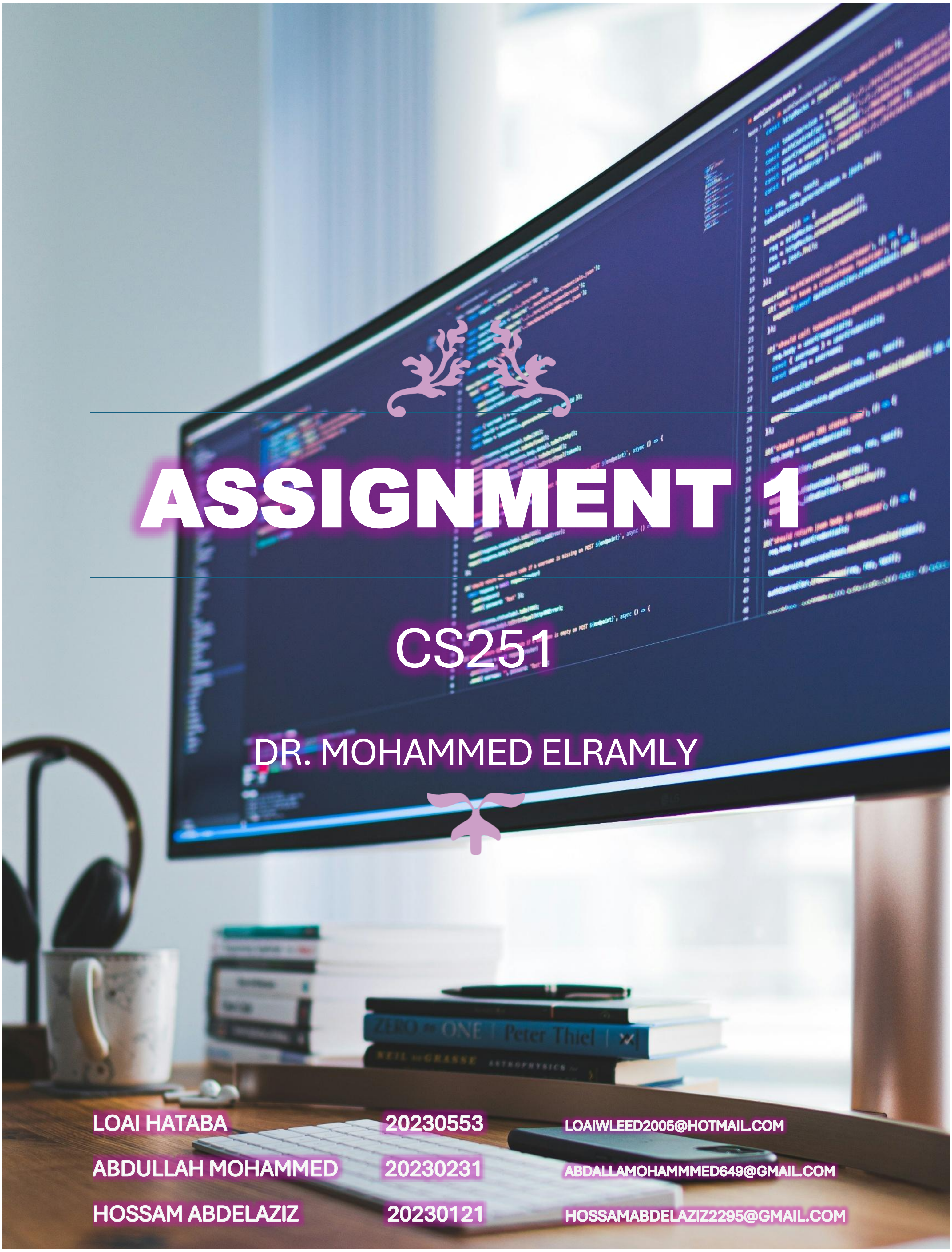


ASSIGNMENT 1

CS251

DR. MOHAMMED ELRAMLY

LOAI HATABA	20230553	LOAIWLEED2005@HOTMAIL.COM
ABDULLAH MOHAMMED	20230231	ABDALLAMOHAMMMED649@GMAIL.COM
HOSSAM ABDELAZIZ	20230121	HOSSAMABDELAZIZ2295@GMAIL.COM

Languages:

Loai → Java ☕

Abdullah → Java ☕

Hossam → Java ☕

Learning:

Name	Duration	Sources
Loai	8 Hours	https://youtube.com/playlist?list=PLJhTWoCm8I6DXaq7XECfyGKtsq4Z6fWZr&si=w_PmOmUhKEH6EyCl https://youtu.be/drQK8ciCAjY?si=cmx5cv4of_BQn4k5
Abdullah	4 Days	https://www.tutorialspoint.com/java/index.htm https://www.youtube.com/watch?v=mNvJipMTKSM&list=PLCInYL3l2AajYlZGzU_LVrHdoouf8W6ZN
Hossam	1 Day	https://www.youtube.com/watch?v=mNvJipMTKSM&list=PLCInYL3l2AajYlZGzU_LVrHdoouf8W6ZN https://www.youtube.com/playlist?list=PLJhTWoCm8I6DXaq7XECfyGKtsq4Z6fWZr

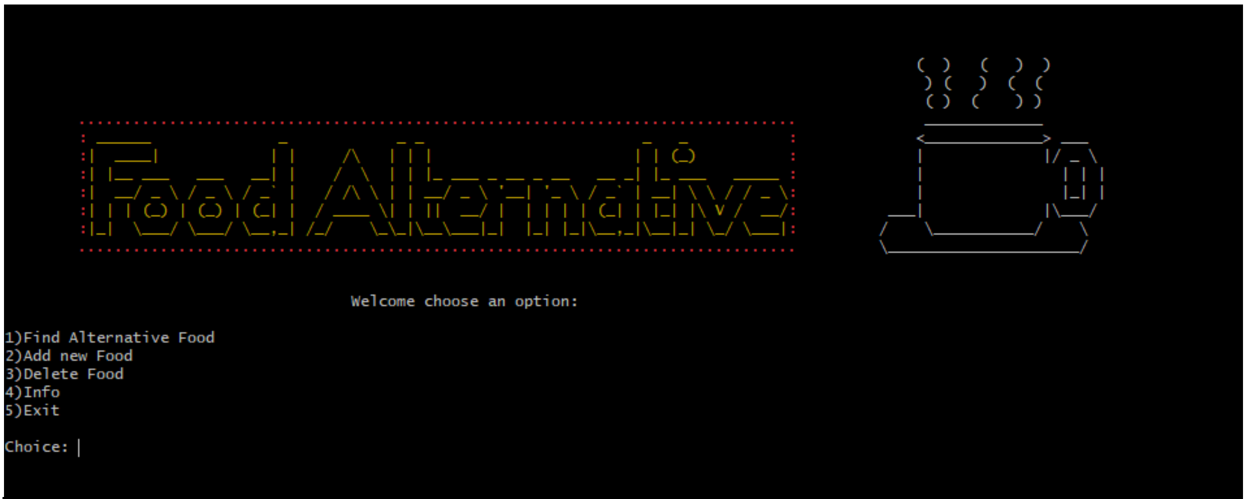
Food Alternative (App 1 Loai):

Main Function:

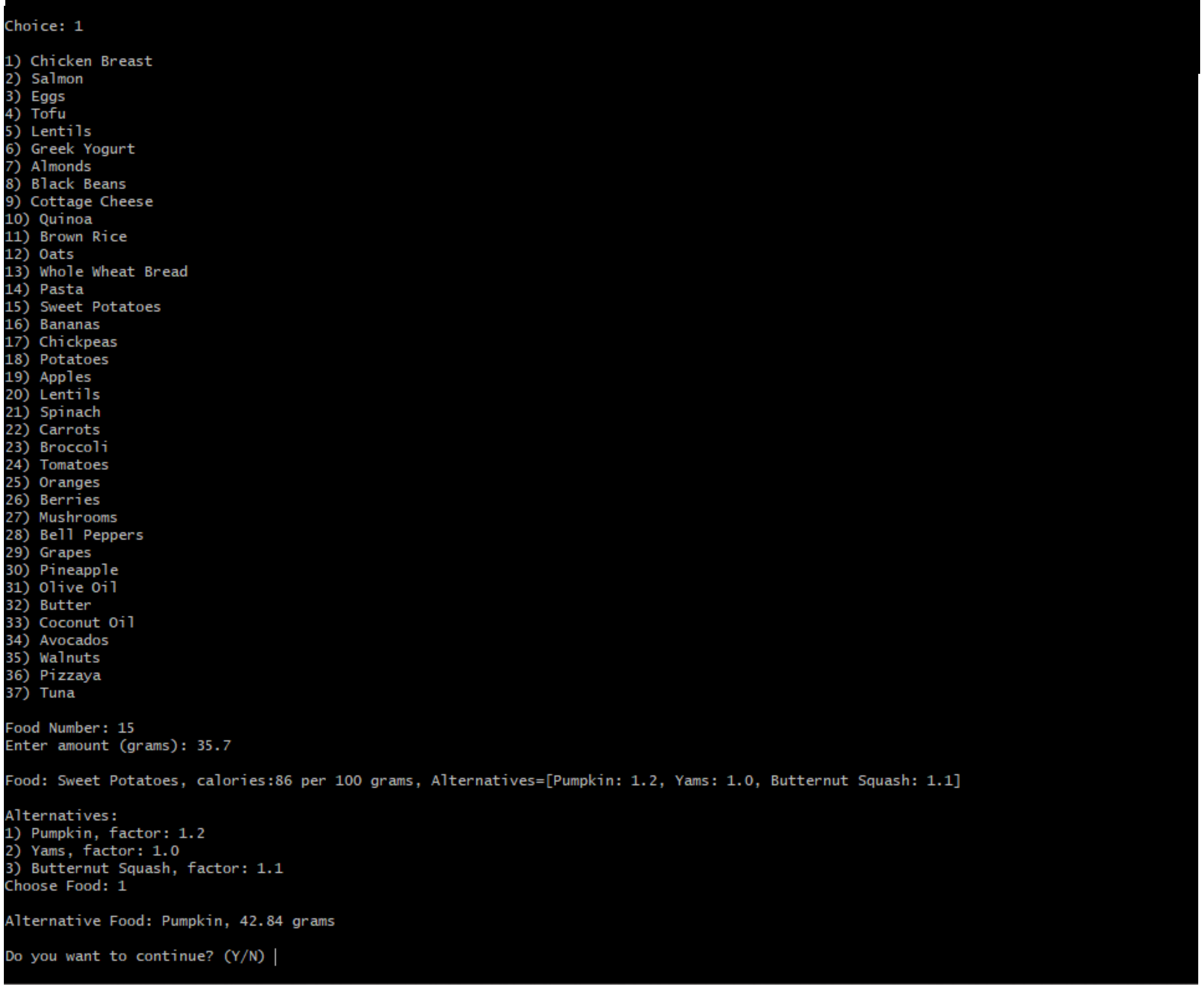
```
Scanner scanner = new Scanner(System.in);

// Load food from JSON

String jsonPath = "food/foodDictionary.json";

List<FoodItem> foodList = GsonTool.loadFood(jsonPath);

while (true){

      printBanner();

      int menu = optionsMenu(scanner);

      switch(menu)

      {

        // Alternative Food

        case 1:

          foodMenu(scanner, foodList);

          int ans = continueApp(scanner);

          if (ans == 0){

            scanner.close();

            System.exit(0);

          }

          break;

        //Add new Food

        case 2:

          addFood(scanner, foodList, jsonPath);

          int ans2 = continueApp(scanner);

          if (ans2 == 0){

            scanner.close();

            System.exit(0);

          }
```

```
            break;

          // Delete Food
          case 3:
            deleteFood(scanner, foodList, jsonPath);
            int ans3 = continueApp(scanner);
            if (ans3 == 0){
              scanner.close();
              System.exit(0);
            }
            break;

          case 4:
            prinInfoBanner();
            int ans4 = continueApp(scanner);
            if (ans4 == 0){
              scanner.close();
              System.exit(0);
            }
            break;

          case 5:
            System.out.println("\nGoodbye!!");
            scanner.close();
            System.exit(0);
            break;
        }
      }
    }
```

