



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

<b>The Process of Learning .....</b>	<b>3</b>
Aly ElDeen Program .....	3
Nagham Program .....	9
Fatema El-Zhraa Program .....	14
<b>Low Code No Code Tool .....</b>	<b>22</b>
Evaluation for LCNC Tools .....	22
The potential of LCNC Tools.....	22
What LCNC Tools Can Do .....	23
Benefits of LCNC Tools .....	23
Will LCNC Tools Take the Job of Developers? .....	24
<b>Compare between The two LCNC Tools .....</b>	<b>25</b>

## The Process of Learning :

### Aly El-Deen Yasser Aly:

- Hours of Study: 18 hours
- Source of study

[https://youtube.com/playlist?list=PLCIInYL3l2AajYlZGzU\\_LVrHdoouf8W6ZN&si=tfRlF2iNxOtJehum](https://youtube.com/playlist?list=PLCIInYL3l2AajYlZGzU_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();
        }
    }
}
```

```

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

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

```

- Images For the code:

```

public class Main {
    public static void main(String[] args) {
        // Initialize manager list and add a default manager
        ArrayList<Manager> managers = new ArrayList<>();
        managers.add(new Manager( name:"Aly", salary: 20000, FieldType: "General", Username: "aly", Password: "11114"));

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

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

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

```

```

package Workers;

/**
 * The {@code Worker} class represents an employee with basic attributes such as
 * name, salary, and field type. It provides methods to retrieve worker information
 * and adjust the salary.
 *
 * <p>This class can serve as a base class for more specialized worker types
 * (e.g., Manager, JuniorWorker, SeniorWorker) using inheritance.</p>
 *
 * @author Aly El-Deen Yasser
 * @version 1.0
 */
public class Worker { 5 usages 4 inheritors
    private final String name; 2 usages
    private double salary; 3 usages
    /** The field or department type the worker belongs to (e.g., HRs, Sales and Marketing, PRs,
     * Web Developer, Data Scientists, APP Developers ).**/ 2 usages
    private final String fieldType; 2 usages
    /**
     * Constructs a new {@code Worker} with the specified name, salary, and field type.
     *
     * @param name The name of the worker.
     * @param salary The initial salary of the worker.
     * @param fieldType The department or field the worker belongs to.
     */
    public Worker(String name, double salary, String fieldType) { 2 usages
        this.name = name;
        this.fieldType = FieldType;
        this.salary = salary;
    }
}

```

The screenshot shows the IntelliJ IDEA interface with the project navigation bar at the top. The current file is `Normal.java`, which is part of the `Workers` package. The code defines a class `Normal` that extends `Worker`. The class has a private attribute `Penalties` and a private attribute `Bounces`. The code includes detailed Javadoc comments describing the class's features and its relationship to the `Worker` class.

```
1 /**
2 * The {@code Normal} class extends the {@link Worker} class and represents a regular worker
3 * with advanced task management, penalties, bonuses, and salary adjustment features.
4 *
5 * <p><strong>Features:</strong></p>
6 * <ul>
7 *   <li>Manage tasks with descriptions, priorities, bounces, and deadlines.</li>
8 *   <li>Enforce maximum task limits per worker.</li>
9 *   <li>Apply penalties (as percentage deductions) and bonuses to the salary.</li>
10 *   <li>Validate user inputs for all task attributes, ensuring data consistency.</li>
11 *   <li>Adjust salary through penalties, bonuses, and raises.</li>
12 *   <li>Comprehensive checkout system to finalize payments based on tasks.</li>
13 * </ul>
14 *
15 * @version 1.5
16 * @author Aly El-Deen Yasser Ali
17 */
18
19 package Workers.Normal;
20
21 import Tasks.Tasks;
22 import Workers.Worker;
23
24 import java.time.LocalDate;
25 import java.time.format.DateTimeFormatter;
26 import java.time.format.DateTimeParseException;
27 import java.util.ArrayList;
28 import java.util.Objects;
29 import java.util.Scanner;
30
31 // Edit | Explain | Test | Document | Fix
32 public class Normal extends Worker {
33     private double Penalties = 0, Bounces = 0; 5 usages
34
35 }
```

The screenshot shows the IntelliJ IDEA interface with the project navigation bar at the top. The current file is `Manger.java`, which is part of the `Workers` package. The code defines a class `Manger` that extends `Worker`. The class has private attributes `Username` and `Password`. The code includes detailed Javadoc comments describing the class's authentication features and its relationship to the `Worker` class.

