Cairo University

Faculty of Computers and Artificial Intelligence

Software Engineering

Sent to: Dr. Mohamed El-Ramly

CS251

- **Assignment: A1**
- **Task: T1 &T2**
- **Date: 2025/2/25**
- **Section: S14**
- **Team Programming Language: Java**
- **Team Leader Phone Number: +20 128 696 4627**
- **Name, IDs, and E-mails:**

Name	IDs	E-Mails.
Aly El-Deen Yasser Ali	20231109	ali.el.badry.747@gmail.com
Nagham Wael Mohamed	20231189	naghamw63@gmail.com
Fatema El-Zhraa Ahmed Mohamed El-Fiky	20230280	fatmaelfeky922@gmail.com

---

Table of Content

The Process of Learning :

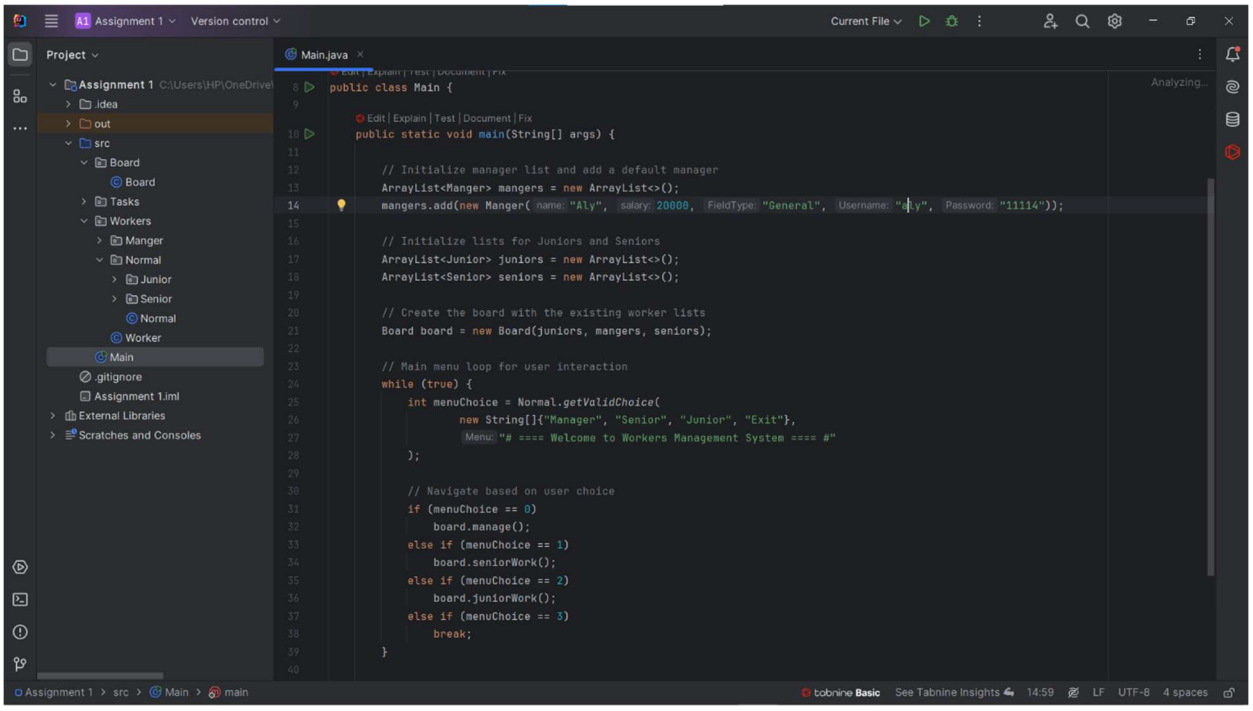
Aly El-Deen Yasser Aly:

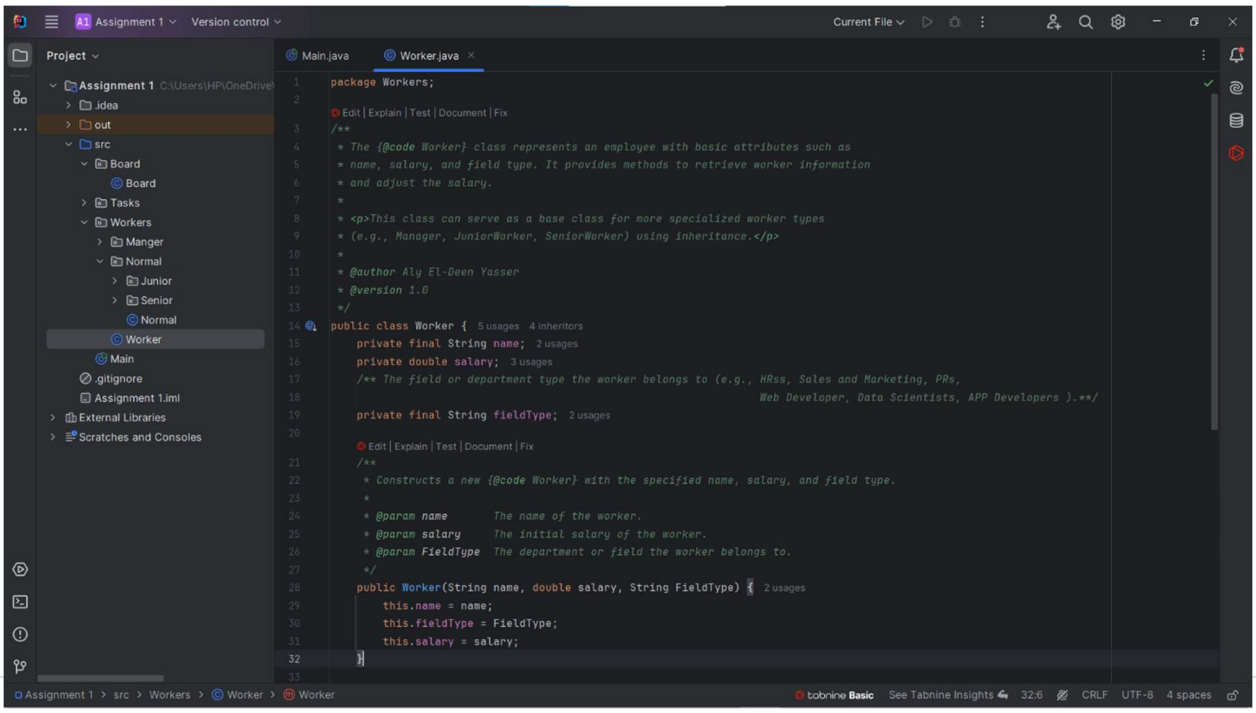
- Hours of Study: 18 hours
- Source of study https://youtube.com/playlist?list=PLCInYL3l2AajYlZGzU_LVrHdoouf8W6ZN&si=tfRlF2iNxOtJehum
- Main Logic of code:

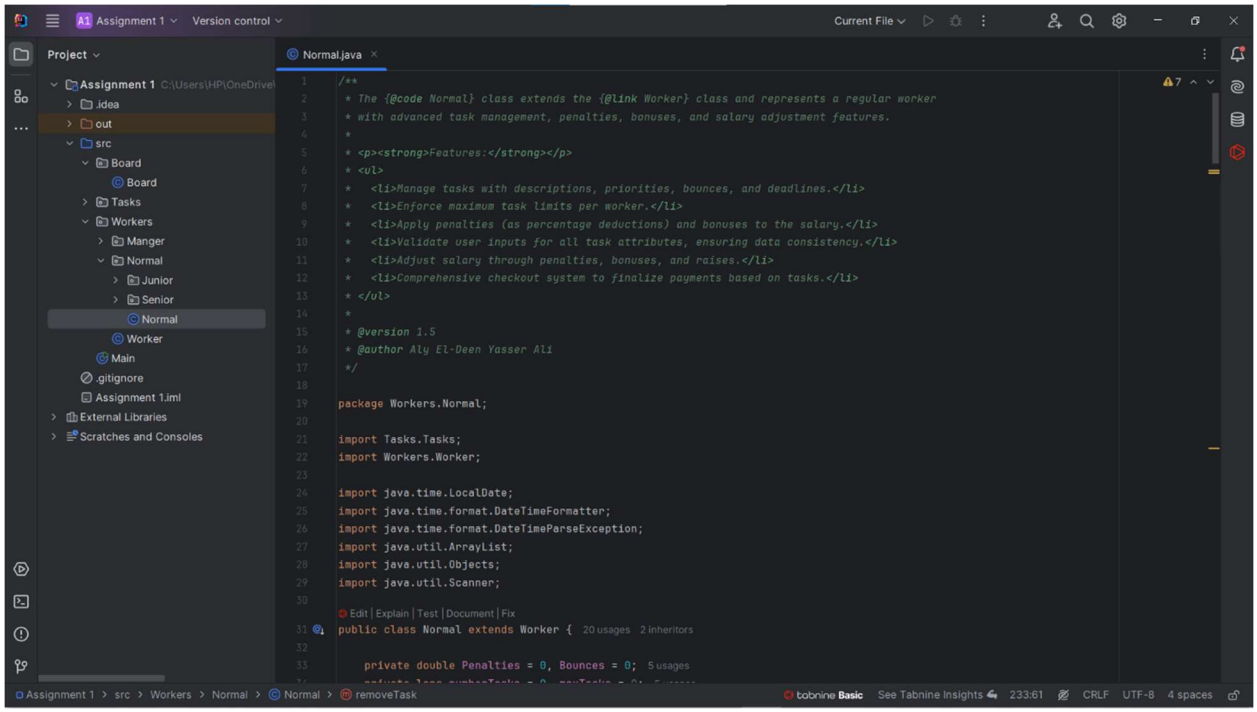
```
import java.util.ArrayList;
import Workers.Manger.Manger;
import Workers.Normal.Junior.Junior;
import Workers.Normal.Normal;
import Workers.Normal.Senior.Senior;
import Board.Board;

public class Main {

    public static void main(String[] args) {

        // Initialize manager list and add a default manager
        ArrayList<Manger> mangers = new ArrayList<>();
        mangers.add(new Manger("Aly", 20000, "General", "aly",
"11114"));

        // Initialize lists for Juniors and Seniors
        ArrayList<Junior> juniors = new ArrayList<>();
        ArrayList<Senior> seniors = new ArrayList<>();

        // Create the board with the existing worker lists
        Board board = new Board(juniors, mangers, seniors);

        // Main menu loop for user interaction
        while (true) {
            int menuChoice = Normal.getValidChoice(
                    new String[]{"Manager", "Senior", "Junior",
"Exit"},
                    "# ==== Welcome to Workers Management System ====
#"
            );

            // Navigate based on user choice
            if (menuChoice == 0)
                board.manage();
            else if (menuChoice == 1)
                board.seniorWork();
            else if (menuChoice == 2)
```

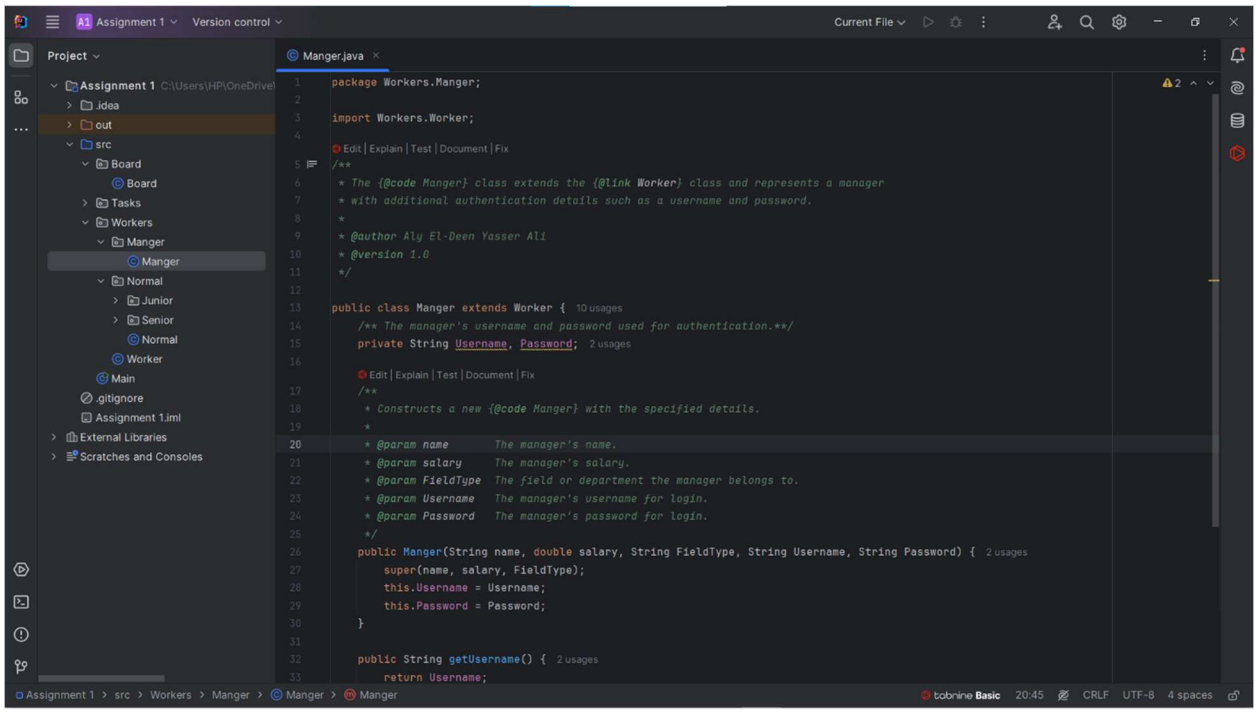
```
                board.juniorWork();
            else if (menuChoice == 3)
                break;
        }

        System.out.println("\nThanks For Using Our Program");
    }
}
```

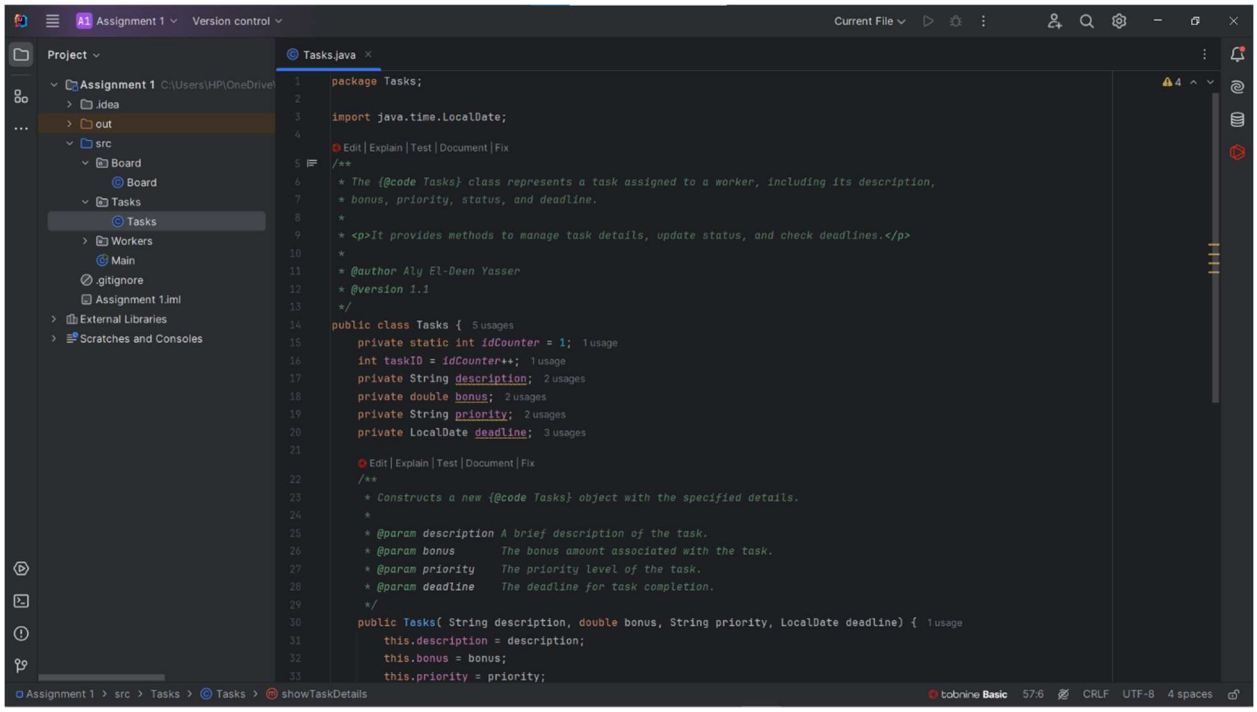
- Images For the code:

