RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : SREYA G
Year/Branch/Section : II/CSE/D
Register No. : 230701334
Semester : IV
Academic Year: 2024-25

INDEX

Reg. No :2116230701334 **Name : SREYA G**

Branch : CSE **Year/Section : II / FD**

LIST OF EXPERIMENTS

Experiment No:	Title	Tools
1	Design a UI where users recall visual elements (e.g., icons or text chunks). Evaluate the effect of chunking on user memory.	Figma.
2.	Develop and compare CLI, GUI, and Voice User Interfaces (VUI) for the same task and assess user satisfaction.	Python (Tkinterfor GUI, Speech Recognition for VUI) / Terminal
3	A) Create a prototype with familiar and unfamiliar navigation elements. Evaluate ease of use with different user groups.	Proto.io
	B)Create a prototype with familiar and unfamiliar navigation elements. Evaluate ease of use with different user groups.	Wireflow
4	A)Conduct task analysis for an app (e.g., online shopping) and document user flows. Create corresponding wireframes.	Lucid chart (free tier)
	B)Conduct task analysis for an app (e.g., online shopping) and document user flows. Create corresponding wireframes.	Dia (open source).
5.	A)Simulate the lifecycle stages for UI design using the RAD model and develop a small interactive interface.	Axure RP
	B)Simulate the lifecycle stages for UI design using the RAD model and develop a small interactive interface.	OpenProj.

6.	Experiment with different layouts and color schemes for an app. Collect user feedback on aesthetics and usability.	GIMP (open source for graphics).
7.	A)Develop low-fidelity paper prototypes for a banking app and convert them into digital wireframes.	Pencil Project
	B)Develop low-fidelity paper prototypes for a banking app and convert them into digital wireframes.	Inkscape.
8.	A) Create storyboards to represent the user flow for a mobile app (e.g., food delivery app).	Balsamiq
	B) Create storyboards to represent the user flow for a mobile app (e.g., food delivery app).	OpenBoard
9.	Design input forms that validate data (e.g., email, phone number) and display error messages.	HTML/CSS, JavaScript (with Validator.js).
10.	Create a data visualization (e.g., pie charts, bar graphs) for an inventory management system.	Java Script

Ex. No. : a **Date : 25.01.2025**

Evaluating Good vs. Bad UI Design in Figma

Aim:

To analyse the impact of good and bad UI design principles on user experience by creating two versions of a mobile login screen in Figma.

Procedure:

1. **Set Up the Experiment:**
    - Open Figma and create a new project.
    - Design two mobile login screens: one following good UI/UX principles and the other with bad design choices.

2. **Designing the Good UI Version:**
    - Use a clean and consistent layout with proper spacing.
    - Apply a clear visual hierarchy with readable font sizes.
    - Use a high-contrast color scheme for accessibility.
    - Add clear input fields with labels and placeholders.
    - Provide a properly styled login button with a distinct color.
    - Implement feedback mechanisms like error messages.
    - Ensure mobile responsiveness and touch-friendly elements.

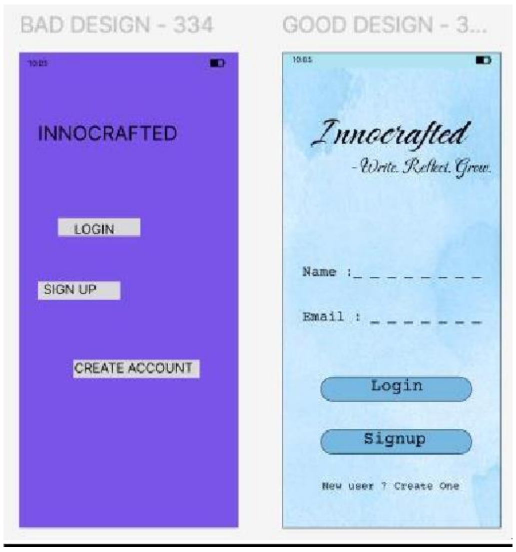
3. **Designing the Bad UI Version:**
    - Use inconsistent fonts and poor color contrast.
    - Place elements in a cluttered and misaligned manner.
    - Remove labels from input fields, relying only on placeholder text.
    - Use small buttons that are difficult to tap on mobile.
    - Provide no error handling or feedback mechanisms.
    - Ignore accessibility considerations like color blindness support.

4. **User Testing:**
    - Recruit 5-10 participants to interact with both designs.
    - Ask them to complete a simple login task in both versions.
    - Record their time taken, errors made, and overall satisfaction.

5. **Analyze Results:**

- Measure the usability of both designs using metrics like:
    - Time taken to complete login.
    - Number of errors (e.g., missed input fields).
    - User ratings on a scale of 1-10 for ease of use.

Output :



Results:

- **Good UI Design:** Users completed the login process faster, made fewer errors, and reported a smoother experience.
- **Bad UI Design:** Users struggled with navigation, took longer to complete tasks, and expressed frustration over unclear elements.

Link :

https://www.figma.com/design/cuhGONS98y1ErYPQjKIMBh/230701334---SREYA-G---EXP- 1 --- GOOD-DESIGN-VS-BAD-DESIGN?t=nEKZ0jUAHck25XSy-1

Ex. No. : 1 **Date : 12.02.2025**

Design a UI where users recall visual elements (e.g., icons or text chunks). Evaluate the effect of chunking on user memory

Aim:

To examine how chunking (grouping visual elements such as icons or text) affects users' ability to recall information in a UI environment designed in Figma.

Procedure:

Step 1: Setting Up the UI in Figma

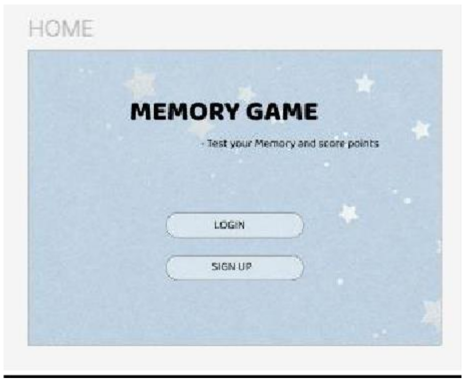
1. **Create a Home Screen (Instruction Page)**
    - Open Figma and create a **new frame** (1024x768px for desktop view).
    - Add a **heading**: "Memory Recall Task."
    - Provide **instructions** explaining that users will view grouped icons/text and recall them later.
    - Create a **Start button** using a rectangle and link it to the next screen using Figma's **Prototype feature**.
2. **Chunking Phase (Display Chunked Items)**
    - Create a **new frame** to show the items users will memorize.
    - Design two versions:
        - **Chunked Design:** Group icons or text into 3-5 item clusters using boxes.
        - **Unchunked Design:** Display items randomly without clear separation.
    - Set up a **5-second delay** to automatically transition to the next screen.
3. **Recall Phase (User Memory Test)**
    - Create a **new frame** for recall.
    - Design two options for user input:
        - **Multiple-choice selection:** Users select from a set of options.

Text input fields: Users type the items they remember.

    - Add a **Submit button** to move to the results screen.
4. **Result Screen (Feedback and Analysis)**

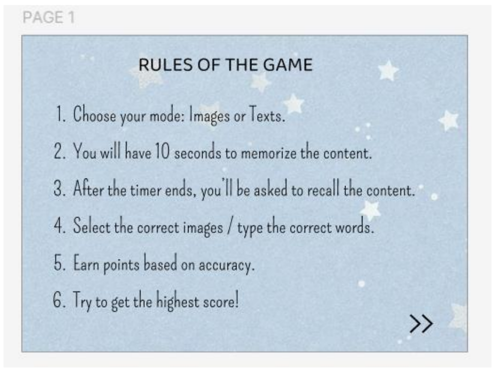
- Show feedback like: “You recalled 4/5 items correctly!”
- Record user performance based on the number of correct answers.
- Compare results for chunked vs. unchunked groups and icons vs. text-based chunks.

Output


HOME

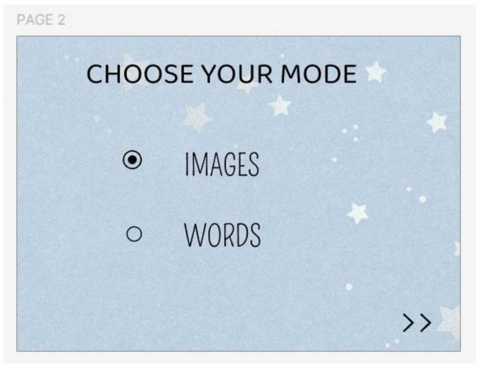
MEMORY GAME

- Test your Memory and score points


PAGE 1

RULES OF THE GAME

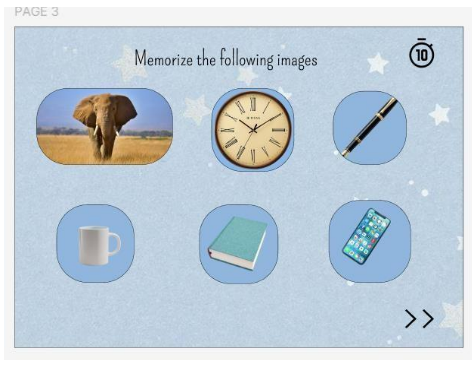
1. Choose your mode: Images or Texts.
2. You will have 10 seconds to memorize the content.
3. After the timer ends, you'll be asked to recall the content.
4. Select the correct images / type the correct words.
5. Earn points based on accuracy.
6. Try to get the highest score!


PAGE 2

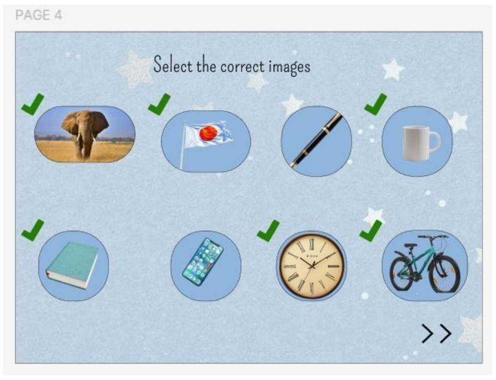
CHOOSE YOUR MODE

[✓ IMAGES]

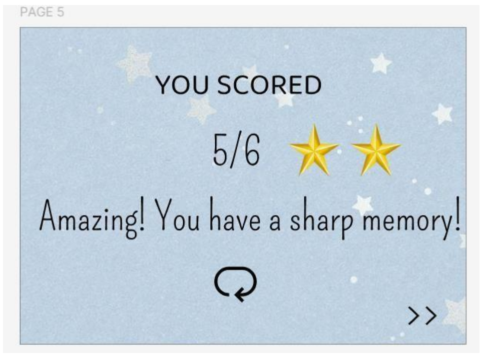
WORDS


PAGE 3

Memorize the following images


PAGE 4

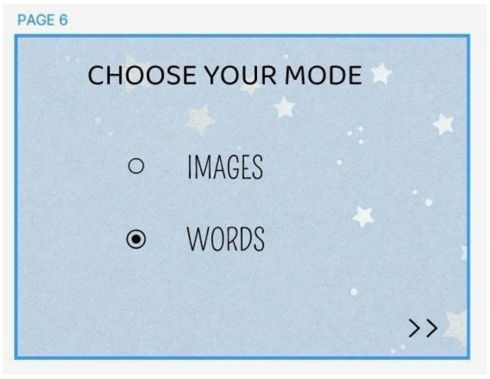
Select the correct images


PAGE 5

YOU SCORED

5/6

Amazing! You have a sharp memory!