```
1 package Workers.Manger;
2
3 import Workers.Worker;
4
5 // Edit | Explain | Test | Document | Fix
6 /**
7 * The {@code Manger} class extends the {@link Worker} class and represents a manager
8 * with additional authentication details such as a username and password.
9 *
10 * @author Aly El-Deen Yasser Ali
11 * @version 1.0
12 */
13
14 public class Manger extends Worker { 10 usages
15     /** The manager's username and password used for authentication.**/
16     private String Username, Password; 2 usages
17
18     // Edit | Explain | Test | Document | Fix
19     /**
20      * Constructs a new {@code Manger} with the specified details.
21      *
22      * @param name      The manager's name.
23      * @param salary    The manager's salary.
24      * @param FieldType The field or department the manager belongs to.
25      * @param Username  The manager's username for login.
26      * @param Password  The manager's password for login.
27      */
28
29     public Manger(String name, double salary, String FieldType, String Username, String Password) { 2 usages
30         super(name, salary, FieldType);
31         this.Username = Username;
32         this.Password = Password;
33     }
34
35     public String getUsername() { 2 usages
36         return Username;
37     }
38 }
```

The screenshot shows the IntelliJ IDEA interface with the project 'Assignment 1' open. The left sidebar displays the project structure, including a 'src' folder containing 'Board', 'Tasks', 'Workers', 'Main', '.gitignore', and 'Assignment 1.iml'. The 'Tasks' folder is currently selected. The main editor window shows the 'Tasks.java' file with the following code:

```
1 package Tasks;
2
3 import java.time.LocalDate;
4
5 /**
6  * The {@code Tasks} class represents a task assigned to a worker, including its description,
7  * bonus, priority, status, and deadline.
8  *
9  * <p>It provides methods to manage task details, update status, and check deadlines.</p>
10 *
11 * @author Aly El-Deen Yasser
12 * @version 1.1
13 */
14 public class Tasks { 5 usages
15     private static int idCounter = 1; 1 usage
16     int taskId = idCounter++; 1 usage
17     private String description; 2 usages
18     private double bonus; 2 usages
19     private String priority; 2 usages
20     private LocalDate deadline; 3 usages
21
22     /**
23      * Constructs a new {@code Tasks} object with the specified details.
24      *
25      * @param description A brief description of the task.
26      * @param bonus The bonus amount associated with the task.
27      * @param priority The priority level of the task.
28      * @param deadline The deadline for task completion.
29      */
30     public Tasks( String description, double bonus, String priority, LocalDate deadline) { 1 usage
31         this.description = description;
32         this.bonus = bonus;
33         this.priority = priority;
34     }
35
36     /**
37      * Constructor to initialize a Normal worker.
38      *
39      * @param Name The worker's name.
40      * @param Salary The base salary.
41      * @param FieldType The department or field.
42      * @param MaxTasks The maximum number of tasks.
43      */
44     public Normal(String Name, double Salary, String FieldType, long MaxTasks) { 2 usages
45         super(Name, Salary, FieldType);
46         this.maxTasks = MaxTasks;
47         id = lastId++;
48     }
49
50     /**
51      * Displays a menu and returns a validated user choice.
52      *
53      * @param options The menu options.
54      * @param Menu The menu title.
55      * @return The validated choice.
56      */
57     public static int getValidChoice(String[] options, String Menu) { 12 usages
58         Scanner sc = new Scanner(System.in);
59         int choice = -1;
60     }
61
62 }
```

The status bar at the bottom indicates the file is 'tobnine Basic' with 57:6 lines, using CRLF line endings, and 4 spaces for indentation.

The screenshot shows the IntelliJ IDEA interface with the project 'Assignment 1' open. The left sidebar displays the project structure, including a 'src' folder containing 'Board', 'Tasks', 'Workers', 'Main', '.gitignore', and 'Assignment 1.iml'. The 'Workers' folder is currently selected. The main editor window shows the 'Normal.java' file with the following code:

```
1 public class Normal extends Worker { 20 usages 2 inheritors
2
3     private double Penalties = 0, Bounces = 0; 5 usages
4     private long numberTasks = 0, maxTasks = 0; 5 usages
5     // adding id
6     private static int lastId = 0; 1 usage
7     private int id; 2 usages
8     private ArrayList<Tasks> TasksToDo = new ArrayList<Tasks>(); 11 usages
9
10    /**
11     * Constructor to initialize a Normal worker.
12     *
13     * @param Name The worker's name.
14     * @param Salary The base salary.
15     * @param FieldType The department or field.
16     * @param MaxTasks The maximum number of tasks.
17     */
18    public Normal(String Name, double Salary, String FieldType, long MaxTasks) { 2 usages
19        super(Name, Salary, FieldType);
20        this.maxTasks = MaxTasks;
21        id = lastId++;
22    }
23
24    /**
25     * Displays a menu and returns a validated user choice.
26     *
27     * @param options The menu options.
28     * @param Menu The menu title.
29     * @return The validated choice.
30     */
31    public static int getValidChoice(String[] options, String Menu) { 12 usages
32        Scanner sc = new Scanner(System.in);
33        int choice = -1;
34    }
35
36 }
```

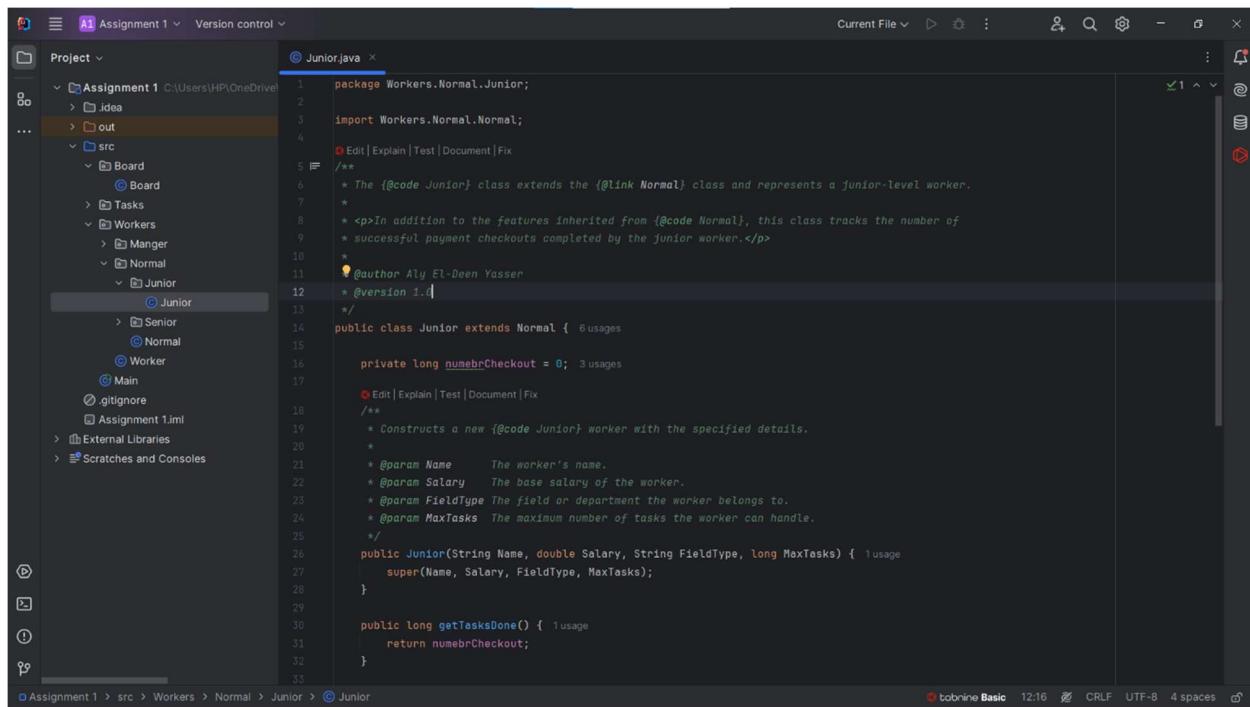
The status bar at the bottom indicates the file is 'tobnine Basic' with 233:61 lines, using CRLF line endings, and 4 spaces for indentation.

The screenshot shows the IntelliJ IDEA interface with the project 'Assignment 1' open. The file 'Senior.java' is selected in the editor. The code defines a class 'Senior' that extends 'Normal'. It includes annotations for parameters and methods, and a constructor that initializes 'yearsExperience'. The 'showAllInfo()' method is overridden to print the years of experience.

```
1 package Workers.Normal.Senior;
2
3 import Workers.Normal.Normal;
4
5 public class Senior extends Normal {
6     private long yearsExperience;
7
8     /**
9      * Constructs a new {@code Normal} worker with the specified details.
10     *
11     * @param Name      The worker's name.
12     * @param Salary    The base salary of the worker.
13     * @param FieldType The field or department the worker belongs to.
14     * @param MaxTasks The maximum number of tasks the worker can handle.
15     */
16     public Senior(String Name, double Salary, String FieldType, long MaxTasks, long YearsExperience) {
17         super(Name, Salary, FieldType, MaxTasks);
18         this.yearsExperience = YearsExperience;
19     }
20
21     /**
22      * Displays all worker info and adds completed checkouts for juniors.
23     */
24     @Override
25     public void showAllInfo() {
26         super.showAllInfo();
27         System.out.println("Years of experience: " + yearsExperience);
28     }
29
30 }
31
```

The screenshot shows the IntelliJ IDEA interface with the project 'Assignment 1' open. The file 'Board.java' is selected in the editor. The code defines a class 'Board' that manages workers, tasks, and penalties. It includes imports for various worker classes and utility classes like ArrayList and Scanner. The constructor initializes lists for Juniors, Managers, and Seniors.

```
1 /**
2  * The Board class manages the operations of Juniors, Seniors, and Managers within an organization.
3  * It includes functionalities to add workers, assign tasks, apply penalties, give raises, promote Juniors,
4  * and authenticate managers for administrative operations.
5 */
6
7 package Board;
8
9 import Workers.Manger.Manger;
10 import Workers.Normal.Junior.Junior;
11 import Workers.Normal.Normal;
12 import Workers.Normal.Senior.Senior;
13
14 import java.util.ArrayList;
15 import java.util.List;
16 import java.util.Scanner;
17
18 public class Board {
19     private ArrayList<Junior> juniors;
20     private ArrayList<Manger> mangers;
21     private ArrayList<Senior> seniors;
22     Scanner sc = new Scanner(System.in);
23
24     /**
25      * Constructor to initialize Board with existing lists of Juniors, Managers, and Seniors.
26     */
27     public Board(ArrayList<Junior> juniors, ArrayList<Manger> mangers, ArrayList<Senior> Seniors) {
28         this.juniors = juniors;
29         this.mangers = mangers;
30         this.seniors = Seniors;
31     }
32 }
```



The screenshot shows a Java code editor in an IDE. The project structure on the left is as follows:

- Assignment 1
- src
  - Board
  - Tasks
  - Workers
    - Manger
    - Normal
      - Junior
      - Senior
  - Main
- .gitignore
- Assignment 1.iml
- External Libraries
- Scratches and Consoles

The current file is `Junior.java` located in the `Workers.Normal` package. The code is as follows:

```
1 package Workers.Normal.Junior;
2
3 import Workers.Normal.Normal;
4
5 /**
6  * The {@code Junior} class extends the {@link Normal} class and represents a junior-level worker.
7  *
8  * <p>In addition to the features inherited from {@code Normal}, this class tracks the number of
9  * successful payment checkouts completed by the junior worker.</p>
10 *
11 * @author Aly El-Deen Yasser
12 * @version 1.0
13 */
14 public class Junior extends Normal {
15     private long numbrCheckout = 0;
16
17     /**
18      * Constructs a new {@code Junior} worker with the specified details.
19      *
20      * @param Name      The worker's name.
21      * @param Salary    The base salary of the worker.
22      * @param FieldType The field or department the worker belongs to.
23      * @param MaxTasks  The maximum number of tasks the worker can handle.
24      */
25     public Junior(String Name, double Salary, String FieldType, long MaxTasks) {
26         super(Name, Salary, FieldType, MaxTasks);
27     }
28
29     public long getTasksDone() {
30         return numbrCheckout;
31     }
32 }
```

The status bar at the bottom indicates the file is in Cobol Basic mode, was saved at 12:16, uses CRLF line endings, is in UTF-8 encoding, and has 4 spaces per indentation.

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&s\\_i=tfRlF2iNxOtJehum](https://youtube.com/playlist?list=PLCInYL3l2AajYlZGzU_LVrHdoouf8W6ZN&s_i=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 :

The image displays two screenshots of a Java code editor, likely Microsoft Visual Studio Code, showing the code for a file named Main.java. The code is part of a project named 'to\_do\_list'.

**Top Screenshot:**

```

public class Main {
    ...
    //edit task through the edit task menu option
    private static void editTask() { 1usage
        //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) {
                            break;
                        }
                    } catch (NumberFormatException e) {
                        System.out.println("Invalid input. Please enter a valid task number.");
                    }
                }
                task.setName(name);
                task.setDescription(description);
                task.setPriority(priority);
            }
        } catch (InputMismatchException e) {
            System.out.println("Error: Invalid input. Please enter a valid task number.");
        }
    }
}

```

**Bottom Screenshot:**

```

public class Main {
    ...
    //display task details through display task menu option
    private static void displayTaskDetails() { 1usage
        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.");
        }
    }
}

```

```
public class Main {
    ...
    //take task info through the add task menu option
    private static void addTask() { 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 {
                dueDate = LocalDate.parse(dateInput);
                break;
            } catch (DateTimeParseException e) {
                System.out.println("Invalid date format. Please enter a valid date (dd-mm-yyyy).");
            }
        }
    }

    public static void main(String[] args) {
        ...
        //function for printing the menu
        private static void printMenu() { 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() { 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) {

```

```
22  public class Main {  
23      //the program's starting point(main function)  
24      public static void main(String[] args) {  
...  
108      boolean running = true;  
109  
110      //display menu while the program is running(user didn't choose exit)  
111      while (running) {  
112          printMenu();  
113          String choice = scanner.nextLine().trim();  
114          switch (choice) {  
115              case "1":  
116                  addTask();  
117                  break;  
118              case "2":  
119                  removeTask();  
120                  break;  
121              case "3":  
122                  displayTaskDetails();  
123                  break;  
124              case "4":  
125                  editTask();  
126                  break;  
127              case "5":  
128                  sortTasks();  
129                  break;  
130              case "6":  
131                  displayList();  
132                  break;  
133              case "7":  
134                  clearList();  
135                  break;  
136              case "8":  
137                  break;  
138          }  
139      }  
140  }  
141  
142  static class Task { 8 usages  
143      //getters  
144      public String getName() { return name; }  
145  
146      //this is not used yet but is here for further development in the future  
147      public String getDescription() { return description; }  
148  
149      public int getPriority() { return priority; }  
150  
151      public LocalDate getDueDate() { return dueDate; }  
152  
153      public void setName(String name) { this.name = name; }  
154  
155      public void setDescription(String description) { this.description = description; }  
156  
157      public void setPriority(int priority) { this.priority = priority; }  
158  
159      public void setDueDate(LocalDate dueDate) { this.dueDate = dueDate; }  
160  
161      @Override  
162      public String toString() {  
163          String priorityString;  
164          switch(priority){  
165              case 1:  
166                  priorityString = "Low Priority";  
167                  break;  
168              case 2:  
169                  priorityString = "Medium Priority";  
170                  break;  
171              case 3:  
172                  priorityString = "High Priority";  
173                  break;  
174          }  
175          return "Task{" + "Name: " + name + ", Description: " + description + ", Priority: " + priorityString + ", Due Date: " + dueDate + '}';  
176      }  
177  }  
178  
179  public class Main {  
180      static class Task { 8 usages  
181          //getters  
182          public String getName() { return name; }  
183  
184          //this is not used yet but is here for further development in the future  
185          public String getDescription() { return description; }  
186  
187          public int getPriority() { return priority; }  
188  
189          public LocalDate getDueDate() { return dueDate; }  
190  
191          public void setName(String name) { this.name = name; }  
192  
193          public void setDescription(String description) { this.description = description; }  
194  
195          public void setPriority(int priority) { this.priority = priority; }  
196  
197          public void setDueDate(LocalDate dueDate) { this.dueDate = dueDate; }  
198  
199          @Override  
200          public String toString() {  
201              String priorityString;  
202              switch(priority){  
203                  case 1:  
204                      priorityString = "Low Priority";  
205                      break;  
206                  case 2:  
207                      priorityString = "Medium Priority";  
208                      break;  
209                  case 3:  
210                      priorityString = "High Priority";  
211                      break;  
212              }  
213              return "Task{" + "Name: " + name + ", Description: " + description + ", Priority: " + priorityString + ", Due Date: " + dueDate + '}';  
214          }  
215      }  
216  }
```

The screenshot shows a Java code editor with the file `Main.java` open. The code defines a `Task` class and a `Main` class. The `Task` class has private fields for name, description, priority, and dueDate, and a constructor that initializes these fields. The `Main` class contains a static block that initializes a `LocalDate` object and a `Scanner` object. The code also includes imports for `java.io`, `java.time`, and `java.util`.