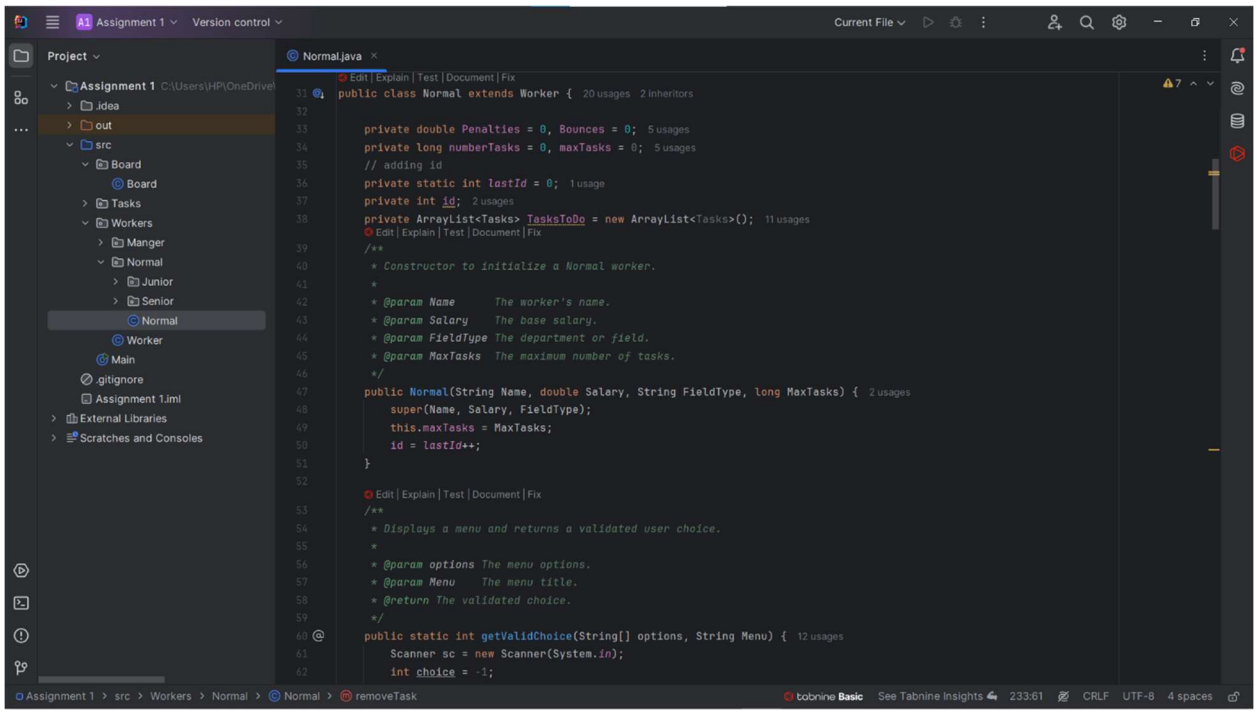


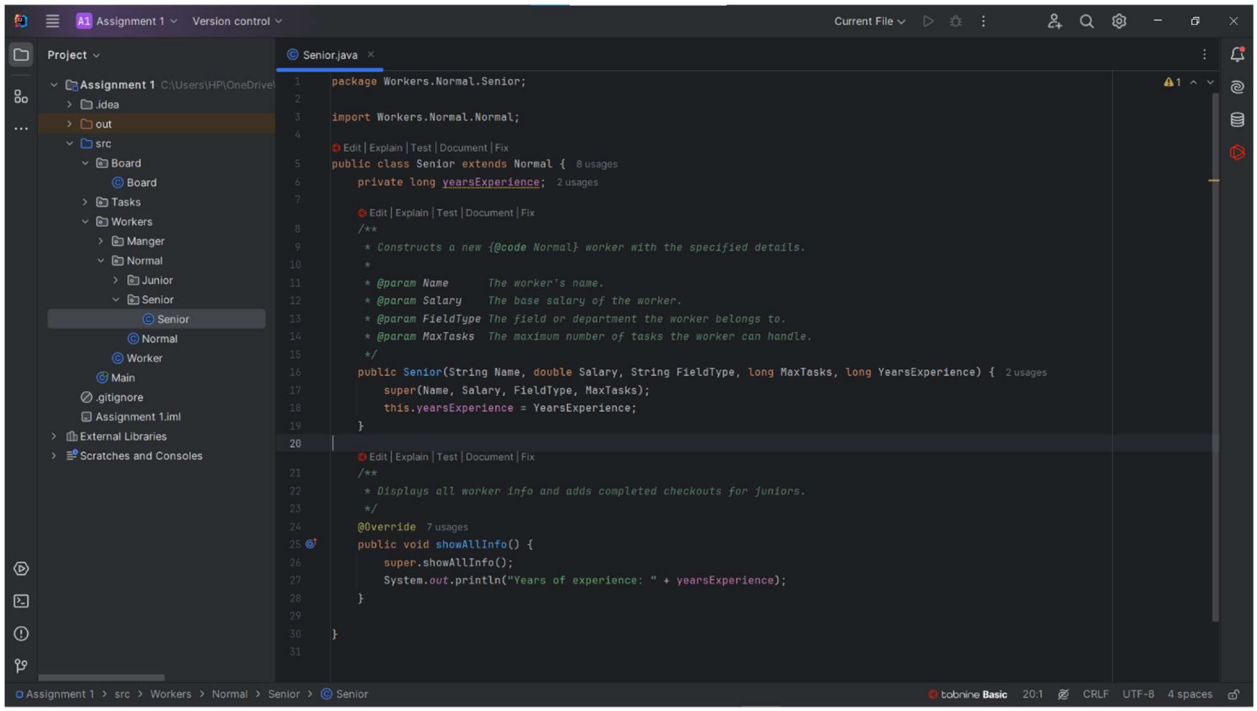


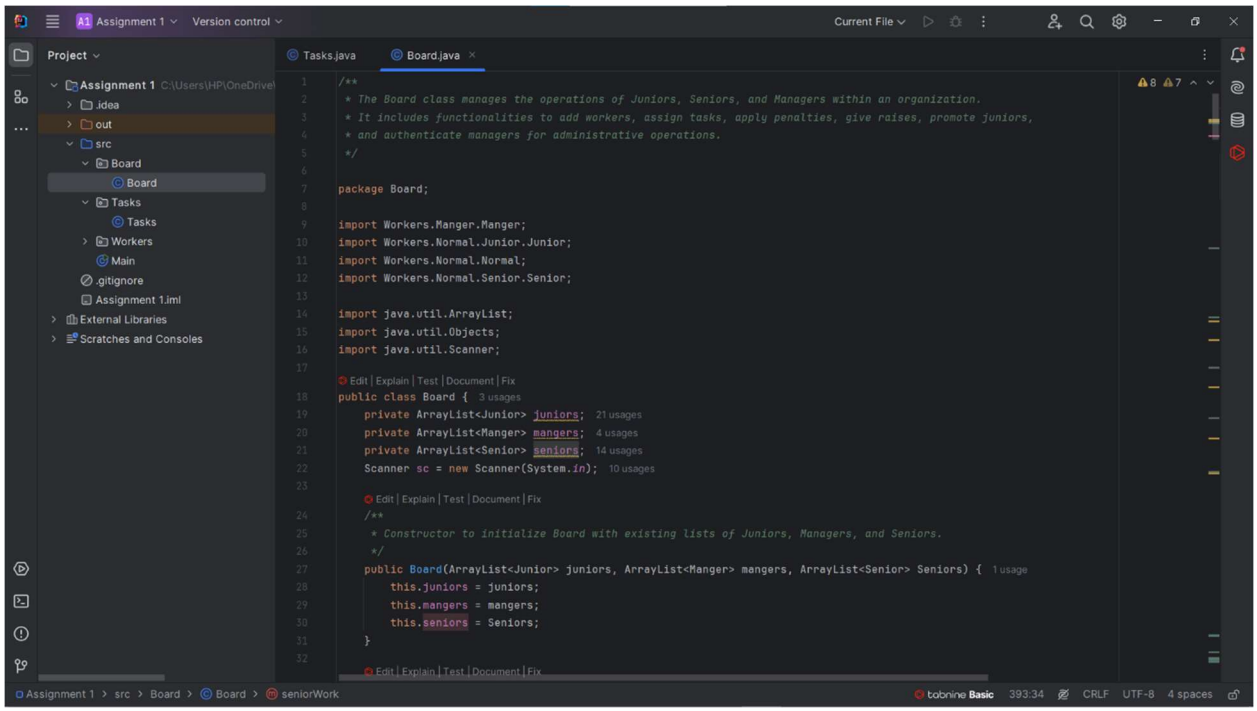

Assignment 1
Version control
Current File
Project
Normal.java
Assignment 1 C:\Users\HP\OneDrive\
.idea
out
src
Board
Board
Tasks
Workers
Manger
Normal
Junior
Senior
Normal
Worker
Main
.gitignore
Assignment 1.iml
External Libraries
Scratches and Consoles
/**
* The {@code Normal} class extends the {@link Worker} class and represents a regular worker
* with advanced task management, penalties, bonuses, and salary adjustment features.
*
* Features:
*
* Manage tasks with descriptions, priorities, bounces, and deadlines.
* Enforce maximum task limits per worker.
* Apply penalties (as percentage deductions) and bonuses to the salary.
* Validate user inputs for all task attributes, ensuring data consistency.
* Adjust salary through penalties, bonuses, and raises.
* Comprehensive checkout system to finalize payments based on tasks.
*
*
* @version 1.5
* @author Aly El-Deen Yasser Ali
*/
package Workers.Normal;
import Tasks.Tasks;
import Workers.Worker;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;
Edit | Explain | Test | Document | Fix
public class Normal extends Worker { 20 usages 2 inheritors
private double Penalties = 0, Bounces = 0; 5 usages
Assignment 1 > src > Workers > Normal > Normal > removeTask
tabnine Basic
See Tabnine Insights
233:61
CRLF
UTF-8
4 spaces

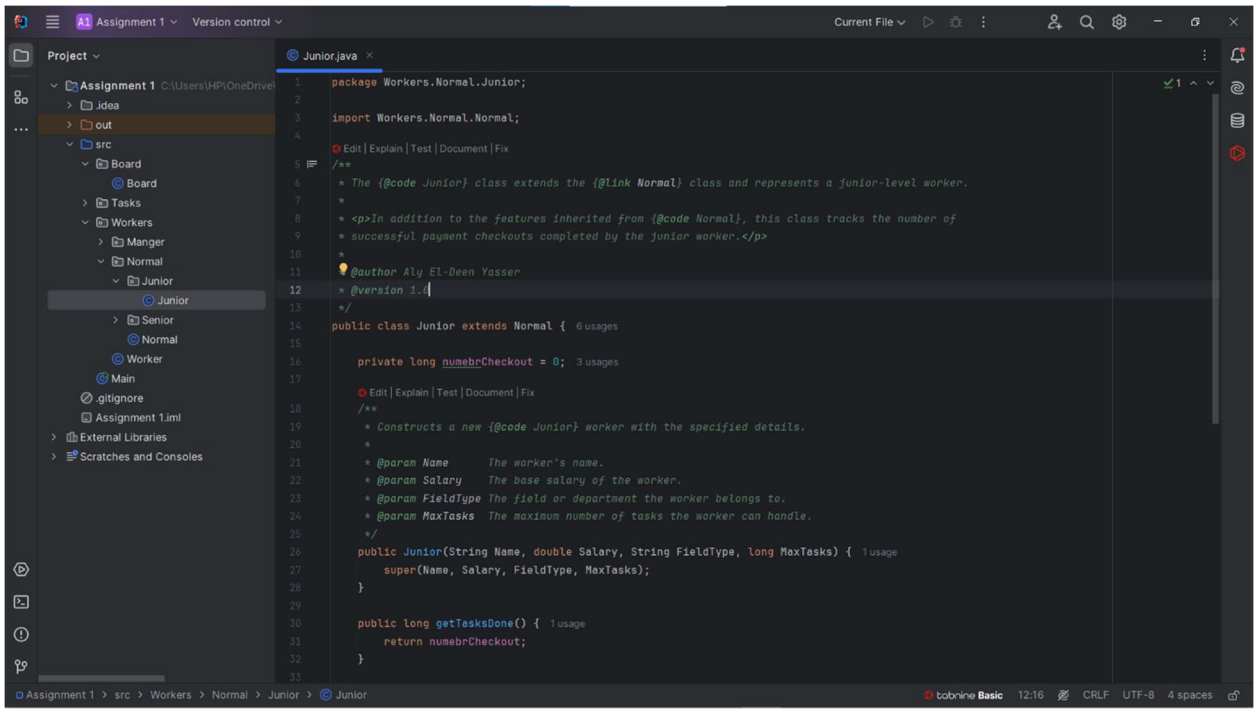

Assignment 1
Version control
Current File
Project
Manger.java
Assignment 1 C:\Users\HP\OneDrive\
.idea
out
src
Board
Board
Tasks
Workers
Manger
Manger
Normal
Junior
Senior
Normal
Worker
Main
.gitignore
Assignment 1.iml
External Libraries
Scratches and Consoles
package Workers.Manger;
import Workers.Worker;
Edit | Explain | Test | Document | Fix
/**
* The {@code Manger} class extends the {@link Worker} class and represents a manager
* with additional authentication details such as a username and password.
*
* @author Aly El-Deen Yasser Ali
* @version 1.0
*/
public class Manger extends Worker { 10 usages
/** The manager's username and password used for authentication.**/
private String Username, Password; 2 usages
Edit | Explain | Test | Document | Fix
/**
* Constructs a new {@code Manger} with the specified details.
*
* @param name The manager's name.
* @param salary The manager's salary.
* @param FieldType The field or department the manager belongs to.
* @param Username The manager's username for login.
* @param Password The manager's password for login.
*/
public Manger(String name, double salary, String FieldType, String Username, String Password) { 2 usages
super(name, salary, FieldType);
this.Username = Username;
this.Password = Password;
public String getUsername() { 2 usages
return Username;
Assignment 1 > src > Workers > Manger > Manger > Manger
tabnine Basic
20:45
CRLF
UTF-8
4 spaces