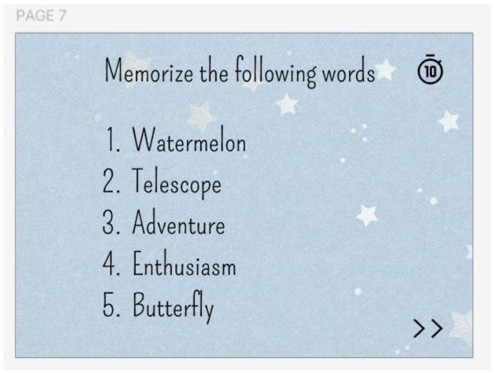

PAGE 6

CHOOSE YOUR MODE

○ IMAGES

[✓ WORDS]

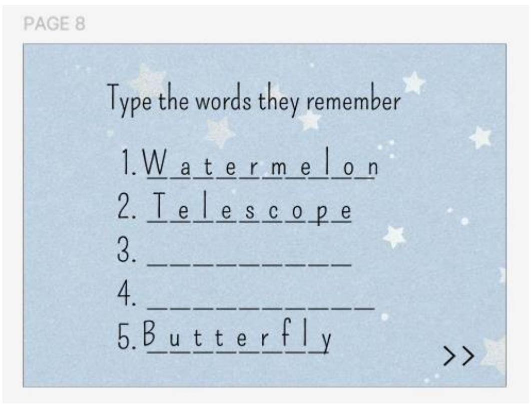
>>


PAGE 7

Memorize the following words

1. Watermelon
2. Telescope
3. Adventure
4. Enthusiasm
5. Butterfly

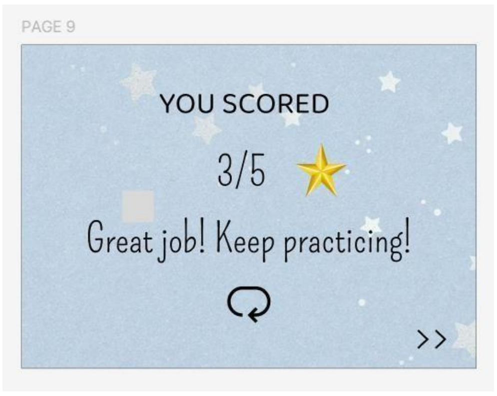
>>


PAGE 8

Type the words they remember

1. Watermelon
2. Telescope
3. __________
4. ___________
5. Butterfly

>>

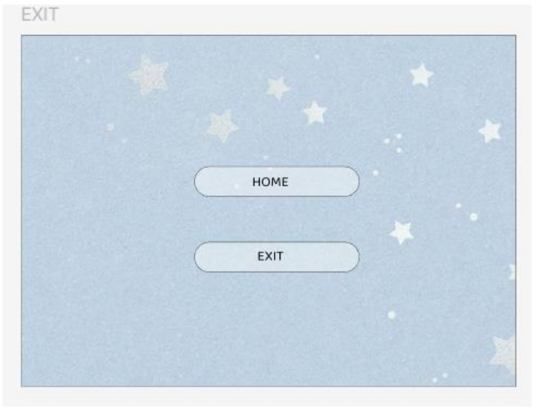

PAGE 9

YOU SCORED

3/5

Great job! Keep practicing!

>>


EXIT

HOME

EXIT

Results:

Users recalled chunked items better than unstructured ones, with icons being more memorable than text. The optimal chunk size was 3-5 items, as recall dropped beyond this. Multiple-choice input was easier, but text input led to better memory retention.

Link:

https://www.figma.com/design/nuZJ8HXygO5tsX8EaguDqf/230701334---

SREYA-G--- MEMORY-GAME?t=nEKZ0jUAHck25XSy-1

Ex. No. : 2 **Date : 22.02.2025**

Develop and compare CLI,GUI, and Voice User Interfaces (VUI) for the same task and assess user satisfaction using Python (Tkinter for GUI , Speech Recognition for VUI),Terminal

AIM:

The aim is to develop and compare Command Line Interface (CLI), Graphical User Interface (GUI), and Voice User Interface (VUI) for the same task, and assess user satisfaction using Python (with Tkinter for GUI and Speech Recognition for VUI) and Terminal.

PROCEDURE:

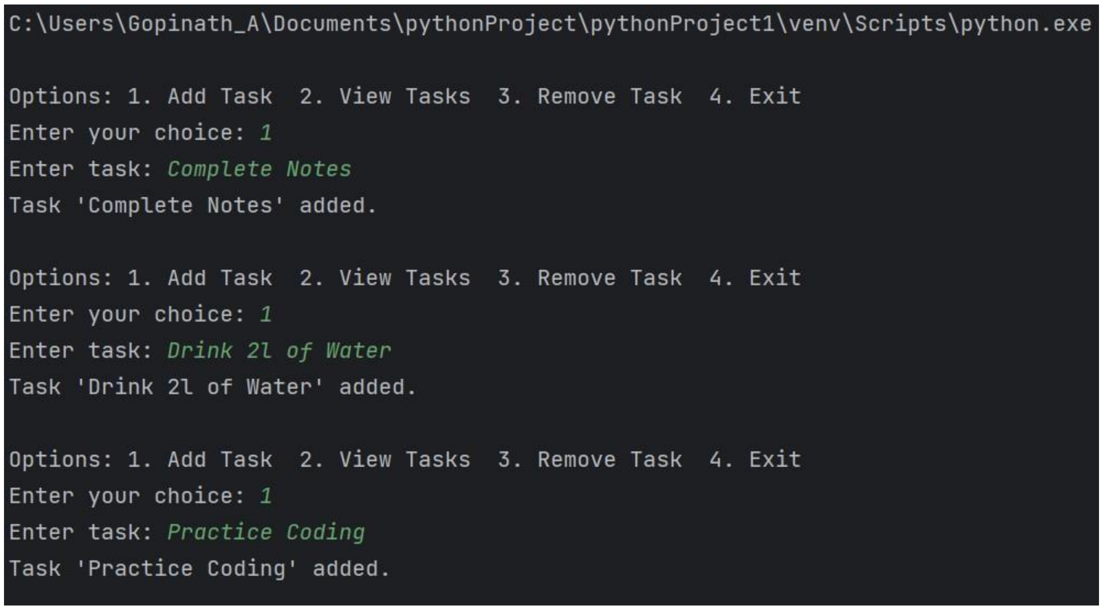
i) CLI (Command Line Interface)

CLI implementation where users can add, view, and remove tasks using the terminal.

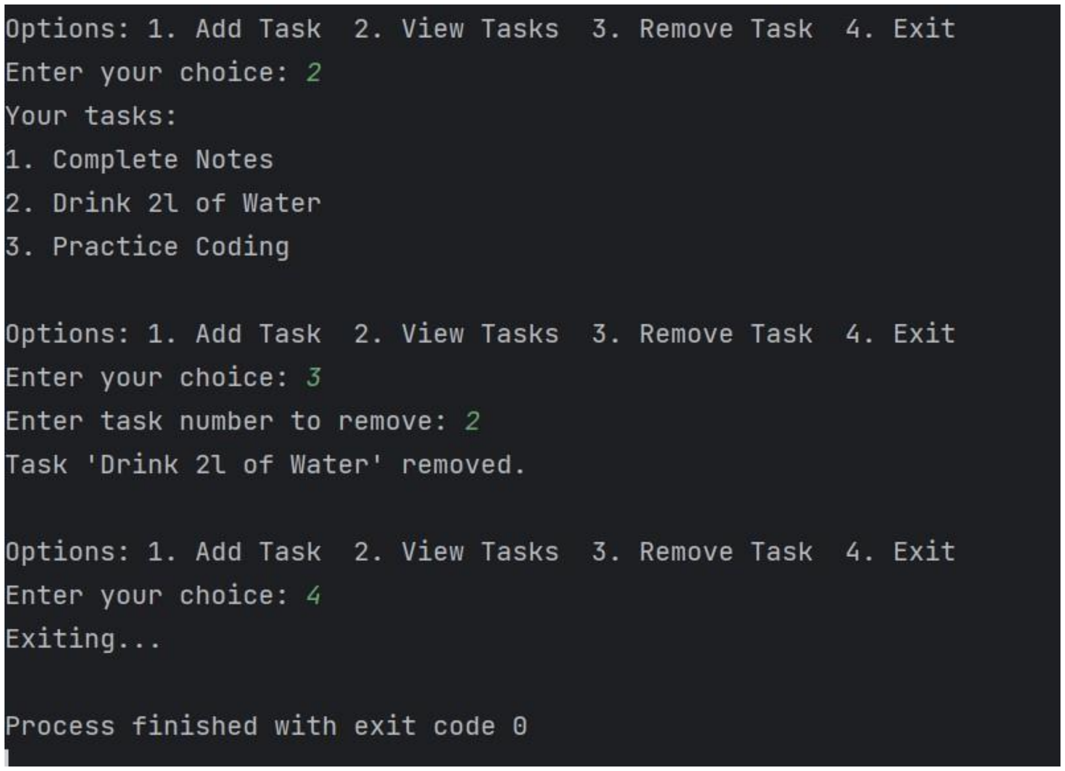
```
□ tasks = []
def add_task(task):
    tasks.append(task)
    print(f"Task '{task}' added.")

def view_tasks():
    if tasks:
        print("Your tasks:")
        for idx, task in enumerate(tasks, 1):
            print(f"{idx}. {task}")
    else:
        print("No tasks to show.")

def remove_task(task_number):
    if 0 < task_number <= len(tasks):
        removed_task = tasks.pop(task_number - 1)
        print(f"Task '{removed_task}' removed.")
    else:
        print("Invalid task number.")
```

```
def main():
    while True:
        print("\nOptions: 1.Add Task 2.View Tasks 3.Remove Task 4.Exit")
        choice = input("Enter your choice: ")

        if choice == '1.':
            task = input("Enter task: ")
            add_task(task)
        elif choice == '2.':
            view_tasks()
        elif choice == '3':
            task_number = int(input("Enter task number to remove: "))
            remove_task(task_number)
        elif choice == '4':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")

if _name____== "_main_":
    main()
```

OUTPUT:



```
C:\Users\Gopinath_A\Documents\pythonProject\pythonProject1\venv\Scripts\python.exe

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice: 1
Enter task: Complete Notes
Task 'Complete Notes' added.

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice: 1
Enter task: Drink 2l of Water
Task 'Drink 2l of Water' added.

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice: 1
Enter task: Practice Coding
Task 'Practice Coding' added.
```



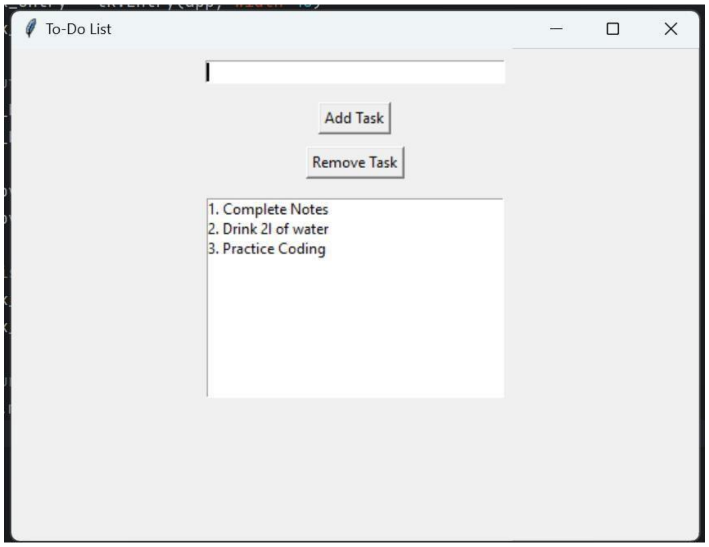
```
Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice: 2
Your tasks:
1. Complete Notes
2. Drink 2l of Water
3. Practice Coding

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice: 3
Enter task number to remove: 2
Task 'Drink 2l of Water' removed.

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice: 4
Exiting...

Process finished with exit code 0
```

ii) GUI (Graphical User Interface)

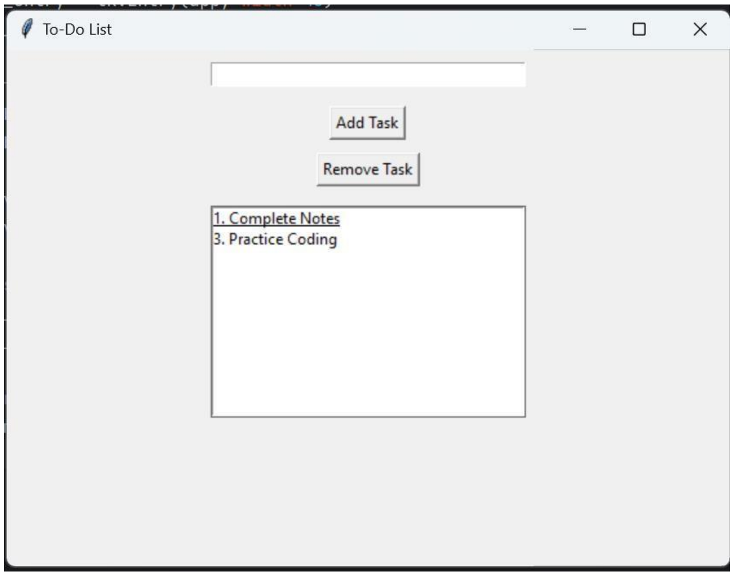
Tkinter to create a simple GUI for our To-Do List application.

```
import tkinter as tk
from tkinter import messagebox

tasks = []

def add_task():
    task = task_entry.get()
    if task:
        tasks.append(task)
        task_entry.delete(0, tk.END)
        update_task_list()
    else:
        messagebox.showwarning("Warning", "Task cannot be empty")

def update_task_list():
    task_list.delete(0, tk.END)
    for task in tasks:
        task_list.insert(tk.END, task)


def remove_task():
    selected_task_index = task_list.curselection()
    if selected_task_index:
        task_list.delete(selected_task_index)
        tasks.pop(selected_task_index[0])

app = tk.Tk()
app.title("To-Do List")

task_entry = tk.Entry(app, width=40)
task_entry.pack(pady=10)

add_button = tk.Button(app, text="Add Task", command=add_task)
add_button.pack(pady=5)

remove_button = tk.Button(app, text="Remove Task", command=remove_task)
remove_button.pack(pady=5)

task_list = tk.Listbox(app, width=40, height=10)
task_list.pack(pady=10)

app.mainloop()
```