```
//Author: Nagham Wael Mohamed Elsayed
//ID: 29231189
//version: V1.0
//First Modified: 22 Feb 2025
//Last Modified: 24 Feb 2025
//Purpose: a to-do list console program
//Features: Add a task , Remove a task , Display task details , Edit a task , Sort tasks , Display list , Clear list , Save list to a file

import java.io.FileWriter;
import java.io.IOException;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Scanner;

public class Main {

    // Task class definition
    static class Task {
        private String name; 4 usages
        private String description; 4 usages
        private int priority; // 1 = low, 2 = average, 3 = high 4 usages
        private LocalDate dueDate; 4 usages

        public Task(String name, String description, int priority, LocalDate dueDate) { 1 usage
            this.name = name;
        }
    }

    static {
        LocalDate.now();
        Scanner scanner;
    }
}
```

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](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\nnEnter your choice :
",Choices);

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

    }

    return false;
}

isFirst=false;
return true;

}
```

## Screen Shots :

**Main.java**

```

1 //Author: Fatma El-Zhraa Ahmed Mohamed Elfiky
2 //ID: 20230280
3
4 //version: V1.0
5 //First Modified: 24 Feb 2025
6 //Last Modified: 25 Feb 2025
7
8 //Purpose: a Contact Management System
9
10 /**
11  * Add a contact
12  * delete a contact by phone number ,
13  * Display contacts
14  * save file
15  * search for contact by name or by phone number
16 */
17
18 public class Main {
19     public static void main(String[] args) {
20
21         System.out.println("##Welcome to the contact management system ##\n\n");
22
23         // the class is managing the flow of the program
24         UI Userinterface= new UI();
25         while (UserInterface.MainMenu());
26
27         System.out.println("##Thanks for using our program##");
28
29     }
30
31 }

```

**UI.java**

```

1 import ContactsPackage.*;
2
3
4 import java.util.ArrayList;
5 import java.util.Arrays;
6 import java.util.Scanner;
7
8 public class UI {
9
10     static private boolean isSaved=false; 5 usages
11     static private boolean isFirst=true; 2 usages
12     private ContactManager manager; 8 usages
13     private String NameOfFile; no usages
14
15     private String check_menu( String menuText , ArrayList<String> choices){...}
16
17
18     public boolean MainMenu(){ 1 usage
19
20         Scanner in = new Scanner(System.in);
21
22         if (isFirst)
23             manager=new ContactManager();
24
25
26         ArrayList<String> choices=new ArrayList<String>(Arrays.asList("1","2","3","4","5","6"));
27         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
28
29     }
30
31 }

```

CS Contact System Version control

```

public class UI {
    public boolean MainMenu() {
        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 all contacts\n5.Save contacts in file\n6.Exit");
        if(choice.equals("1")){
            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")){
            if(manager.Delete())
                isSaved=false;
        }
        else if( choice.equals("3")){
            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
                    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")){
            manager.Display();
        }
        else if( choice.equals("5")){
            if(!isSaved)
                manager_Save();
        }
    }
}

```

123:1 CRLF UTF-8 4 spaces 6:29 PM 2/25/2025

CS Contact System Version control

```

public class UI {
    public boolean MainMenu() {
        // 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")){
            manager.Display();
        }
        else if( choice.equals("5")){
            if(!isSaved)
                manager_Save();
        }
    }
}

```

123:1 CRLF UTF-8 4 spaces 6:29 PM 2/25/2025

The image displays two side-by-side screenshots of a Java code editor, likely IntelliJ IDEA, showing the same code for a `Contact` class. Both screenshots show the code in a dark-themed interface.