Assignment 1
Version control
Current File
Project
Tasks.java
Assignment 1 C:\Users\HP\OneDrive\
.idea
out
src
Board
Board
Tasks
Tasks
Workers
Main
.gitignore
Assignment 1.iml
External Libraries
Scratches and Consoles
package Tasks;
import java.time.LocalDate;
Edit | Explain | Test | Document | Fix
/**
* The {@code Tasks} class represents a task assigned to a worker, including its description,
* bonus, priority, status, and deadline.
*
* It provides methods to manage task details, update status, and check deadlines.
*
* @author Aly El-Deen Yasser
* @version 1.1
*/
public class Tasks { 5 usages
private static int idCounter = 1; 1 usage
int taskID = idCounter++; 1 usage
private String description; 2 usages
private double bonus; 2 usages
private String priority; 2 usages
private LocalDate deadline; 3 usages
Edit | Explain | Test | Document | Fix
/**
* Constructs a new {@code Tasks} object with the specified details.
*
* @param description A brief description of the task.
* @param bonus The bonus amount associated with the task.
* @param priority The priority level of the task.
* @param deadline The deadline for task completion.
*/
public Tasks(String description, double bonus, String priority, LocalDate deadline) { 1 usage
this.description = description;
this.bonus = bonus;
this.priority = priority;
Assignment 1 › src › Tasks › Tasks › showTaskDetails
tabnine Basic
57:6
CRLF
UTF-8
4 spaces


Assignment 1
Version control
Current File
Project
Normal.java
Assignment 1 C:\Users\HP\OneDrive\
.idea
out
src
Board
Board
Tasks
Workers
Manger
Normal
Junior
Senior
Normal
Worker
Main
.gitignore
Assignment 1.iml
External Libraries
Scratches and Consoles
Edit | Explain | Test | Document | Fix
public class Normal extends Worker { 20 usages 2 inheritors
private double Penalties = 0, Bounces = 0; 5 usages
private long numberTasks = 0, maxTasks = 0; 5 usages
// adding id
private static int lastId = 0; 1 usage
private int id; 2 usages
private ArrayList<Tasks> TasksToDo = new ArrayList<Tasks>(); 11 usages
Edit | Explain | Test | Document | Fix
/**
* Constructor to initialize a Normal worker.
*
* @param Name The worker's name.
* @param Salary The base salary.
* @param FieldType The department or field.
* @param MaxTasks The maximum number of tasks.
*/
public Normal(String Name, double Salary, String FieldType, long MaxTasks) { 2 usages
super(Name, Salary, FieldType);
this.maxTasks = MaxTasks;
id = lastId++;
}
Edit | Explain | Test | Document | Fix
/**
* Displays a menu and returns a validated user choice.
*
* @param options The menu options.
* @param Menu The menu title.
* @return The validated choice.
*/
public static int getValidChoice(String[] options, String Menu) { 12 usages
Scanner sc = new Scanner(System.in);
int choice = -1;
Assignment 1 › src › Workers › Normal › Normal › removeTask
tabnine Basic
See Tabnine Insights
233:61
CRLF
UTF-8
4 spaces


Assignment 1
Version control
Current File
Project
Senior.java
Assignment 1 C:\Users\HP\OneDrive\
.idea
out
src
Board
Board
Tasks
Workers
Manger
Normal
Junior
Senior
Senior
Normal
Worker
Main
.gitignore
Assignment 1.iml
External Libraries
Scratches and Consoles
package Workers.Normal.Senior;
import Workers.Normal.Normal;
Edit | Explain | Test | Document | Fix
public class Senior extends Normal { 8 usages
private long yearsExperience; 2 usages
Edit | Explain | Test | Document | Fix
/**
* Constructs a new {@code Normal} worker with the specified details.
*
* @param Name The worker's name.
* @param Salary The base salary of the worker.
* @param FieldType The field or department the worker belongs to.
* @param MaxTasks The maximum number of tasks the worker can handle.
*/
public Senior(String Name, double Salary, String FieldType, long MaxTasks, long YearsExperience) { 2 usages
super(Name, Salary, FieldType, MaxTasks);
this.yearsExperience = YearsExperience;
}
Edit | Explain | Test | Document | Fix
/**
* Displays all worker info and adds completed checkouts for juniors.
*/
@Override 7 usages
public void showAllInfo() {
super.showAllInfo();
System.out.println("Years of experience: " + yearsExperience);
}
}
Assignment 1 > src > Workers > Normal > Senior > Senior
tabnine Basic
20:1
CRLF
UTF-8
4 spaces


Assignment 1
Version control
Current File
Project
Tasks.java
Board.java
Assignment 1 C:\Users\HP\OneDrive\
.idea
out
src
Board
Board
Tasks
Tasks
Workers
Main
.gitignore
Assignment 1.iml
External Libraries
Scratches and Consoles
/**
* The Board class manages the operations of Juniors, Seniors, and Managers within an organization.
* It includes functionalities to add workers, assign tasks, apply penalties, give raises, promote juniors,
* and authenticate managers for administrative operations.
*/
package Board;
import Workers.Manger.Manger;
import Workers.Normal.Junior.Junior;
import Workers.Normal.Normal;
import Workers.Normal.Senior.Senior;
import java.util.ArrayList;
import java.util.Objects;
import java.util.Scanner;
Edit | Explain | Test | Document | Fix
public class Board { 3 usages
private ArrayList<Junior> juniors; 21 usages
private ArrayList<Manger> mangers; 4 usages
private ArrayList<Senior> seniors; 14 usages
Scanner sc = new Scanner(System.in); 10 usages
Edit | Explain | Test | Document | Fix
/**
* Constructor to initialize Board with existing lists of Juniors, Managers, and Seniors.
*/
public Board(ArrayList<Junior> juniors, ArrayList<Manger> mangers, ArrayList<Senior> Seniors) { 1 usage
this.juniors = juniors;
this.mangers = mangers;
this.seniors = Seniors;
}
Edit | Explain | Test | Document | Fix
Assignment 1 > src > Board > Board > seniorWork
tabnine Basic
393:34
CRLF
UTF-8
4 spaces



Video Link: https://youtu.be/-brwmS8jSZo

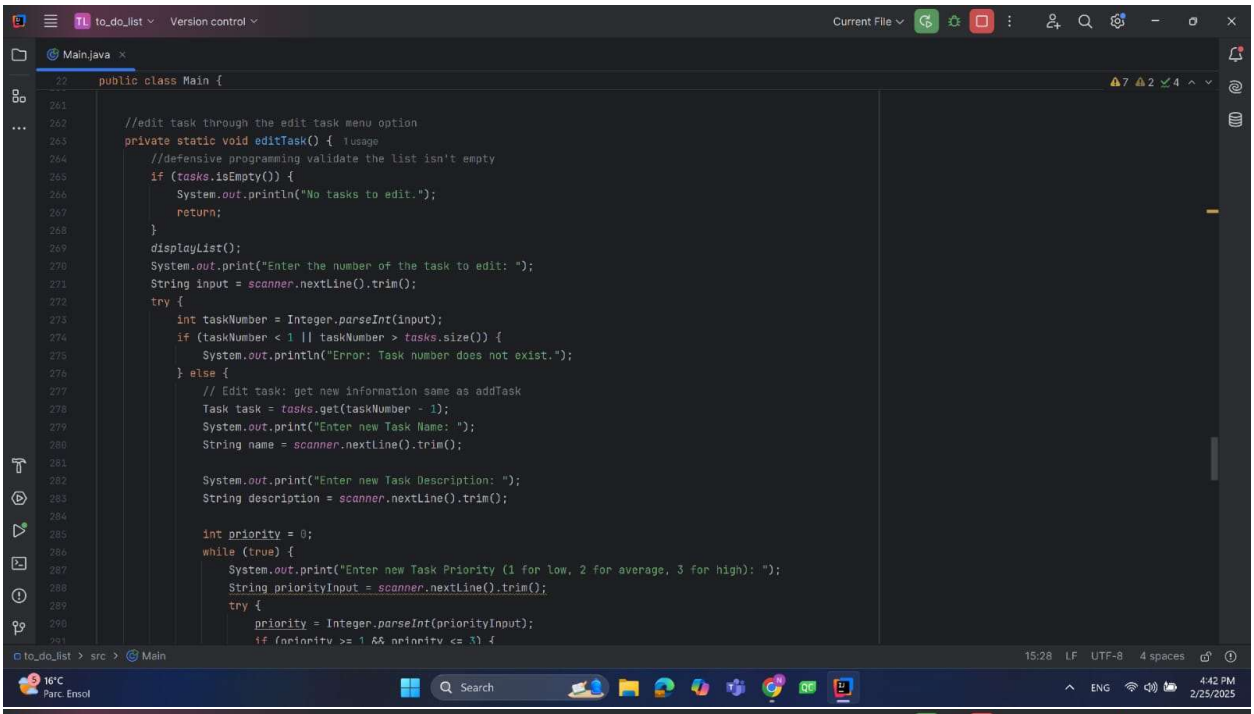
Nagham Wael:

- Hours of Study: 12h
- Source of Study: Geeks For Geeks + https://youtube.com/playlist?list=PLCInYL3l2AajYlZGzU_LVrHdoouf8W6ZN&si=tfRlF2iNxOtJehum
- Main Code Logic:

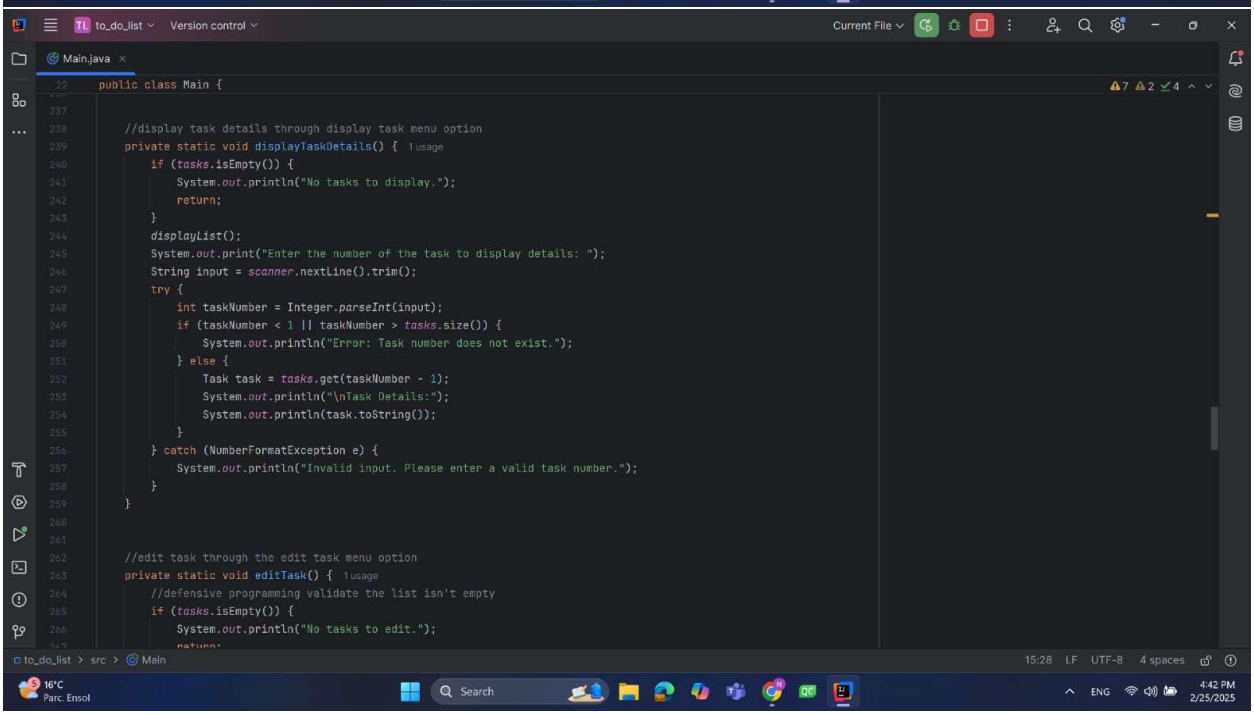
```
•  public static void main(String[] args) {
        boolean running = true;

        //display menu while the program is running(user didn't choose exit)
        while (running) {
            printMenu();
            String choice = scanner.nextLine().trim();
            switch (choice) {
                case "1":
                    addTask();
                    break;
                case "2":
                    removeTask();
                    break;
                case "3":
                    displayTaskDetails();
                    break;
                case "4":
                    editTask();
                    break;
                case "5":
                    sortTasks();
                    break;
                case "6":
                    displayList();
                    break;
                case "7":
                    clearList();
                    break;
                case "8":
                    saveListToFile();
                    break;
                case "9":
                    running = false;
                    System.out.println("Exiting program...");
                    break;
                default:
                    System.out.println("Invalid option. Please choose from 1
    to 9.");
            }
```

```
        }

        scanner.close();
    }
```

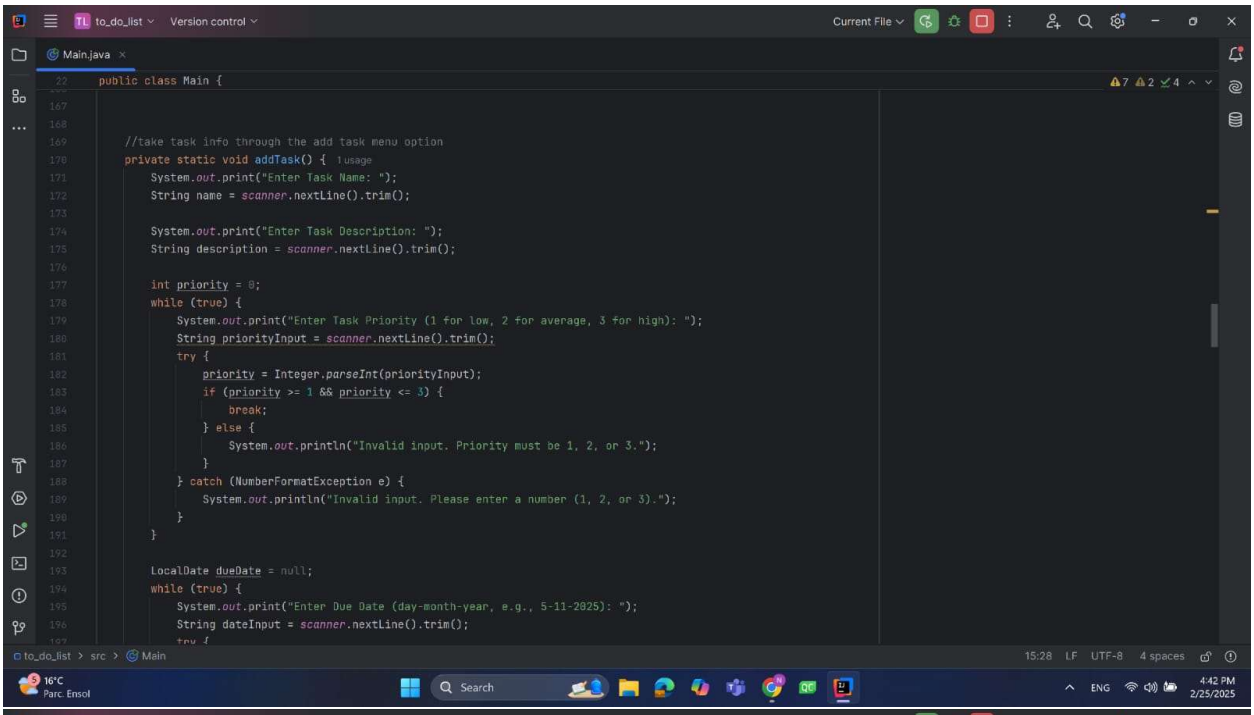
- Images for code :