OUTPUT:



```
1. Complete Notes
2. Drink 2l of water
3. Practice Coding
```



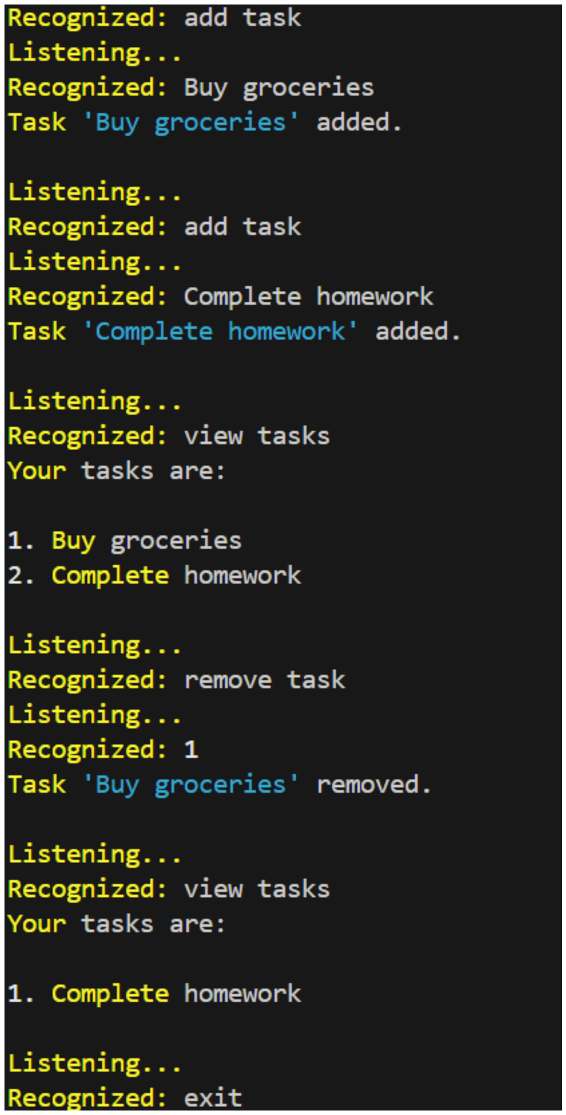
```
1. Complete Notes
3. Practice Coding
```

iii) VUI (Voice User Interface)

speech_recognition library for voice input and the pyttsx3 library for text-to-speech output. Make sure you have these libraries installed (pip install SpeechRecognition pyttsx3).

```
import speech_recognition as sr
import pyttsx3

tasks = []
recognizer = sr.Recognizer()
engine = pyttsx3.init()

def add_task(task):
    tasks.append(task)
    engine.say(f"Task {task} added")
    engine.runAndWait()

def view_tasks():
    if tasks:
        engine.say("Your tasks are")
        for task in tasks:
            engine.say(task)
    else:
        engine.say("No tasks to show")
    engine.runAndWait()

def remove_task(task_number):
    if 0 < task_number <= len(tasks):
        removed_task = tasks.pop(task_number - 1)
        engine.say(f"Task {removed_task} removed")
    else:
        engine.say("Invalid task number")
    engine.runAndWait()

def recognize_speech():
    with sr.Microphone() as source:
        print("Listening...")
        audio = recognizer.listen(source)
        try:
            command = recognizer.recognize_google(audio)
            return command
        except sr.UnknownValueError:
            engine.say("Sorry, I did not understand that")
            engine.runAndWait()
            return None

def main():
    while True:
        engine.say("Options: add task, view tasks, remove task, or exit")
        engine.runAndWait()

        command = recognize_speech()
        if not command:
            continue

        if "add task" in command:
            engine.say("What is the task?")
            engine.runAndWait()
            task = recognize_speech()
            if task:
                add_task(task)
        elif "view tasks" in command:
            view_tasks()
        elif "remove task" in command:
            engine.say("Which task number to remove?")
            engine.runAndWait()
            task_number = recognize_speech()
            if task_number:
                remove_task(int(task_number))
        elif "exit" in command:
            engine.say("Exiting...")
            engine.runAndWait()
            break
        else:
            engine.say("Invalid option. Please try again.")
            engine.runAndWait()

if _name____== "_main_":
    main()
```

OUTPUT:



```
Recognized: add task
Listening...
Recognized: Buy groceries
Task 'Buy groceries' added.

Listening...
Recognized: add task
Listening...
Recognized: Complete homework
Task 'Complete homework' added.

Listening...
Recognized: view tasks
Your tasks are:

1. Buy groceries
2. Complete homework

Listening...
Recognized: remove task
Listening...
Recognized: 1
Task 'Buy groceries' removed.

Listening...
Recognized: view tasks
Your tasks are:

1. Complete homework

Listening...
Recognized: exit
```

RESULT:

Thus the codes to develop and compare CLI, GUI, and Voice User Interfaces (VUI) for the same task and assess user satisfaction using Python (Tkinter for GUI, Speech Recognition for VUI), Terminal have been executed successfully.

Ex. No. : 3a **Date : 22.02.2025**

Create a prototype with familiar and unfamiliar navigation elements. Evaluate ease of use with different user groups using proto.io

AIM:

The aim is to develop a prototype incorporating both familiar and novel navigation elements and assess usability among diverse user groups using Proto.io.

PROCEDURE:

Tool Link: https://proto.io/

Step 1: Sign Up and Log In

1. Go to proto.io.
2. Sign up for a new account or log in if you already have one.

Step 2: Create a New Project

1. Click on "Create New Project."
2. Give your project a name (e.g., "Simple App Example").
3. Select the device type (e.g., Mobile - iPhone X).
4. Click "Create" to start the project.

Step 3: Design the Home Screen

1. Add a New Screen:

○ Click on the "+" button in the left panel to add a new screen.
○ Choose "Blank" and name it "Home."

2.Add Elements to the Home Screen:

○ Drag a "Header" widget from the "Widgets" panel to the top of the screen.

○ Double-click the header to edit the text and change it to "Home Screen."

○ Drag a "Button" widget onto the screen. Place it in the center.

○ Double-click the button to edit the text and change it to "Go to Profile."

3.Add Interaction:

○ Select the button and click on the "Interactions" tab on the right panel.

○ Click "+ Add Interaction."

○ Set the trigger to "Tap/Click."

○ Set the action to "Navigate to Screen" and choose "New Screen."

○ Create a new screen and name it "Profile."

Step 4: Design the Profile Screen

1. **Add Elements to the Profile Screen:**

○ On the newly created Profile screen, drag a "Header" widget to the top of the screen.

○ Double-click the header to edit the text and change it to "Profile Screen."

○ Drag an "Image" widget onto the screen. Place it below the header.

○ Double-click the image to upload a profile picture or any placeholder image.

○ Drag a "Text" widget onto the screen to add some profile information (e.g., "John Doe, Software Engineer").

2. **Add Back Button:**

- Drag a "Button" widget onto the screen.
- Double-click the button to edit the text and change it to "Back to Home."

3. **Add Interaction:**

- Select the button and click on the "Interactions" tab on the right panel.
- Click "+ Add Interaction."
- Set the trigger to "Tap/Click."
- Set the action to "Navigate to Screen" and choose "Home."

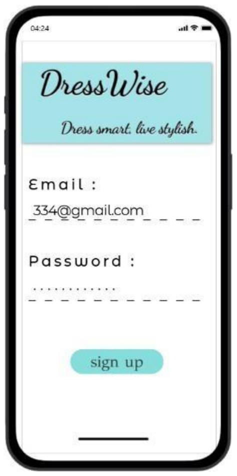
Step 5: Preview the Prototype

1. Click on the "Preview" button in the top-right corner.
2. Interact with the prototype by clicking on the buttons to navigate between the Home and Profile screens.

Step 6: Share the Prototype

1. Click on the "Share" button in the top-right corner.
2. Copy the shareable link and send it to others for feedback.

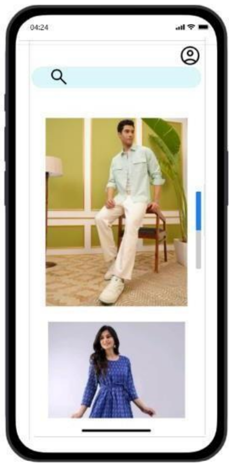
Output :

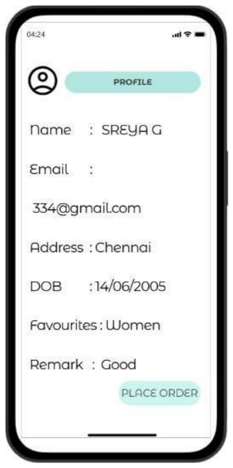

DressWise

Dress smart. live stylish.

Email :
334@gmail.com

Password :
....




Name : SREYA G

Email :

334@gmail.com

Address : Chennai

DOB : 14/06/2005

Favourites : Women

Remark : Good

Result :

Hence, creating a prototype with familiar and unfamiliar navigation and using different user groups using prto.io has been successfully executed.

Ex. No. : 3b **Date : 22.02.2025**

Create a prototype with familiar and unfamiliar navigation elements. Evaluate ease of use with different user groups using wireflow

Aim:

The aim is to design a prototype with both well-known and new navigation elements and measure user-friendliness across different user groups using Wireflow.

Procedure:

Step 1: Plan Your Prototype

1. Define Navigation Elements:
   - *Familiar*: Standard menus, top bars, footers, and sidebar navigation.
   - *Unfamiliar*: Novel features such as hidden menus, gesture-based navigation, or custom swipes.
2. Sketch Your Layout:
   - Start with paper sketches or use tools like Figma or Sketch to visualize your design concepts.

Step 2: Set Up Your Wireflow Project

1. Sign Up/Log In:
   - Head to Wireflow and create an account or log in if you already have one.
2. Start a New Project:
   - Click on "New Project" and name it. Choose a template or start from scratch.

Step 3: Design the Prototype

1. Add Familiar Navigation Elements:
   - Drag and drop components like menus, header bars, buttons, etc., into your screens.
2. Incorporate Unfamiliar Elements:
   - Introduce hidden menus, unique gestures, or unexpected interactions.
3. Link Screens:
   - Use Wireflow's linking tools to create connections and transitions between screens.

Step 4: Prepare for Usability Testing

1. Identify User Groups:
    - Segment users based on age, tech-savviness, or previous experience with similar products.
2. Recruit Participants:
    - Use online tools like UserTesting, forums, or social media to find participants.

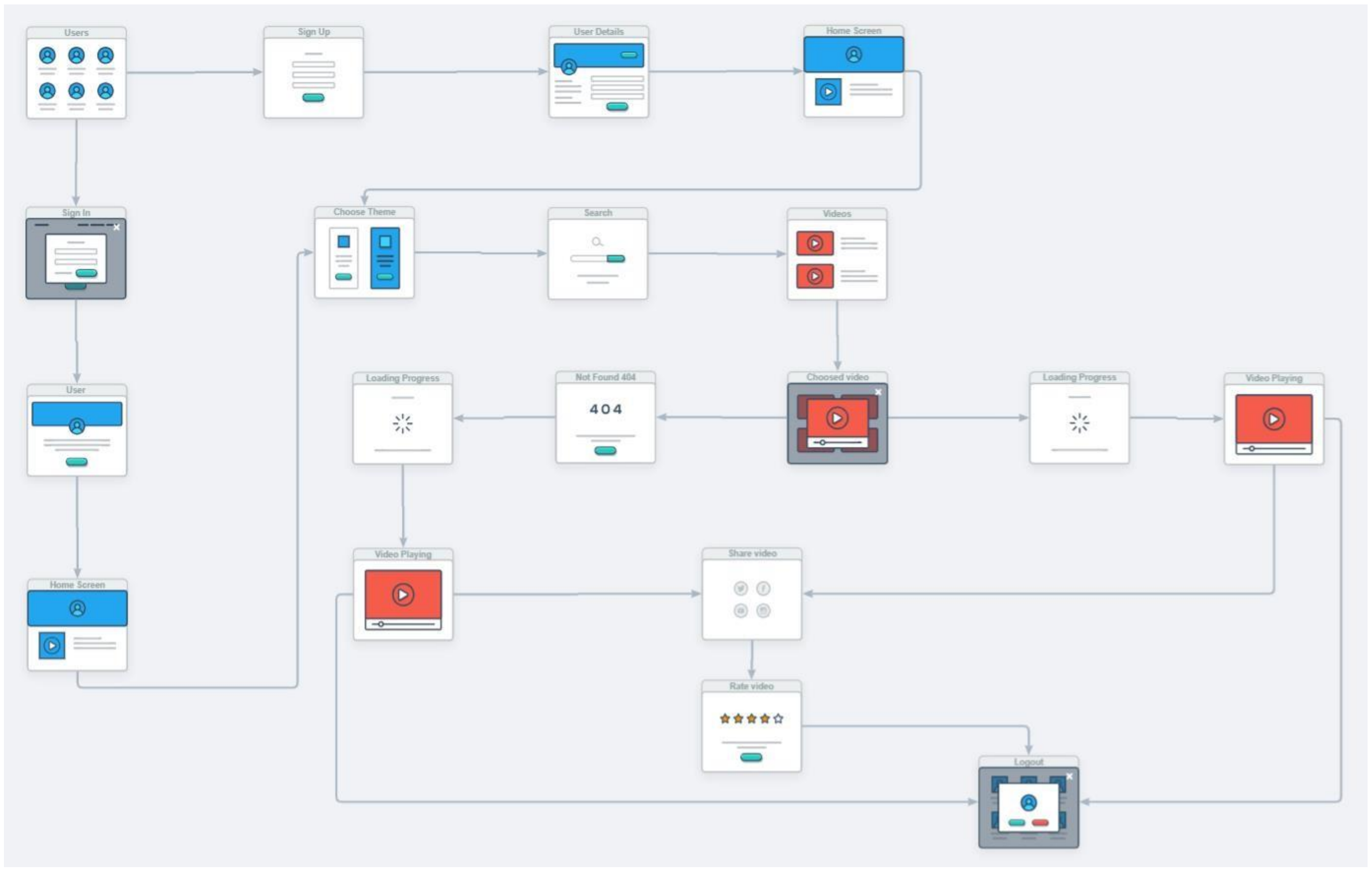
Step 5: Conduct Testing

1. Share the Prototype:
    - Invite users to interact with your prototype via a shareable link from Wireflow.
2. Test Sessions:
    - Ask users to complete tasks using both types of navigation. Observe their interactions and collect feedback.
3. Collect Feedback:
    - Utilize Wireflow's feedback features or conduct follow-up interviews to gather detailed responses.