**Project Structure:**

- Project: Contact System
- src folder contains:
  - file.txt
  - ContactManager.java
  - Main.java
  - contact.java (highlighted)
  - Ui.java
- ContactsPackage folder contains:
  - contact.java
  - ContactManager.java
  - file.txt
  - Main.java
  - Ui.java
- Other files: .gitignore, Contact System.iml, External Libraries, Scratches and Console

**Code (Contact.java):**

```
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){ 3 usages
        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);
    }
}
```

**Code (Contact.java - Second Screenshot):**

```
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){ 3 usages
        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);
    }
}
```

**Environment:**

- Operating System: Windows (taskbar icons)
- Time: 6:30 PM
- Date: 2/25/2025
- IDE Version: 32:18 CRLF UTF-8 4 spaces

The image shows two side-by-side screenshots of a Java IDE interface, likely IntelliJ IDEA, displaying two files: ContactManager.java and Main.java.

**ContactManager.java (Top Window):**

```
1 package ContactsPackage;
2
3 import java.io.IOException;
4
5 import java.util.ArrayList;
6 import java.util.Scanner;
7
8 import java.nio.file.*;
9 import java.io.*;
10
11
12 public class ContactManager {
13
14     private ArrayList<contact>Contacts; 19 usages
15     private String File; 2 usages
16
17     //Validation codes for data and files
18
19     @
20
21     private String validNum(String num){ 2 usages
22
23         Scanner in =new Scanner(System.in) ;
24         String NumberRegex = "^(\\D\\\\d{9})\\\\\\d{3}$";
25
26
27         while (!num.matches(NumberRegex)){
28             System.out.print("Please enter correct format of phone number : ");
29             num= in.nextLine();
30         }
31
32         return num;
33     }
34 }
```

**Main.java (Bottom Window):**

```
14 public class ContactManager { 2 usages
32
33
34     private String validName(String name){ 2 usages
35
36         Scanner in =new Scanner(System.in) ;
37         String NameRegex ="[A-Za-z -]$";
38
39
40         while (name.trim().isEmpty()||!name.matches(NameRegex)){
41             System.out.print("Please Enter a valid name : ");
42             name= in.nextLine();
43         }
44
45         return name;
46     }
47
48     private String validEmail(String email){ 1 usage
49
50         Scanner in = new Scanner(System.in);
51
52         String emailPattern = "[a-zA-Z0-9.%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
53
54         while (!email.matches(emailPattern)){
55             System.out.print("Please enter correct email : ");
56             email= in.nextLine();
57         }
58
59         return email;
60     }
61
62     private boolean ValidationData(String txt , int stage){ 3 usages
63
64
65 }
```

The screenshot shows a Java development environment with two code editors displaying the same file, `ContactManager.java`, from a project named "Contact System".

**Top Editor Content:**

```
public class ContactManager { 2 usages
}
private boolean ValidationData(String txt , int stage){ 3 usages
    String regex ;
    if(stage==1)
        regex = "(^([01]\\d{9})\\d{5}$";
    else if (stage==2)
        regex ="^[-\\w+ -]*$";
    else
        regex = "^[a-zA-Z0-9_.%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
    return txt.matches(regex);
}

// Loading at the beginning of the program the file
private boolean LoadData(){ 1 usage
    //String NameOfFile =FileManagement();
    Scanner in = new Scanner(System.in);

    System.out.println("Please Enter the file name : ");
    String NameOfFile = in.nextLine();
    Path path = Paths.get(NameOfFile);

    if (!Files.exists(path) || !Files.isRegularFile(path)) {
        System.out.println("File does not exist or is not a regular file");
        return false;
    }
    try {
        BufferedReader reader = Files.newBufferedReader(path);
        String line;
        while ((line = reader.readLine()) != null) {
            String[] parts = line.split(",");
            if (parts.length == 3) {
                contact.add(new Contact(parts[0], parts[1], parts[2]));
            } else {
                System.out.println("Line format is incorrect: " + line);
            }
        }
        reader.close();
    } catch (IOException e) {
        System.out.println("Error reading file: " + e.getMessage());
        return false;
    }
    return true;
}
```

**Bottom Editor Content:**

```
public class ContactManager { 2 usages
}
private boolean LoadData(){ 1 usage
    File=NameOfFile;
    if(line==null)
        return true;
    Contacts=new ArrayList<>();
    while (line!=null){
        String name="", phone="", email="";
        StringBuilder transition= new StringBuilder();
        boolean needAppend=true;
        for (int index = 0, order=0; index < line.length(); index++) {
            char c = line.charAt(index);
            if(c==','){
                order++;
                if (order==1)
                    name=transition.toString();
                else if (order==2)
                    phone=transition.toString();
                else
                    email= transition.toString();
                transition.setLength(0);
                needAppend=false;
            }
            transition.append(c);
        }
        if(needAppend)
            Contacts.add(new Contact(name, phone, email));
        line=reader.readLine();
    }
    reader.close();
    return true;
}
```

## Video Link :

[https://www.youtube.com/watch?v=fuV9C\\_FRDf8](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.