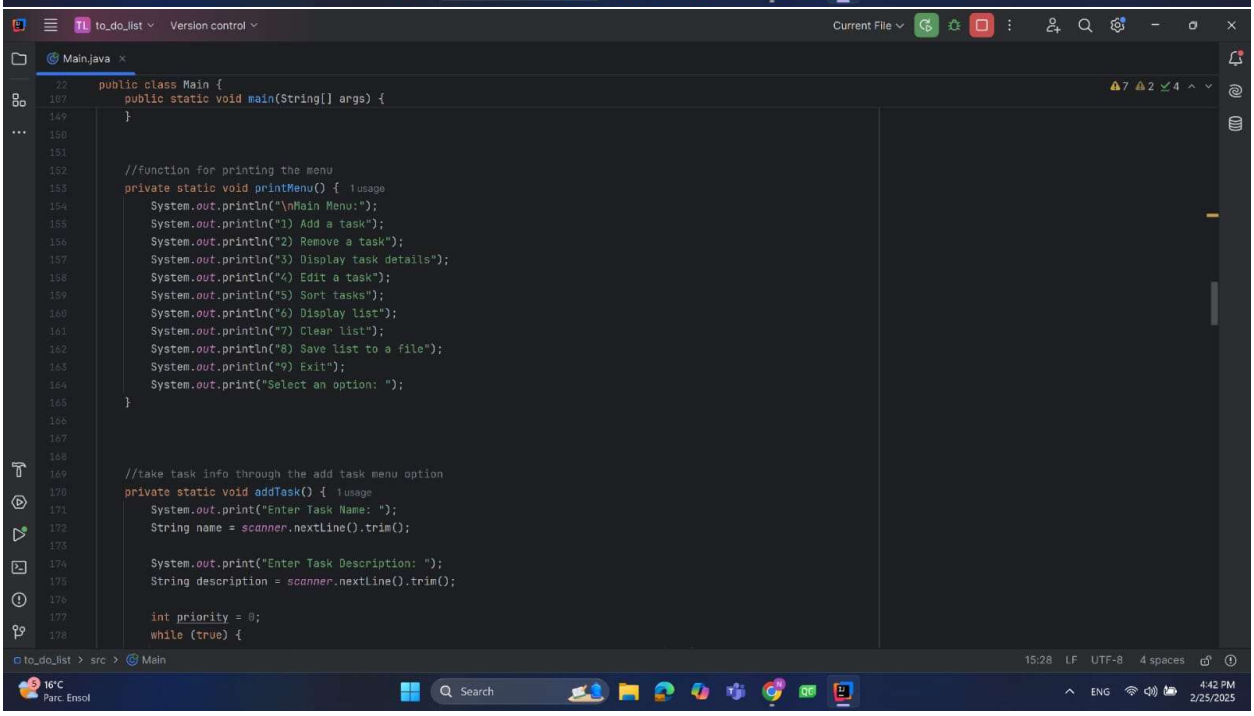
```
    public class Main {


        //edit task through the edit task menu option
        private static void editTask() {  1 usage
            //defensive programming validate the list isn't empty
            if (tasks.isEmpty()) {
                System.out.println("No tasks to edit.");
                return;
            }
            displayList();
            System.out.print("Enter the number of the task to edit: ");
            String input = scanner.nextLine().trim();
            try {
                int taskNumber = Integer.parseInt(input);
                if (taskNumber < 1 || taskNumber > tasks.size()) {
                    System.out.println("Error: Task number does not exist.");
                } else {
                    // Edit task: get new information same as addTask
                    Task task = tasks.get(taskNumber - 1);
                    System.out.print("Enter new Task Name: ");
                    String name = scanner.nextLine().trim();

                    System.out.print("Enter new Task Description: ");
                    String description = scanner.nextLine().trim();

                    int priority = 0;
                    while (true) {
                        System.out.print("Enter new Task Priority (1 for low, 2 for average, 3 for high): ");
                        String priorityInput = scanner.nextLine().trim();
                        try {
                            priority = Integer.parseInt(priorityInput);
                            if (priority >= 1 && priority <= 3) {
```



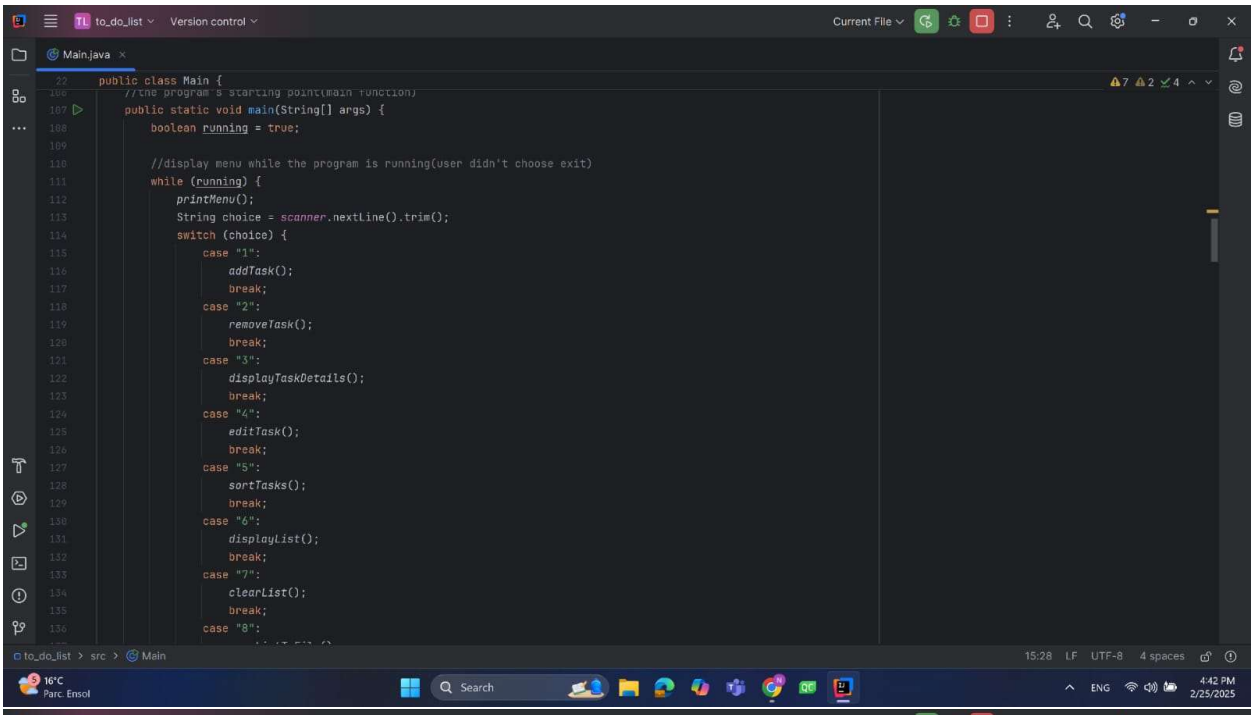
```
    public class Main {


        //display task details through display task menu option
        private static void displayTaskDetails() {  1 usage
            if (tasks.isEmpty()) {
                System.out.println("No tasks to display.");
                return;
            }
            displayList();
            System.out.print("Enter the number of the task to display details: ");
            String input = scanner.nextLine().trim();
            try {
                int taskNumber = Integer.parseInt(input);
                if (taskNumber < 1 || taskNumber > tasks.size()) {
                    System.out.println("Error: Task number does not exist.");
                } else {
                    Task task = tasks.get(taskNumber - 1);
                    System.out.println("\nTask Details:");
                    System.out.println(task.toString());
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a valid task number.");
            }
        }


        //edit task through the edit task menu option
        private static void editTask() {  1 usage
            //defensive programming validate the list isn't empty
            if (tasks.isEmpty()) {
                System.out.println("No tasks to edit.");
```



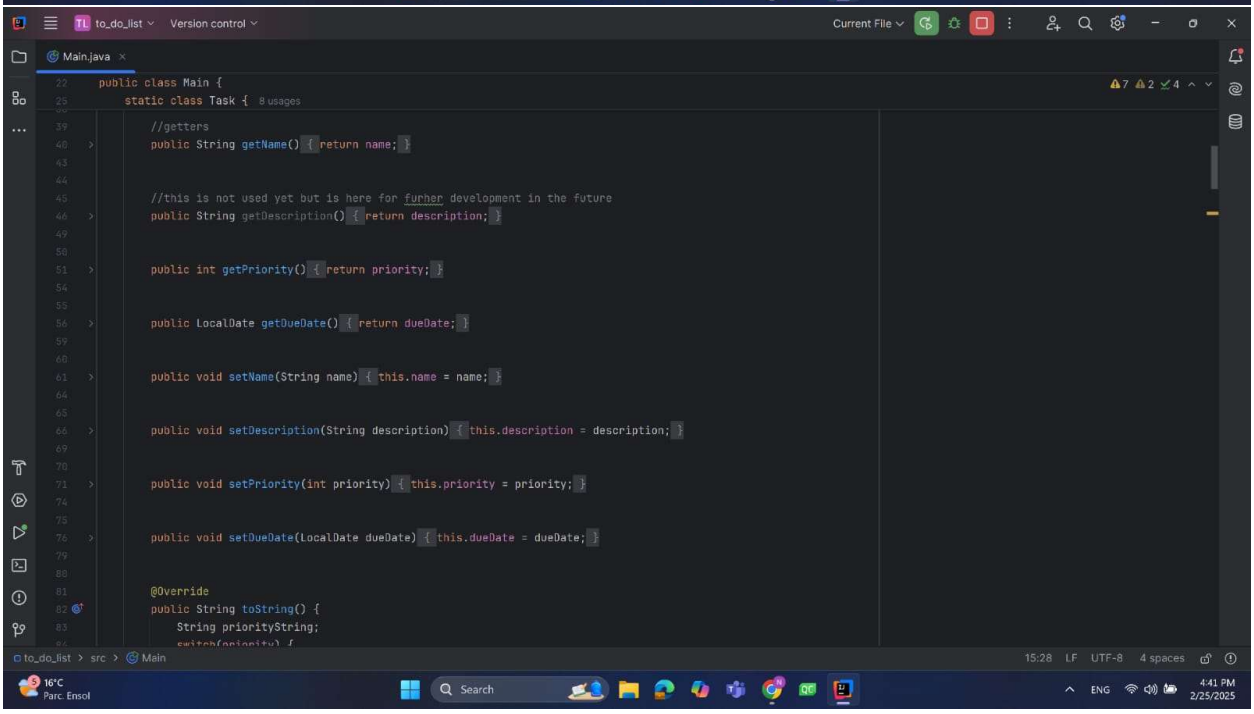
```
public class Main {



    //take task info through the add task menu option
    private static void addTask() {  1 usage
        System.out.print("Enter Task Name: ");
        String name = scanner.nextLine().trim();

        System.out.print("Enter Task Description: ");
        String description = scanner.nextLine().trim();

        int priority = 0;
        while (true) {
            System.out.print("Enter Task Priority (1 for low, 2 for average, 3 for high): ");
            String priorityInput = scanner.nextLine().trim();
            try {
                priority = Integer.parseInt(priorityInput);
                if (priority >= 1 && priority <= 3) {
                    break;
                } else {
                    System.out.println("Invalid input. Priority must be 1, 2, or 3.");
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a number (1, 2, or 3).");
            }
        }

        LocalDate dueDate = null;
        while (true) {
            System.out.print("Enter Due Date (day-month-year, e.g., 5-11-2025): ");
            String dateInput = scanner.nextLine().trim();
            try {
```



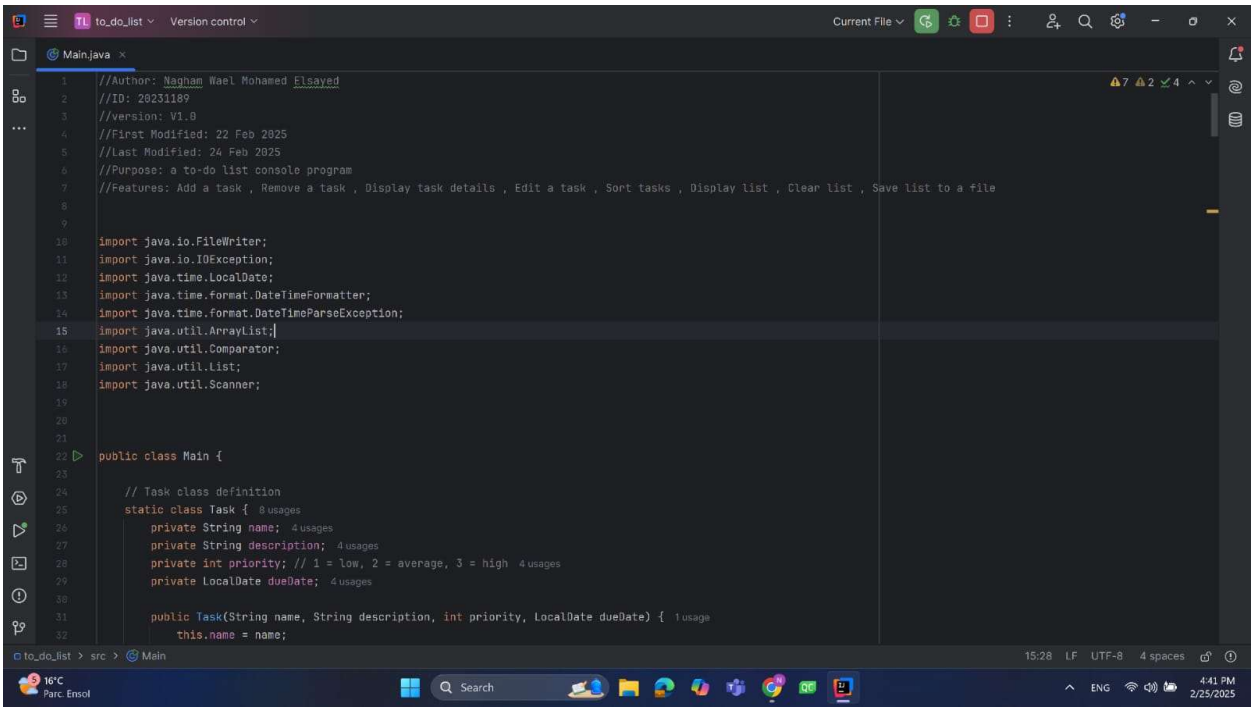
```
public class Main {
    public static void main(String[] args) {
    }



    //function for printing the menu
    private static void printMenu() {  1 usage
        System.out.println("\nMain Menu:");
        System.out.println("1) Add a task");
        System.out.println("2) Remove a task");
        System.out.println("3) Display task details");
        System.out.println("4) Edit a task");
        System.out.println("5) Sort tasks");
        System.out.println("6) Display list");
        System.out.println("7) Clear list");
        System.out.println("8) Save list to a file");
        System.out.println("9) Exit");
        System.out.print("Select an option: ");
    }



    //take task info through the add task menu option
    private static void addTask() {  1 usage
        System.out.print("Enter Task Name: ");
        String name = scanner.nextLine().trim();

        System.out.print("Enter Task Description: ");
        String description = scanner.nextLine().trim();

        int priority = 0;
        while (true) {
```



```
public class Main {
    //the program's starting point(main function)
    public static void main(String[] args) {
        boolean running = true;

        //display menu while the program is running(user didn't choose exit)
        while (running) {
            printMenu();
            String choice = scanner.nextLine().trim();
            switch (choice) {
                case "1":
                    addTask();
                    break;
                case "2":
                    removeTask();
                    break;
                case "3":
                    displayTaskDetails();
                    break;
                case "4":
                    editTask();
                    break;
                case "5":
                    sortTasks();
                    break;
                case "6":
                    displayList();
                    break;
                case "7":
                    clearList();
                    break;
                case "8":
```



```
public class Main {
    static class Task {  8 usages

        //getters
        public String getName() { return name; }


        //this is not used yet but is here for furher development in the future
        public String getDescription() { return description; }


        public int getPriority() { return priority; }


        public LocalDate getDueDate() { return dueDate; }


        public void setName(String name) { this.name = name; }


        public void setDescription(String description) { this.description = description; }


        public void setPriority(int priority) { this.priority = priority; }


        public void setDueDate(LocalDate dueDate) { this.dueDate = dueDate; }


        @Override
        public String toString() {
            String priorityString;
```