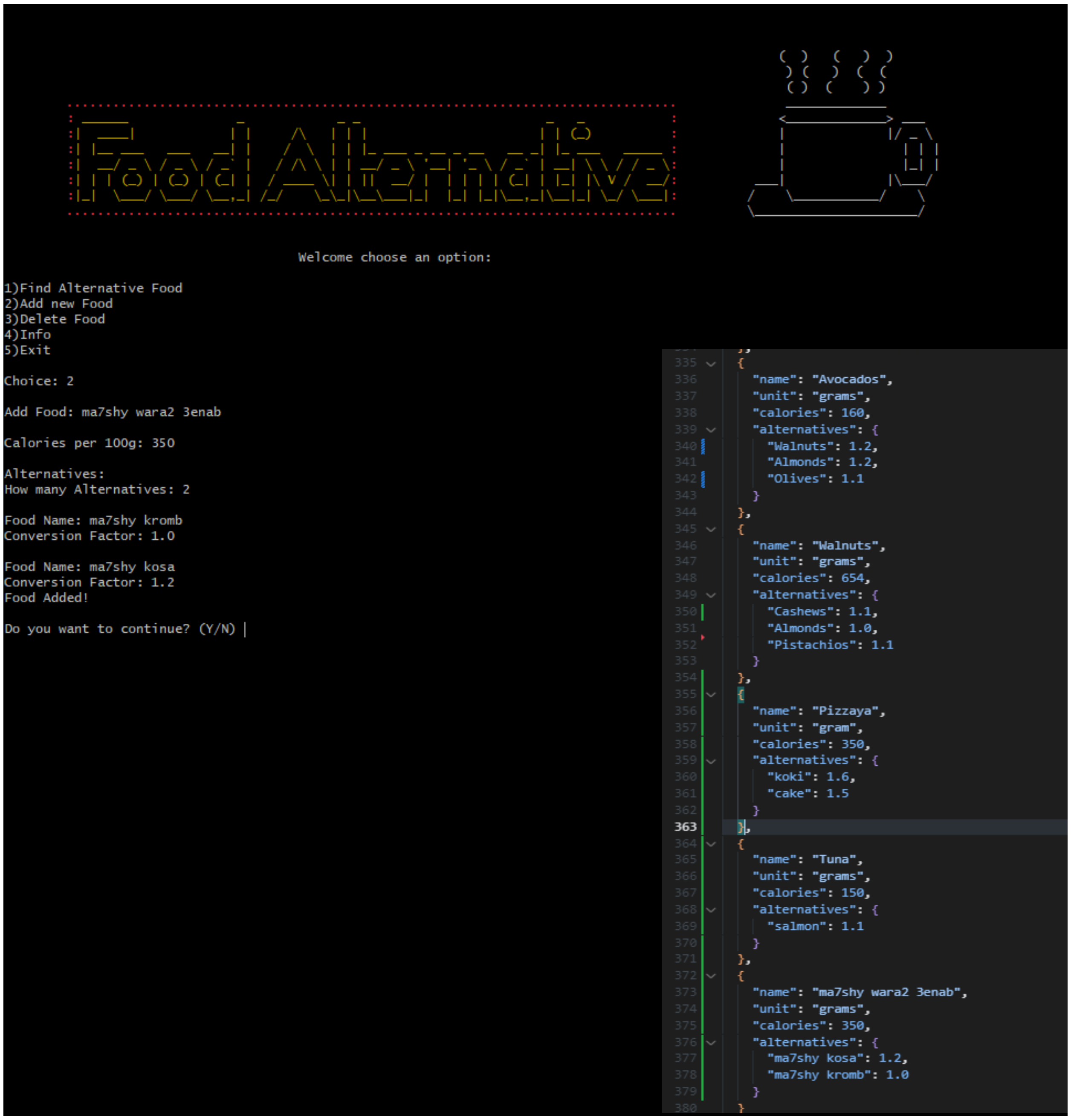
Screenshots:



```
                                   Welcome choose an option:

1)Find Alternative Food
2)Add new Food
3)Delete Food
4)Info
5)Exit

Choice: |
```

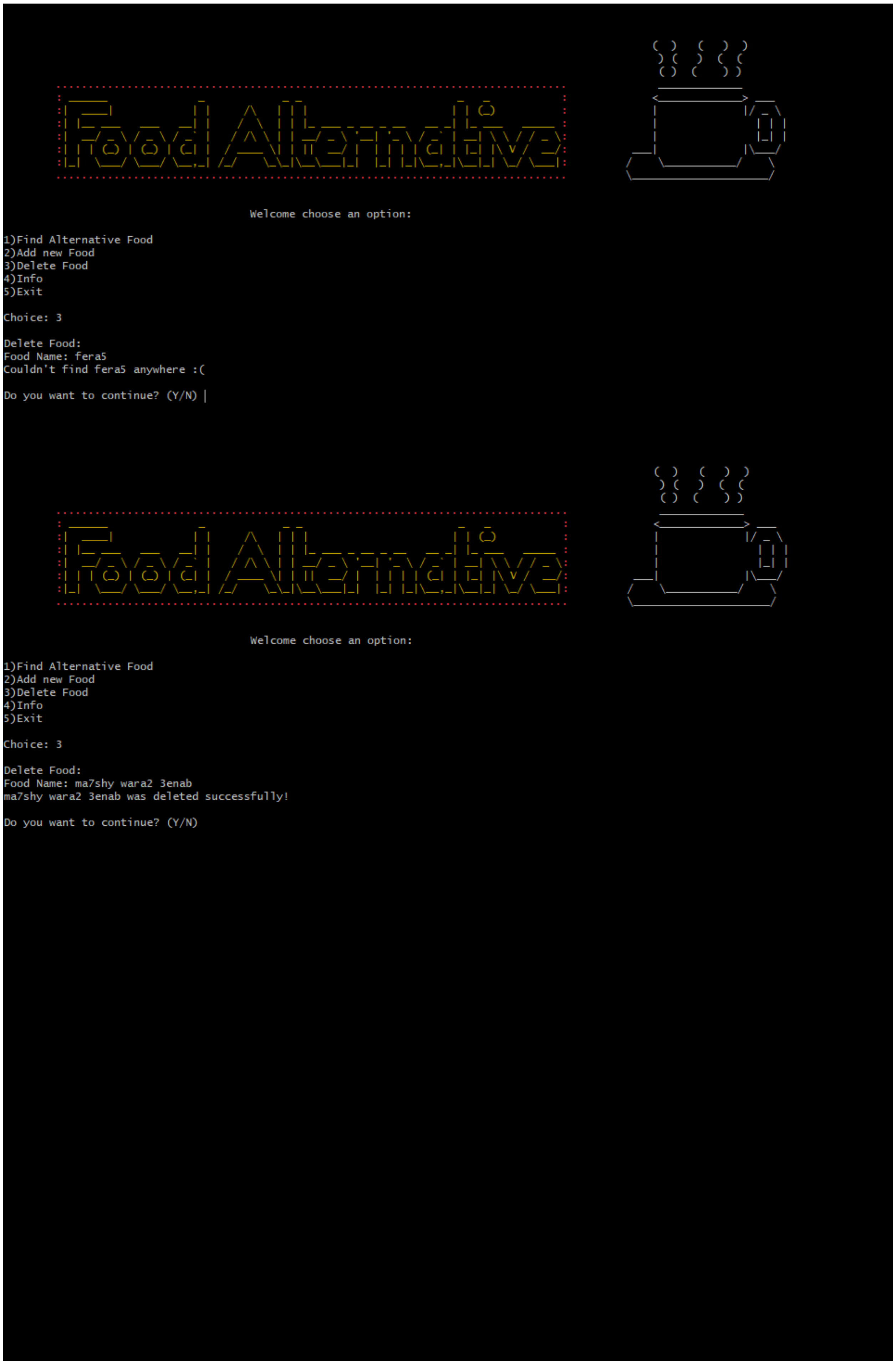


```
Choice: 1

1) Chicken Breast
2) Salmon
3) Eggs
4) Tofu
5) Lentils
6) Greek Yogurt
7) Almonds
8) Black Beans
9) Cottage Cheese
10) Quinoa
11) Brown Rice
12) Oats
13) Whole Wheat Bread
14) Pasta
15) Sweet Potatoes
16) Bananas
17) Chickpeas
18) Potatoes
19) Apples
20) Lentils
21) Spinach
22) Carrots
23) Broccoli
24) Tomatoes
25) Oranges
26) Berries
27) Mushrooms
28) Bell Peppers
29) Grapes
30) Pineapple
31) Olive Oil
32) Butter
33) Coconut Oil
34) Avocados
35) Walnuts
36) Pizzaya
37) Tuna

Food Number: 15
Enter amount (grams): 35.7

Food: Sweet Potatoes, calories:86 per 100 grams, Alternatives=[Pumpkin: 1.2, Yams: 1.0, Butternut Squash: 1.1]

Alternatives:
1) Pumpkin, factor: 1.2
2) Yams, factor: 1.0
3) Butternut Squash, factor: 1.1
Choose Food: 1

Alternative Food: Pumpkin, 42.84 grams

Do you want to continue? (Y/N) |
```



