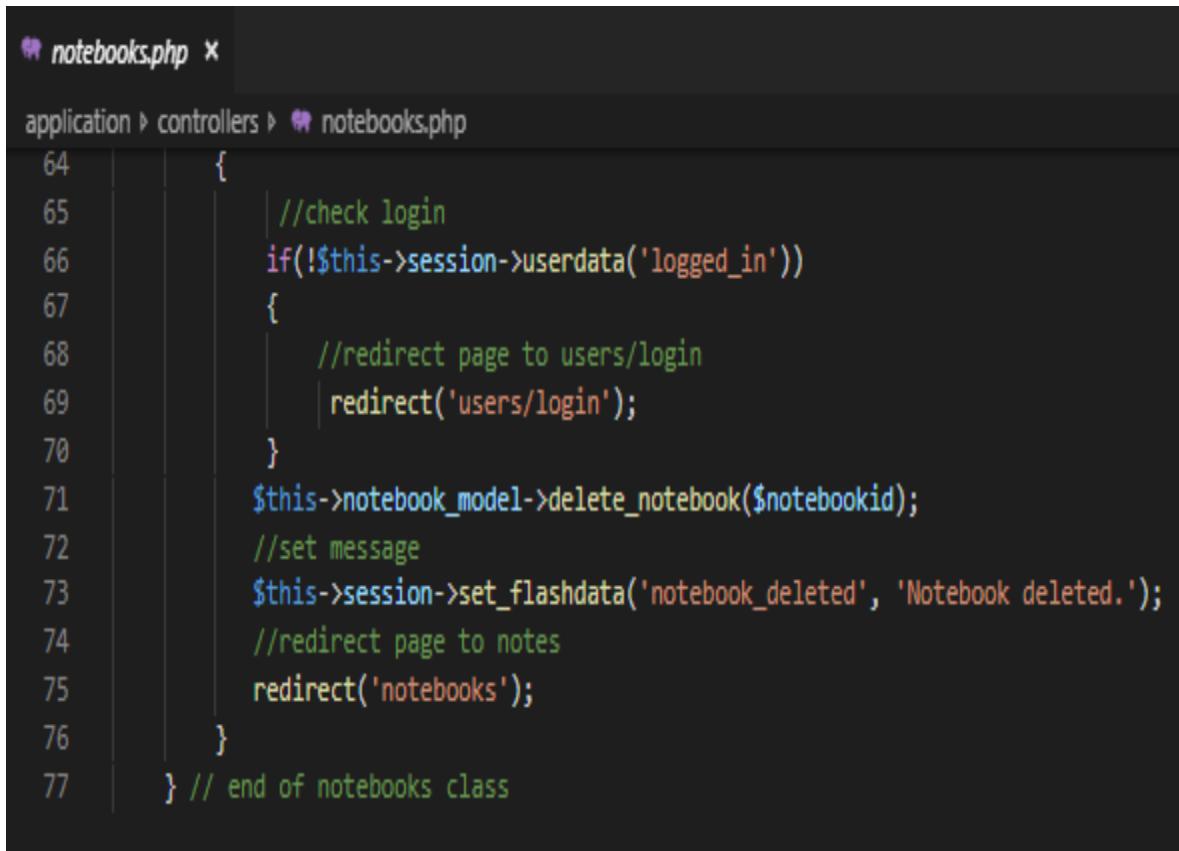
```
notebooks.php *  
application > controllers > notebooks.php  
1  <!--this is a controller class of notebook-->  
2  <?php  
3  class Notebooks extends CI_Controller  
4  {  
5      //function which loads following data when called  
6      public function index()  
7      {  
8          //check login  
9          if(!$this->session->userdata('logged_in'))  
10         {  
11             redirect('users/login');  
12         }  
13         $data['title'] = "Notebooks";  
14         $data['pinn'] = $this->note_model->get_pin();  
15         $data['notebooks'] = $this->notebook_model->get_notebooks();  
16         //load following assets  
17         $this->load->view('includes/header', $data);  
18         $this->load->view('notebooks/index', $data);  
19         $this->load->view('includes/footer');  
20     } // end of function  
21  
22     //function to add new notebook  
23     public function add()  
24     {  
25         //check login  
26         if(!$this->session->userdata('logged_in'))  
27         {  
28             redirect('users/login');  
29         }  
30         $data['title'] = "Add Notebook";  
31         $this->form_validation->set_rules('name', 'Name', 'required');  
32         if($this->form_validation->run() == FALSE)  
33         {  
34             $data['pinn'] = $this->note_model->get_pin();  
35             $this->load->view('includes/header', $data);  
36             $this->load->view('notebooks/add', $data);  
37             $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);
72 }
```

Figure 79 code of notebook controller part2



The screenshot shows a code editor window with the file 'notebooks.php' open. The code is part of a controller class. It includes a login check, deletion of a notebook, setting a flash message, and redirecting to the notes page. The code is color-coded for readability.

```
notebooks.php X
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  
1  <!-- this is a controller class of notes-->  
2  <?php  
3  class Notes extends CI_Controller  
4  {  
5      //function that calls model function when called  
6      public function index($offset = 0)  
7      {  
8          //pagination config  
9          $config['base_url'] = base_url() . 'notes/index/';  
10         $config['total_rows'] = $this->db->count_all('notes');  
11         $config['per_page'] = 3;  
12         $config['uri_segment'] = 3;  
13         // Produces: class="pagination-link"  
14         $config['attributes'] = array('class' => 'pagination-link');  
15  
16         //initiate pagination  
17         $this->pagination->initialize($config);  
18  
19         //check login  
20         if(!$this->session->userdata('logged_in'))  
21         {  
22             redirect('users/login');  
23         }  
24         $data['title'] = 'Latest notes';  
25         $data['notify'] = $this->todo_model->notify_tasks();  
26         $data['notes'] = $this->note_model->get_notes(FALSE,  
27             $config['per_page'], $offset);  
28         $data['pin'] = $this->note_model->get_pin();  
29         $this->load->view('includes/header', $data);  
30         $this->load->view('notes/index', $data);  
31         $this->load->view('includes/footer');  
32     }  
33  
34     //function that calls model function when called  
35     public function view($slug = NULL)  
36     {  
37         --
```

Figure 81 code of note controller part1



The screenshot shows a code editor window with the file 'Notes.php' open. The code is written in PHP and defines a controller class for managing notes. The code includes logic for checking user login, retrieving notes from a database, and displaying them. It also includes a function for adding new notes, which involves validating form data and setting rules for title and note detail fields.

```
//check login
if(!$this->session->userdata('logged_in'))
{
    redirect('users/login');
}
$data['note']= $this->note_model->get_notes($slug);

if (empty($data['note']))
{
    show_404();
}
$data['notename'] = $data['note']['notename'];
$data['pinn'] = $this->note_model->get_pin();
$this->load->view('includes/header', $data);
$this->load->view('notes/view', $data);
$this->load->view('includes/footer');

//function that calls model function when called
public function add()
{
    //check login
    if(!$this->session->userdata('logged_in'))
    {
        redirect('users/login');
    }
    $data['title'] = 'Add note';
    $data['notebooks'] = $this->note_model->get_notebooks();
    $this->form_validation->set_rules
    ('notename', 'Notename', 'required');
    $this->form_validation->set_rules
    ('notedetail', 'Notedetail', 'required');

    if($this->form_validation->run() === FALSE)
    {
        $data['pinn'] = $this->note_model->get_pin();
        $this->load->view('includes/header', $data);
    }
}
```

Figure 82 code of note controller part2



The screenshot shows a code editor window with the file 'Notes.php' open. The code is part of a note controller. It includes logic for adding a new note, handling file uploads, and saving the note to the database. The code uses the CodeIgniter framework's built-in upload library.

```
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

The screenshot shows a code editor window with the file name 'Notes.php' at the top. The code is written in PHP and contains two main functions: 'delete(\$noteid)' and 'edit(\$slug)'. The 'delete' function checks if the user is logged in, then inserts the note into a trash table, deletes it from the main table, sets a flash message, and redirects to the notes page. The 'edit' function checks if the user is logged in, retrieves the note by slug, checks if the user is the owner, and then retrieves the user's notebooks.