Video Link:

https://youtu.be/VqOBuBvfRiw?si=WCep92XFi12x7LnN

Fatema El-Zhraa Ahmed:

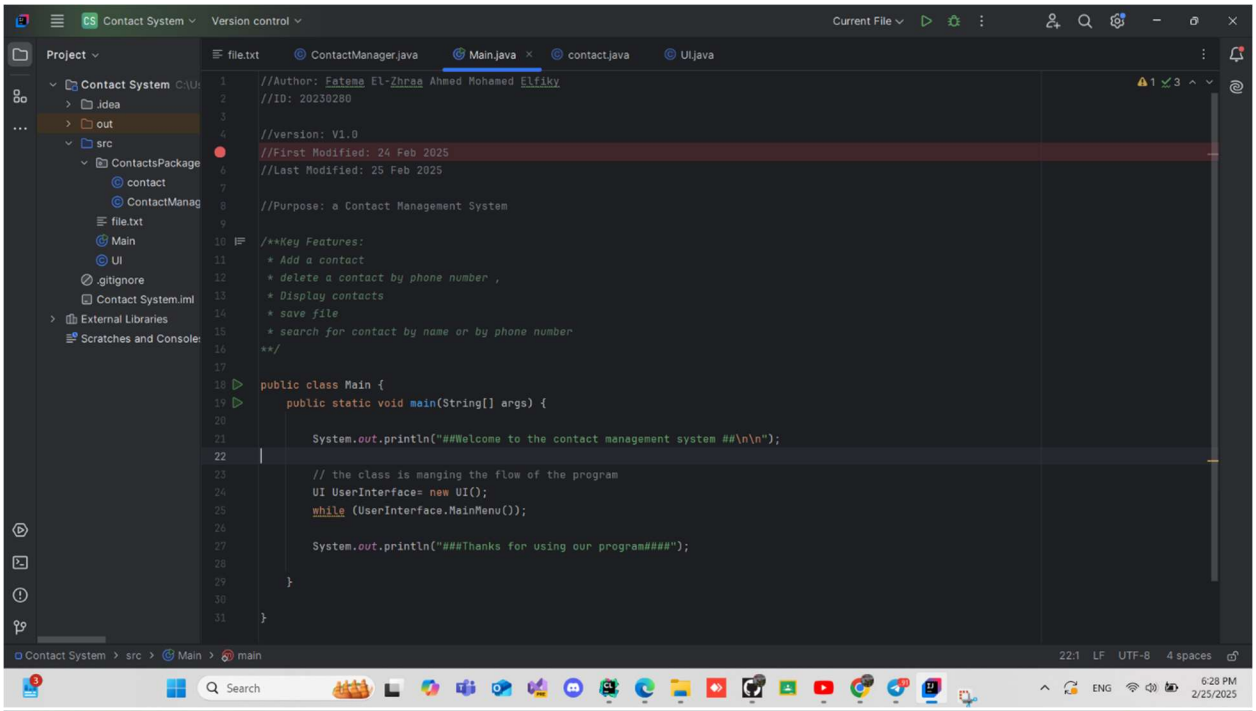
- Hours of Study:12 hours
- Source of Study : https://www.youtube.com/playlist?list=PLCInYL3l2AajYlZGzU_LVrHdoouf8W6ZN
- Main Code Logic:

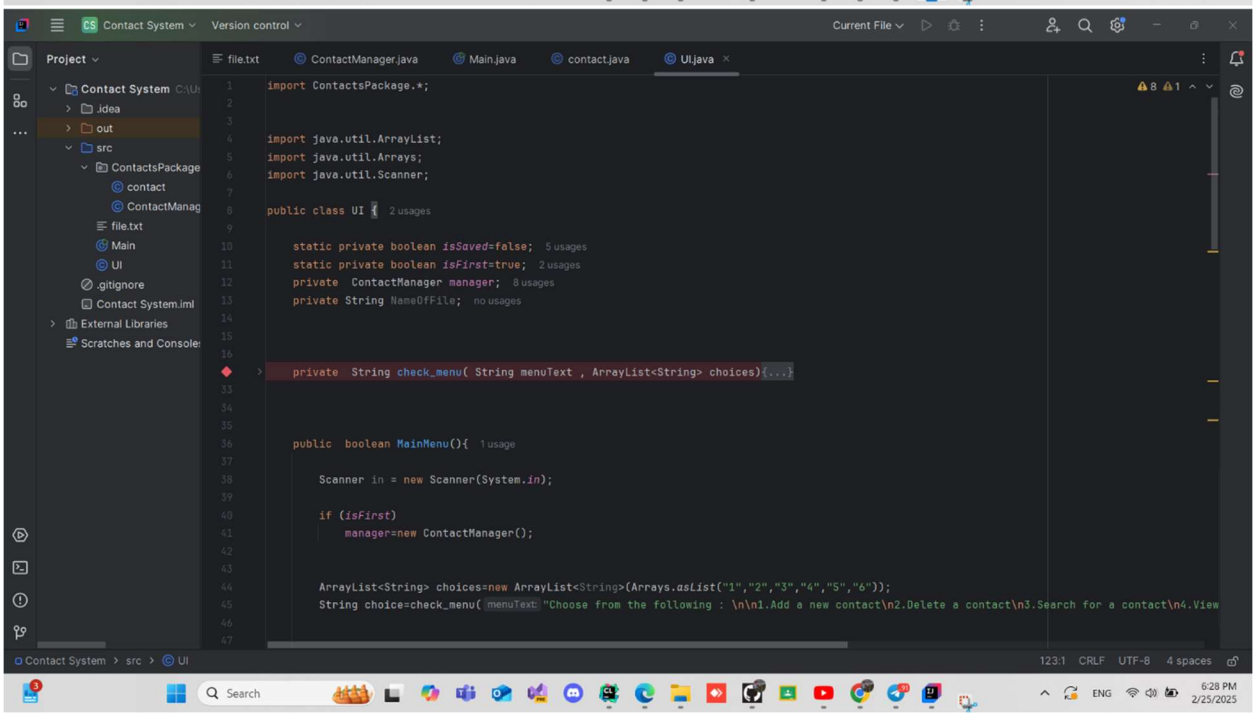
```
• public  boolean MainMenu(){

    Scanner in = new Scanner(System.in);

    if (isFirst)
        manager=new ContactManager();


    ArrayList<String> choices=new
ArrayList<String>(Arrays.asList("1","2","3","4","5","6"));
    String choice=check_menu("Choose from the following :
\n\n1.Add a new contact\n2.Delete a contact\n3.Search for a
contact\n4.View all contacts\n5.Save contacts in a
file\n6.Exit\n\nEnter your answer : ",choices );


    if(choice.equals("1")){//Add

        boolean isRepeated=manager.Add();
        if(!isRepeated)
            System.out.println("Sorry , it will not be added
for repeating phone number or email");
        else
            isSaved=false;

    }
    else if( choice.equals("2")){//Delete

        if(manager.Delete())
            isSaved=false;

    }
    else if( choice.equals("3")){//search for (by name ,
```

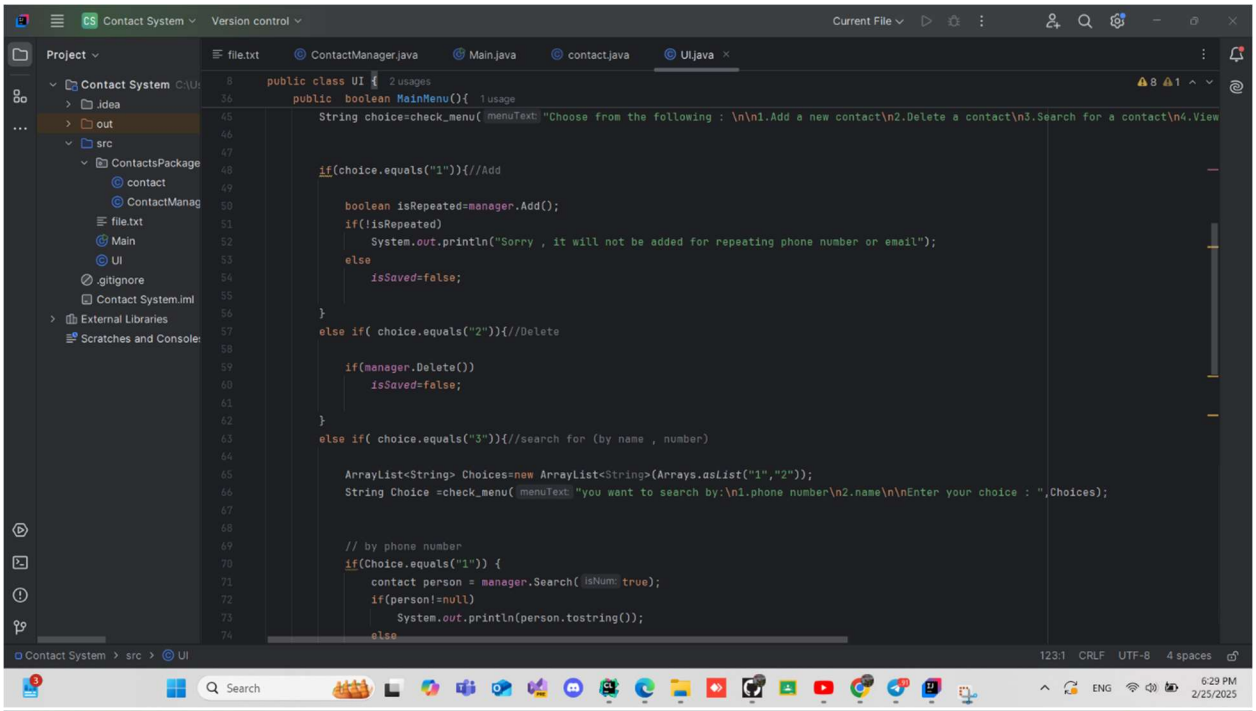
```
number)

        ArrayList<String> Choices=new
ArrayList<String>(Arrays.asList("1","2"));
        String Choice =check_menu("you want to search
by:\n1.phone number\n2.name\n\nEnter your choice :
",Choices);

        if(Choice.equals("1")) {
            contact person = manager.Search(true);
            if(person!=null)
                System.out.println(person.tostring());
            else
                System.out.printf("Sorry the contact is not
found\n\n");

        }
        else {
            contact person = manager.Search(false);
            if(person!=null)
                System.out.println(person.tostring());
            else
                System.out.printf("Sorry the contact is not
found\n\n");

        }

    }
    else if( choice.equals("4")){//view all contacts

       manager.Display();

    }
    else if( choice.equals("5")){//Save contacts in file

        if(!isSaved)
            manager.Save();

        isSaved=true;
```

```
    }else{//Exit

        if(!isSaved){

            ArrayList<String>Choices=new
ArrayList<String>(Arrays.asList("1","2"));
            String Choice=check_menu("Do you want to save
changes before exiting?\n1.Yes\n2.No\n\nEnter your choice :
",Choices);

            if(Choice.equals("1"))// as task 5
                manager.Save();

        }

        return false;
    }

    isFirst=false;
    return true;

}
```

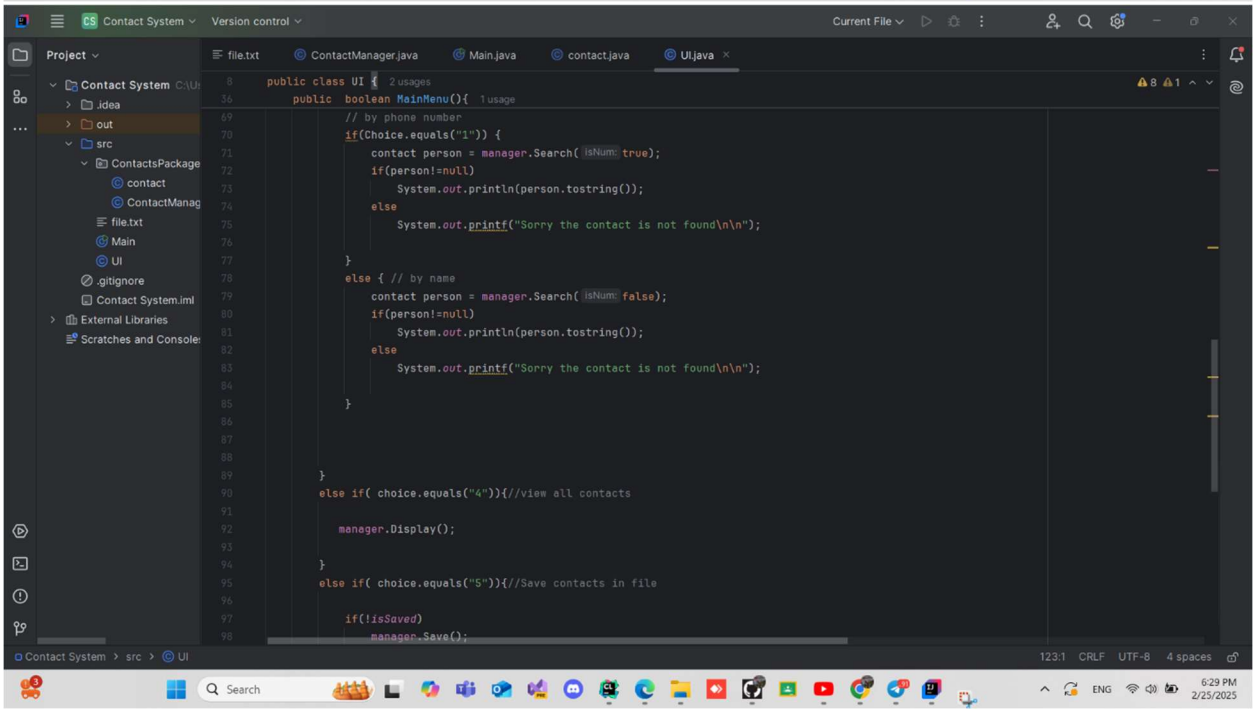
Screen Shots :


Contact System
Version control
Current File
Project
file.txt
ContactManager.java
Main.java
contact.java
UI.java
//Author: Fateme El-Zhraa Ahmed Mohamed Elfiky
//ID: 20230280
//version: V1.0
//First Modified: 24 Feb 2025
//Last Modified: 25 Feb 2025
//Purpose: a Contact Management System
/**Key Features:
* Add a contact
* delete a contact by phone number ,
* Display contacts
* save file
* search for contact by name or by phone number
**/
public class Main {
public static void main(String[] args) {
System.out.println("##Welcome to the contact management system ##\n\n");
// the class is manging the flow of the program
UI UserInterface= new UI();
while (UserInterface.MainMenu());
System.out.println("###Thanks for using our program####");
}
}
Contact System › src › Main › main
22:1 LF UTF-8 4 spaces
Search
ENG
6:28 PM
2/25/2025