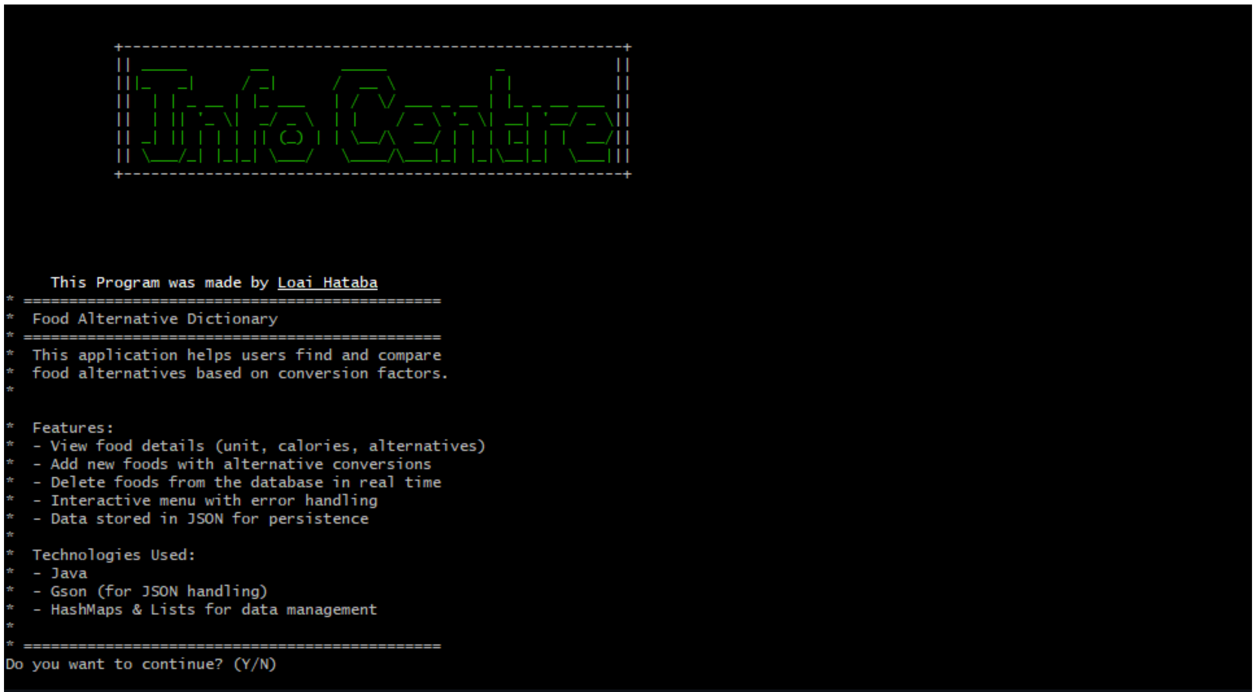
```
                              Welcome choose an option:

1)Find Alternative Food
2)Add new Food
3)Delete Food
4)Info
5)Exit

Choice: 2

Add Food: ma7shy wara2 3enab

Calories per 100g: 350

Alternatives:
How many Alternatives: 2

Food Name: ma7shy kromb
Conversion Factor: 1.0

Food Name: ma7shy kosa
Conversion Factor: 1.2
Food Added!

Do you want to continue? (Y/N) |
```

```json
  {
    "name": "Avocados",
    "unit": "grams",
    "calories": 160,
    "alternatives": {
      "Walnuts": 1.2,
      "Almonds": 1.2,
      "Olives": 1.1
    }
  },
  {
    "name": "Walnuts",
    "unit": "grams",
    "calories": 654,
    "alternatives": {
      "Cashews": 1.1,
      "Almonds": 1.0,
      "Pistachios": 1.1
    }
  },
  {
    "name": "Pizzaya",
    "unit": "gram",
    "calories": 350,
    "alternatives": {
      "koki": 1.6,
      "cake": 1.5
    }
  },
  {
    "name": "Tuna",
    "unit": "grams",
    "calories": 150,
    "alternatives": {
      "salmon": 1.1
    }
  },
  {
    "name": "ma7shy wara2 3enab",
    "unit": "grams",
    "calories": 350,
    "alternatives": {
      "ma7shy kosa": 1.2,
      "ma7shy kromb": 1.0
    }
  }
```

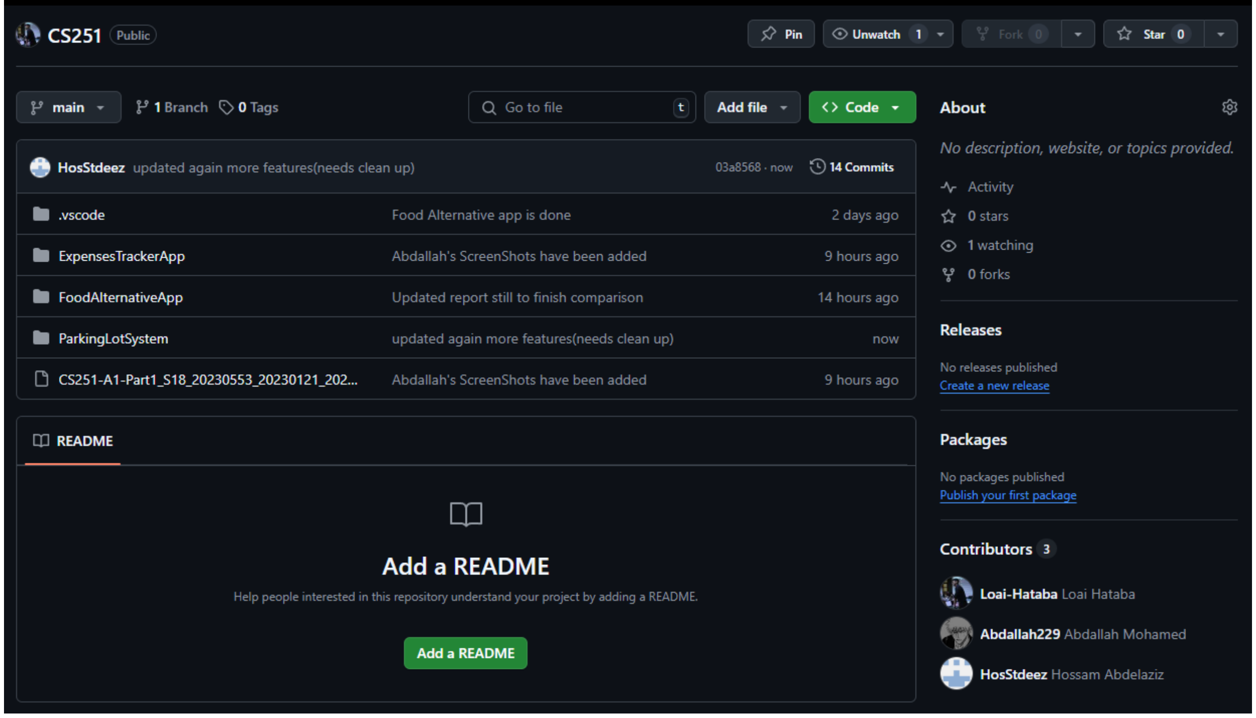


```
                                    Welcome choose an option:

1)Find Alternative Food
2)Add new Food
3)Delete Food
4)Info
5)Exit

Choice: 3

Delete Food:
Food Name: fera5
Couldn't find fera5 anywhere :(

Do you want to continue? (Y/N) |
```

```
                                    Welcome choose an option:

1)Find Alternative Food
2)Add new Food
3)Delete Food
4)Info
5)Exit

Choice: 3

Delete Food:
Food Name: ma7shy wara2 3enab
ma7shy wara2 3enab was deleted successfully!

Do you want to continue? (Y/N)
```



```
        +----------------------------------------------------------+
        ||  _____        __          _____           _              ||
        || |_   _|      / _|        / ____|         | |             ||
        ||   | |  _ __ | |_ ___    | |     ___ _ __ | |_ _ __ ___   ||
        ||   | | | '_ \|  _/ _ \   | |    / _ \ '_ \| __| '__/ _ \  ||
        ||  _| |_| | | | || (_) |  | |___|  __/ | | | |_| | |  __/  ||
        || |_____|_| |_|_| \___/    \_____\___|_| |_|\__|_|  \___|  ||
        +----------------------------------------------------------+



    This Program was made by Loai Hataba
* ==============================================
*  Food Alternative Dictionary
* ==============================================
*  This application helps users find and compare
*  food alternatives based on conversion factors.
*

*  Features:
*  - View food details (unit, calories, alternatives)
*  - Add new foods with alternative conversions
*  - Delete foods from the database in real time
*  - Interactive menu with error handling
*  - Data stored in JSON for persistence
*
*  Technologies Used:
*  - Java
*  - Gson (for JSON handling)
*  - HashMaps & Lists for data management
*
* ==============================================
Do you want to continue? (Y/N)
```



CS251 Public

Pin · Unwatch 1 · Fork 0 · Star 0

main · 1 Branch · 0 Tags · Go to file · Add file · Code

HosStdeez updated again more features(needs clean up) · 03a8568 · now · 14 Commits

.vscode	Food Alternative app is done	2 days ago
ExpensesTrackerApp	Abdallah's ScreenShots have been added	9 hours ago
FoodAlternativeApp	Updated report still to finish comparison	14 hours ago
ParkingLotSystem	updated again more features(needs clean up)	now
CS251-A1-Part1_S18_20230553_20230121_202...	Abdallah's ScreenShots have been added	9 hours ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

Add a README

About

No description, website, or topics provided.

Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 3

Loai-Hataba Loai Hataba
Abdallah229 Abdallah Mohamed
HosStdeez Hossam Abdelaziz

Video Link:

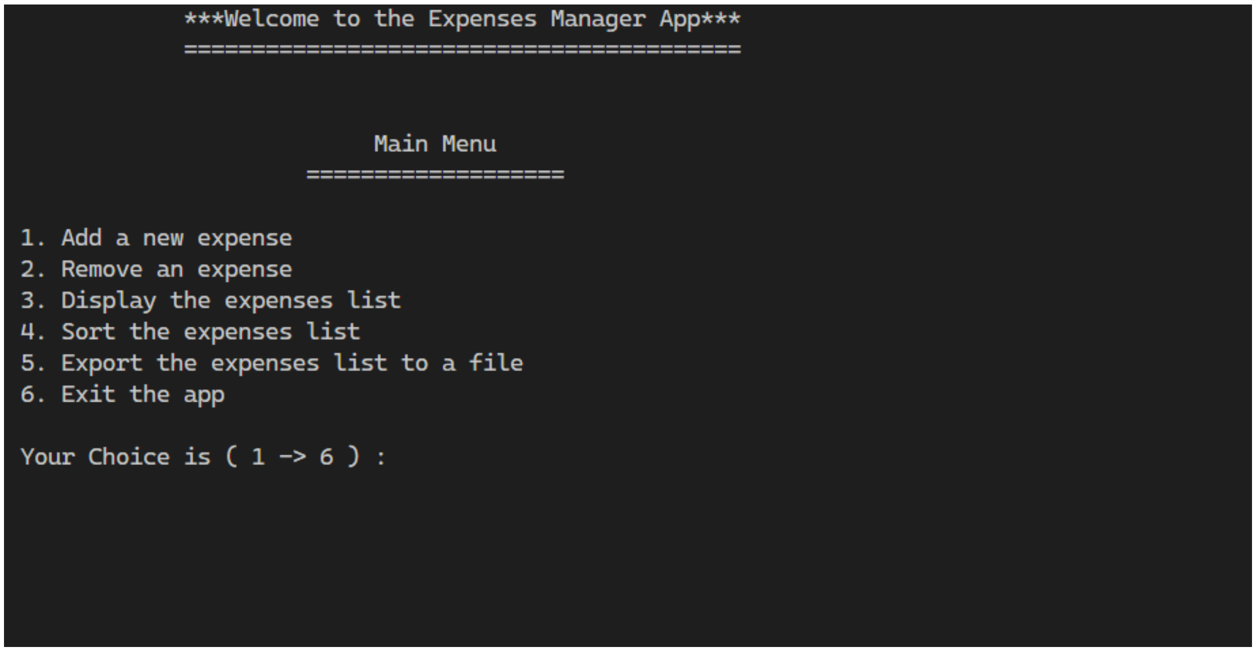
https://drive.google.com/file/d/17hZjf0I-YBnwHZ371pdMD7uhwbIYaQP3/view?usp=sharing