```
Notes.php
application > controllers > Notes.php
public function delete($noteid)
{
    //check login
    if(!$this->session->userdata('logged_in'))
    {
        //redirect page to users/login
        redirect('users/login');
    }
    //insert into trash table
    $this->trash_model->trash_note($noteid);
    //delete selected note
    $this->note_model->delete_note($noteid);
    //set message
    $this->session->set_flashdata('note_deleted',
        'Note deleted.');
    //redirect page to notes
    redirect('notes');
} //end of the function

//function that calls model function when called
public function edit($slug)
{
    //check login
    if(!$this->session->userdata('logged_in'))
    {
        //redirect page to users/login
        redirect('users/login');
    }
    $data['note']= $this->note_model->get_notes($slug);
    //check user
    if($this->session->userdata('user_id') != $this->note_model->get_notes($slug)['user_id'])
    {
        //redirect page to users/login
        redirect('users/login');
    }
    $data['notebooks'] = $this->note_model->
        get_notebooks();
```

Figure 84 code of note controller part4

```
Notes.php x

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
```

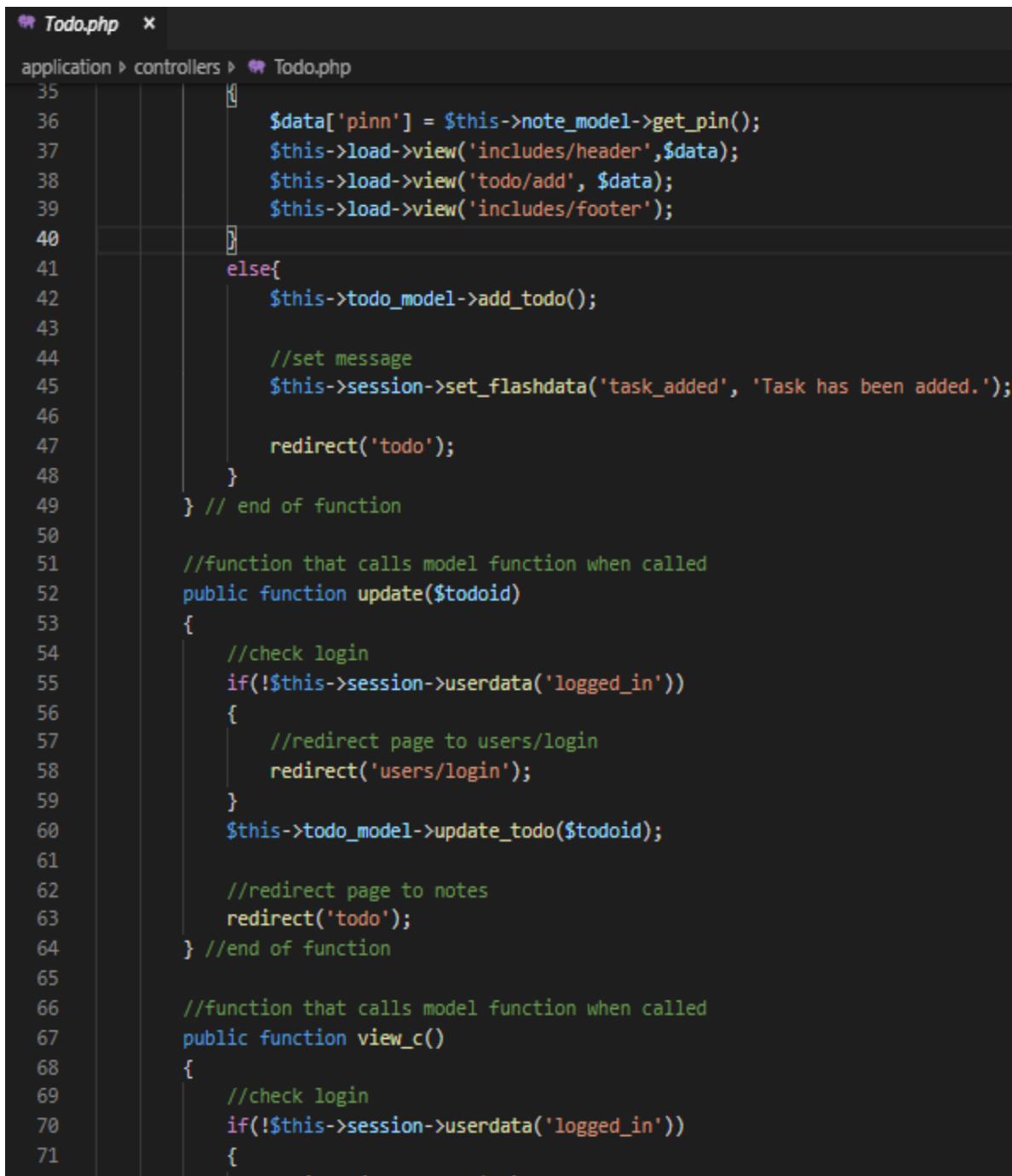
Figure 85 code of note controller part4

```
Profile.php x  
application > controllers > Profile.php  
1  <!-- this is a controller class of profile-->  
2  <?php  
3  class Profile extends CI_Controller  
4  {  
5      //function that calls model function when called  
6      public function index()  
7      {  
8          $this->load->model('user_model');  
9          //check login  
10         if(!$this->session->userdata('logged_in'))  
11         {  
12             redirect('users/login');  
13         }  
14         $data['pin'] = $this->note_model->get_pin();  
15         $this->load->view('includes/header', $data);  
16         $data['title'] = 'Profile';  
17         $id = $this->session->userdata('user_id');  
18         $data['row'] = $this->user_model->count_rows($id);  
19         $data['task'] = $this->user_model->count_ctasks($id);  
20         $this->load->view('users/profile', $data);  
21         $this->load->view('includes/footer');  
22     }  
23 } //end of function
```

Figure 86 code of profile controller part1

```
* Todo.php  x
application > controllers > Todo.php
1  <!-- this is a controller class of todo-->
2  <?php
3  class Todo extends CI_Controller
4  {
5      //function that calls model function when called
6      public function index()
7      {
8          //check login
9          if(!$this->session->userdata('logged_in'))
10         {
11             redirect('users/login');
12         }
13         $data['title'] = "Your's today's perform list ";
14         $data['todos'] = $this->todo_model->get_todo();
15         $data['tos'] = $this->todo_model->get_todo7();
16         $data['pinn'] = $this->note_model->get_pin();
17         //load following assets
18         $this->load->view('includes/header', $data);
19         $this->load->view('todo/index', $data);
20         $this->load->view('includes/footer');
21     } // end of function
22
23     //function that calls model function when called
24     public function add()
25     {
26         //check login
27         if(!$this->session->userdata('logged_in'))
28         {
29             redirect('users/login');
30         }
31         $data['title'] = 'Add Todo';
32         $this->form_validation->set_rules('todoname','Todoname', 'required');
33         $this->form_validation->set_rules('date','Tododate', 'required');
34         if($this->form_validation->run() === FALSE)
35         {
36             $data['pinn'] = $this->note_model->get_pin();
37             $this->load->view('includes/header',$data);
38         }
39     }
40 }
```

Figure 87 code of task controller part1

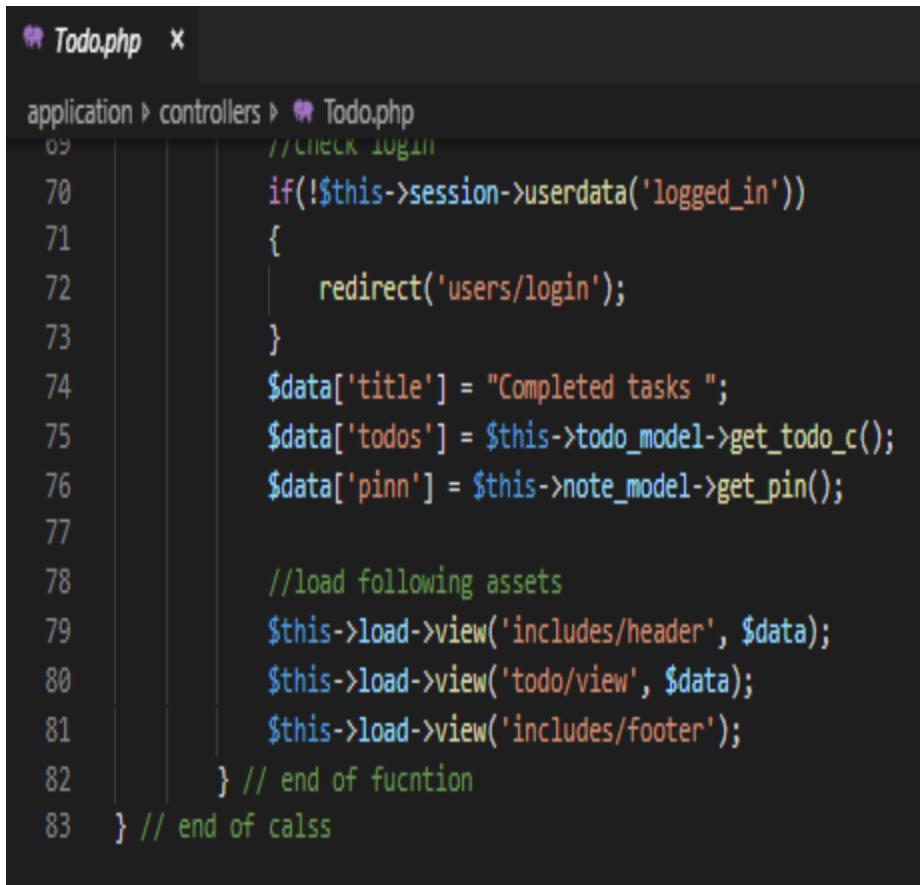


The screenshot shows a code editor window with the file name 'Todo.php' at the top. The code is written in PHP and defines three functions: add(), update(\$todoid), and view_c(). The 'add()' function handles adding a new task to the database and redirecting to the 'todo' page. It includes logic to check if a user is logged in. The 'update(\$todoid)' function updates an existing task and redirects to the 'todo' page, also checking for login. The 'view_c()' function, which is partially visible, likely handles displaying a list of tasks.