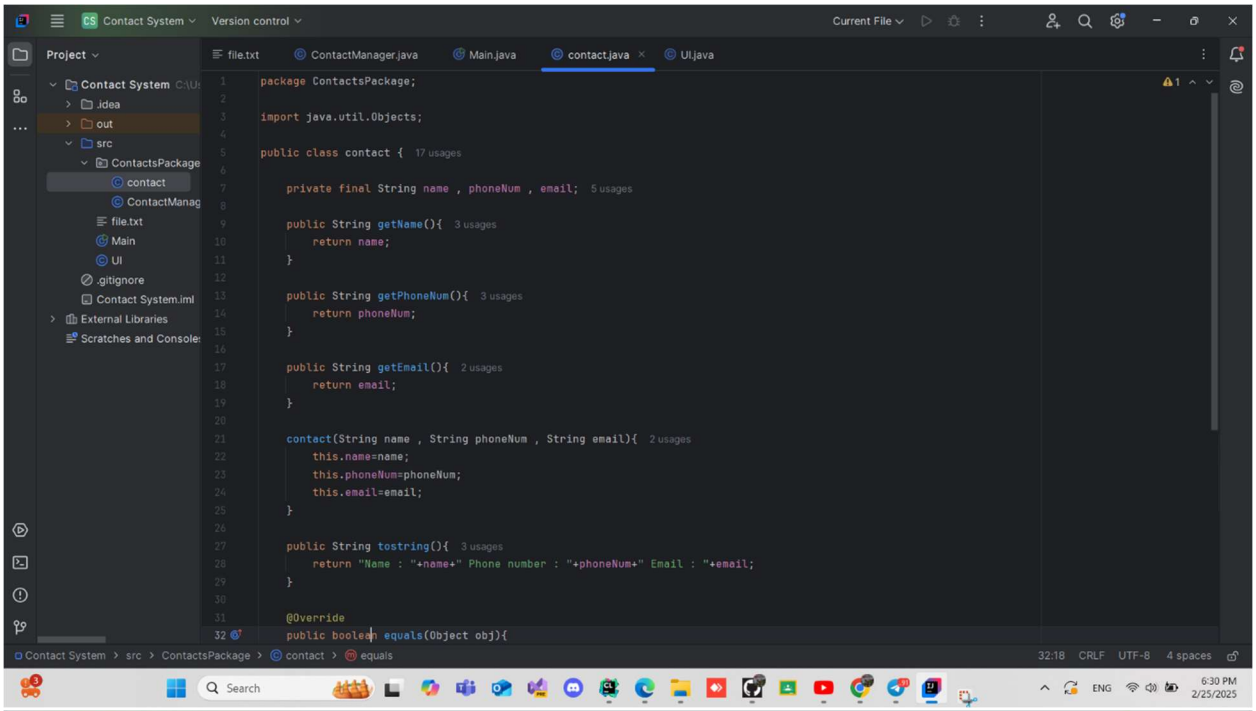

Contact System
Version control
Current File
Project
file.txt
ContactManager.java
Main.java
contact.java
UI.java
import ContactsPackage.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;
public class UI {
static private boolean isSaved=false;
static private boolean isFirst=true;
private ContactManager manager;
private String NameOfFile;
private String check_menu(String menuText , ArrayList<String> choices){...}
public boolean MainMenu(){
Scanner in = new Scanner(System.in);
if (isFirst)
manager=new ContactManager();
ArrayList<String> choices=new ArrayList<String>(Arrays.asList("1","2","3","4","5","6"));
String choice=check_menu(menuText: "Choose from the following : \n\n1.Add a new contact\n2.Delete a contact\n3.Search for a contact\n4.View
Contact System › src › UI
123:1 CRLF UTF-8 4 spaces
Search
ENG
6:28 PM
2/25/2025

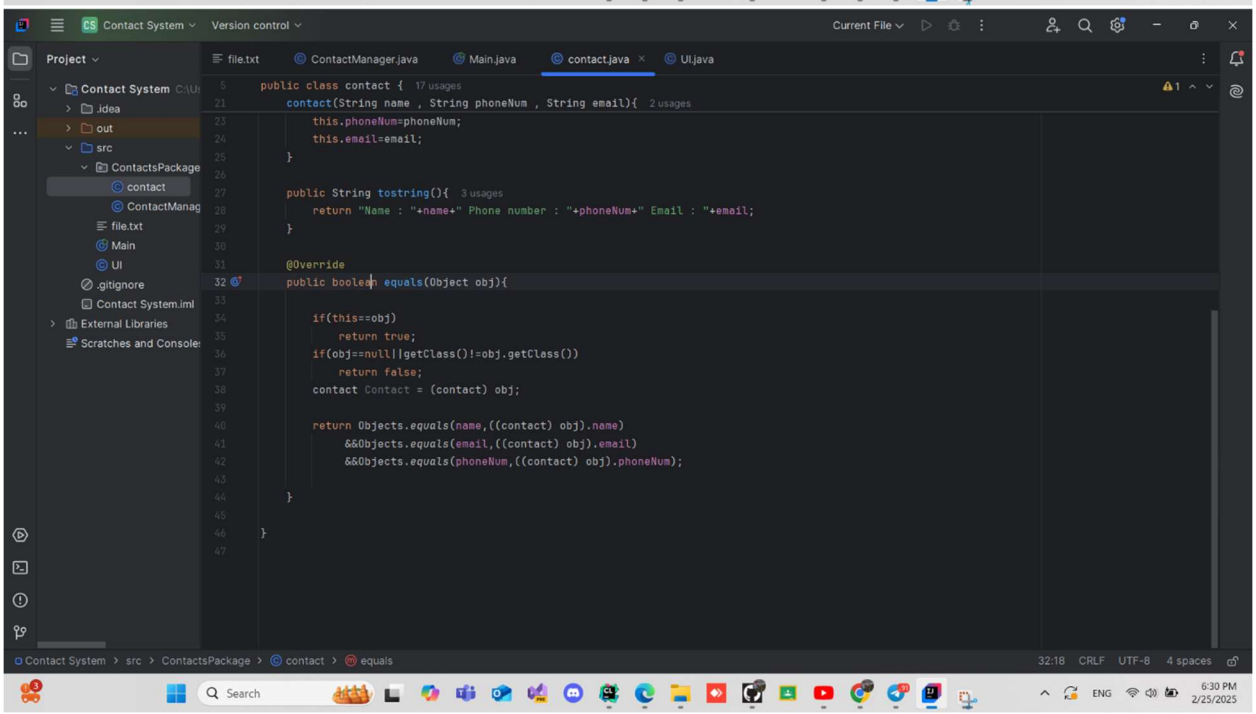


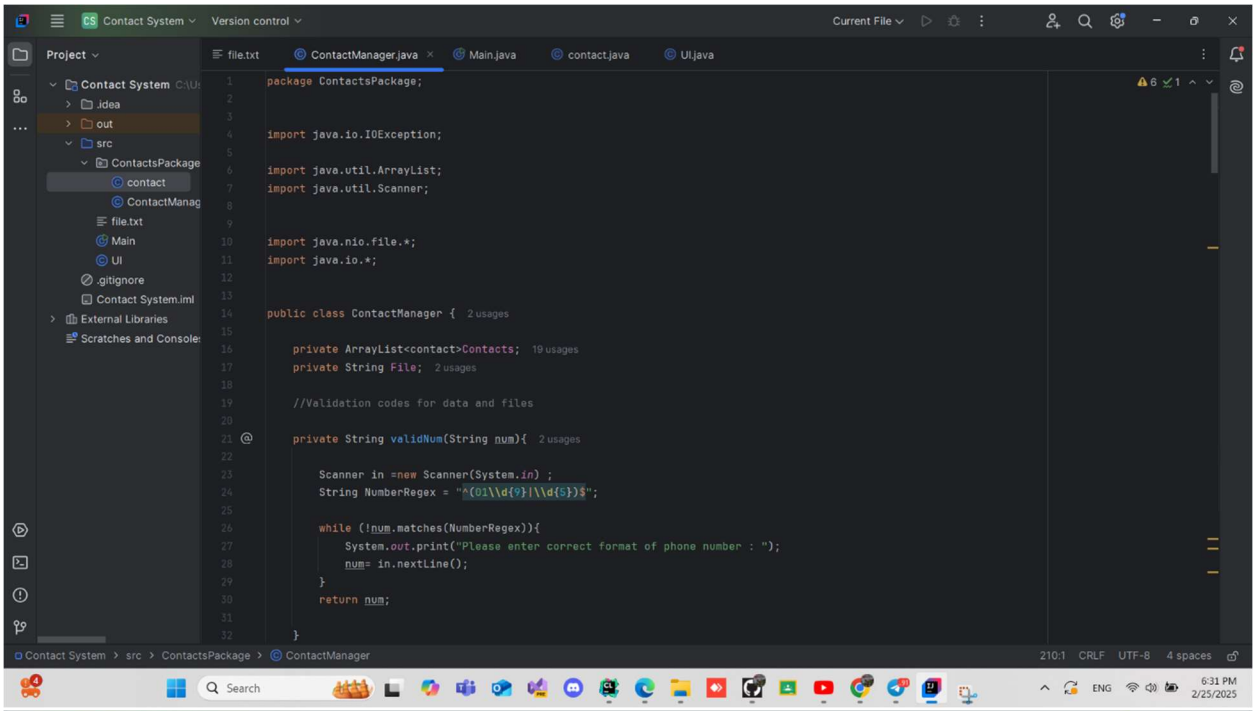
```
public class UI {  2 usages
    public  boolean MainMenu(){  1 usage
        String choice=check_menu( menuText: "Choose from the following : \n\n1.Add a new contact\n2.Delete a contact\n3.Search for a contact\n4.View

        if(choice.equals("1")){//Add

            boolean isRepeated=manager.Add();
            if(!isRepeated)
                System.out.println("Sorry , it will not be added for repeating phone number or email");
            else
                isSaved=false;

        }
        else if( choice.equals("2")){//Delete

            if(manager.Delete())
                isSaved=false;

        }
        else if( choice.equals("3")){//search for (by name , number)

            ArrayList<String> Choices=new ArrayList<String>(Arrays.asList("1","2"));
            String Choice =check_menu( menuText: "you want to search by:\n1.phone number\n2.name\n\nEnter your choice : ",Choices);


            // by phone number
            if(Choice.equals("1")) {
                contact person = manager.Search( isNum: true);
                if(person!=null)
                    System.out.println(person.tostring());
                else
```

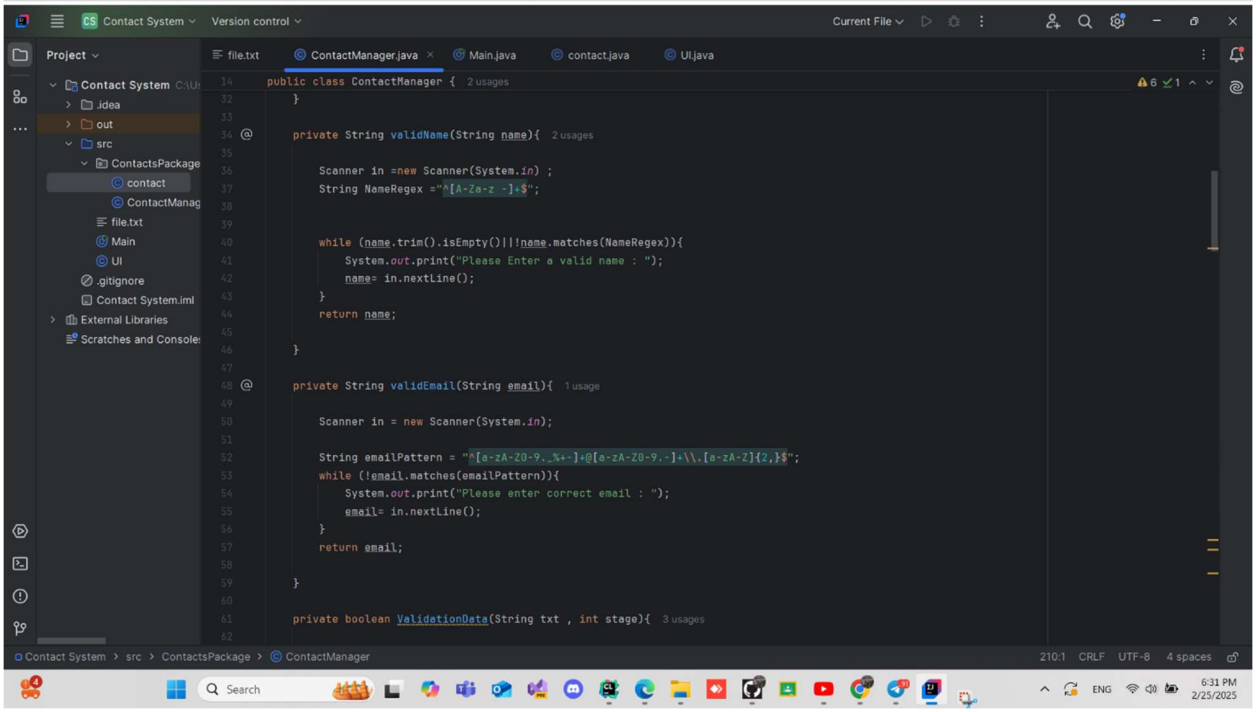


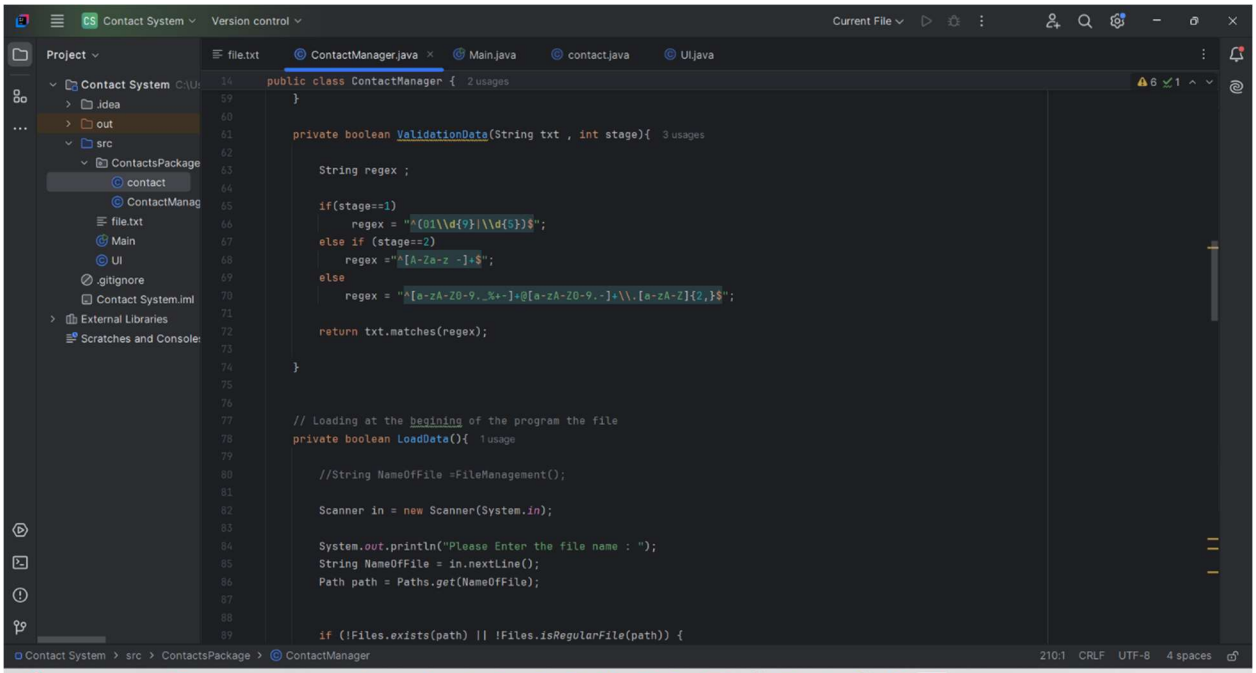
```
public class UI {  2 usages
    public  boolean MainMenu(){  1 usage
            // by phone number
            if(Choice.equals("1")) {
                contact person = manager.Search( isNum: true);
                if(person!=null)
                    System.out.println(person.tostring());
                else
                    System.out.printf("Sorry the contact is not found\n\n");

            }
            else { // by name
                contact person = manager.Search( isNum: false);
                if(person!=null)
                    System.out.println(person.tostring());
                else
                    System.out.printf("Sorry the contact is not found\n\n");


            }



        }
        else if( choice.equals("4")){//view all contacts

            manager.Display();

        }
        else if( choice.equals("5")){//Save contacts in file

            if(!isSaved)
                manager.Save();
```