Budget Tracker (App 2 Abdullah):

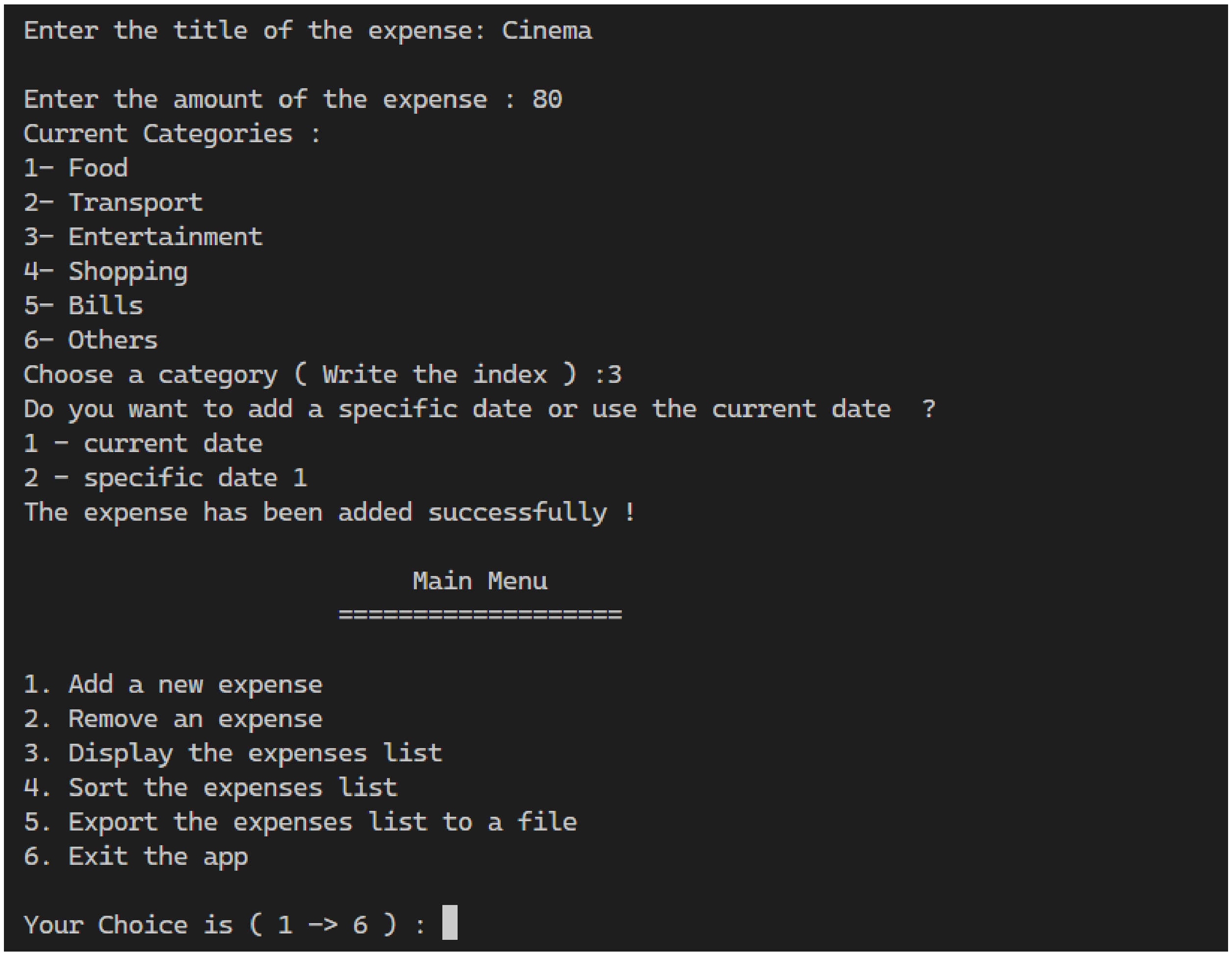
Main Function:

```
   System.out.println(
"            ***Welcome to the Expenses Manager App***");
    System.out.println(
"            =========================================\n");
    // Create an instance of the expenses list
    final ExpensesList myExpenses = new ExpensesList();
    //The app menu :
    while (true) {
        System.out.println("\n                          Main Menu  ");
        System.out.println("                     ===================\n");
        // Main menu options :
        // 1 Adding an expense :
        System.out.println("1. Add a new expense ");
        // 2 removing an expense :
        System.out.println("2. Remove an expense ");
        // 3 Display the expenses list :
        System.out.println("3. Display the expenses list ");
        // 4 Sort the expenses list :
        System.out.println("4. Sort the expenses list ");
        // 5 Export the expenses list to a file :
        System.out.println("5. Export the expenses list to a file ");
        // 6 Exit the app :
        System.out.println("6. Exit the app ");
        //read the user choice  :

        final int choice = validInput.getValidInt("\nYour Choice is ( 1 -> 6
            ) : ", "Error : Invalid Choice !!", 1, 6);
        switch (choice) {
            case 1 ->
                myExpenses.addExpense();
            case 2 ->
                myExpenses.removeExpense();
            case 3 ->
                myExpenses.displayExpenses();
            case 4 ->
                myExpenses.sortExpenses();
            case 5 ->
                myExpenses.exportExpenses();
            case 6 -> {
final int ch = validInput.getValidInt("Do you want to saving before closing
?\n1)Yes\n2)No ", "Error : Invalid Choice !!", 1, 2);
  if (ch == 1) {
       myExpenses.exportExpenses();
       }
  System.out.println("Terminating the program :(");
          return;
            }
    default ->
        throw new AssertionError();
        }
    }
```

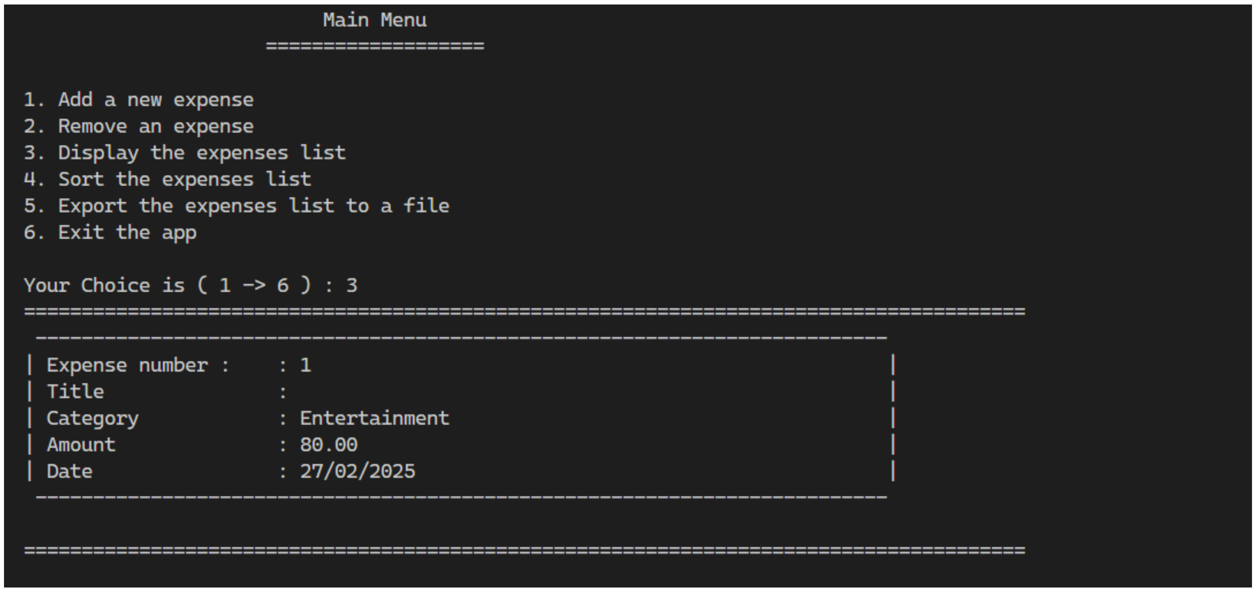
Screenshots:



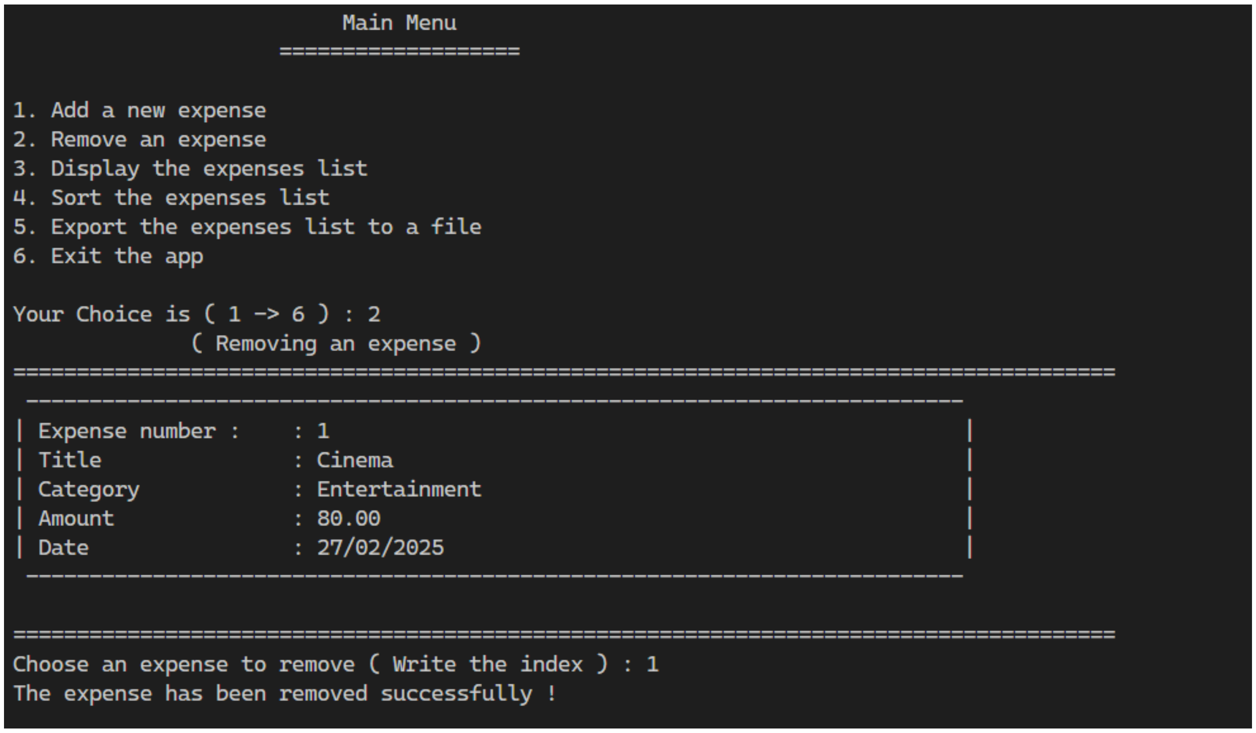
```
            ***Welcome to the Expenses Manager App***
            =========================================


                         Main Menu
                    ===================

1. Add a new expense
2. Remove an expense
3. Display the expenses list
4. Sort the expenses list
5. Export the expenses list to a file
6. Exit the app

Your Choice is ( 1 -> 6 ) :
```