```
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



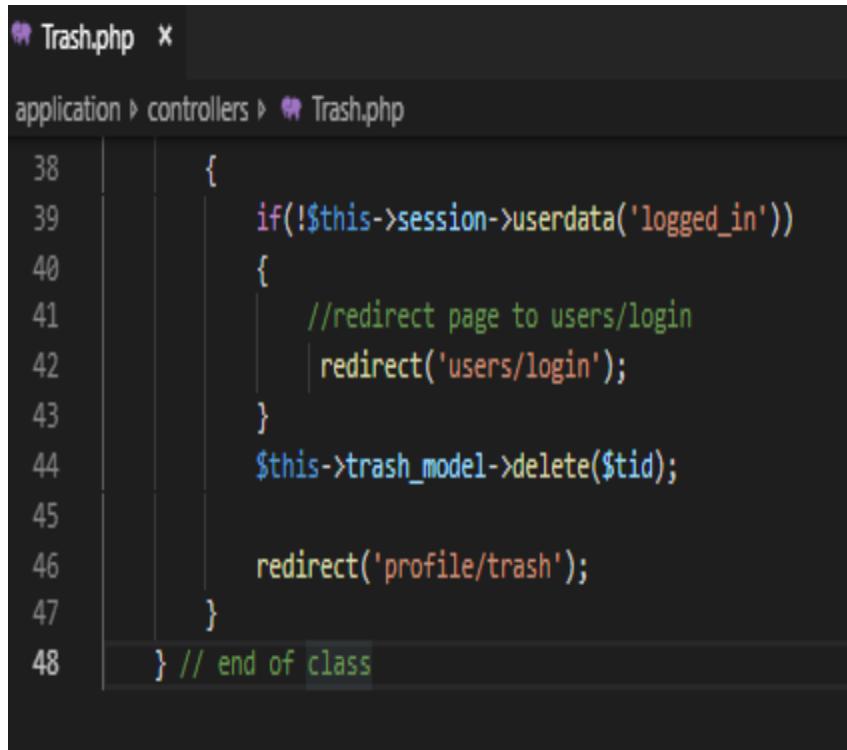
The screenshot shows a code editor window with the file 'Todo.php' open. The code is part of a task controller. It includes logic to check if the user is logged in, loads data for completed tasks and pinned notes, and then displays these using the 'load' view.

```
09     //CHECK LOGIN
10     if(!$this->session->userdata('logged_in'))
11     {
12         redirect('users/login');
13     }
14     $data['title'] = "Completed tasks ";
15     $data['todos'] = $this->todo_model->get_todo_c();
16     $data['pinn'] = $this->note_model->get_pin();
17
18     //load following assets
19     $this->load->view('includes/header', $data);
20     $this->load->view('todo/view', $data);
21     $this->load->view('includes/footer');
22 }
23 } // end of calss
```

Figure 89 code of task controller part3

```
Trash.php x
application > controllers > Trash.php
1  <!-- this is a controller of trash page-->
2  <?php
3  class Trash extends CI_Controller
4  {
5      //this function controls data from from trash view to model
6      function index()
7      {
8          //check login
9          if(!$this->session->userdata('logged_in'))
10         {
11             redirect('users/login');
12         }
13         $data['title'] = 'Trash';
14         $data['trash'] = $this->trash_model->view_trash();
15         $data['pinn'] = $this->note_model->get_pin();
16         $this->load->view('includes/header', $data);
17         $this->load->view('notes/trash', $data);
18         $this->load->view('includes/footer');
19     }
20 }
21
22 //this function receivs $id from view and sends that data to model
23 function recover($tid)
24 {
25     if(!$this->session->userdata('logged_in'))
26     {
27         //redirect page to users/login
28         redirect('users/login');
29     }
30     $this->trash_model->recover($tid);
31     $this->trash_model->delete($tid);
32
33     redirect('profile/trash');
34 }
35
36 //this function receivs $id from view and sends that data to model
37 function delete($tid)
--
```

Figure 90 code of trash controller part1



The screenshot shows a code editor window with the file 'Trash.php' open. The file is located in the 'application/controllers' directory. The code is part of a class definition:

```
38     {
39         if(!$this->session->userdata('logged_in'))
40         {
41             //redirect page to users/login
42             | redirect('users/login');
43         }
44         $this->trash_model->delete($tid);
45
46         redirect('profile/trash');
47     }
48 } // end of class
```

Figure 91 code of trash controller part2

```
users.php *  
application > controllers > users.php  
1  <!-- this is a controller class of user-->  
2  <?php  
3  class Users extends CI_Controller  
4  {  
5      //register user  
6      public function register()  
7      {  
8          $data['title'] = 'Sign Up';  
9          // setting validation on name, email, password, and password2  
10         $this->form_validation->set_rules('name', 'Name', 'required');  
11         $this->form_validation->set_rules('email', 'Email', 'required|callback_check_email_exists');  
12         $this->form_validation->set_rules('password', 'Password', 'required');  
13         $this->form_validation->set_rules('password2', 'Confirm Password', 'matches[password]');  
14         $password = $this->input->post('password');  
15         // Validate password strength  
16         $uppercase = preg_match('@[A-Z]@', $password);  
17         $lowercase = preg_match('@[a-z]@', $password);  
18         $number = preg_match('@[0-9]@', $password);  
19         $specialChars = preg_match('@[^\\w]@', $password);  
20         //start of if condition  
21         if($this->form_validation->run() === FALSE || !$uppercase || !$lowercase  
22             || !$number || !$specialChars || strlen($password) < 8)  
23         {  
24             //load following assets if above condition match  
25             $this->load->view('includes/header');  
26             $this->load->view('users/register', $data);  
27             $this->load->view('includes/footer');  
28         }  
29         else  
30         {  
31             //encrypt password  
32             $enc_password = md5($password);  
33             //sends encrypted password in user_model register function  
34             $this->user_model->register($enc_password);  
35             //set message  
36             $this->session->set_flashdata('user_registered', 'Registration complete.  
37             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  
  
75     //create session  
76     $user_data= array(  
77         'user_id'=> $user_id,  
78         'email' => $email,  
79         'password' => $password,  
80         'logged_in' => true  
81     );  
82     //stores above array data so it can be access anytime we need  
83     $this->session->set_userdata($user_data);  
84     $this->user_model->login_history_set();  
85     //set message  
86     $this->session->set_flashdata('user_loggedin', 'Login sucessfull.');//  
87     //redirecr page to notes  
88     redirect('notes');  
89  
90 }  
91 else  
92 {  
93     //set message  
94     $this->session->set_flashdata('login_failed', 'Username or password incorrect');//  
95  
96     //redurect page to users/login  
97     redirect('users/login');  
98 }  
99     // end of nested if condition  
100    // end of if condition  
101 }  
102 } // end of function  
103  
104 //Log user out  
105 public function logout()  
106 {  
107     //unset userdata(session)  
108     $this->session->unset_userdata('logged_in');  
109     $this->session->unset_userdata('user_id');  
110     $this->session->unset_userdata('email');  
111     $this->session->unset_userdata('password');
```