Step 6: Analyze and Report

1. Analyze Data:
    - Review the feedback and data collected. Look for patterns in ease of use and user preferences.
2. Compare Results:
    - Compare how different user groups interacted with familiar vs. unfamiliar navigation.
3. Create a Report:
    - Summarize your findings, highlighting insights, challenges, and recommendations

Output:



Result:

Hence the design of a prototype with both well-known and new navigation elements and measure user-friendliness across different user groups using Wireflow has been successfully studied and executed.

Ex. No. : 4a **Date : 08.03.2025**

Conduct task analysis for an app (e.g., online shopping) and document user flows. Create corresponding wireframes using Lucidchart

---

Aim:

To understand and document the steps a user takes to complete the main tasks within an online shopping app.

Procedure:

Step 1: Assigning Tasks

1. Browsing Products
2. Searching for a Specific Product
3. Adding a Product to the Cart
4. Checking Out

Step 2: Document User Flows

1. Browsing Products

   1. Home Screen: User lands on the home page with product categories.
   2. Product Categories: User taps on a category to view products.
   3. Product List: User scrolls through the product list.
   4. Product Details: User taps on a specific product to see details. Home Screen ->

Product Categories -> Product List -> Product Details

2. Searching for a Specific Product

   1. Search: User taps the search bar or icon.
   2. Enter Query: User types the product name or keyword.
   3. Search Results: User reviews matching items.
   4. Product Details: User taps on a specific product to see details. Search ->

Enter Query -> Search Results -> Product Details

3. Adding a Product to the Cart

1. View Products: User browses or searches for a product.
2. Product Details: User taps on the product to see more info.
3. Add to Cart: User clicks "Add to Cart". View

Products -> Product Details -> Add to Cart

4. Checking Out

1. Open Cart: User taps on the cart icon.
2. Review Cart: User checks all products.
3. Proceed to Checkout: User clicks "Checkout".
4. Enter Shipping Info: User provides shipping details.
5. Enter Payment Info: User provides payment details.
6. Place Order: User clicks "Place Order".

Open Cart -> Review Cart -> Proceed to Checkout -> Enter Shipping Info -
> Enter Payment Info -> Place Order

Step-by-Step Procedure to Create User Flows in Lucidchart

1. Create a New Document

- Go to Lucidchart and sign in or sign up if you don't have an account.
- Click on + Document or Create New Diagram.

2. Select a Template

- You can start with a blank document or select a flowchart template.
- For this example, let's start with a blank document.

3. Add Shapes for Each Step

- Drag and drop shapes from the left sidebar to represent different steps in your flow (e.g., rectangles for actions, diamonds for decisions).
- Name each shape based on the steps from the task analysis:
    - Login/Register
    - Browsing Products
    - Adding Products to Cart
    - Managing Cart
    - Checkout Process
    - Tracking Orders

4. Connect the Shapes

- Use connectors to link the shapes, indicating the flow from one step to the next.
- Add arrows to show the direction of the flow.

5. Add Details to Each Step

- Double-click on each shape to add text describing the action or decision.
- For example, for the "Login/Register" step, you might add:
  - Open the app
  - Click on "Sign Up" or "Login"
  - Enter details (username, email, password)
  - Click "Submit"
  - Verification through email or phone (if required)
  - Redirect to the home screen upon successful login

6. Use Different Shapes for Different Actions

- Use rectangles for general actions.
- Use diamonds for decision points (e.g., "Is the user logged in?").
- Use ovals for start and end points.

7. Customize and Organize Your Flowchart

- Arrange the shapes and connectors logically.
- Use different colors to distinguish between types of steps or user roles.
- Group related steps into sections for better clarity.

8. Review and Save Your Flowchart

- Review the flowchart to ensure all steps are included and connected correctly.
- Save your flowchart by clicking on File -> Save.

9. Share and Collaborate

  - Click on the Share button to collaborate with others.
  - You can also export your flowchart as an image or PDF for presentation purposes.

Example Flowchart Breakdown:

Login/Register Flow

- Steps:
  - Open the app
  - Click on "Login" or "Register"
  - Enter details

  - Verify (if required)
  - Redirect to the home screen

Browse and Search Flow

- Steps:
  - Navigate to categories or use search bar
  - Apply filters/sorting options
  - View product details Add

to Cart Flow

- Steps:
  - View product details
  - Select options (size, color, quantity)
  - Add product to cart

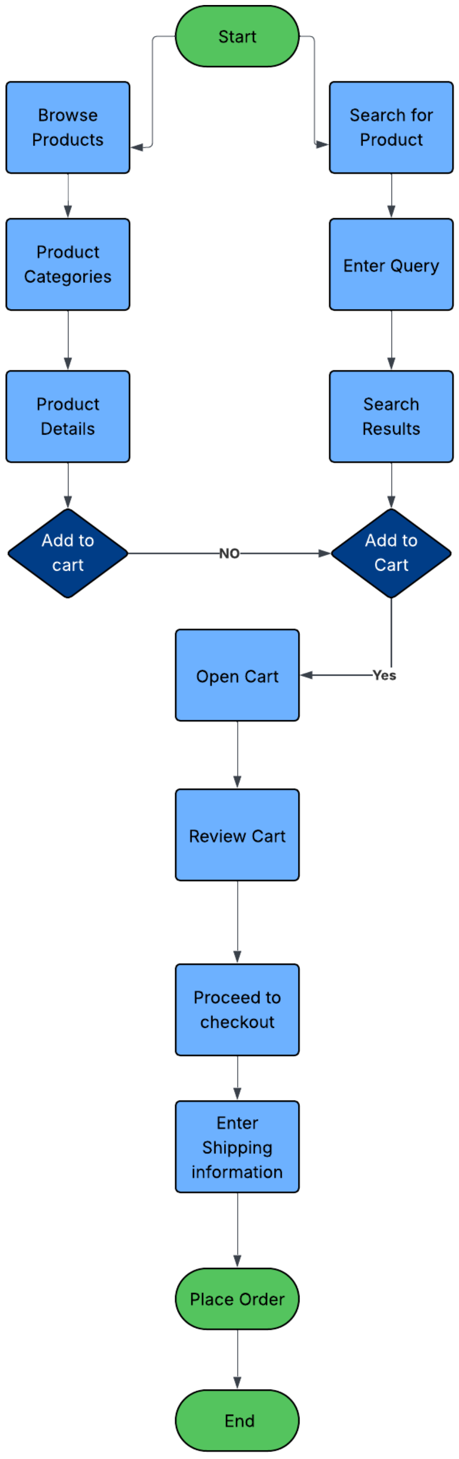
Checkout Flow

- Steps:
  - Review cart
  - Proceed to checkout
  - Enter shipping information
  - Select payment method
  - Confirm and place order

Order Tracking Flow

- Steps:
  - Navigate to "My Orders"
  - Select order to track
  - View tracking details

Output:


Start
Browse Products
Search for Product
Product Categories
Enter Query
Product Details
Search Results
Add to cart
NO
Add to Cart
Yes
Open Cart
Review Cart
Proceed to checkout
Enter Shipping information
Place Order
End

Result:

Hence conducted the task analysis for an app (e.g., online shopping) and documented the user flows. Created corresponding wireframes using Lucidchart has been successfully done.

LINK : https://lucid.app/lucidchart/33506edd-f02c-4927-8690-64a582109bc7/edit?viewport_loc=-1023%2C-39%2C4583%2C2002%2C0_0&invitationId=inv_f13c88b5-da2b-4dcd- aae4-f5533e499d2e

Ex. No. : 4b **Date : 22.03.2025**

Conduct task analysis for an app (e.g., online shopping) and document user flows. Create corresponding wireframes using dia

AIM:

The aim is to perform task analysis for an app, such as online shopping, document user flows, and create corresponding wireframes using Dia.

PROCEDURE:

Tool link: http://dia-installer.de/

1. Install Dia:

○ Download Dia from the official website (http://dia- installer.de/)

○ Install Dia on your computer

○ Open Dia:

○ Launch the Dia application.

2. Create New Diagram:

○ Go to File , New Diagram.

○ select Flowchart as the diagram type.

3. Add Shapes:

○ Use the shape tools (rectangles, ellipses, etc.) to create wireframes for

each screen.

■ For example:

■ Home Page: Rectangle

- Product Categories: Rectangle
- Product Listings: Rectangle
- Product Details: Rectangle
- Cart: Rectangle

- Checkout: Rectangle
- Order Confirmation: Rectangle
- Order History: Rectangle

4. Connect Shapes:

  - Use the line tool to connect shapes, representing the user flows.

- For example:
- Home Page ,Product Categories
- Product Categories ,Product Listings
- Product Listings ,Product Details
- Product Details ,Cart
- Cart ,Checkout
- Checkout ,Order Confirmation
- Order Confirmation ,Order History

5. Label Shapes:

○ Double-click on each shape to add labels.

■ For example:

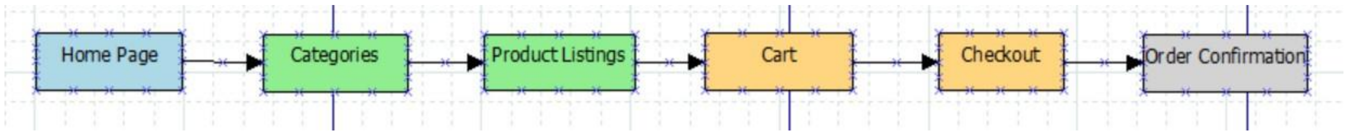
■ Label the rectangle as “Home Page”, “Categories”, “Product Listings”, “Product Details”, “Cart”, “Checkout”, “Order Confirmation”, “Order History”.

6. Save the Diagram:

○ Go to File ,Save As.

○ Save the diagram with a meaningful name, such as “Online Shopping App User Flows”.

OUTPUT:



RESULT:

Thus, the task analysis and user flow for an online shopping app were effectively represented in Dia using structured wireframes and logical connections.

Ex. No. : 5a **Date : 22.03.2025**

Simulate the lifecycle stages for UI design using the RAD model and develop a small interactive interface using Axure RP.

Aim:

The aim is to demonstrate the lifecycle stages of UI design via the RAD model and develop a small interactive interface employing Axure RP.

Procedure:

RAD Model (Rapid Application Development): The RAD model emphasizes quick development and iteration. It consists of the following phases:

1. Requirements Planning:
   - Gather initial requirements and identify key features of the UI.
   - Engage stakeholders to understand their needs and expectations.
2. User Design:
   - Create initial prototypes and wireframes.
   - Conduct user feedback sessions to refine the designs.
   - Use tools like Axure RP to develop interactive prototypes.
3. Construction:
   - Develop the actual UI based on the refined designs.
   - Perform iterative testing and feedback cycles.
4. Cutover:
   - Deploy the final UI.
   - Conduct user training and support.

Axure RP Interactive Interface Development

Phase 1: Requirements Planning

1. Identify Key Features:
   - Navigation (Home, Product Categories, Product Details, Cart, Checkout, Order Confirmation, Order History)
   - User actions (Browsing, Searching, Adding to Cart, Checkout, Tracking Orders)
2. Create a Requirements Document:
   - List all features and functionalities.

- Document user stories and use cases.