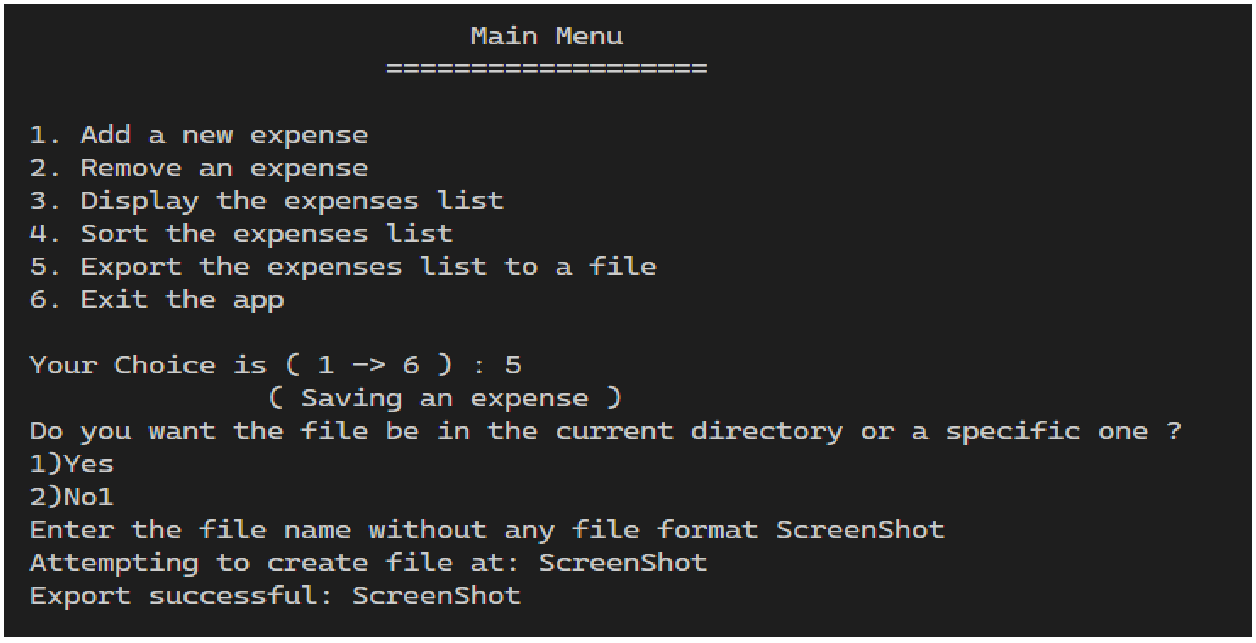
```
Enter the title of the expense: Cinema

Enter the amount of the expense : 80
Current Categories :
1- Food
2- Transport
3- Entertainment
4- Shopping
5- Bills
6- Others
Choose a category ( Write the index ) :3
Do you want to add a specific date or use the current date  ?
1 - current date
2 - specific date 1
The expense has been added successfully !

                          Main Menu
                     ===================


1. Add a new expense
2. Remove an expense
3. Display the expenses list
4. Sort the expenses list
5. Export the expenses list to a file
6. Exit the app

Your Choice is ( 1 -> 6 ) :
```



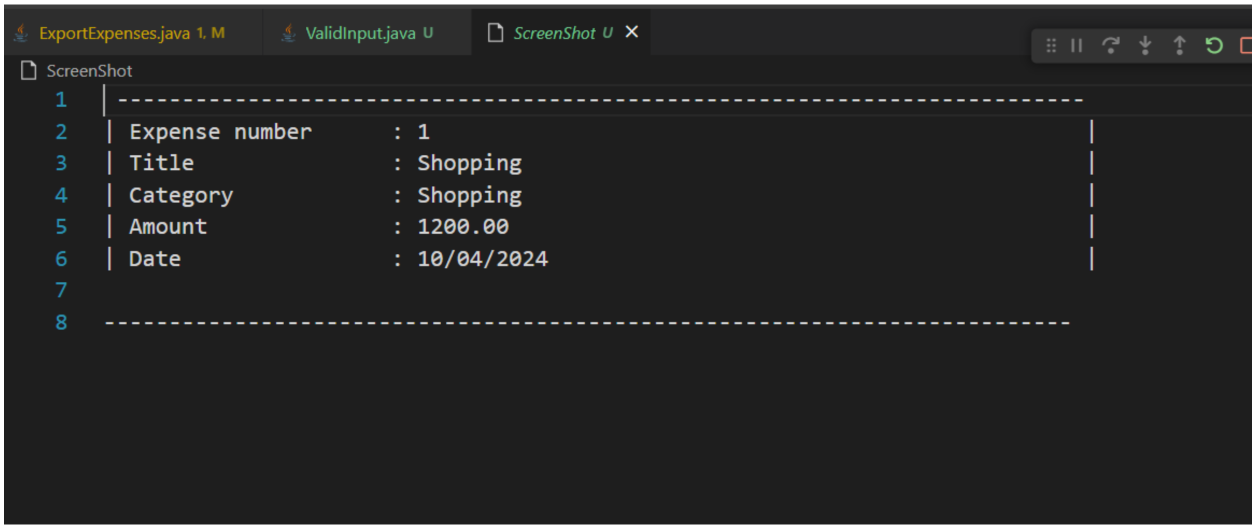
```
                          Main Menu
                     ===================

1. Add a new expense
2. Remove an expense
3. Display the expenses list
4. Sort the expenses list
5. Export the expenses list to a file
6. Exit the app

Your Choice is ( 1 -> 6 ) : 3
=========================================================================================
 ---------------------------------------------------------------------------
| Expense number :    : 1                                                   |
| Title               :                                                     |
| Category            : Entertainment                                       |
| Amount              : 80.00                                               |
| Date                : 27/02/2025                                          |
 ---------------------------------------------------------------------------

=========================================================================================
```



```
                          Main Menu
                     ===================

1. Add a new expense
2. Remove an expense
3. Display the expenses list
4. Sort the expenses list
5. Export the expenses list to a file
6. Exit the app

Your Choice is ( 1 -> 6 ) : 2
              ( Removing an expense )
=======================================================================================
 --------------------------------------------------------------------------
| Expense number :    : 1                                                  |
| Title               : Cinema                                             |
| Category            : Entertainment                                      |
| Amount              : 80.00                                              |
| Date                : 27/02/2025                                         |
 --------------------------------------------------------------------------

=======================================================================================
Choose an expense to remove ( Write the index ) : 1
The expense has been removed successfully !
```



```
                           Main Menu
                      ===================

 1. Add a new expense
 2. Remove an expense
 3. Display the expenses list
 4. Sort the expenses list
 5. Export the expenses list to a file
 6. Exit the app

 Your Choice is ( 1 -> 6 ) : 5
               ( Saving an expense )
 Do you want the file be in the current directory or a specific one ?
 1)Yes
 2)No1
 Enter the file name without any file format ScreenShot
 Attempting to create file at: ScreenShot
 Export successful: ScreenShot
```



```
 ------------------------------------------------------------------------
| Expense number      : 1                                                  |
| Title               : Shopping                                           |
| Category            : Shopping                                           |
| Amount              : 1200.00                                            |
| Date                : 10/04/2024                                         |

------------------------------------------------------------------------
```

Video Link:

https://drive.google.com/file/d/1_kMyuxn59d8yfKYCnpGyDzSnsc1SPGdx/view?usp=drive_link

Parking System (App 3 Hossam):

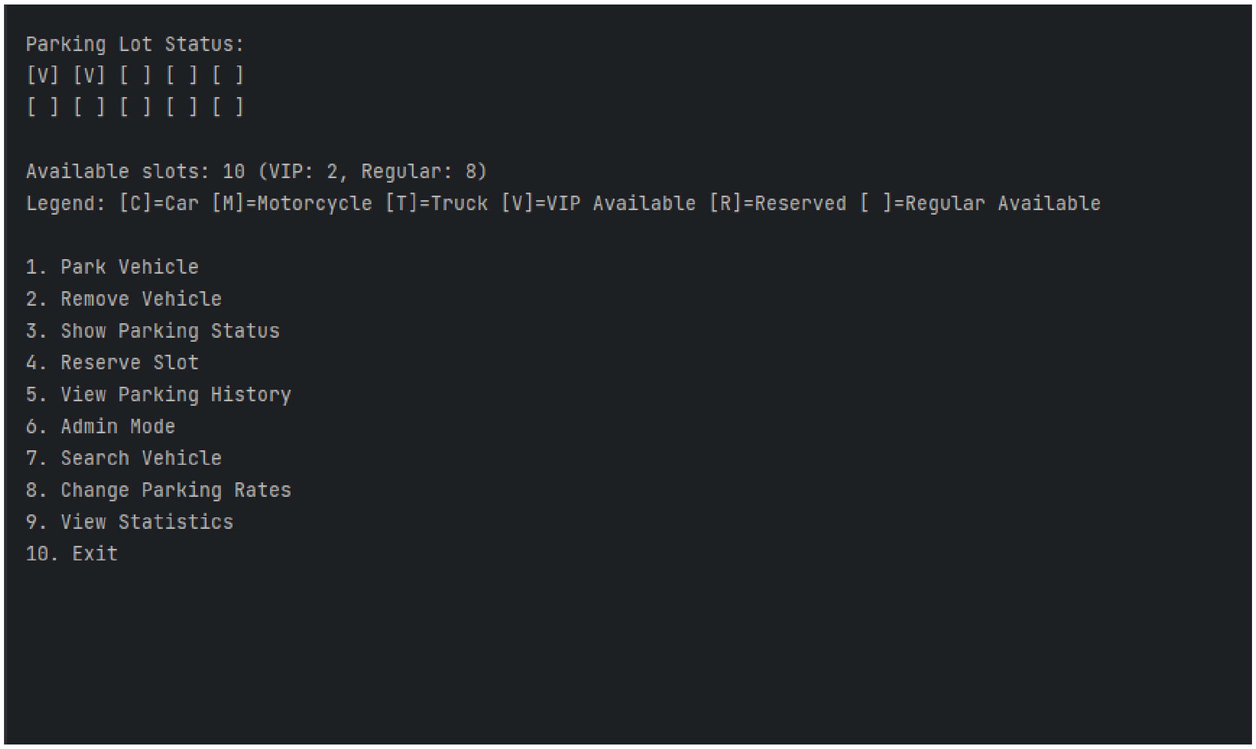
Main Function:

```
ParkingLot parkingLot = new ParkingLot(10);
Scanner scanner = new Scanner(System.in);

parkingLot.displayGrid();

// Track the last time we checked for expired reservations
long lastReservationCheck = System.currentTimeMillis();

while (true) {
  // Check for expired reservations every 5 seconds
  long currentTime = System.currentTimeMillis();
  if (currentTime - lastReservationCheck > 5000) { // 5 seconds
    parkingLot.checkReservations();
    lastReservationCheck = currentTime;
  }

  // Main menu options
  System.out.println("" +
        "\n1. Park Vehicle" +
        "\n2. Remove Vehicle" +
        "\n3. Show Parking Status" +
        "\n4. Reserve Slot" +
        "\n5. View Parking History" +
        "\n6. Admin Mode" +
        "\n7. Search Vehicle" +
        "\n8. Change Parking Rates" +
        "\n9. View Statistics" +
        "\n10. Exit");
int choice = scanner.nextInt();
scanner.nextLine();

// Check for expired reservations after any user action
parkingLot.checkReservations();

switch (choice) {
    case 1:
        System.out.print("Enter license plate: ");
        String plate = scanner.nextLine();
        System.out.print("VIP Slot? (yes/no): ");
        boolean isVIP = scanner.nextLine().equalsIgnoreCase("yes");
        System.out.print("Vehicle type (car/motorcycle/truck): ");
        String vehicleType = scanner.nextLine().toLowerCase();
        parkingLot.parkVehicle(plate, isVIP, vehicleType);
        parkingLot.displayGrid();
        break;
    case 2:
        System.out.print("Enter license plate to remove: ");
        plate = scanner.nextLine();
        parkingLot.removeVehicle(plate);
        parkingLot.displayGrid();
        break;
    case 3:
        parkingLot.displayGrid();
        break;
    case 4:
```