Figure 94 code of users controller part3

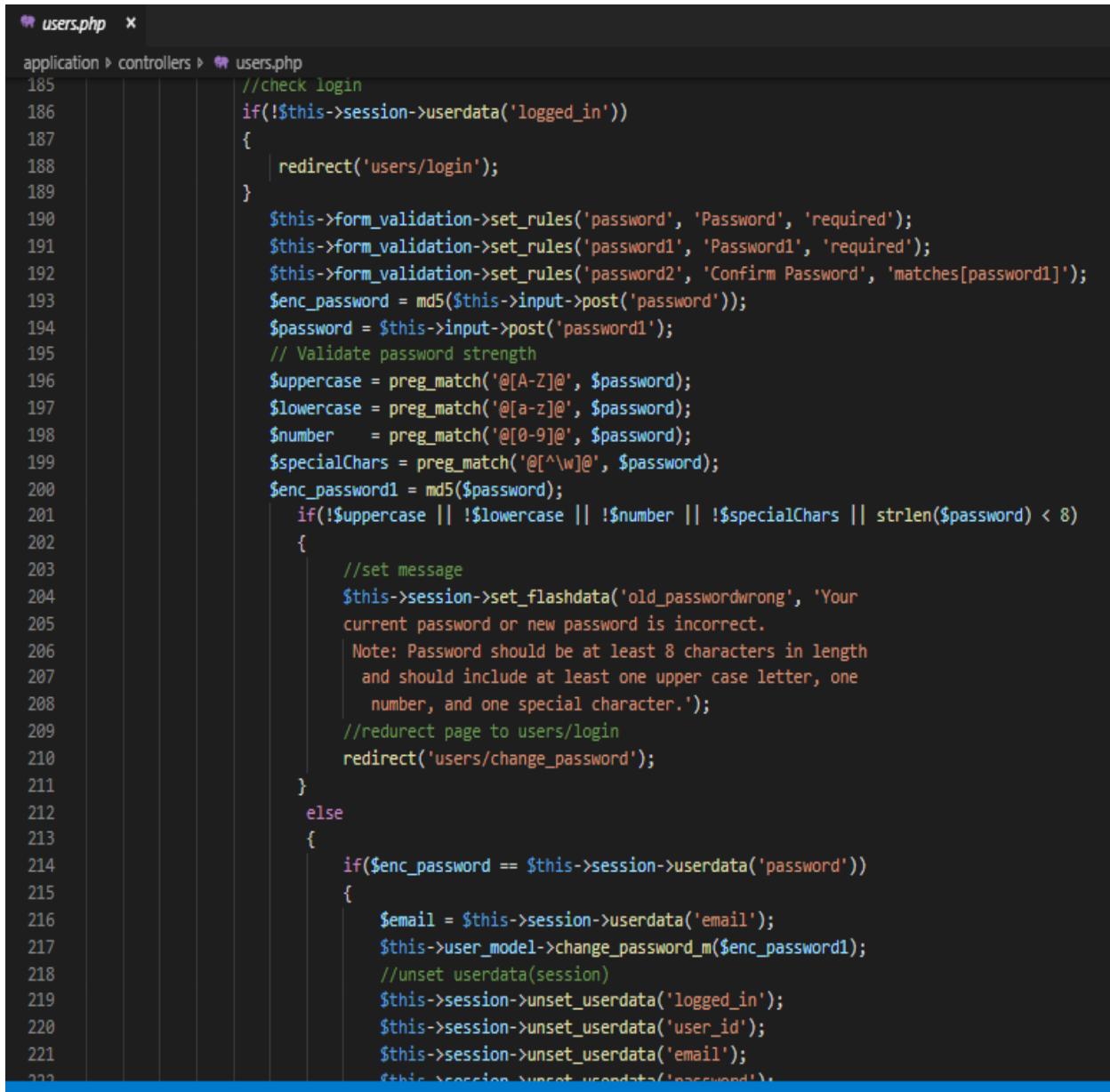
```
users.php x
application > controllers > users.php
+---+-----+-----+-----+-----+
112 |           //set message
113 |           $this->session->set_flashdata('user_logout', 'logout complete');
114 |           //redurect page to users/login
115 |           redirect('users/login');
116 |       } // end of fucnition
117 |
118 |
119 |       //check email exists or not
120 |       public function check_email_exists($email)
121 |       {
122 |           $this->form_validation->set_message('check_email_exists', 'Email is taken');
123 |           if($this->user_model->check_email_exists($email))
124 |           {
125 |               return TRUE;
126 |           }
127 |           else
128 |           {
129 |               return FALSE;
130 |           }
131 |       } // end of fucnition
132 |
133 |       //function that calls model function when called
134 |       public function edit_profile()
135 |       {
136 |           //check login
137 |           if(!$this->session->userdata('logged_in'))
138 |           {
139 |               //redirecrt page to users/login
140 |               redirect('users/login');
141 |           }
142 |           $data['user']= $this->user_model->get_user();
143 |           $data['title'] = 'Edit Profile';
144 |           $data['pin'] = $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 fucnition
```

Figure 95 code of users controller part4

```
users.php x

application > controllers > users.php
148     } // end of function
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['pin'] = $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



The screenshot shows a code editor window with the file 'users.php' open. The code is part of a controller for a web application. It includes logic for password validation, session handling, and user data modification. The code is well-structured with comments explaining the purpose of various sections.