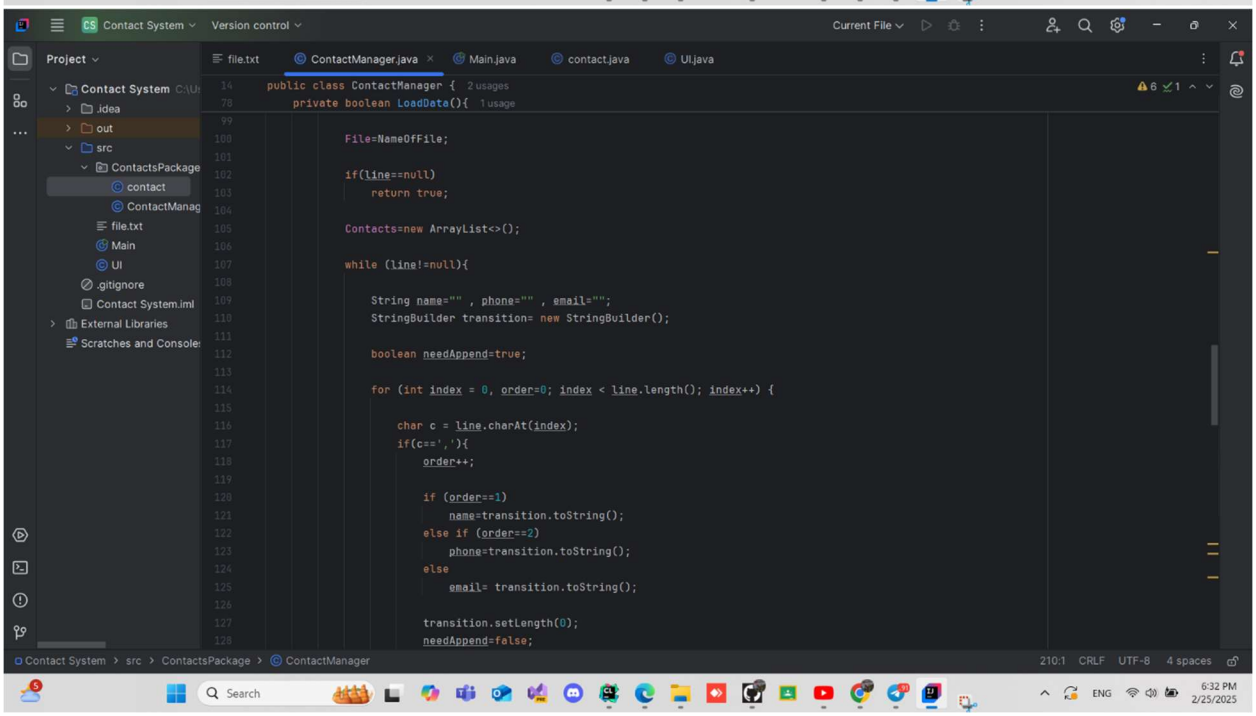

Contact System
Version control
Current File
Project
file.txt
ContactManager.java
Main.java
contact.java
UI.java
Contact System
.idea
out
src
ContactsPackage
contact
ContactManag
file.txt
Main
UI
.gitignore
Contact System.iml
External Libraries
Scratches and Consoles
package ContactsPackage;
import java.util.Objects;
public class contact { 17 usages
private final String name , phoneNum , email; 5 usages
public String getName(){ 3 usages
return name;
}
public String getPhoneNum(){ 3 usages
return phoneNum;
}
public String getEmail(){ 2 usages
return email;
}
contact(String name , String phoneNum , String email){ 2 usages
this.name=name;
this.phoneNum=phoneNum;
this.email=email;
}
public String tostring(){ 3 usages
return "Name : "+name+" Phone number : "+phoneNum+" Email : "+email;
}
@Override
public boolean equals(Object obj){
Contact System › src › ContactsPackage › contact › equals
32:18 CRLF UTF-8 4 spaces
Search
ENG
6:30 PM
2/25/2025


Contact System
Version control
Current File
Project
file.txt
ContactManager.java
Main.java
contact.java
UI.java
public class contact { 17 usages
contact(String name , String phoneNum , String email){ 2 usages
this.phoneNum=phoneNum;
this.email=email;
}
public String tostring(){ 3 usages
return "Name : "+name+" Phone number : "+phoneNum+" Email : "+email;
}
@Override
public boolean equals(Object obj){
if(this==obj)
return true;
if(obj==null||getClass()!=obj.getClass())
return false;
contact Contact = (contact) obj;
return Objects.equals(name,((contact) obj).name)
&&Objects.equals(email,((contact) obj).email)
&&Objects.equals(phoneNum,((contact) obj).phoneNum);
}
}
Contact System › src › ContactsPackage › contact › equals
32:18 CRLF UTF-8 4 spaces
Search
ENG
6:30 PM
2/25/2025


Contact System
Version control
Current File
Project
file.txt
ContactManager.java
Main.java
contact.java
UI.java
Contact System
.idea
out
src
ContactsPackage
contact
ContactManag
file.txt
Main
UI
.gitignore
Contact System.iml
External Libraries
Scratches and Consoles
package ContactsPackage;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.nio.file.*;
import java.io.*;
public class ContactManager { 2 usages
private ArrayList<contact>Contacts; 19 usages
private String File; 2 usages
//Validation codes for data and files
private String validNum(String num){ 2 usages
Scanner in =new Scanner(System.in) ;
String NumberRegex = "^(01\\d{9}|\\d{5})$";
while (!num.matches(NumberRegex)){
System.out.print("Please enter correct format of phone number : ");
num= in.nextLine();
}
return num;
}
Contact System › src › ContactsPackage › ContactManager
210:1 CRLF UTF-8 4 spaces
Search
ENG
6:31 PM
2/25/2025


Contact System
Version control
Current File
Project
file.txt
ContactManager.java
Main.java
contact.java
UI.java
Contact System
.idea
out
src
ContactsPackage
contact
ContactManag
file.txt
Main
UI
.gitignore
Contact System.iml
External Libraries
Scratches and Consoles
public class ContactManager { 2 usages
}
private String validName(String name){ 2 usages
Scanner in =new Scanner(System.in) ;
String NameRegex ="^[A-Za-z -]+$";
while (name.trim().isEmpty()||!name.matches(NameRegex)){
System.out.print("Please Enter a valid name : ");
name= in.nextLine();
}
return name;
}
private String validEmail(String email){ 1 usage
Scanner in = new Scanner(System.in);
String emailPattern = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
while (!email.matches(emailPattern)){
System.out.print("Please enter correct email : ");
email= in.nextLine();
}
return email;
}
private boolean ValidationData(String txt , int stage){ 3 usages
Contact System › src › ContactsPackage › ContactManager
210:1 CRLF UTF-8 4 spaces
Search
ENG
6:31 PM
2/25/2025





Video Link :

https://www.youtube.com/watch?v=fuV9C_FRDf8

Low-Code/No-Code (LCNC) Tools: Evaluation, Potential, and Impact

Evaluation of LCNC Tools:

As for Thunkable:

Thunkable is a popular LCNC platform for building mobile applications with a visual drag-and-drop interface.

It allows users to create cross-platform iOS and Android apps without deep programming knowledge. While it excels in simplicity and rapid development, it has limitations in customization and performance for highly complex applications.

As for Webflow:

Webflow is a powerful LCNC tool for building responsive websites without writing code. It provides extensive design customization, CMS integration, and e-commerce capabilities.

However, for more advanced backend functionality, developers may still need to use custom code or external integrations.

The potential of LCNC Tools:

As for Thunkable:

Thunkable has great potential for facilitating mobile app development, particularly for non-developers and small businesses.

It enables rapid prototyping and deployment of functional mobile applications, making app development more accessible.

As for Webflow:

Webflow empowers designers and entrepreneurs to create sophisticated websites without relying on developers.

It is particularly useful for startups and businesses that require dynamic, responsive websites with CMS capabilities but lack coding expertise.

What LCNC Tools Can Do:

As for Thunkable:

- Create cross-platform mobile applications.
- Integrate APIs and external services.
- Implement UI/UX elements with a drag-and-drop editor.
- Directly publish apps to the Apple App Store and Google Play Store.

As for Webflow:

- Build highly customizable, responsive websites.
- Integrate CMS and e-commerce features.
- Use pre-built animations and interactions.
- Export clean HTML, CSS, and JavaScript for further customization.

Benefits of LCNC Tools:

As for Thunkable:

- Simplifies mobile app development.
- Reduces costs and time-to-market.
- Provides a user-friendly interface for non-technical users.
- Allows for quick prototyping and iteration.

As for Webflow:

- Enables high-quality web design without coding.
- Offers powerful customization and design flexibility.
- Integrates seamlessly with CMS and third-party tools.
- Generates clean and exportable code for further modifications.

The Quality of the Systems They Produce:

As for Thunkable:

Thunkable produces functional and user-friendly mobile applications suitable for MVPs, small business apps, and educational purposes.

However, apps developed with Thunkable may face performance and scalability challenges compared to those built with native development.

As for Webflow:

Webflow websites are visually stunning and optimized for performance. The platform ensures high-quality front-end development, but for complex backend functionalities, additional integrations or custom coding may be required.

Will LCNC Tools Take the Job of Developers?

As for Thunkable:

While Thunkable simplifies app development, it will not replace developers. Advanced mobile applications still require custom coding, optimization, and backend functionalities that LCNC tools cannot fully provide.

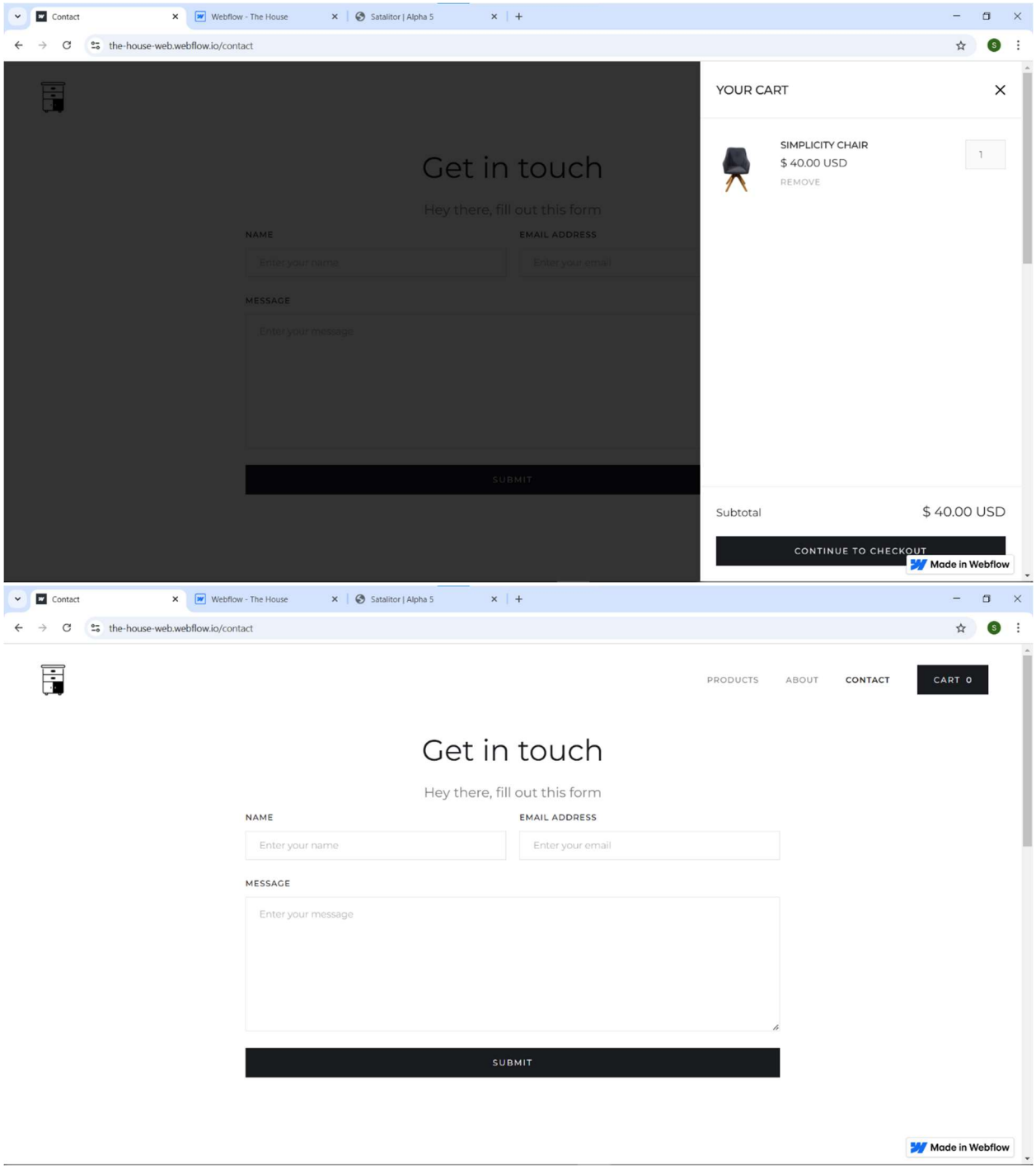
As for Webflow:

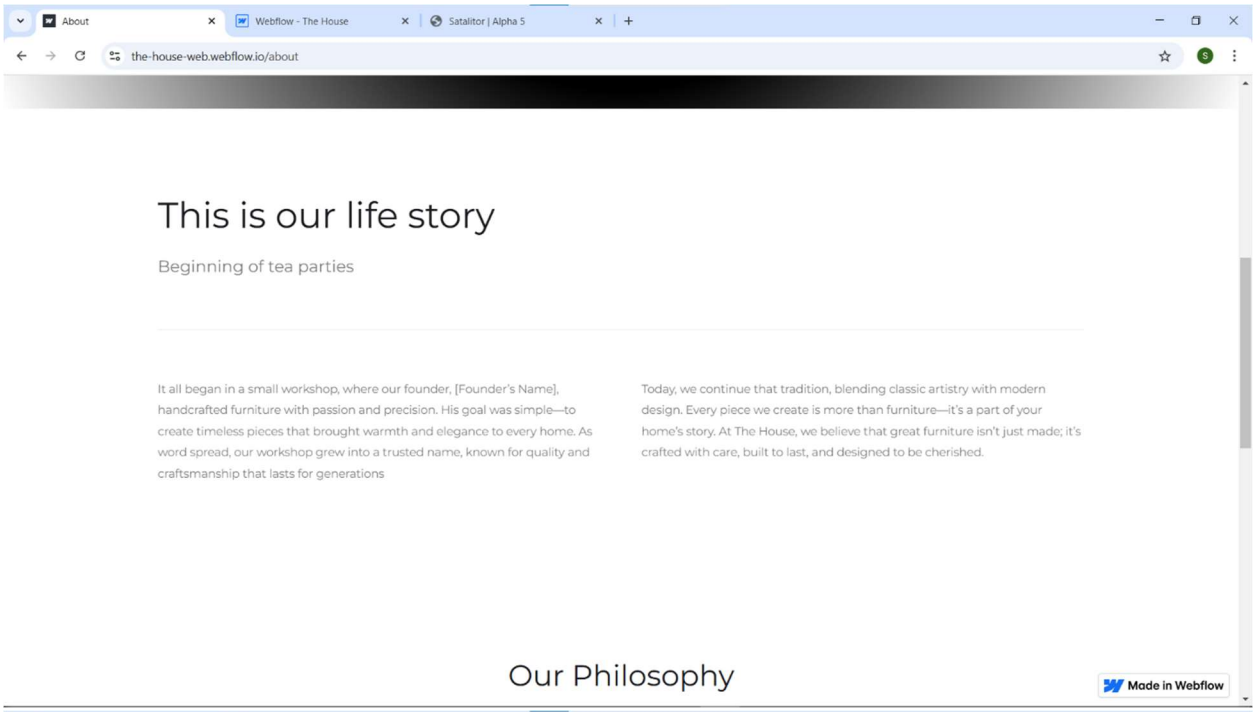
Webflow empowers designers and marketers to build and maintain websites without developers, but for highly dynamic and database-driven applications, developers are still essential. Webflow acts as a tool that enhances productivity rather than replacing professional developers.