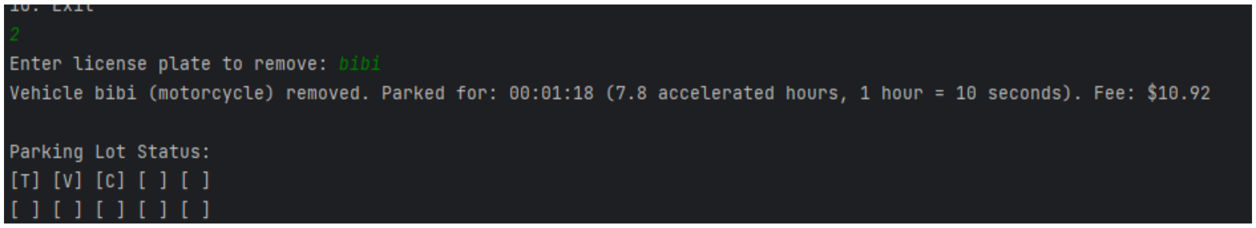
```
    System.out.print("Enter slot number to reserve: ");
    int slotNum = scanner.nextInt();
    scanner.nextLine();
    System.out.print("Enter reservation duration (hours): ");
    int hours = scanner.nextInt();
    scanner.nextLine();
    parkingLot.reserveSlot(slotNum, hours);
    break;
  case 5:
    parkingLot.displayParkingHistory();
    break;
  case 6:
    // Simple password protection for admin mode
    System.out.print("Enter admin password: ");
    String password = scanner.nextLine();
    if (password.equals("hoss123")) {
      parkingLot.adminMode();
    } else {
      System.out.println("Incorrect password!");
    }
    break;
  case 7:
    System.out.print("Enter license plate to search: ");
    plate = scanner.nextLine();
    parkingLot.searchVehicle(plate);
    break;
  case 8:
    // Password protection for changing rates
    System.out.print("Enter admin password: ");
```

```
            password = scanner.nextLine();
            if (password.equals("hoss123")) {
                System.out.print("Enter new regular rate: ");
                double regularRate = scanner.nextDouble();
                System.out.print("Enter new VIP rate: ");
                double vipRate = scanner.nextDouble();
                scanner.nextLine();
                ParkingFeeCalc.updateRates(regularRate, vipRate);
                System.out.println("Rates updated successfully!");
            } else {
                System.out.println("Incorrect password!");
            }
            break;
        case 9:
            parkingLot.displayStatistics();
            break;
        case 10:
            System.out.println("Exiting...");
            return;
        default:
            System.out.println("Invalid option!");
    }
}
```

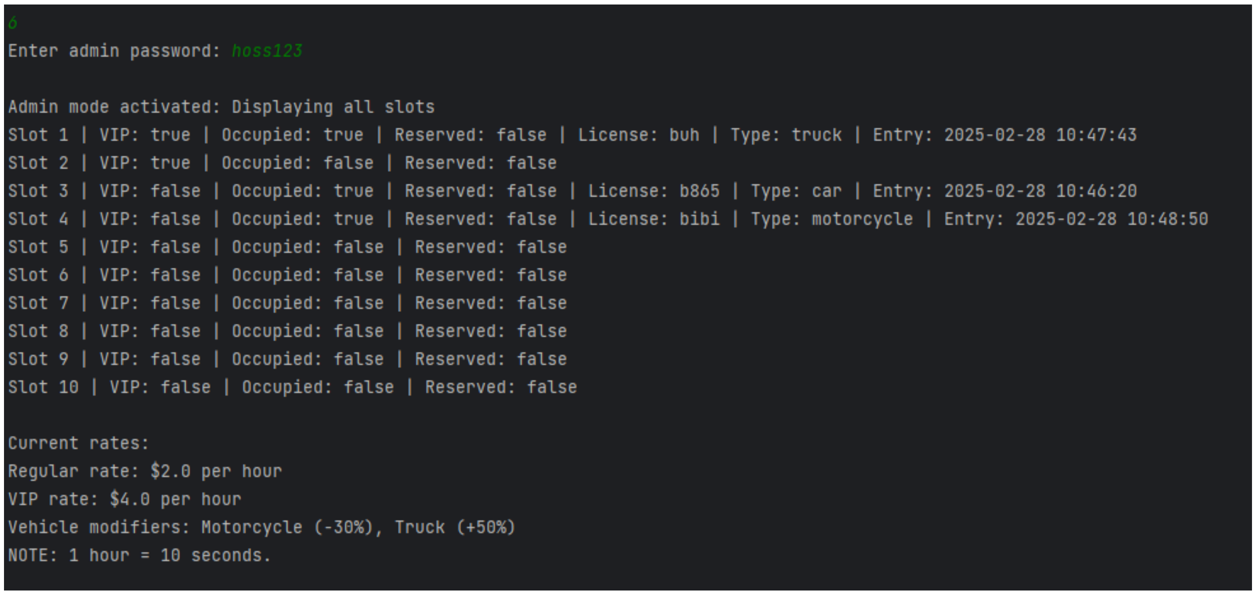
Screenshots:



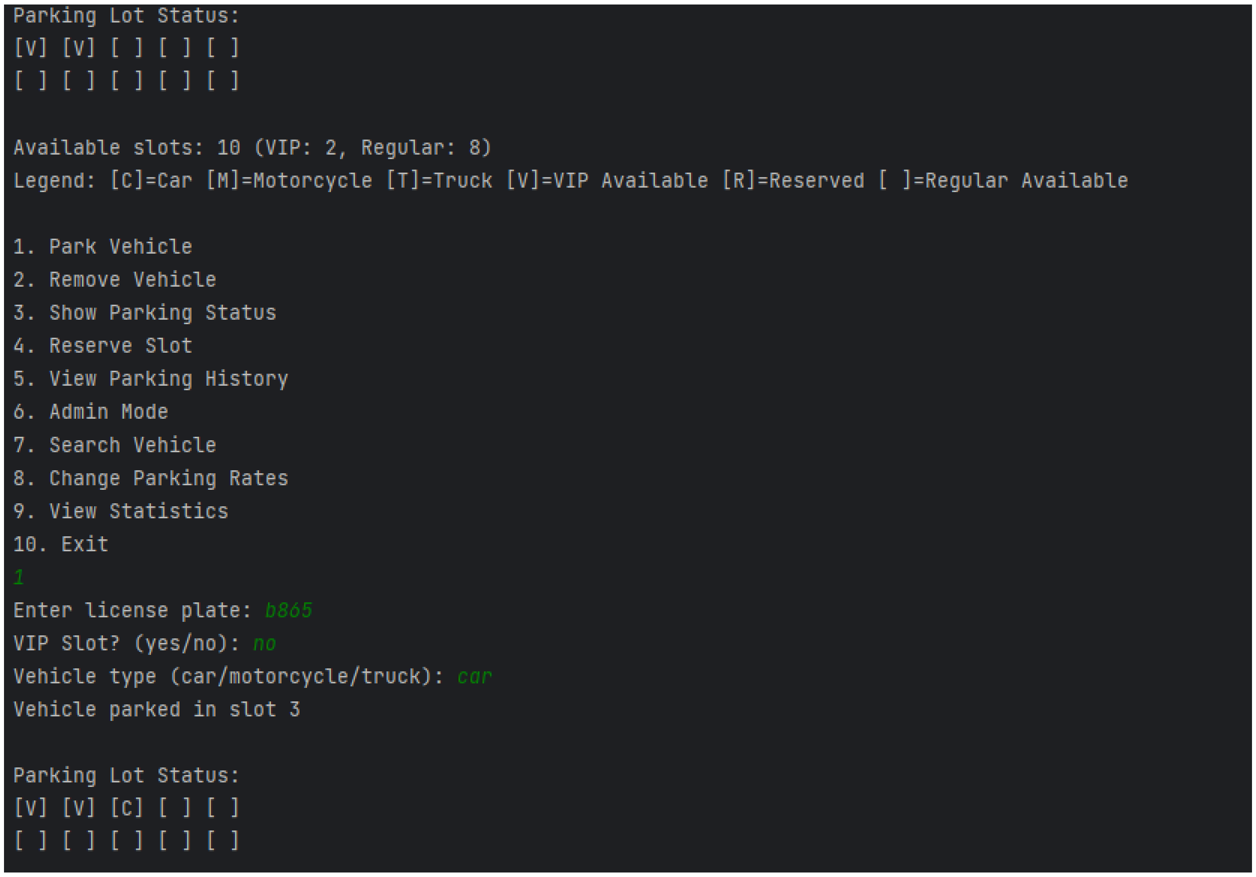
```
Parking Lot Status:
[V] [V] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ]

Available slots: 10 (VIP: 2, Regular: 8)
Legend: [C]=Car [M]=Motorcycle [T]=Truck [V]=VIP Available [R]=Reserved [ ]=Regular Available

1. Park Vehicle
2. Remove Vehicle
3. Show Parking Status
4. Reserve Slot
5. View Parking History
6. Admin Mode
7. Search Vehicle
8. Change Parking Rates
9. View Statistics
10. Exit
```