Phase 2: User Design

1. Install and Launch Axure RP:
   - Download and install Axure RP from Axure's official website.
   - Launch the application.
2. Create a New Project:
   - Go to File -> New to create a new project.
   - Name the project (e.g., "Shopping App Interface").
3. Create Wireframes:
   - Use the widget library to drag and drop elements onto the canvas.
   - Design wireframes for each screen:
     - Home Page
     - Product Categories
     - Product Listings
     - Product Details
     - Cart
     - Checkout
     - Order Confirmation
4. Add Interactions:
   - Select an element (e.g., button) and go to the Properties panel.
   - Click on Interactions and choose an interaction (e.g., OnClick).
   - Define the action (e.g., navigate to another screen).
5. Create Masters:
   - Create reusable components (e.g., headers, footers) using Masters.
   - Drag and drop masters onto the wireframes.
6. Add Annotations:
   - Add notes to describe each element's purpose and functionality.
   - Use the Notes panel to add detailed annotations.

Phase 3: Construction

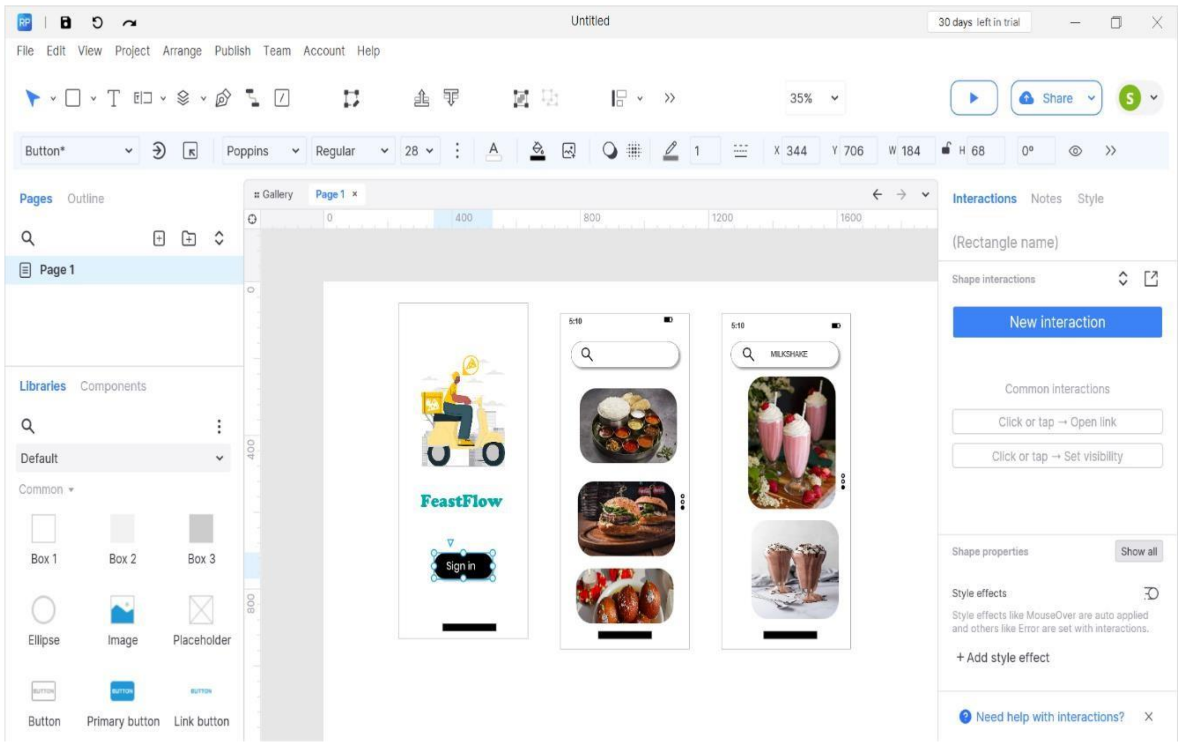
1. Develop Interactive Prototypes:
   - Convert wireframes into interactive prototypes by adding interactions and transitions.
   - Use dynamic panels to create interactive elements (e.g., carousels, pop-ups).
2. Test and Iterate:

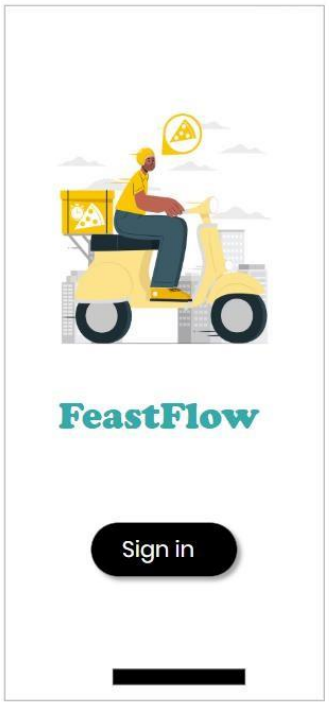
- Preview the prototype using the Preview button.
- Gather feedback from users and stakeholders.
- Make necessary adjustments based on feedback.

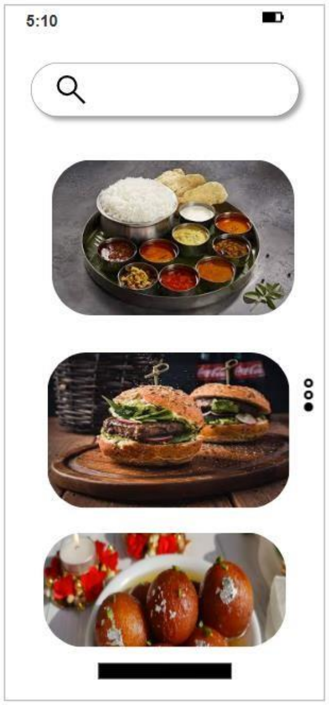
Phase 4: Cutover

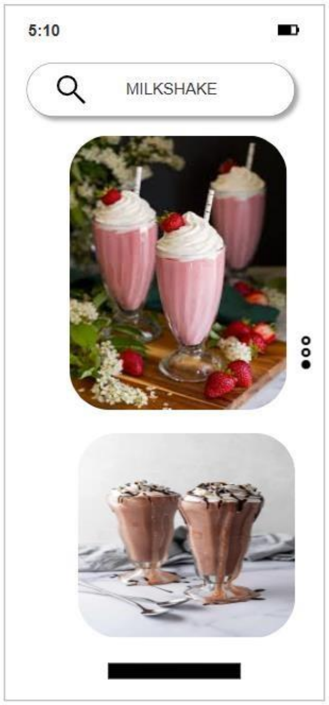
1. Finalize and Export:
   - Finalize the design and interactions.
   - Export the prototype as an HTML file or share it via Axure Cloud.
2. User Training and Support:
   - Conduct training sessions to familiarize users with the new interface.
   - Provide documentation and support for any issues.

Output:









Result:

Thus the demonstration the lifecycle stages of UI design via the RAD model and successfully developed a small interactive interface employing Axure RP.

Ex. No. : 5b **Date : 29.03.2025**

Simulate the life cycle stages for UI design using the RAD model and develop a small interactive interface using OpenProj

AIM:

The aim is to recreate the lifecycle stages of UI design using the RAD model and design a small interactive interface with OpenProj

PROCEDURE:

Tool Link: https://sourceforge.net/projects/openproj/

Step 1: Requirements Planning

1. Gather Requirements:
   - Identify key features and functionalities needed for your interface.
   - Example: A simple "Login" and "Register" interface with debug logs.
2. Define Use Cases:
   - Specify use cases for user login and registration.
   - Example: User logs in with valid credentials, user registers with a new account.

Output in OpenProj:

- Create a new project.
- Add tasks: "Gather Requirements" and "Define Use Cases." ● Set

durations and dependencies for each task. **Step 2: User Design**

1. Sketch Initial Designs:

○ Draw rough sketches of the "Login" and "Register" screens on paper.

2. Create Digital Wireframes:

○ Use a tool like Figma or Sketch to create digital wireframes.

Example Wireframes:

1. Login Screen: Username field, Password field, Login button, Register link.

2. Register Screen: Username field, Email field, Password field, Confirm Password field, Register button.

Output in OpenProj:

● Add tasks: "Sketch Initial Designs" and "Create Digital Wireframes."
● Allocate time and resources to complete these tasks. **Step 3:**

Rapid Prototyping

1. Develop Prototypes:

○ Use a tool like Axure RP to convert wireframes into interactive prototypes.

2. Test Prototypes:

○ Share prototypes with stakeholders for feedback. ○ Collect

feedback and iterate on the design.

Output:

● Interactive prototypes for "Login" and "Register" screens. Output in

OpenProj:

● Add tasks: "Develop Prototypes" and "Test Prototypes." ● Set

dependencies and milestones. **Step 4: User Acceptance/Testing**

1. Review Prototype:

   ○ Conduct user and stakeholder reviews.

2. Conduct Usability Testing:

   ○ Perform usability testing and document feedback. Output:

● Documented feedback and test results. Output in

OpenProj:

● Add tasks: "Review Prototype" and "Usability Testing." ●

Track progress and resources.

Step 5: Implementation

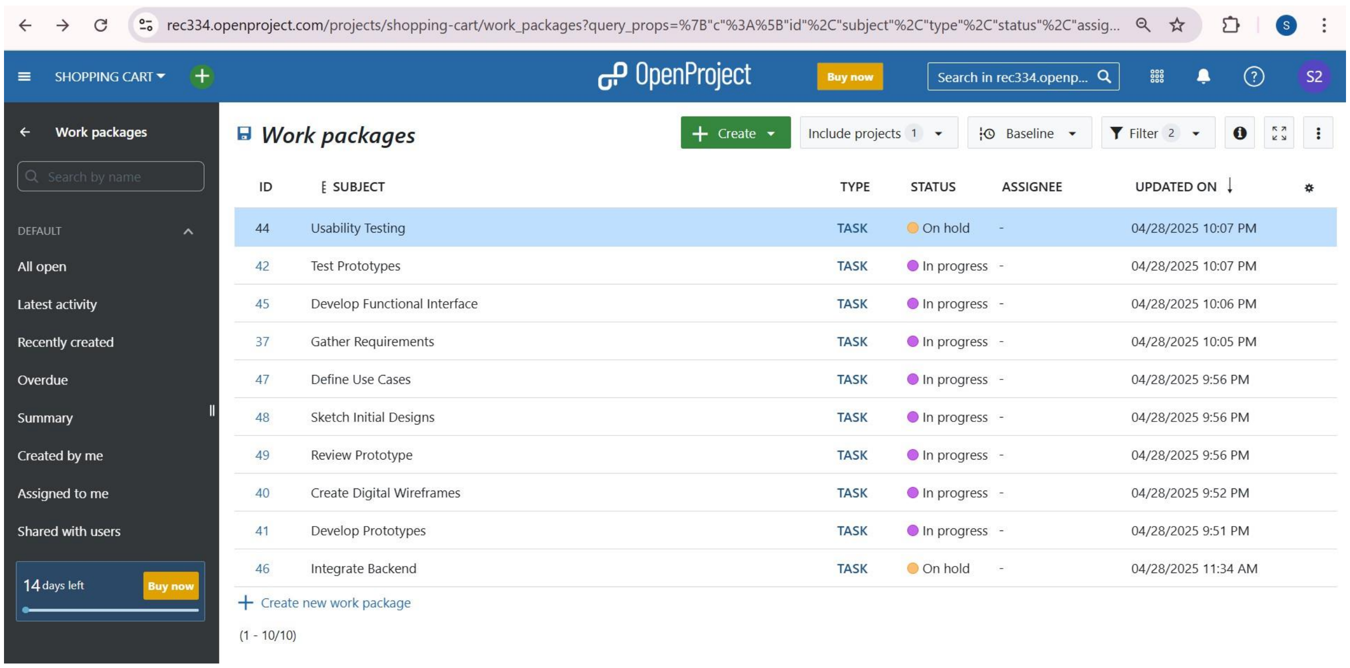
1. Develop Functional Interface:

   ○ Implement final designs and functionalities based on feedback.

2. Integrate Backend (if required):

   ○ Connect the UI with backend services for tasks like user authentication.