```
application > controllers > users.php
185     //check login
186     if(!$this->session->userdata('logged_in'))
187     {
188         | redirect('users/login');
189     }
190     $this->form_validation->set_rules('password', 'Password', 'required');
191     $this->form_validation->set_rules('password1', 'Password1', 'required');
192     $this->form_validation->set_rules('password2', 'Confirm Password', 'matches[password1]');
193     $enc_password = md5($this->input->post('password'));
194     $password = $this->input->post('password1');
195     // Validate password strength
196     $uppercase = preg_match('@[A-Z]@', $password);
197     $lowercase = preg_match('@[a-z]@', $password);
198     $number   = preg_match('@[0-9]@', $password);
199     $specialChars = preg_match('@[^\\w]@', $password);
200     $enc_password1 = md5($password);
201     if(!$uppercase || !$lowercase || !$number || !$specialChars || strlen($password) < 8)
202     {
203         //set message
204         $this->session->set_flashdata('old_passwordwrong', 'Your
205         current password or new password is incorrect.
206         Note: Password should be at least 8 characters in length
207         and should include at least one upper case letter, one
208         number, and one special character.');
209         //redirect page to users/login
210         redirect('users/change_password');
211     }
212     else
213     {
214         if($enc_password == $this->session->userdata('password'))
215         {
216             $email = $this->session->userdata('email');
217             $this->user_model->change_password_m($enc_password1);
218             //unset userdata(session)
219             $this->session->unset_userdata('logged_in');
220             $this->session->unset_userdata('user_id');
221             $this->session->unset_userdata('email');
222             $this->session->unset_userdata('password');
```

Figure 97 code of users controller part6

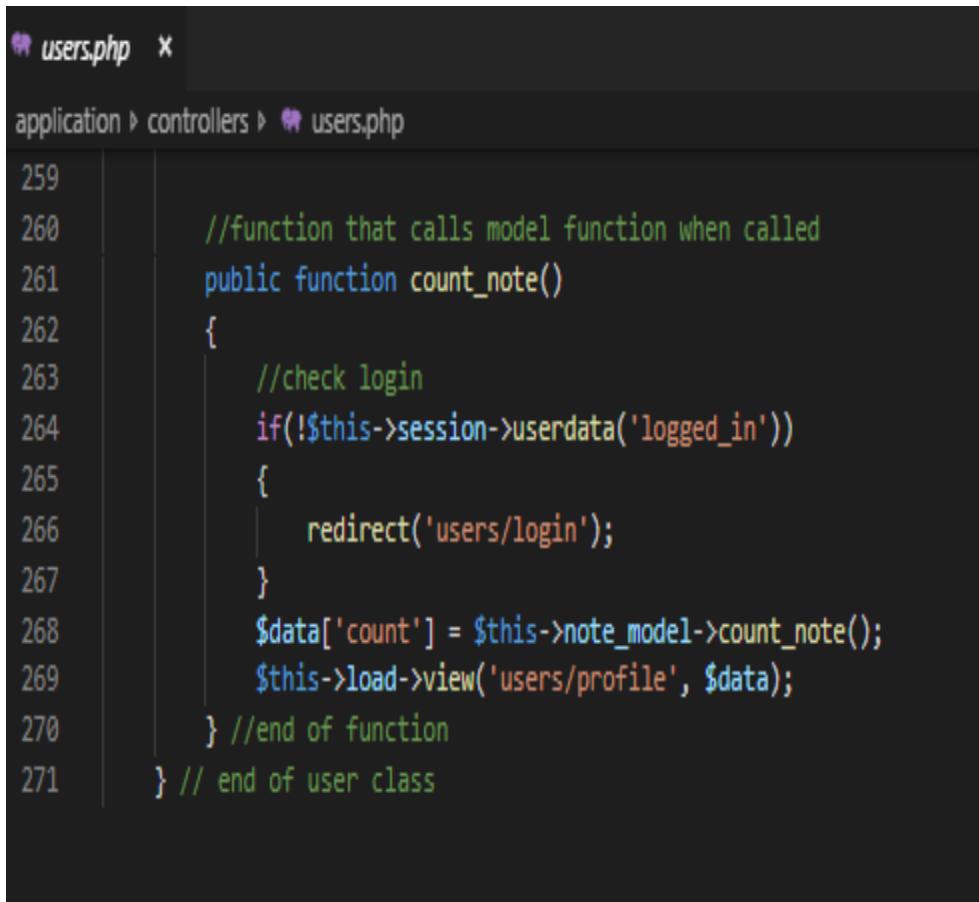


The screenshot shows a code editor window with the file 'users.php' open. The code is written in PHP and contains two main functions: 'change_password' and 'login_history'. The 'change_password' function handles password updates, setting session messages and redirecting based on success or failure. The 'login_history' function checks if the user is logged in, sets the page title, and loads the login history view.

```
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
```

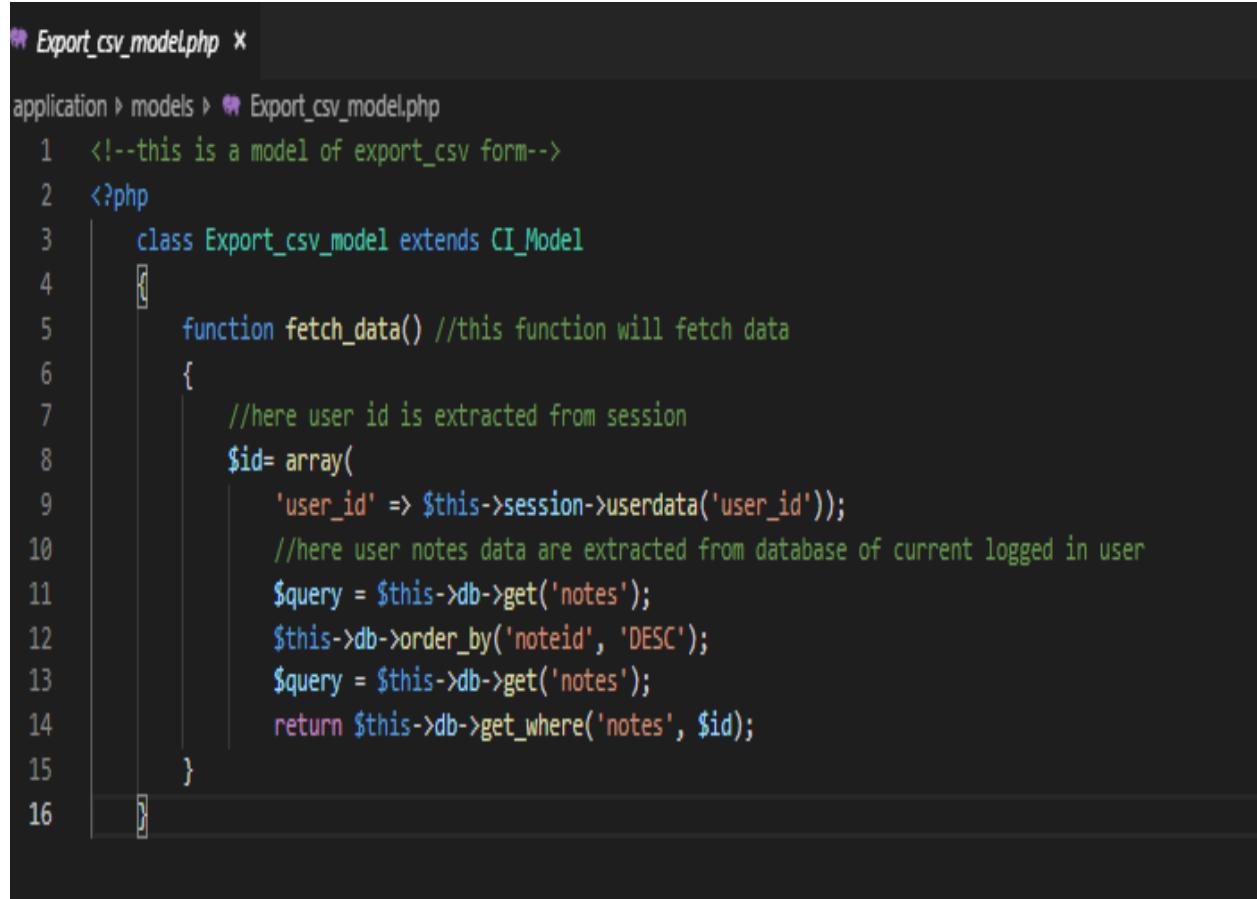
Figure 98 code of users controller part7



The screenshot shows a code editor window with the file 'users.php' open. The code is part of a controller class. It includes a function 'count_note()' which checks if the user is logged in before proceeding. If not, it redirects to the login page. Otherwise, it counts the notes and loads the profile view with the data.

```
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



The screenshot shows a code editor window with the file 'Export_csv_model.php' open. The code is written in PHP and defines a model class for fetching data from a database. The code includes comments explaining the purpose of each section.

```
1  <!--this is a model of export_csv form-->
2  <?php
3  class Export_csv_model extends CI_Model
4  {
5      function fetch_data() //this function will fetch data
6      {
7          //here user id is extracted from session
8          $id= array(
9              'user_id' => $this->session->userdata('user_id'));
10         //here user notes data are extracted from database of current logged in user
11         $query = $this->db->get('notes');
12         $this->db->order_by('noteid', 'DESC');
13         $query = $this->db->get('notes');
14         return $this->db->get_where('notes', $id);
15     }
16 }
```