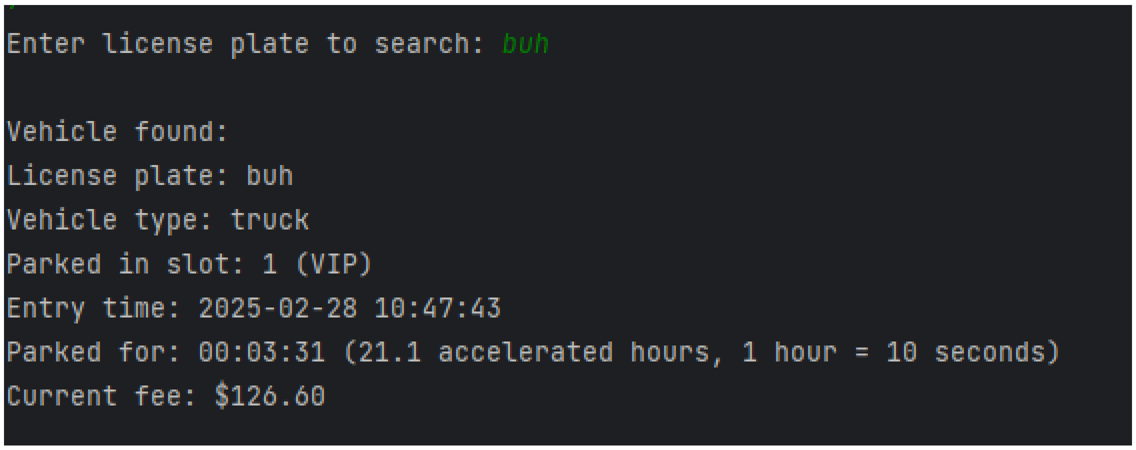
```
10. Exit
2
Enter license plate to remove: bibi
Vehicle bibi (motorcycle) removed. Parked for: 00:01:18 (7.8 accelerated hours, 1 hour = 10 seconds). Fee: $10.92

Parking Lot Status:
[T] [V] [C] [ ] [ ]
[ ] [ ] [ ] [ ] [ ]
```



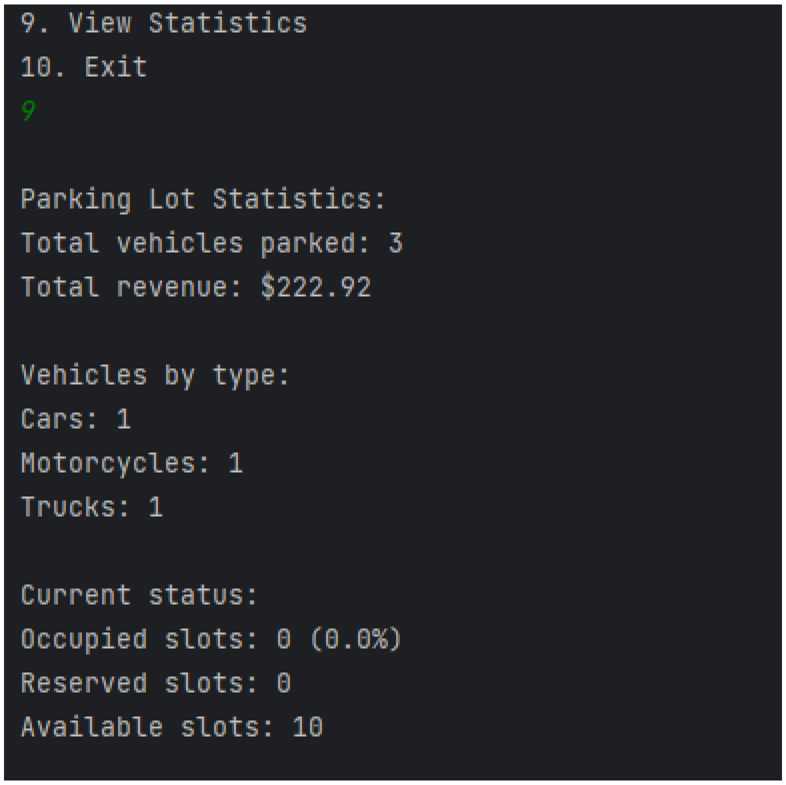
```
6
Enter admin password: hoss123

Admin mode activated: Displaying all slots
Slot 1 | VIP: true | Occupied: true | Reserved: false | License: buh | Type: truck | Entry: 2025-02-28 10:47:43
Slot 2 | VIP: true | Occupied: false | Reserved: false
Slot 3 | VIP: false | Occupied: true | Reserved: false | License: b865 | Type: car | Entry: 2025-02-28 10:46:20
Slot 4 | VIP: false | Occupied: true | Reserved: false | License: bibi | Type: motorcycle | Entry: 2025-02-28 10:48:50
Slot 5 | VIP: false | Occupied: false | Reserved: false
Slot 6 | VIP: false | Occupied: false | Reserved: false
Slot 7 | VIP: false | Occupied: false | Reserved: false
Slot 8 | VIP: false | Occupied: false | Reserved: false
Slot 9 | VIP: false | Occupied: false | Reserved: false
Slot 10 | VIP: false | Occupied: false | Reserved: false

Current rates:
Regular rate: $2.0 per hour
VIP rate: $4.0 per hour
Vehicle modifiers: Motorcycle (-30%), Truck (+50%)
NOTE: 1 hour = 10 seconds.
```



```
Parking Lot Status:
[V] [V] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ]

Available slots: 10 (VIP: 2, Regular: 8)
Legend: [C]=Car [M]=Motorcycle [T]=Truck [V]=VIP Available [R]=Reserved [ ]=Regular Available

1. Park Vehicle
2. Remove Vehicle
3. Show Parking Status
4. Reserve Slot
5. View Parking History
6. Admin Mode
7. Search Vehicle
8. Change Parking Rates
9. View Statistics
10. Exit
1
Enter license plate: b865
VIP Slot? (yes/no): no
Vehicle type (car/motorcycle/truck): car
Vehicle parked in slot 3

Parking Lot Status:
[V] [V] [C] [ ] [ ]
[ ] [ ] [ ] [ ] [ ]
```



```
Enter license plate to search: buh

Vehicle found:
License plate: buh
Vehicle type: truck
Parked in slot: 1 (VIP)
Entry time: 2025-02-28 10:47:43
Parked for: 00:03:31 (21.1 accelerated hours, 1 hour = 10 seconds)
Current fee: $126.60
```



```
9. View Statistics
10. Exit
9

Parking Lot Statistics:
Total vehicles parked: 3
Total revenue: $222.92

Vehicles by type:
Cars: 1
Motorcycles: 1
Trucks: 1

Current status:
Occupied slots: 0 (0.0%)
Reserved slots: 0
Available slots: 10
```

Video Link:

Link:

https://drive.google.com/file/d/1E3eVMdre3xwt6Xg1BuPAEF-Zgqf1_IS_/view?usp=sharing

LCNC Analysis: (AppGyver vs Glide)

1. Introduction

Low-code and no-code development platforms have gained popularity for enabling non-developers and businesses to create applications with minimal coding.

Among these platforms, **AppGyver** and **Glide** stand out as powerful tools for building apps efficiently. This report analyzes AppGyver's and Glide's usability, benefits, and system quality while comparing them to determine their strengths and suitability for different use cases.

---

2. Evaluation of AppGyver

Overview

AppGyver is a professional-grade no-code development platform that allows users to create web and mobile applications without writing traditional code. It is particularly known for its flexibility and extensive customization options.

Usability and Features

- **Drag-and-Drop Interface**: Offers an intuitive builder for designing app layouts and functionalities.
- **Extensive Components**: Provides pre-built UI elements and logic modules.
- **Data Integration**: Supports REST APIs, databases, and third-party services.
- **Multi-Platform Deployment**: Applications can be deployed on web, iOS, and Android.
- **Logic and Automation**: Allows users to define workflows and dynamic logic visually.

Benefits and System Quality

- **Customization**: Unlike many no-code tools, AppGyver enables deep customization, making it ideal for complex applications.
- **Performance**: Apps built with AppGyver are optimized for high performance, especially on mobile devices.

- **Scalability**: Supports scalable applications, making it suitable for startups and enterprises.
- **Security**: Provides robust authentication and data security features.

Impact on Developers' Roles

While no-code platforms like AppGyver simplify application development, they are unlikely to replace traditional developers entirely. Instead, they serve as **enhancement tools** that allow developers to prototype faster and focus on more complex backend logic. Additionally, organizations can leverage no-code platforms for internal tools without needing a dedicated development team.

---

3. Evaluation of Glide

Overview

Glide is a no-code development platform that specializes in creating simple, data-driven applications using Google Sheets as a backend. It is widely used for lightweight business applications, internal tools, and prototypes.

Usability and Features

- **Google Sheets Integration**: Data is dynamically synced with Google Sheets, making data management straightforward.
- **Pre-Built Templates**: Provides templates to accelerate app creation.
- **Mobile-First Design**: Optimized for mobile applications with responsive design.
- **Drag-and-Drop Builder**: Users can easily add and arrange elements without coding.
- **Limited Logic & Automation**: Basic workflows can be set up, but advanced logic is limited.

Benefits and System Quality

- **Ease of Use**: Designed for non-technical users, making app creation accessible to a wide audience.
- **Speed of Development**: Apps can be created and deployed in minutes with minimal effort.
- **Cloud-Based**: Eliminates the need for complex hosting or deployment.

- **Data Management**: Real-time updates with Google Sheets ensure seamless synchronization.

Impact on Developers' Roles

Glide significantly lowers the barrier for app creation, allowing businesses to create internal tools without needing a development team. However, its **limited customization and scalability** mean that traditional developers are still essential for building feature-rich, enterprise-level applications. Glide is best suited for rapid prototyping and simple, data-driven applications rather than complex business solutions.

---

4. Comparison of AppGyver and Glide

Feature	AppGyver	Glide
Ease of Use	Moderate learning curve	Very beginner-friendly
Customization	High (supports complex logic and UI design)	Limited (focuses on simplicity)
Scalability	Suitable for enterprise applications	Best for small projects and internal tools
Pricing	Free for solo developers; enterprise pricing available	Free tier with premium plans for business features
Integrations	REST API, third-party services, external databases	Google Sheets, Airtable, Zapier, and limited API support

---

5. Sample To-Do List App

As part of this comparison, we created a **To-Do List App** using Glide. The application enables users to:

- Add new tasks with descriptions.
- Mark tasks as completed.
- Store and retrieve data using Google Sheets.
- Access the app from both web and mobile devices.

Glide's simplicity allowed for quick development, making it an excellent choice for basic applications.