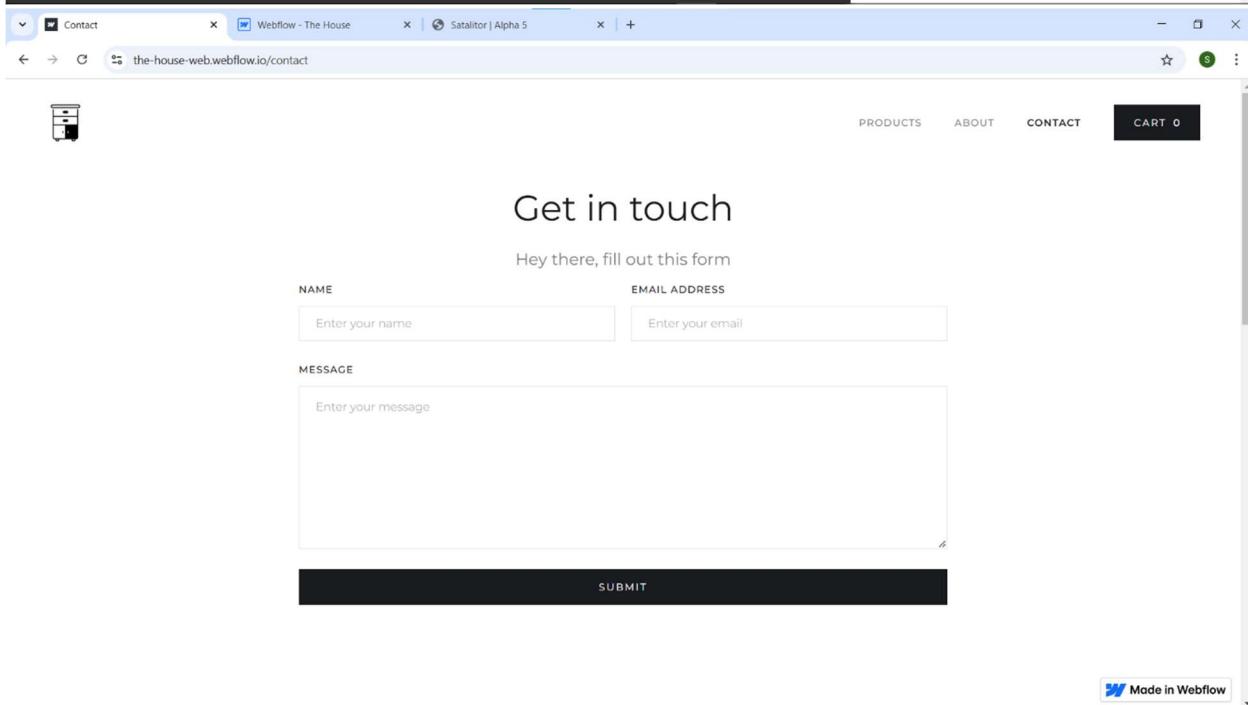
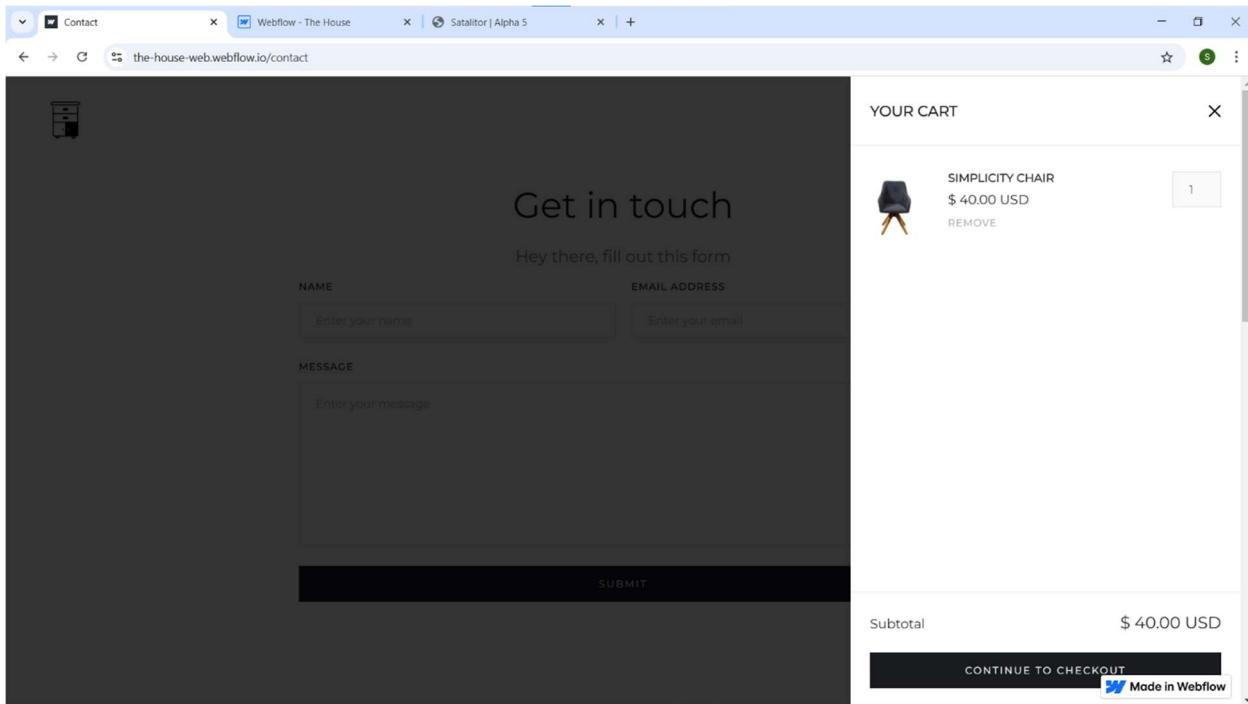
### **As for Webflow:**