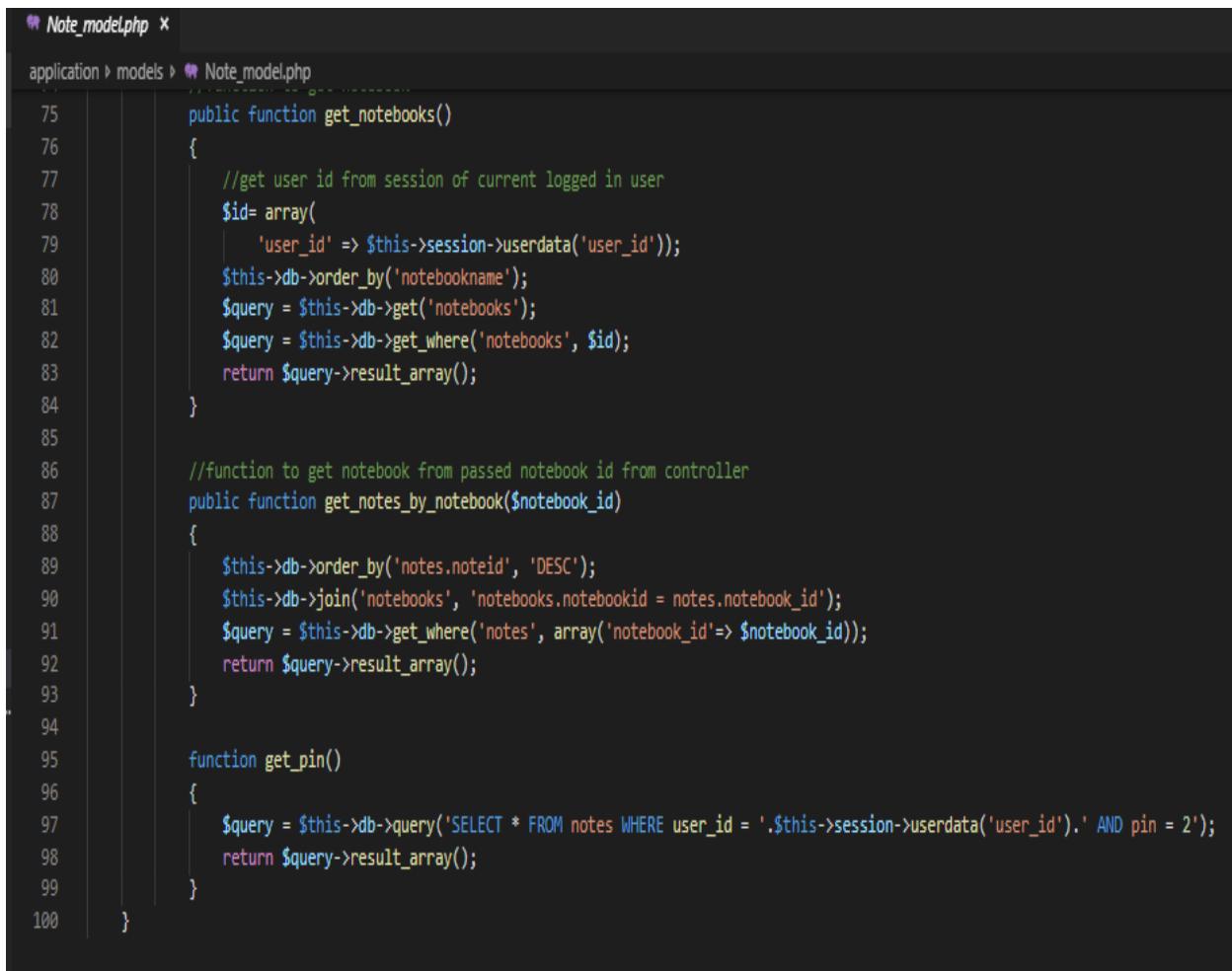
Figure 100 code of export model

```
* Note_model.php *
application > models > * Note_model.php
1  <!--this is a model of note-->
2  <?php
3      class Note_model extends CI_Model
4      {
5          //defulat constructor which load data from database
6          public function __constructor()
7          {
8              $this->load->database();
9          }
10
11         //function to get note from database
12         public function get_notes($slug = FALSE, $limit = FALSE, $offset = FALSE)
13         {
14             if($limit)
15             {
16                 $this->db->limit($limit, $offset);
17             }
18             //user id is taken from session
19             $user_id = $this->session->userdata('user_id');
20             if($slug === FALSE)
21             {
22                 $id= array(
23                     'user_id' => $this->session->userdata('user_id'));
24                     //all notes are selected from database of current logged in user
25                     $this->db->order_by('noteid', 'DESC');
26                     $query = $this->db->get_where('notes', $id);
27                     return $query->result_array(); //returns $query in result_array()
28             }
29             $query = $this->db->get_where('notes', array('slug'=> $slug)); //starts query
30             return $query->row_array();
31         }
32
33         //create new note function where $post_image as parameter
34         public function add_note($post_image)
35         {
36             $slug = url_title($this->input->post('notename'));
37             //array containing many data
```

Figure 101 code of note model part1

```
File 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



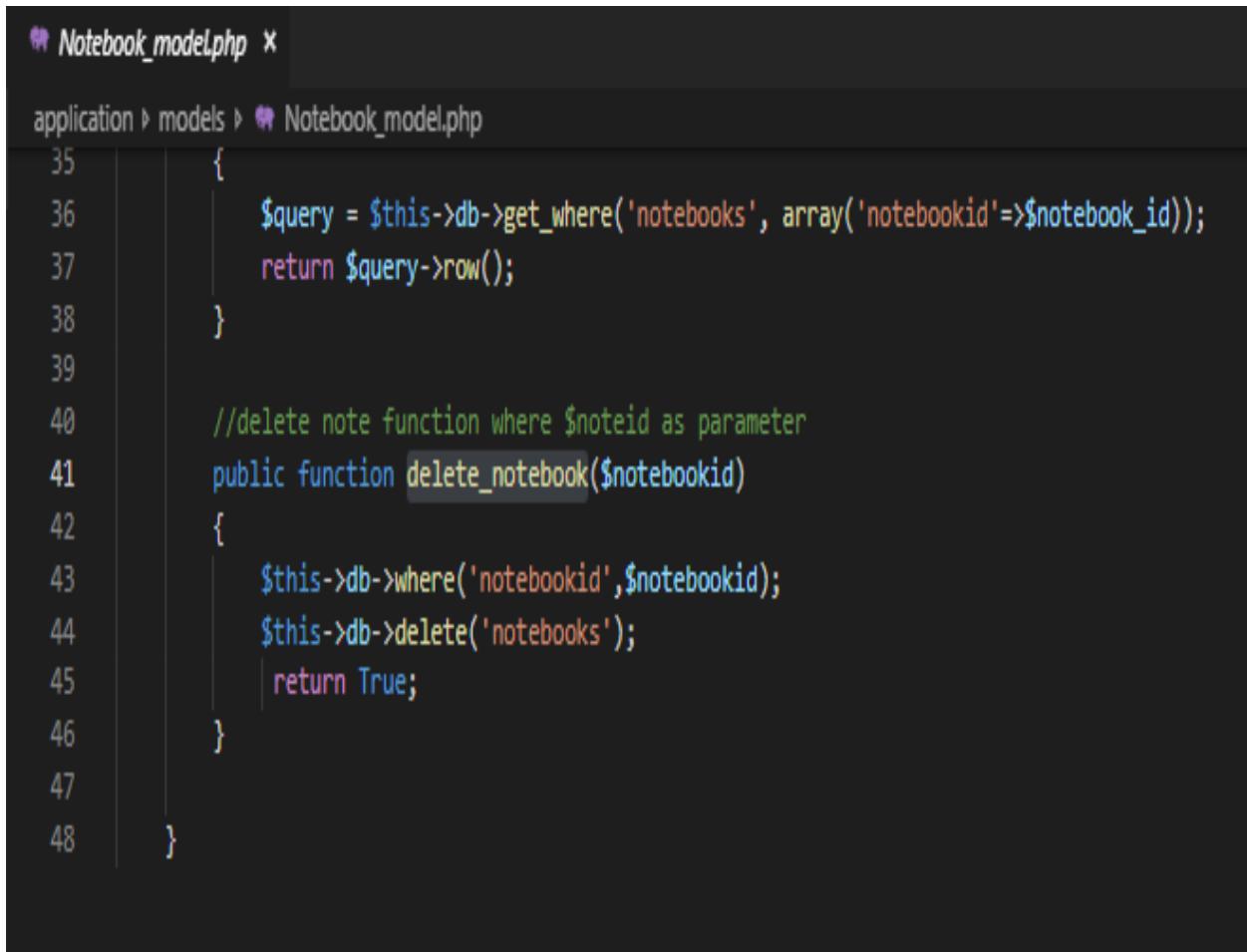
The screenshot shows a code editor window with the file `Note_model.php` open. The code is written in PHP and defines three methods: `get_notebooks()`, `get_notes_by_notebook($notebook_id)`, and `get_pin()`. The code uses MySQLi database queries to retrieve data from tables `notes` and `notebooks`.