Video link:

https://drive.google.com/file/d/1vqT3N6ripz5jJ3y9BYiqkThsBJVXNKQ7/view?usp=drive_link

App link:

https://habit-tracker-app-ills.glide.page

---

6. Conclusion

AppGyver and Glide both offer unique advantages in the no-code development space. Glide is ideal for quick prototyping and simple applications, while AppGyver provides greater flexibility and customization for more complex projects.

Pre-Project :

1. Market and Gap analysis

Global Practices

Personal budgeting software isn't a new ideaa it can be found in many countries, offering comprehensive features to help people manage their finances. An example of those applications include **Mint**, **PocketGuard**, **EveryDollar**, and **GoodBudget**. These apps provide different approaches to budgeting, from detailed manual tracking to automated expense management.

- **Mint**: Automatically syncs with bank accounts and categorizes expenses. It provides visual reports and personalized budgeting recommendations.
- **PocketGuard**: Offers simple budgeting with a focus on preventing overspending by showing how much disposable income is left after accounting for bills and savings.
- **EveryDollar**: Provides zero-based budgeting where users assign every dollar of income to a category, promoting careful financial planning.
- **GoodBudget**: Uses an envelope system where users allocate money to different categories, making it ideal for people who prefer manual control.

Egypt's Market

In the Arab region, personal budgeting apps are less prevalent, but there are some local and regional options that cater to Arabic-speaking users. Popular apps include:

- **Masareef**: A budgeting app in Arabic that helps users track income and expenses.
- **Wally**: A widely used app that supports expense tracking and allows users to set financial goals. It supports both English and Arabic languages.
- **Tajer**: Focuses on expense tracking for small businesses but can be used for personal budgeting.
- **Monefy**: An easy-to-use app that allows quick expense logging and provides visual insights.

These apps offer basic features such as expense tracking and simple reporting, but advanced features like bank account integration, financial goal setting, and predictive analytics aren't really in there.

Gap Analysis

Local apps tend to focus only on expense tracking without offering deeper financial insights or integration with banking systems, the apps are fine and do their job but they could really go an extra mile to offer more features in their apps that would be very beneficial for the users such as integrating your bank account so you can track your spendings more accurately.

The key gaps in the Arab region market include:

- **Bank Account Integration**: Very few apps automatically sync with bank accounts or e-wallets, making users manually enter transactions.
- **Financial Goal Setting**: Limited functionality for setting and tracking financial goals like savings for vacations or debt repayment.
- **Customizable Budgets**: Lack of flexible budgeting tools that allow users to set limits for different categories and adjust them dynamically.
- **Insights and Analytics**: Minimal reporting features to help users understand spending patterns or predict future expenses.
- **User Education**: Few apps offer financial education content to help users improve their financial literacy.

2. Market Segmentation and Research

Customer Segments (Primary Target Users)

1. **Young Adults (20-30 years old)**
    - University students and fresh graduates managing their first income.
    - Early-career professionals trying to build financial discipline.
    - Individuals saving for short-term goals like travel, gadgets, or education.
    - Users who are new to budgeting and need a simple, automated system.

2. **Working Professionals (30-45 years old)**
    - Mid-career employees looking to track their spending and savings.
    - Business owners or self-employed individuals managing personal and business finances.
    - Individuals planning for major expenses like home purchases, marriage, or children's education.

3. **Parents & Household Managers**
    - Families managing monthly household budgets (groceries, rent, utilities, children's education).
    - Parents who need to track expenses for kids' extracurricular activities and tuition.
    - People looking to save for family vacations, medical emergencies, or future investments.

4. **Freelancers & Gig Workers**
    - Workers with irregular income who need better financial planning.
    - Freelancers managing multiple income sources and unpredictable cash flow.
    - Individuals needing expense tracking for tax deductions and financial planning.

5. **Independent Investors & Financially proficient Users**
    - People actively investing in stocks, real estate, or startups.
    - Individuals looking for an all-in-one tool to manage income, savings, and investments.
    - Professionals who want financial reports and trend analysis for decision-making.

Demographic

- **Age Group:**
    - Primary: **20-45 years old** (young adults, professionals, parents).
    - Secondary: **45+ years old** (investors, retirees managing wealth).

- **Income Levels:**
    - **Lower Middle Class (10,000 – 20,000 EGP/month)**: Need tools to track spending, avoid debt.
    - **Middle Class (20,000 – 50,000 EGP/month)**: Managing expenses while saving for long-term goals.
    - **Upper Middle Class (50,000+ EGP/month)**: More focus on investments and wealth management.

- **Education Level:**
    - University students, graduates, and professionals with at least a high school diploma.
    - Higher financial literacy among professionals and investors, but young adults may need simpler tools.

- **Tech Usage Patterns:**
    - **Mobile-First Users**: High smartphone penetration in Egypt and Arab countries makes mobile apps essential.
    - **Frequent Internet Users**: Active on digital banking apps and online financial services.
    - **Preference for Automation**: Users prefer auto-tracking of expenses and AI-driven insights.

Why These Groups Would Be Interested?

1. **Young Adults (20-30 years old)**
    - New to financial independence and need structured budgeting.
    - Struggle with impulse spending and saving habits.
    - Want a simple, gamified, and mobile-friendly solution.

2. **Working Professionals (30-45 years old)**
    - Need tools to track and optimize monthly expenses.
    - Have multiple financial commitments (loans, savings, investments).
    - Prefer an app with financial goal tracking (e.g., home ownership, vacations).

3. **Parents & Household Managers**
    - Require expense tracking for family budgeting.
    - Need a shared budget feature for spouses or household members.
    - Want reminders for bill payments, school fees, and savings plans.

4. **Freelancers & Gig Workers**
    - Need income categorization for tax purposes.
    - Require insights into spending patterns due to fluctuating income.
    - Seek automation in logging and tracking multiple income streams.

5. **Independent Investors & Financially proficient Users**
    - Interested in advanced reporting tools (graphs, trends, AI-driven insights).
    - Require investment tracking and integration with financial platforms.
    - Want security and data privacy for financial transactions.

3. Domain Analysis :

Introduction:

This document describes background information that has been gathered about budgeting and how it is handled. This information is to be used to guide the development of software to automate the process of making people track their expenses.

Glossary:

Term	**Definition**
Income	Money received from salaries, businesses, investments, or other sources.
Expense	Money spent on necessities (rent, food, utilities) and discretionary spending (entertainment, travel)
Budget	A financial plan that allocates funds to different spending categories
Financial Goal	A target for saving money (e.g., buying a car, going on vacation, repaying loans)
Expense Category	A category for costs like groceries, transportation, schooling, or entertainment
Savings	Money set aside for future use, such as emergency funds or investments
Investment	Assets purchased with the goal of generating future income, like stocks or property
Bank Integration	The ability to sync financial transactions automatically with bank accounts or e-wallets
Zero-Based Budgeting	A method where every dollar is assigned to a specific expense or savings category, leaving no unallocated funds
Cash Flow	The movement of money in and out of a person’s finances, tracking income vs. expenses
Net Worth	The total value of a person's assets minus liabilities (debts)

General Knowledge:

- A budget is a financial plan that outlines income sources, expenses, and potential savings goals to support better financial management.
- The budget owner, typically an individual, monitors expenditure and savings. Each budget includes a title, description, and relevant financial details.
- Expenses can be categorized into different groups, allowing for better organization and tracking of spending habits.
- Budgets and financial insights are usually private to the user, but some systems offer options to share reports with financial advisors or family members.

Customers and Users:

- **Potential Clients**: Businesses or organizations that might use the program for their clients, including Budgeting tools as a value-added service provided by banks. Employers provide their staff with financial wellness initiatives. Clients can manage their personal finances with the assistance of financial advising firms.

- **Potential Users:** Individual consumers want to manage their monthly budget and keep tabs on their spending. Employees that want to create financial goals in order to reduce their pay. Young adults or students who are handling their limited resources. Budgets and expenses are planned by families. Self-employed people or freelancers keeping tabs on erratic income and expenses.

The Environment:

The primary users are those maintaining their own financees, with secondary users like financial planners who can see collaborative reports. The software is installable on desktops (Windows, macOS), mobile phones (Android, iOS), or as an internet application, with internet connectivity required for synchronizing transactions but with offline support for manual entry. Integration with bank APIs, e-wallets, and financial data sources enables automated monitoring, and email, SMS, or push notifications notify users. Security measures such as data encryption, and 2FA to ensure data security.

Tasks and procedures currently perform:

Input Expenses:

The system allows users to manually enter their daily, weekly, or monthly expenses. Rent, groceries, entertainment, and other expenses can be grouped together to better monitor spending trends.

Creating a budget plan:

helping the customer create a customized budget plan according to their financial objectives. This includes determining sources of income, classifying spending, establishing savings goals, and making plans for debt payback or future investments.

Setting Financial Goals:

Users choose both short-term and long-term financial goals, like debt repayment, purchasing a car, and vacation savings. The program facilitates progress monitoring and offers information on the monthly savings required to reach the target.

Reporting and Insights:

Financial data, such as spending patterns, budget summaries, and savings progress, are produced by the app. Charts and graphs are examples of visual representations that assist users in analyzing their financial patterns and coming to well-informed conclusions.

Sync with Bank Account:

allowing the program to automatically import transactions by syncing with credit cards, e-wallets, or bank accounts. This guarantees real-time tracking of income and expenses and minimizes human data entering.

4. Proposed Solution

Purpose and Goals:

In order to assist users effectively manage their personal money, the software tracks their income, expenses, and savings. It offers resources for financial goal setting, budgeting, and spending analysis to enhance financial decision-making. Real-time analytics, transaction tracking automation, and safe data management are among the high-level goals.

Key Features and Functionality:

- **Financial Goal Setting:** Users can set short-term and long-term financial goals with progress tracking.
- **Customizable Budgets:** Flexible budgeting tools allowing users to set, adjust, and categorize spending limits dynamically.
- **Expense Categorization:** Automatic and manual categorization of expenses into predefined or custom categories.
- **Insights & Analytics:** Visual reports, and spending trends to help users understand and optimize their financial behavior.
- **Reminders & Notifications:** Alerts for bill payments, savings milestones, and overspending warnings.
- **Gamification Elements:** Reward systems, streaks, and badges to encourage consistent financial habits.
- **Investment Calculator:** Calculate simple and compounded interest.

Target users:

Individual customers wish to monitor their spending and manage their monthly budget. Workers who wish to set financial targets to lower their compensation. Students and young adults managing their limited assets. Families arrange their spending and budgets. freelancers or self-employed individuals monitoring unpredictable revenue and costs.

Technologies:

The **Personal Budgeting Software** will use a **Desktop app** with a possible mobile app developing later. The backend will handle secure transactions, data storage, and AI-driven insights.

- **Frontend:** JavaFx for the GUI.
- **Backend:** Java classes.
- **Database:** SQLite for lightwight, easy, and fast database management.

Technical Decisions:

- **Cloud-Based Solution:** Hosted on AWS or Google Cloud for scalability and real-time syncing.
- **Offline Mode:** Allows users to log expenses without an internet connection, syncing later.