OUTPUT:



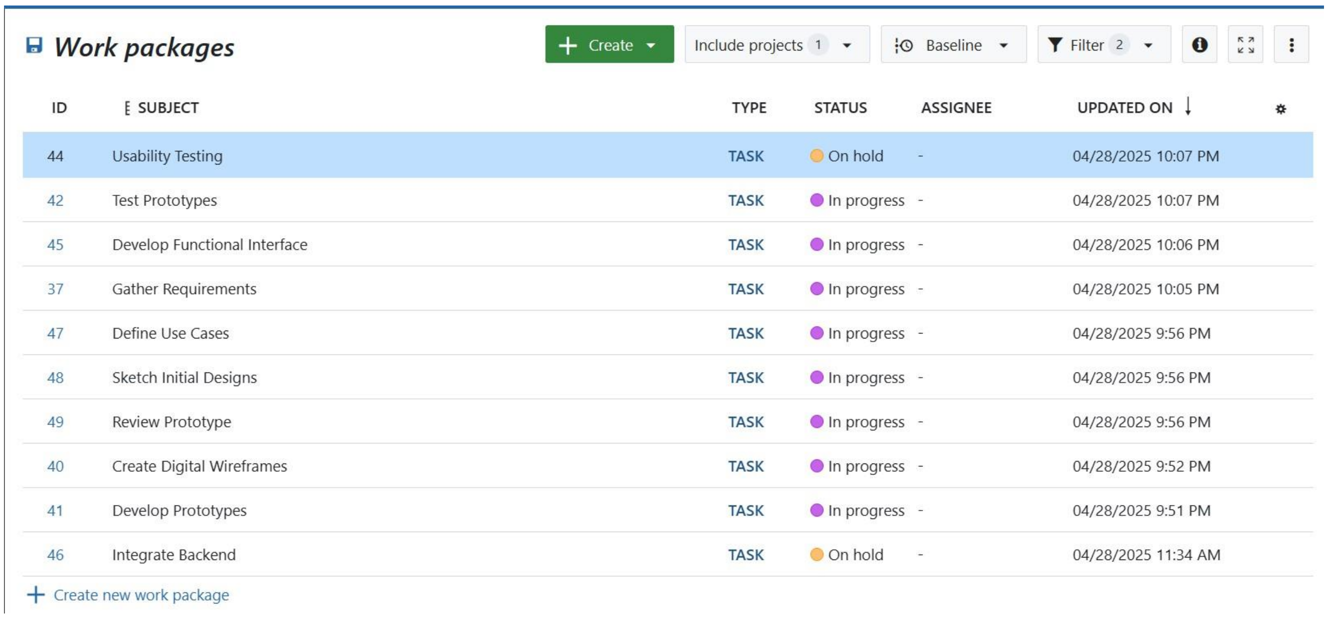
rec334.openproject.com/projects/shopping-cart/work_packages?query_props=%7B"c"%3A%5B"id"%2C"subject"%2C"type"%2C"status"%2C"assig...

Work packages

ID	SUBJECT	TYPE	STATUS	ASSIGNEE	UPDATED ON
44	Usability Testing	TASK	On hold	-	04/28/2025 10:07 PM
42	Test Prototypes	TASK	In progress	-	04/28/2025 10:07 PM
45	Develop Functional Interface	TASK	In progress	-	04/28/2025 10:06 PM
37	Gather Requirements	TASK	In progress	-	04/28/2025 10:05 PM
47	Define Use Cases	TASK	In progress	-	04/28/2025 9:56 PM
48	Sketch Initial Designs	TASK	In progress	-	04/28/2025 9:56 PM
49	Review Prototype	TASK	In progress	-	04/28/2025 9:56 PM
40	Create Digital Wireframes	TASK	In progress	-	04/28/2025 9:52 PM
41	Develop Prototypes	TASK	In progress	-	04/28/2025 9:51 PM
46	Integrate Backend	TASK	On hold	-	04/28/2025 11:34 AM

Create new work package

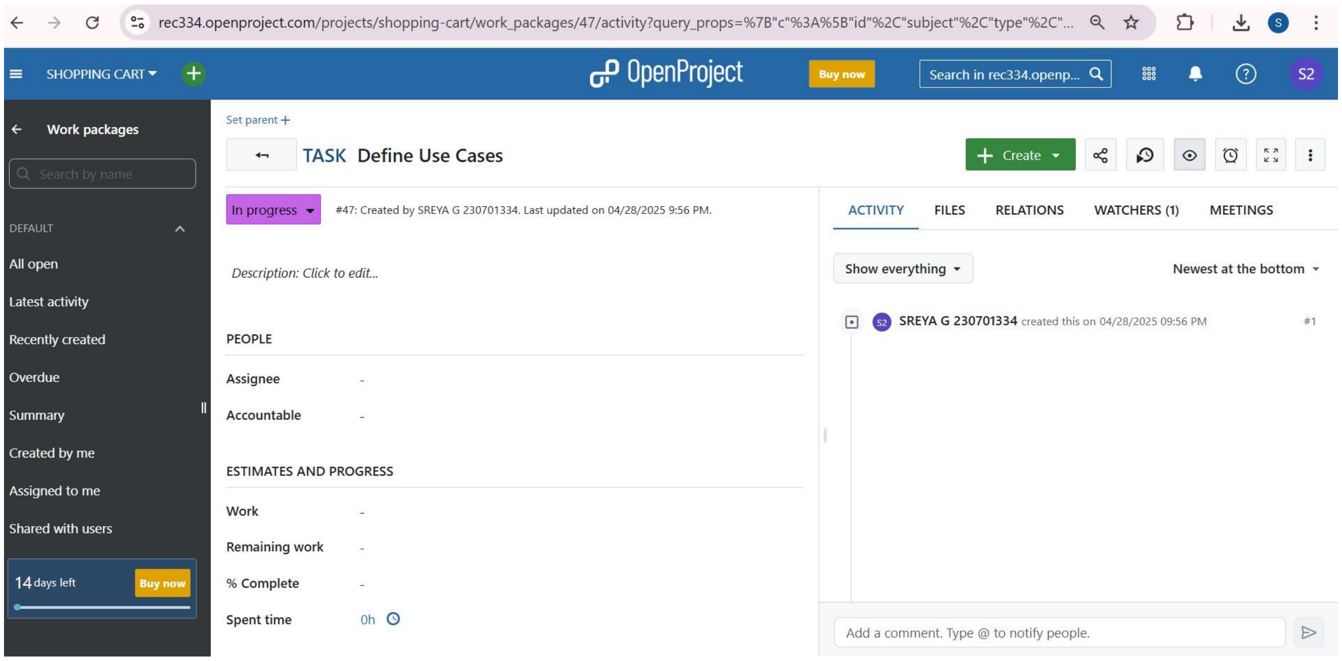
(1 - 10/10)



Work packages

ID	SUBJECT	TYPE	STATUS	ASSIGNEE	UPDATED ON
44	Usability Testing	TASK	On hold	-	04/28/2025 10:07 PM
42	Test Prototypes	TASK	In progress	-	04/28/2025 10:07 PM
45	Develop Functional Interface	TASK	In progress	-	04/28/2025 10:06 PM
37	Gather Requirements	TASK	In progress	-	04/28/2025 10:05 PM
47	Define Use Cases	TASK	In progress	-	04/28/2025 9:56 PM
48	Sketch Initial Designs	TASK	In progress	-	04/28/2025 9:56 PM
49	Review Prototype	TASK	In progress	-	04/28/2025 9:56 PM
40	Create Digital Wireframes	TASK	In progress	-	04/28/2025 9:52 PM
41	Develop Prototypes	TASK	In progress	-	04/28/2025 9:51 PM
46	Integrate Backend	TASK	On hold	-	04/28/2025 11:34 AM

Create new work package


rec334.openproject.com/projects/shopping-cart/work_packages/47/activity?query_props=%7B"c"%3A%5B"id"%2C"subject"%2C"type"%2C"...

SHOPPING CART | OpenProject | Buy now | Search in rec334.openp... | S2

Work packages

DEFAULT
- All open
- Latest activity
- Recently created
- Overdue
- Summary
- Created by me
- Assigned to me
- Shared with users

14 days left | Buy now

Set parent +

TASK Define Use Cases

In progress | #47: Created by SREYA G 230701334. Last updated on 04/28/2025 9:56 PM.

Description: Click to edit...

PEOPLE

Assignee	-
Accountable	-

ESTIMATES AND PROGRESS

Work	-
Remaining work	-
% Complete	-
Spent time	0h

ACTIVITY | FILES | RELATIONS | WATCHERS (1) | MEETINGS

Show everything | Newest at the bottom

SREYA G 230701334 created this on 04/28/2025 09:56 PM #1

Add a comment. Type @ to notify people.

RESULT:

Hence the lifecycle stages of UI design using the RAD model and design of a small interactive interface with OpenProj has been successfully executed.

Ex. No. : 6 **Date : 01.04.2025**

Experiment with different layouts and color schemes for an app. Collect user feedback on aesthetics and usability using GIMP(GNU Image Manipulation Program (GIMP)

AIM:

The aim is to trial different app layouts and color schemes and evaluate user feedback on aesthetics and usability using GIMP.

PROCEDURE:

Tool Link: https://www.gimp.org/ Step

1: Install GIMP

● Download and Install: Download GIMP from GIMP Downloads and install it on your computer.

Step 2: Create a New Project

1. Open GIMP:

○ Launch the GIMP application.

2. Create a New Canvas:

○ Go to File -&gt; New to create a new project.

○ Set the dimensions for your app layout (e.g., 1080x1920 pixels for a standard mobile screen).

Step 3: Design the Base Layout

1. Create the Base Layout:
○ Use the Rectangle Select Tool to create sections for different parts of your app (e.g., header, content area, footer).

○ Fill these sections with basic colors using the Bucket Fill Tool.

Example Output: A base layout with defined sections for header, content, and footer.

2. Add UI Elements:

○ Text Elements: Use the Text Tool to add text elements like headers, buttons, and labels.

○ Interactive Elements: Use the Brush Tool or Shape Tools to draw buttons, input fields, and other interactive elements.

Example Output: A layout with labeled sections and basic UI elements.

3. Organize Layers:

○ Use layers to separate different UI elements. This allows you to easily modify or experiment with individual components.

○ Name each layer according to its content (e.g., Header, Button1, InputField).

Step 4: Experiment with Color Schemes

1. Create Color Variants:
○ Duplicate Layout: Duplicate the base layout by right-clicking on the image tab and selecting Duplicate.

○ Change Colors: Use the Bucket Fill Tool or Colorize Tool to change the colors of the UI elements in each duplicate.
Example Output: Multiple color variants of the same layout.

2. Save Each Variant:

○ Save each color variant as a separate file (e.g., Layout1.png, Layout2.png, etc.).

○ Go to File -> Export As and choose the file format (e.g., PNG).

Step 5: Collect User Feedback

1. Prepare a Feedback Form:

○ Create Form: Create a feedback form using tools like Google Forms or Microsoft Forms.

○ Include Questions: Include questions about the aesthetics and usability of each layout and color scheme.

2. Share the Variants:

○ Distribute Files: Share the image files of the different layouts and color schemes with your users.

○ Provide Instructions: Provide clear instructions on how to view each variant and how to fill out the feedback form.

3. Gather Feedback:

○ Collect responses from users regarding their preferences and suggestions.

○ Analyze the feedback to determine which layout and color scheme are most preferred.

Step 6: Iterate and Refine

1. Refine the Design:

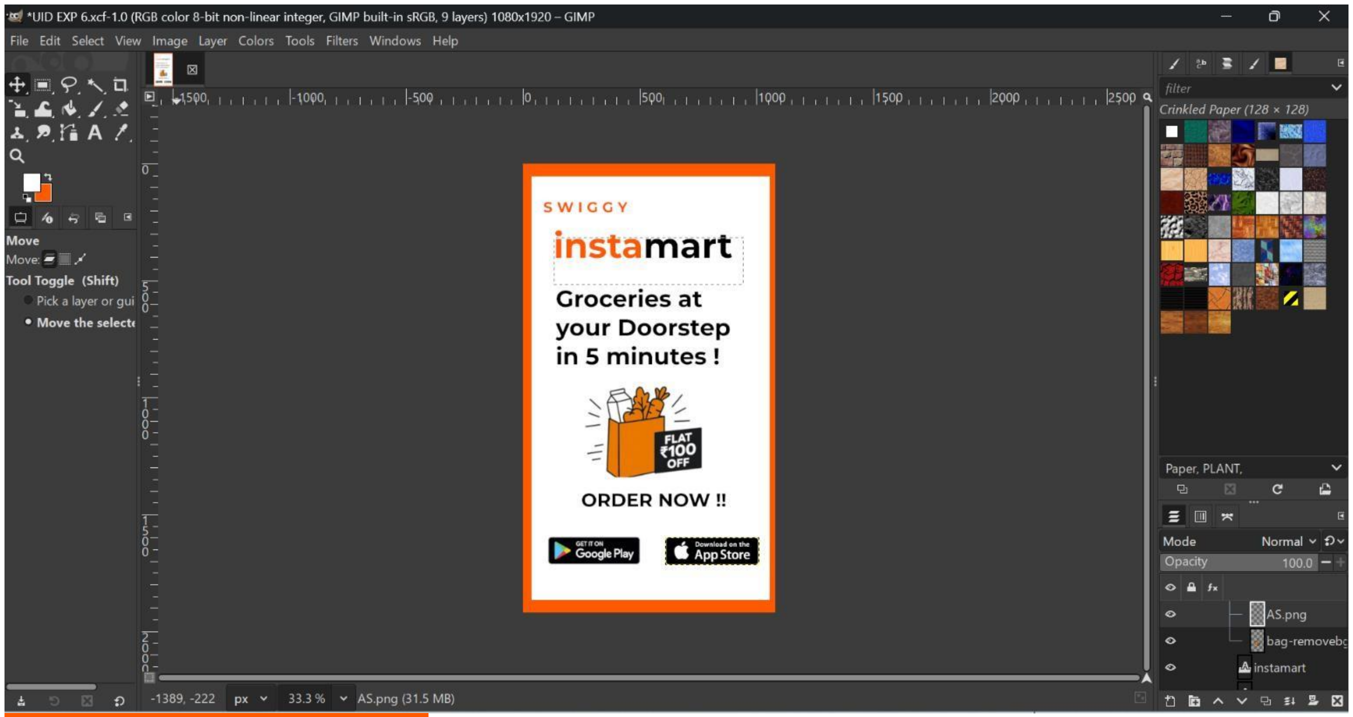
○ Based on the feedback, make necessary adjustments to the layout and color scheme.