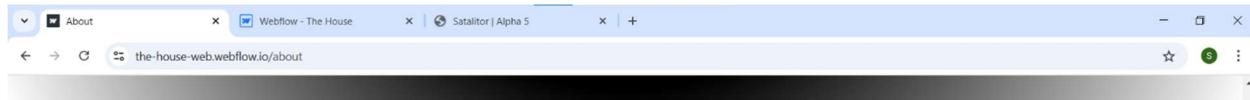
Webflow empowers designers and marketers to build and maintain websites without developers, but for highly dynamic and database-driven applications, developers are still essential. Webflow acts as a tool that enhances productivity rather than replacing professional developers.

## 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 :





## 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

### Monthly Newsletter

Sign up to receive updates from our shop, including new tea selections and upcoming events.

Powered by Webflow

Made in Webflow

MENU	CATEGORIES	HELP	FOLLOW
Home	Chairs	Shipping	Instagram
About	Sofas	Returns & Exchange	Facebook
Contact	Tables	Product Care	Twitter
Products			

The screenshot shows a web browser window with three tabs: 'My House Store | Main Page', 'Webflow - The House', and 'Satalitor | Alpha 5'. The main content area is titled 'Featured Products' with the sub-instruction 'Check out new and popular products'. It displays three furniture items: 'LA MALE SOFA' (a light-colored sofa against a dark wall), 'CHAIR 1' (a blue upholstered chair on a dark background), and 'SIMPLICITY CHAIR' (a blue upholstered chair on a wooden tripod base). A 'Made in Webflow' logo is visible in the bottom right corner.

## Who Are we ?

Modern Design With Best Price in world of Furniture

Modern Designs for Every Home

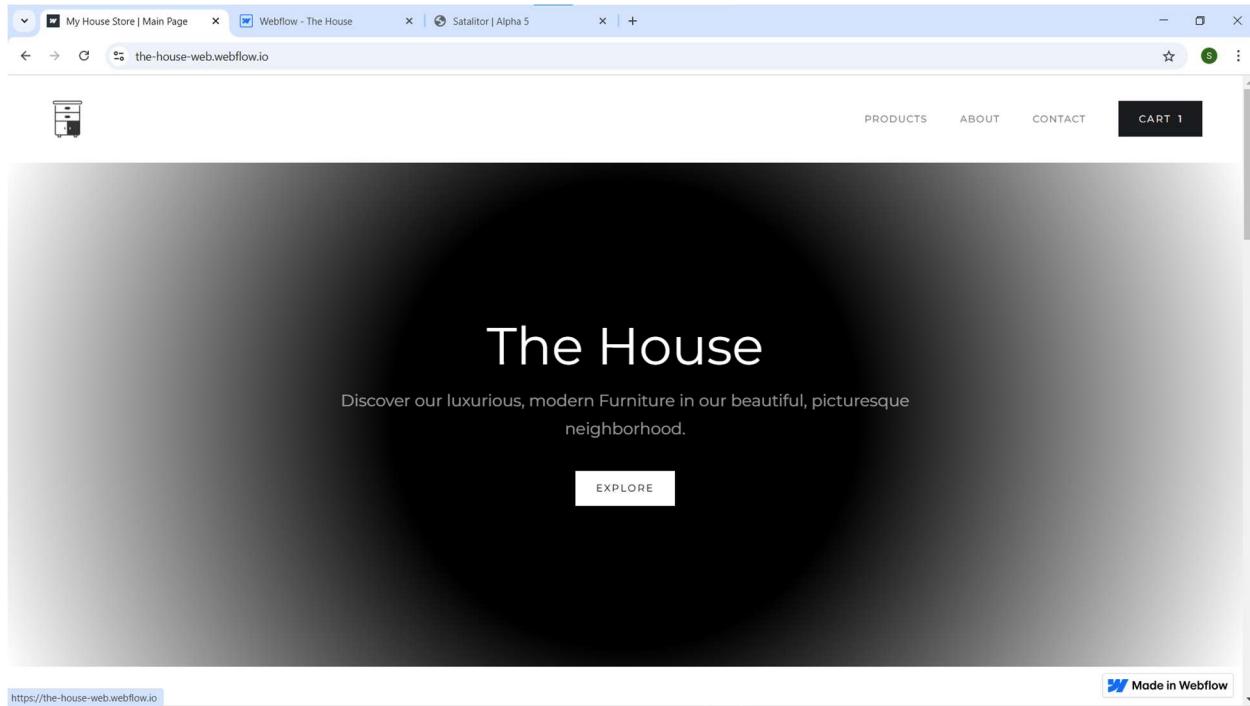
Explore our curated collection of contemporary furniture that combines style and functionality. From sleek sofas to elegant dining sets, we have everything to transform your living spaces.

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](https://youtu.be/c071WQAifNM?si=sWq_lknsMG3GbJbN)