```
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  
1   <!--this is a model of notebook-->  
2   <?php  
3       class Notebook_model extends CI_Model  
4       {  
5           //defulat constructor which load data from database  
6           public function __construct()  
7           {  
8               $this->load->database();  
9           }  
10  
11           //function to get notebook from database  
12           public function get_notebooks()  
13           {  
14               $id= array(  
15                   'user_id' => $this->session->userdata('user_id'));  
16               $this->db->order_by('notebookname');  
17               $query = $this->db->get('notebooks');  
18               $query = $this->db->get_where('notebooks', $id);  
19               return $query->result_array();  
20           }  
21  
22           //function to add new notebook  
23           public function add_notebook()  
24           {  
25               $data = array(  
26                   'notebookname'=> $this->input->post('name'),  
27                   'user_id' => $this->session->userdata('user_id')  
28               );  
29  
30               return $this->db->insert('notebooks', $data);  
31           }  
32  
33           //function to get notebook from databasse and get notebookid from controller  
34           public function get_notebook($notebook_id)  
35           {  
36               $query = $this->db->get_where('notebooks', array('notebookid'=>$notebook_id));  
37               return $query->row();  
38           }  
39       }  
40   }  
41 }  
42 }
```

Figure 104 code of notebook model part1



The screenshot shows a code editor window with the title "Notebook_model.php". The file path is "application > models > Notebook_model.php". The code is as follows:

```
35     {
36         $query = $this->db->get_where('notebooks', array('notebookid'=>$notebook_id));
37         return $query->row();
38     }
39
40     //delete note function where $noteid as parameter
41     public function delete_notebook($notebookid)
42     {
43         $this->db->where('notebookid',$notebookid);
44         $this->db->delete('notebooks');
45         return True;
46     }
47
48 }
```

Figure 105 code of notebook model part2

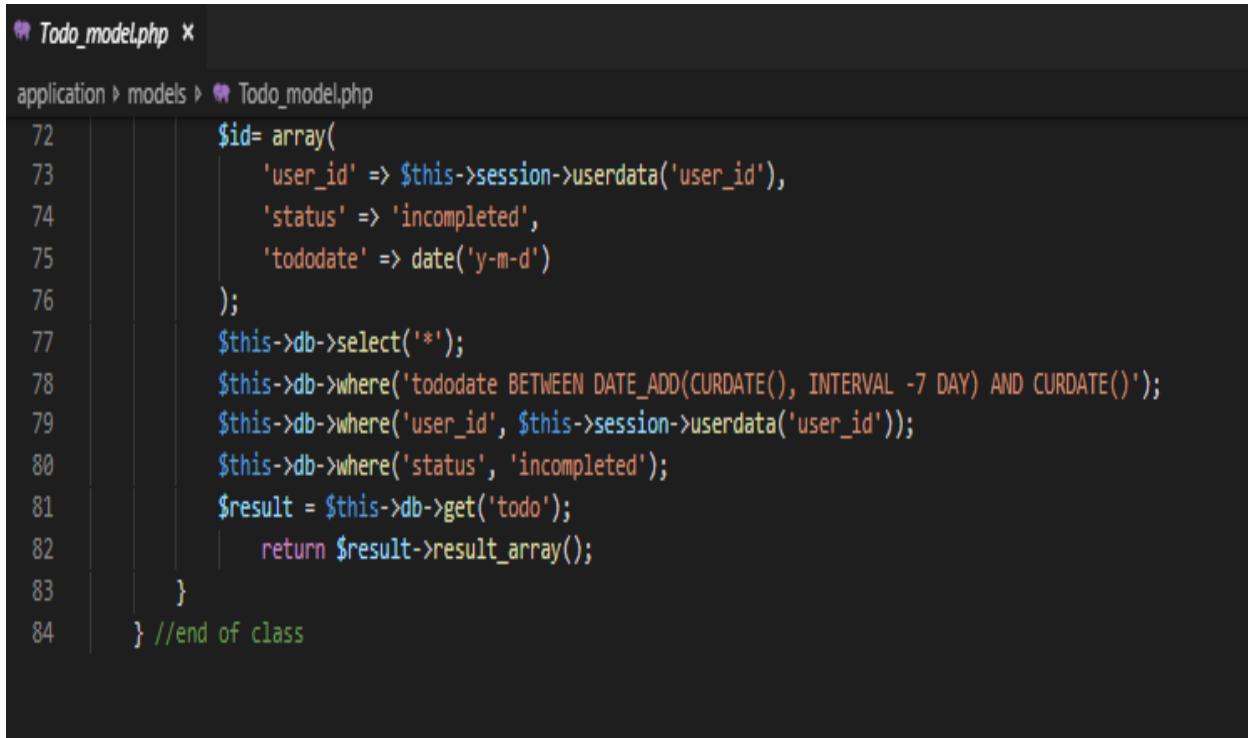
```
Todo_model.php ×

application › models › Todo_model.php
1  <!--this is a model of todo-->
2  <?php
3  class Todo_model extends CI_Model
4  {
5      public function __construct()
6      {
7          $this->load->database();
8      }
9
10     //function to add tasks
11     public function add_todo()
12     {
13         $data = array(
14             'todoname'=> $this->input->post('todoname'),
15             'user_id' => $this->session->userdata('user_id'),
16             'tododate' => $this->input->post('date')
17         );
18         return $this->db->insert('todo', $data);
19     }
20
21     //function to set tasks from database
22     public function get_todo()
23     {
24         $id= array(
25             'user_id' => $this->session->userdata('user_id'),
26             'status' => 'incompleted',
27             'tododate' => date('y-m-d')
28         );
29         $this->db->order_by('todoid', 'DESC');
30         $query = $this->db->get_where('todo', $id);
31         return $query->result_array();
32     }
33
34     //function to get tasks
35     public function get_todo_c()
36     {
37         $id= array(
```

Figure 106 code of todo model part1

```
Todo_model.php *
application > models > Todo_model.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     {
72         //
```

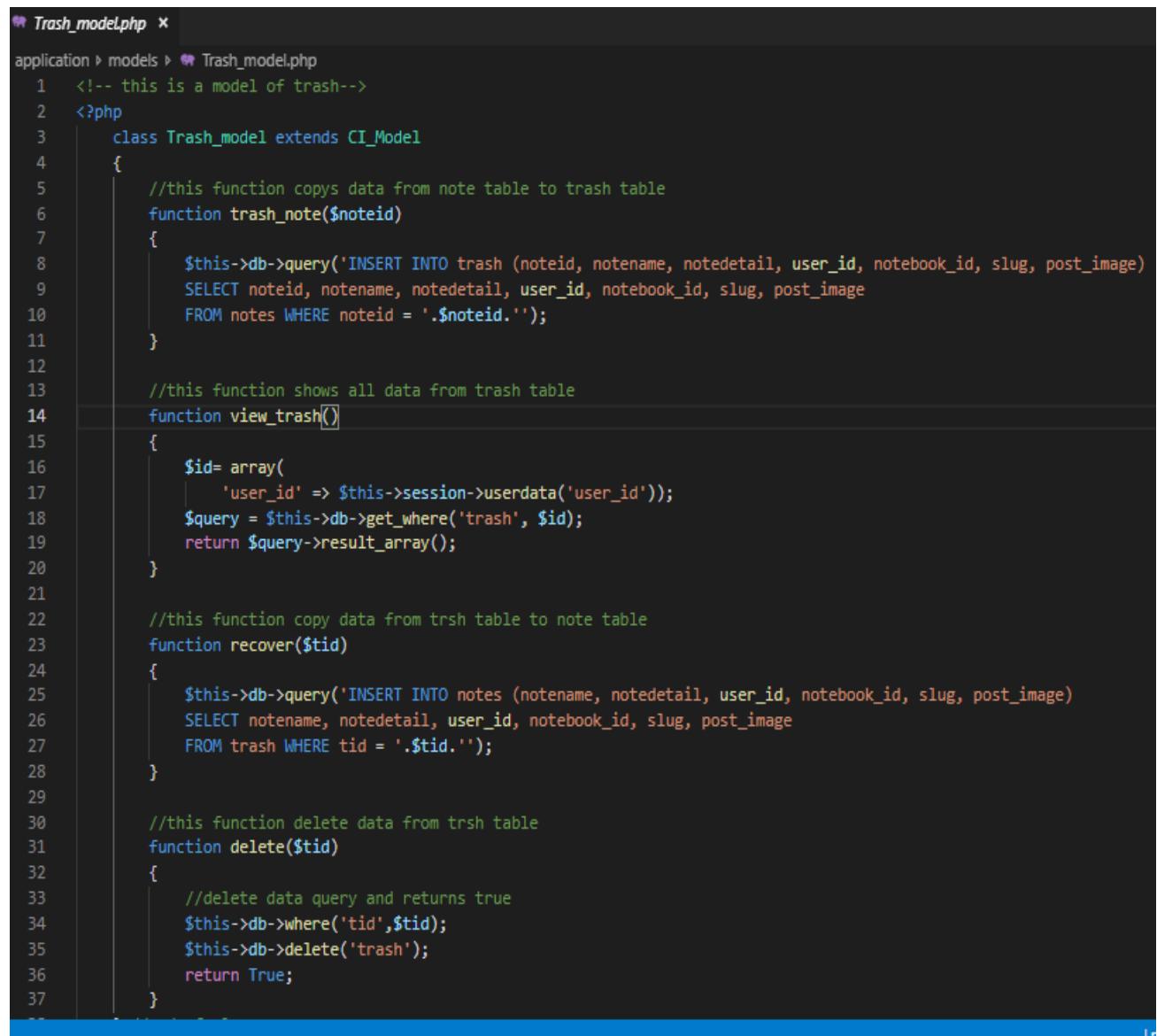
Figure 107 code of todo model part2



The screenshot shows a code editor window with the title "Todo_model.php". The file path is "application > models > Todo_model.php". The code is part of a class definition:

```
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



The screenshot shows a code editor window with the file 'Trash_model.php' open. The file is located in the 'application > models' directory. The code is written in PHP and defines a class 'Trash_model' that extends 'CI_Model'. The class contains four functions: 'trash_note', 'view_trash', 'recover', and 'delete'. The 'trash_note' function inserts data from the 'notes' table into the 'trash' table. The 'view_trash' function retrieves all data from the 'trash' table for a specific user. The 'recover' function inserts data from the 'trash' table back into the 'notes' table. The 'delete' function deletes data from the 'trash' table.

```
1  <!-- this is a model of trash-->
2  <?php
3  class Trash_model extends CI_Model
4  {
5      //this function copy's data from note table to trash table
6      function trash_note($noteid)
7      {
8          $this->db->query('INSERT INTO trash (noteid, notename, notedetail, user_id, notebook_id, slug, post_image)
9              SELECT noteid, notename, notedetail, user_id, notebook_id, slug, post_image
10             FROM notes WHERE noteid = '.$noteid.'');
11      }
12
13      //this function shows all data from trash table
14      function view_trash()
15      {
16          $id= array(
17              'user_id' => $this->session->userdata('user_id'));
18          $query = $this->db->get_where('trash', $id);
19          return $query->result_array();
20      }
21
22      //this function copy data from trsh table to note table
23      function recover($tid)
24      {
25          $this->db->query('INSERT INTO notes (notename, notedetail, user_id, notebook_id, slug, post_image)
26              SELECT notename, notedetail, user_id, notebook_id, slug, post_image
27             FROM trash WHERE tid = '.$tid.'');
28      }
29
30      //this function delete data from trsh table
31      function delete($tid)
32      {
33          //delete data query and returns true
34          $this->db->where('tid',$tid);
35          $this->db->delete('trash');
36          return True;
37      }
38  }
```

Figure 109 code of trash model

```
user_model.php *
application > models > user_model.php
1  <!--this is a model of user-->
2  <?php
3  class User_model extends CI_Model
4  {
5      public function register($enc_password)
6      {
7          //user data array
8          $data = array(
9              'name'=> $this->input->post('name'),
10             'email'=> $this->input->post('email'),
11             'password'=> $enc_password
12         );
13         //insert user
14         return $this->db->insert('users', $data);
15     }
16
17     //LOG USER IN
18     public function login($email, $password)
19     {
20         //validate
21         $this->db->where('email', $email);
22         $this->db->where('password', $password);
23
24         $result= $this->db->get('users');
25         if($result->num_rows() == 1)
26         {
27
28             return $result->row(0)->userid;
29         }
30         else
31         {
32             return FALSE;
33         }
34     }
35
36     //function to get login log from database
37     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

```
72     //function to get users
73     public function get_user()
74     {
75         $id= array(
76             'userid' => $this->session->userdata('user_id'));
77         $query = $this->db->get('users');
78         $query = $this->db->get_where('users', $id);
79         return $query->row_array(); //returns $query in result_array()
80     }
81
82
83     // function to update user
84     public function update_user()
85     {
86         $id = $this->session->userdata('user_id');
87
88         $data = array(
89             'name' => $this->input->post('name'),
90             'email' => $this->input->post('email')
91         );
92
93         $this->db->where('userid', $id);
94         return $this->db->update('users', $data);
95     }
96
97     //function to change password of user
98     public function change_password_m($enc_password1)
99     {
100         $id = $this->session->userdata('user_id');
101         $data = array(
102             'password'=> $enc_password1
103         );
104         $this->db->where('userid', $id);
105         return $this->db->update('users', $data);
106     }
107
108     //function to count notes of that particular user
109     //function to add new note
```

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  
1  <!--this is a index page of note where added notes are shown-->  
2  <!--this part presents cart section-->  
3  <div class="card card-4">  
4      <div class="card-body">  
5          <!--this shows title of this page-->  
6          <div class="row card-header">  
7              <div class="col-md-6">  
8                  <h2 class="text"><strong>Testing</strong></h2>  
9              </div>  
10         </div>  
11         <div class="container">  
12             <?php if($try):  
13                 {  
14                     echo $try;  
15                 } endif; ?>  
16  
17             <?php if($try1):  
18                 {  
19                     echo $try1;  
20                 } endif; ?>  
21  
22             <?php if($try2):  
23                 {  
24                     echo $try2;  
25                 } endif; ?>  
26  
27             <?php if($try3):  
28                 {  
29                     echo $try3;  
30                 } endif; ?>  
31  
32             <?php if($try4):  
33                 {  
34                     echo $try4;  
35                 } endif; ?>  
36
```

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
1  <?php
2  defined('BASEPATH') OR exit('No direct script access allowed');
3  class Testing extends CI_Controller
4  {
5      public function __construct()
6      {
7          parent::__construct();
8          $this->load->library('unit_test');
9      }
10
11     public function index()
12     {
13         $test = $this->check_array();
14         $expected_result = 'is_array';
15         $test_name = 'Array check';
16         $test_note = 'This is a test for checking array input and output';
17
18         $test1 = $this->pass_login('asd@asd', 'Sanamasd@12345');
19         $expected_result1 = 1;
20         $test_name1 = 'Login Test';
21         $test_note1 = 'This is a test for pass login';
22
23         $test2 = $this->fail_login('asdasd@asd', 'Sanamasd@12345');
24         $expected_result2 = True;
25         $test_name2 = 'Login Test';
26         $test_note2 = 'This is a test for fail login';
27
28         $test3 = $this->chk_email('sanam@gmail.com');
29         $expected_result3 = False;
30         $test_name3 = 'Check email';
31         $test_note3 = 'This is a test for checking email exists or not';
32
33         $test4 = $this->chk_email('test@gmail.com');
34         $expected_result4 = True;
35         $test_name4 = 'Check email';
36         $test_note4 = 'This is a test for checking email exists or not';
37     }
}
```

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

The screenshot shows a code editor window with the file name "Testing.php" at the top. The code is written in PHP and defines a class with several methods. The methods include login, fail_login, chk_email, count_notes, count_tasks, and delete_notebook. The code uses object-oriented programming with the \$this variable and various model classes like user_model and notebook_model.

```
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