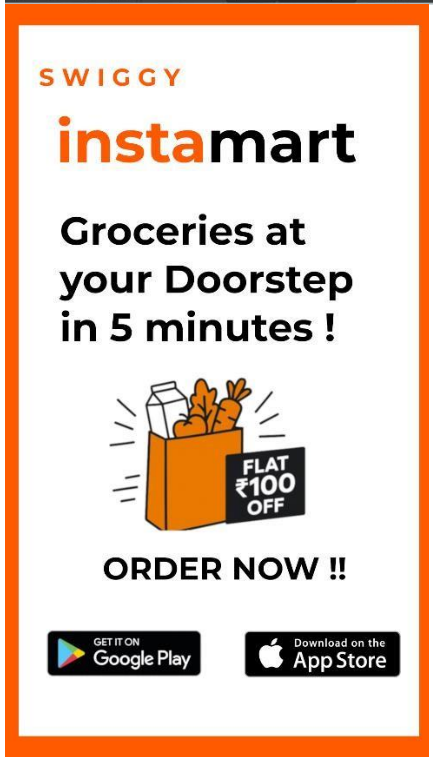
○ Experiment with additional variations if needed.

2. Final Testing:

○ Conduct a final round of testing with the refined design to ensure usability and aesthetic satisfaction.

OUTPUT:


*UID EXP 6.xcf-1.0 (RGB color 8-bit non-linear integer, GIMP built-in sRGB, 9 layers) 1080x1920 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
Move
Tool Toggle (Shift)
Pick a layer or gui
Move the selecte
Crinkled Paper (128 × 128)
Paper, PLANT,
Mode Normal
Opacity 100.0
AS.png
bag-removebg
instamart
-1389, -222 px 33.3 % AS.png (31.5 MB)


SWIGGY
instamart
Groceries at your Doorstep in 5 minutes !
FLAT ₹100 OFF
ORDER NOW !!
GET IT ON Google Play
Download on the App Store

RESULT:

Different layouts and color schemes for an app have been experimented and user feedback on aesthetics and usability using GIMP (GNU Image Manipulation Program (GIMP) has been collected.

Ex. No.: 7a **Date : 01.04.2025**

Develop low-fidelity paper prototypes for a banking app and convert them into digital wireframes using Pencil Project

AIM:

The aim is to develop low-fidelity paper prototypes for a banking app and convert them into digital wireframes with Pencil Project.

PROCEDURE:

Tool Link: https://pencil.evolus.vn/

Step 1: Create Low-Fidelity Paper Prototypes

1. **Define the Purpose and Features**:
   - Identify the core features of the banking app (e.g., login, account balance, transfers, bill payments).
2. **Sketch Basic Layouts**:
   - Use plain paper and pencils to sketch basic screens.
   - Focus on primary elements like buttons, menus, and forms.
3. **Iterate and Refine**:
   - Get feedback from users or stakeholders.
   - Iterate on your sketches to improve clarity and functionality.

Step 2: Convert Paper Prototypes to Digital Wireframes Using Pencil Project

1. **Install Pencil Project**:
   - Download and install Pencil Project from the official website.
2. **Create a New Document**:
   - Open Pencil Project and create a new document.
3. **Add Screens**:
   - Click on the "Add Page" button to create different screens (e.g., Login,

Dashboard, Transfer).

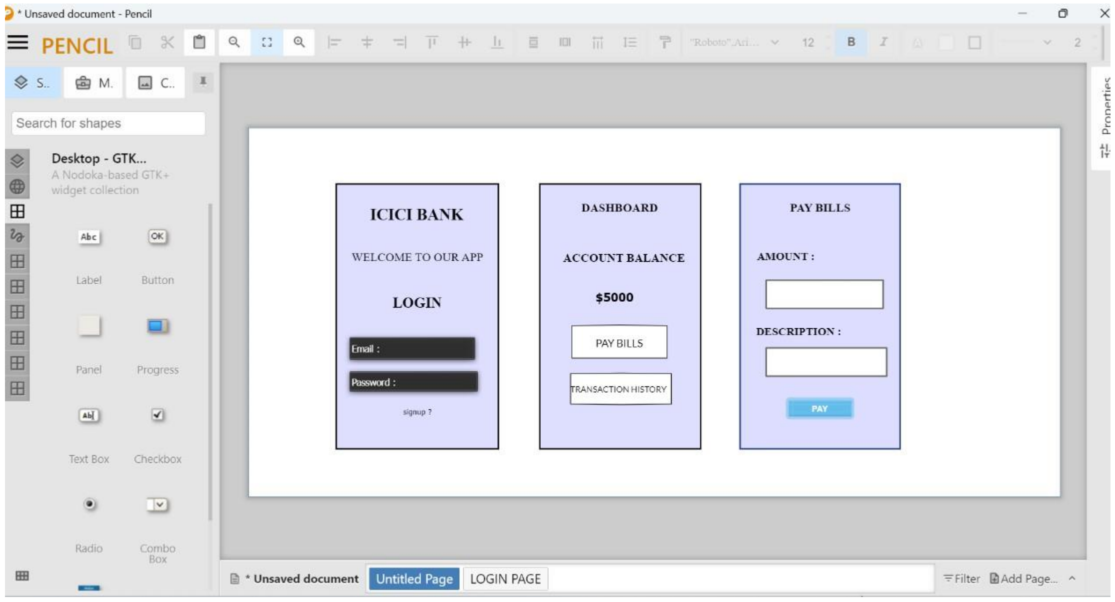
4. **Use Stencils and Shapes**:
   - Use the built-in stencils and shapes to create UI elements.
   - Drag and drop elements like buttons, text fields, and icons onto your canvas.
5. **Organize and Align**:
   - Arrange and align the elements to match your paper prototype.
   - Ensure that the design is user-friendly and intuitive.
6. **Link Screens**:
   - Use connectors to link different screens together.
   - Create navigation flows to show how users will interact with the app. 7.

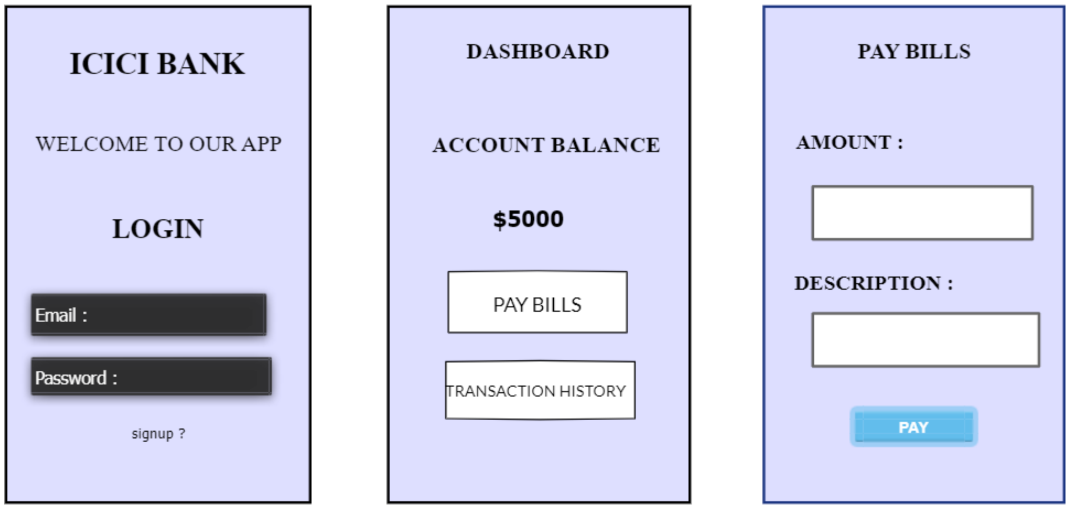
Add Annotations:

   - Include annotations to explain the functionality of different elements.

8. **Export Your Wireframes**:
   - Once satisfied with your digital wireframes, export them in your preferred format (e.g., PNG, PDF).

OUTPUT:





RESULT:

Hence we have developed low-fidelity paper prototypes for a banking app and converted them into digital wireframes with Pencil Project.

Ex. No.: 7b **Date :01.04.2025**

Develop low-fidelity paper prototypes for a banking app and convert them into digital wireframes using Inkscape

AIM:

The aim is to construct low-fidelity paper prototypes for a banking app and digitize them into wireframes using Inkscape.

PROCEDURE:

Tool Link: https://inkscape.org/

Step 1: Create Low-Fidelity Paper Prototypes

1. Identify Core Features:
   - Determine the essential features of the banking app (e.g., login, dashboard, account management, transfers).
2. Sketch Basic Layouts:
   - Use plain paper and pencils to sketch the main screens.
   - Focus on the primary elements like buttons, navigation menus, and input fields.
3. Iterate and Refine:
   - Get feedback from users or stakeholders.
   - Make necessary adjustments to improve clarity and functionality.

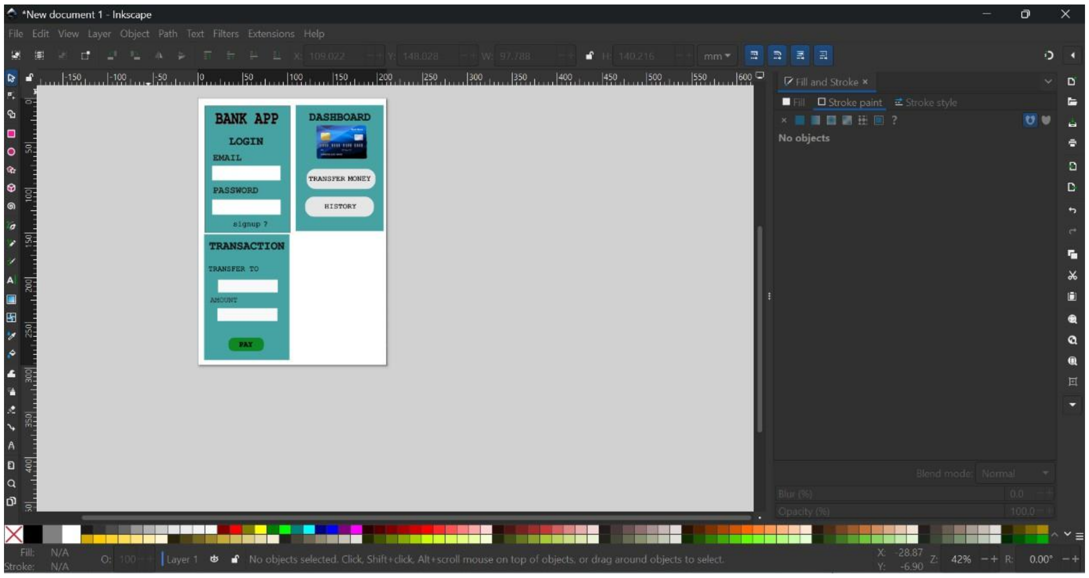
Step 2: Convert Paper Prototypes to Digital Wireframes Using Inkscape

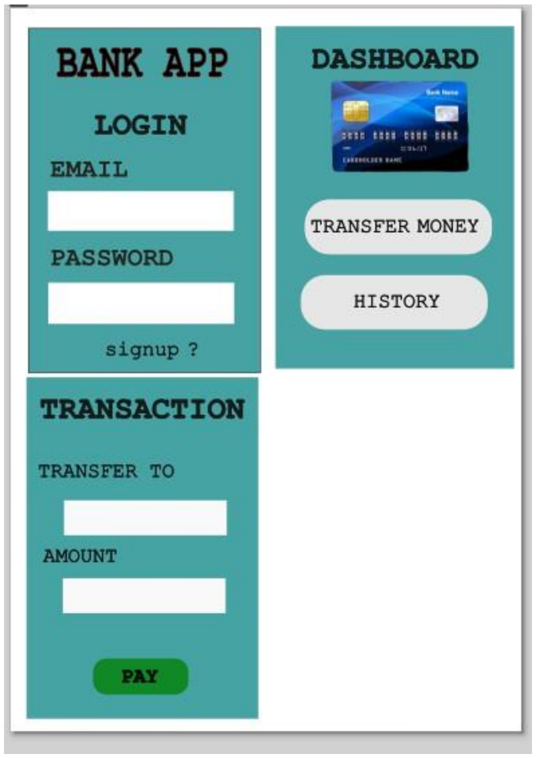
1. Install Inkscape:
   - Download and install Inkscape from the official website.
2. Create a New Document:
   - Open Inkscape and create a new document by clicking on File > New.
3. Set Up the Document:
   - Set the dimensions and grid for your design. Go to File > Document Properties to adjust the size.

- Enable the grid by going to View > Page Grid.