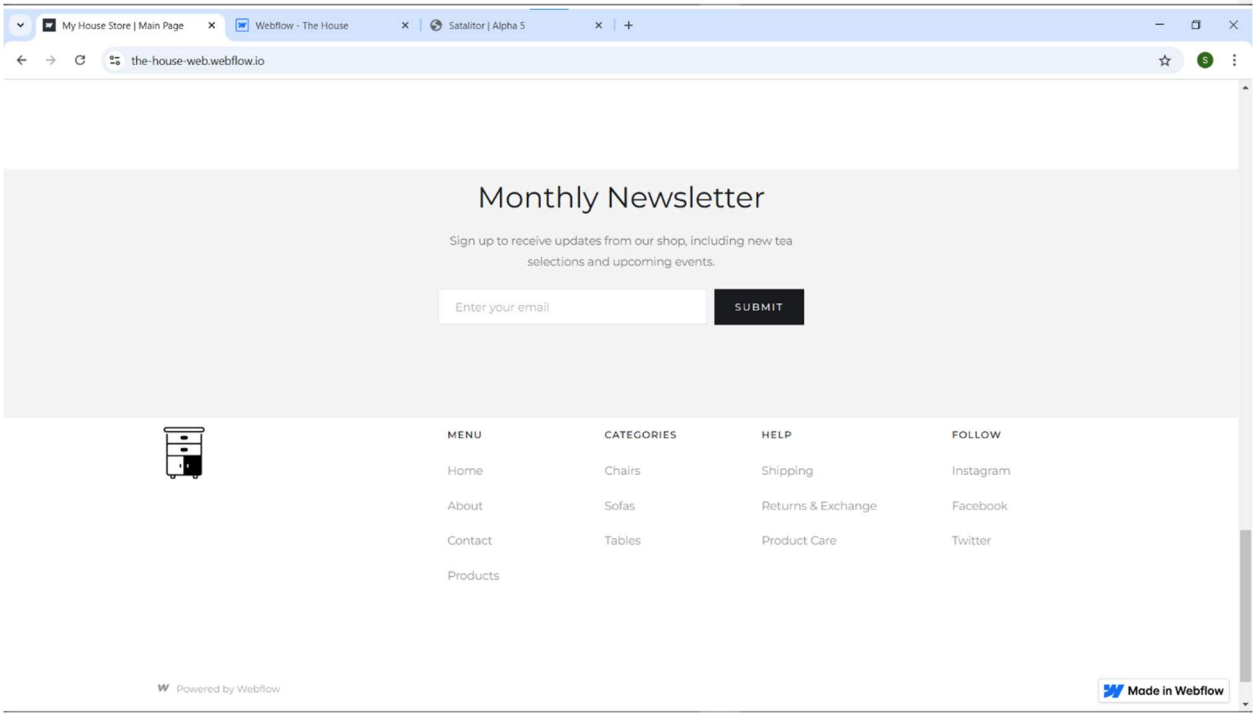
Compare Between Two LCNC Tools:

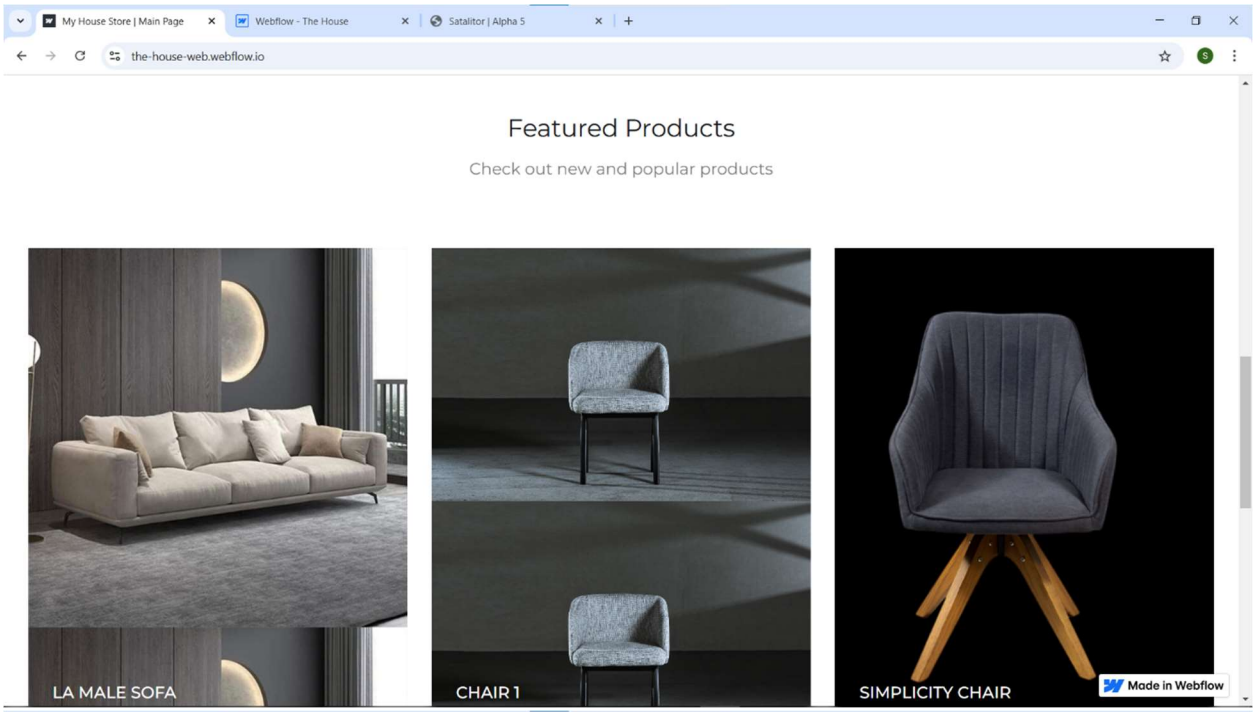
Key of Compare	WebFlow	Thunkable
Purpose	Website design and development	Mobile app development (iOS & Android)
Target Users	Designers, developers, and businesses looking to create responsive websites	Individuals and businesses looking to create mobile apps without coding
Drag-and-Drop Interface	Yes, for building websites visually	Yes, for building mobile apps using a block-based interface
Export Code	Can export HTML, CSS, and JavaScript	No direct code export; apps are built within Thunkable's environment
Use Cases	Landing pages, business websites, e-commerce stores, blogs, and portfolios	Prototyping, MVPs, mobile app development for business, education, and personal projects

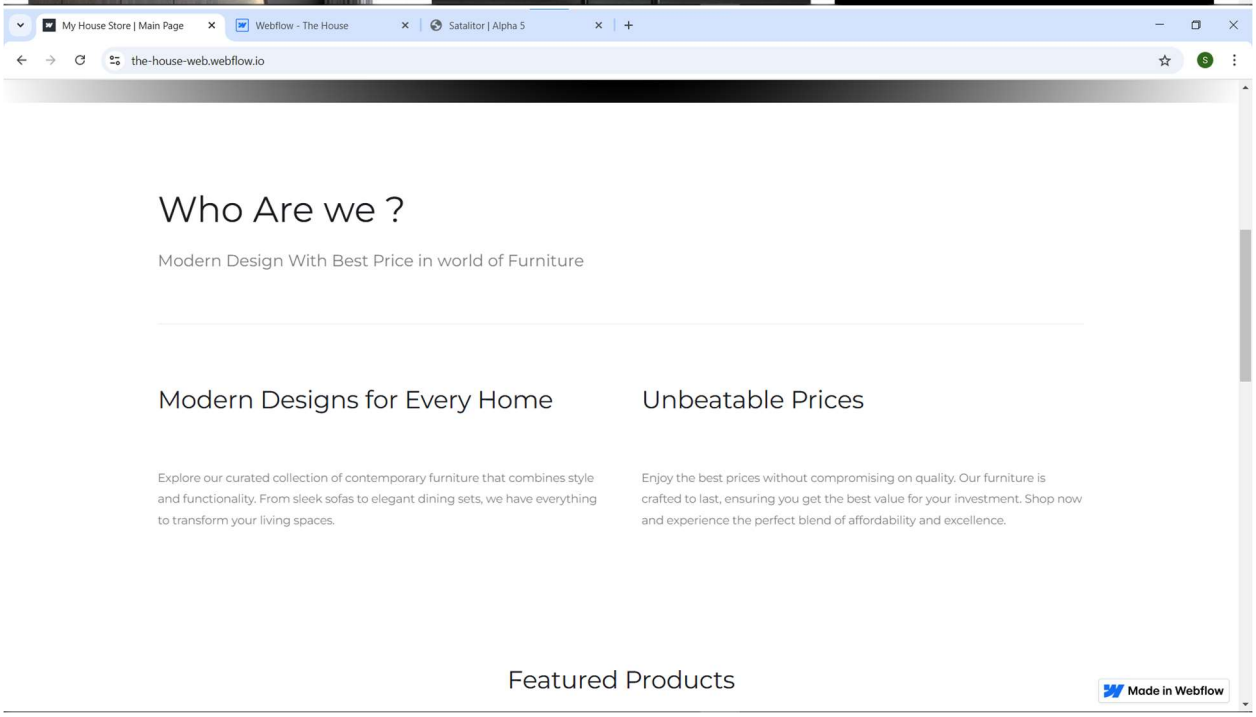
Screenshot of LCNC Code :

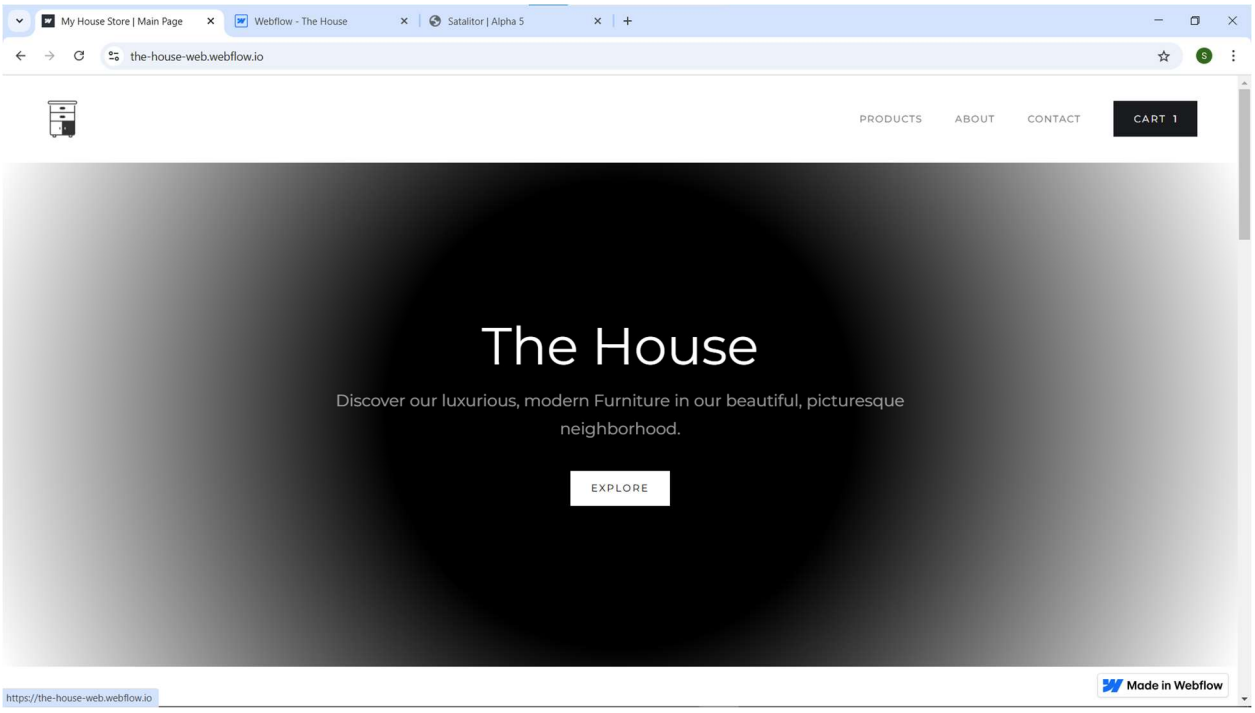



About
Webflow - The House
Satalitor | Alpha 5
the-house-web.webflow.io/about
This is our life story
Beginning of tea parties
It all began in a small workshop, where our founder, [Founder's Name], handcrafted furniture with passion and precision. His goal was simple—to create timeless pieces that brought warmth and elegance to every home. As word spread, our workshop grew into a trusted name, known for quality and craftsmanship that lasts for generations
Today, we continue that tradition, blending classic artistry with modern design. Every piece we create is more than furniture—it's a part of your home's story. At The House, we believe that great furniture isn't just made; it's crafted with care, built to last, and designed to be cherished.
Our Philosophy
Made in Webflow


My House Store | Main Page
Webflow - The House
Satalitor | Alpha 5
the-house-web.webflow.io
Monthly Newsletter
Sign up to receive updates from our shop, including new tea selections and upcoming events.
Enter your email
SUBMIT
MENU
Home
About
Contact
Products
CATEGORIES
Chairs
Sofas
Tables
HELP
Shipping
Returns & Exchange
Product Care
FOLLOW
Instagram
Facebook
Twitter
Powered by Webflow
Made in Webflow


My House Store | Main Page
Webflow - The House
Satalitor | Alpha 5
the-house-web.webflow.io
Featured Products
Check out new and popular products
LA MALE SOFA
CHAIR 1
SIMPLICITY CHAIR
Made in Webflow


My House Store | Main Page
Webflow - The House
Satalitor | Alpha 5
the-house-web.webflow.io
Who Are we ?
Modern Design With Best Price in world of Furniture
Modern Designs for Every Home
Explore our curated collection of contemporary furniture that combines style and functionality. From sleek sofas to elegant dining sets, we have everything to transform your living spaces.
Unbeatable Prices
Enjoy the best prices without compromising on quality. Our furniture is crafted to last, ensuring you get the best value for your investment. Shop now and experience the perfect blend of affordability and excellence.
Featured Products
Made in Webflow



Website link : https://the-house-web.webflow.io/products

Video link:

https://youtu.be/c071WQAifNM?si=sWq_lknsMG3GbJbN