4. Draw Basic Shapes:
   - Use the rectangle and ellipse tools to draw the basic shapes for your UI elements (e.g., buttons, input fields, icons).
5. Add Text:
   - Use the text tool to add labels and placeholder text to your elements.
6. Organize and Align:
   - Arrange and align the elements to match your paper prototype.
   - Use the alignment and distribution tools to keep everything organized.
7. Group Elements:
   - Select related elements and group them together using Object > Group.
   - This helps keep your design organized and easy to edit.
8. Create Multiple Screens:
   - Duplicate your base layout to create different screens (e.g., login, dashboard, transfer).
   - Use Edit > Duplicate to create copies of your elements and arrange them for each screen.
9. Link Screens (Optional):
   - If you want to show navigation flows, you can add arrows or other indicators to demonstrate how users will move between screens.
10. Export Your Wireframes:
    - Once you're satisfied with your digital wireframes, export them by going to File > Export PNG Image.
    - Choose the appropriate settings and export each screen as needed.

OUTPUT:


*New document 1 - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help
Fill and Stroke
Fill
Stroke paint
Stroke style
No objects
BANK APP
LOGIN
EMAIL
PASSWORD
signup ?
DASHBOARD
TRANSFER MONEY
HISTORY
TRANSACTION
TRANSFER TO
AMOUNT
PAY
Layer 1
No objects selected. Click, Shift+click, Alt+scroll mouse on top of objects, or drag around objects to select.



RESULT:

Hence low-fidelity paper prototypes for a banking app and digitize them into wireframes using Inkscape is constructed.

Ex. No.: 8a **Date :08.04.2025**

Create storyboards to represent the user flow for a mobile app (e.g., food delivery app) using Balsamiq

AIM:

The aim is to create storyboards representing the user flow for a mobile app, such as a food delivery app, using Balsamiq.

PROCEDURE:

Tool Link: https://balsamiq.com/

Step 1: Define the User Flow

1. **Identify Key Screens**:
   - List the main screens your app will have (e.g., Home, Menu, Cart, Checkout, Order Confirmation).
2. **Map the User Journey**:
   - Understand the typical user journey through these screens (e.g., browsing menu, adding items to cart, checking out).

Step 2: Create Storyboards Using Balsamiq

1. **Install Balsamiq**:
   - Download and install Balsamiq from the https://balsamiq.com/ website.
2. **Create a New Project**:
   - Open Balsamiq and create a new project.
3. **Add Wireframe Screens**:
   - Use the “+” button to add new wireframe screens for each key screen in your app.
4. **Design Each Screen**:
   - Use Balsamiq's components to design the UI for each screen.
   - Include basic elements like buttons, text fields, and images.

5. **Organize the Flow**:
    - Arrange the screens in the order users will navigate through them.
    - Connect the screens with arrows to represent user actions.

Example Screens for Food Delivery App

1. **Home Screen**:
    - Search bar for finding restaurants
    - Categories for different cuisines
2. **Menu Screen**:
    - List of food items with images, names, and prices
    - Add to Cart buttons
3. **Cart Screen**:
    - Items added to the cart with quantity and total price
    - Checkout button
4. **Checkout Screen**:
    - Delivery address form
    - Payment options
    - Place Order button
5. **Order Confirmation Screen**:
    - Order summary
    - Estimated delivery time

Example Output

Here's how the wireframes might look:

Home Screen

- **Search Bar**: Allows users to search for restaurants.
- **Categories**: Buttons for different cuisines (e.g., Italian, Chinese).

Menu Screen

- **Food Items List**: Displays food items with images, names, and prices.
- **Add to Cart**: Button to add items to the cart.

Cart Screen

- **Items Added**: Lists items added to the cart with quantity and prices.
- **Checkout Button**: Proceed to checkout.

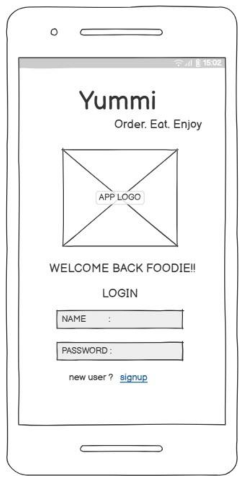
Checkout Screen

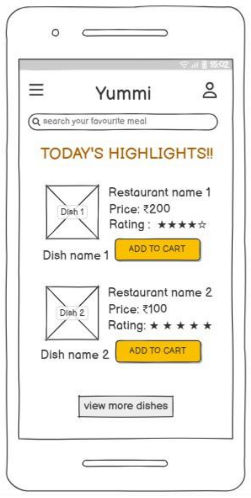
- **Delivery Address Form**: Users enter their delivery address.
- **Payment Options**: Choose between different payment methods.
- **Place Order Button**: Finalize the order.

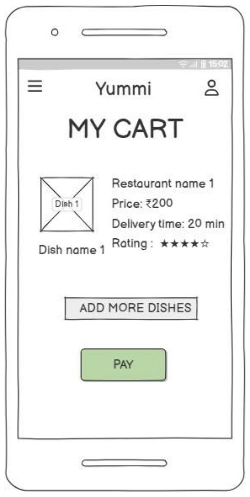
Order Confirmation Screen

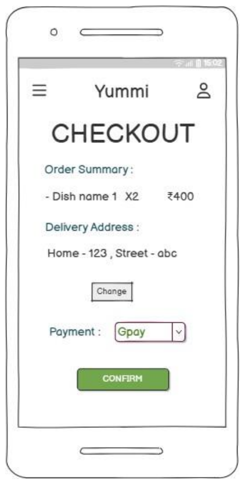
- **Order Summary**: Shows the order details.
- **Estimated Delivery Time**: Provides an estimated delivery time.

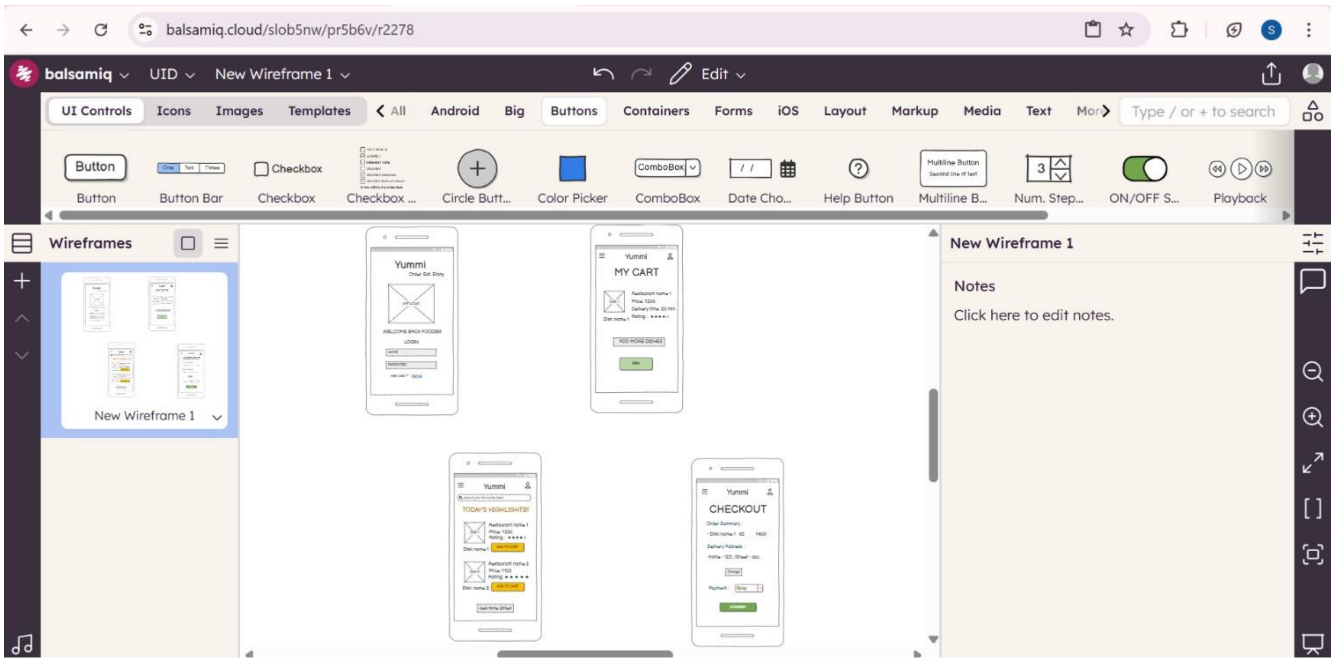
OUTPUT :











RESULT :

Hence we have created storyboards representing the user flow for a mobile app, such as a food delivery app, using Balsamiq.

Ex. No.: 8b **Date : 08.04.2025**

Create storyboards to represent the user flow for a mobile app (e.g., food delivery app) using OpenBoard

AIM:

To map out the user flow for a mobile app (e.g., a food delivery app), storyboards will be designed using OpenBoard.

PROCEDURE:

Tool Link: https://openboard.ch/download.en.html

Step 1: Define the User Flow

1. **Identify Key Screens**:
   - List the main screens your app will have (e.g., Home, Menu, Cart, Checkout, Order Confirmation).
2. **Map the User Journey**:
   - Understand the typical user journey through these screens (e.g., browsing menu, l2adding items to cart, checking out).

Step 2: Create Storyboards Using OpenBoard

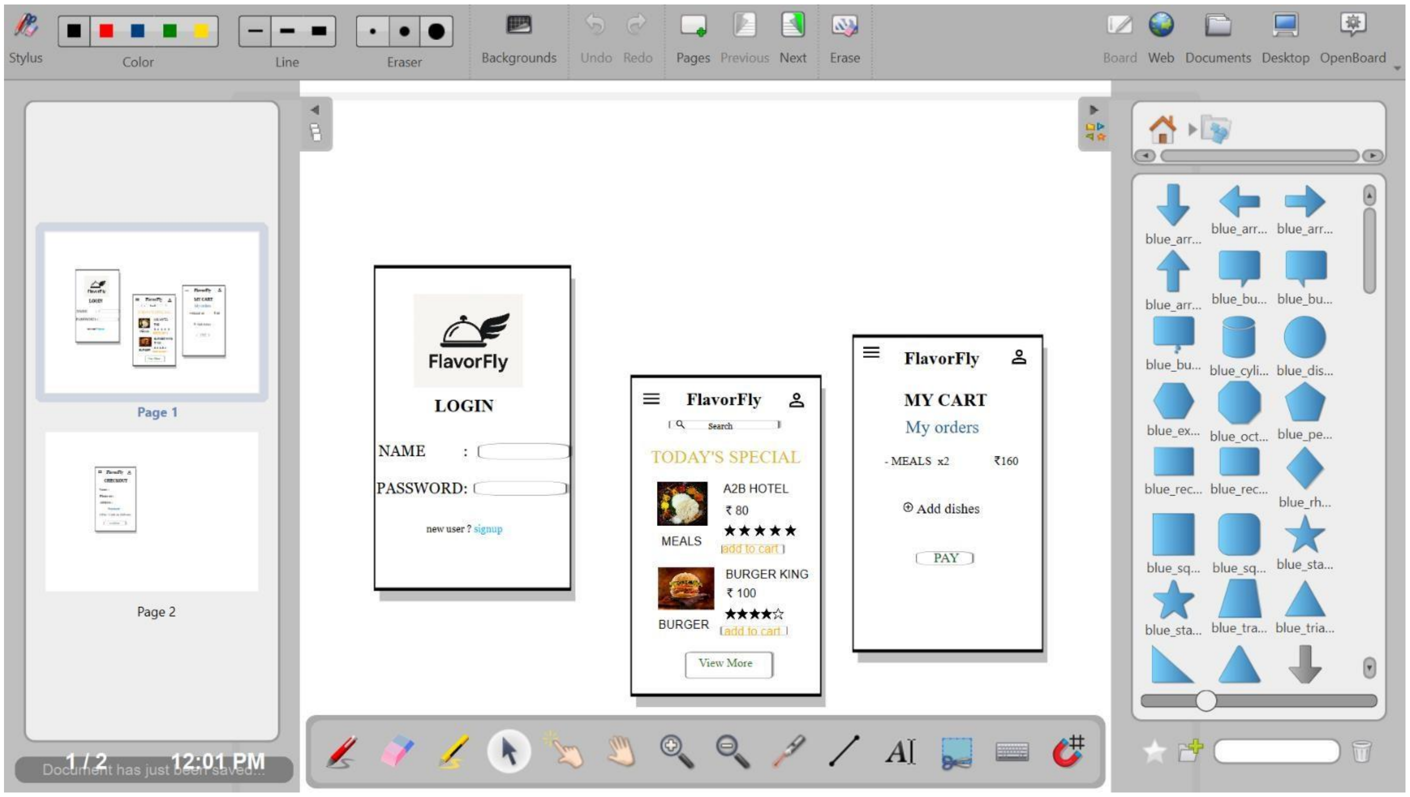
1. **Install OpenBoard**:
   - Download and install OpenBoard from the official website.
2. **Create a New Document**:
   - Open OpenBoard and create a new document.
3. **Add Frames for Each Screen**:
   - Use the drawing tools to create frames representing each key screen of your app.

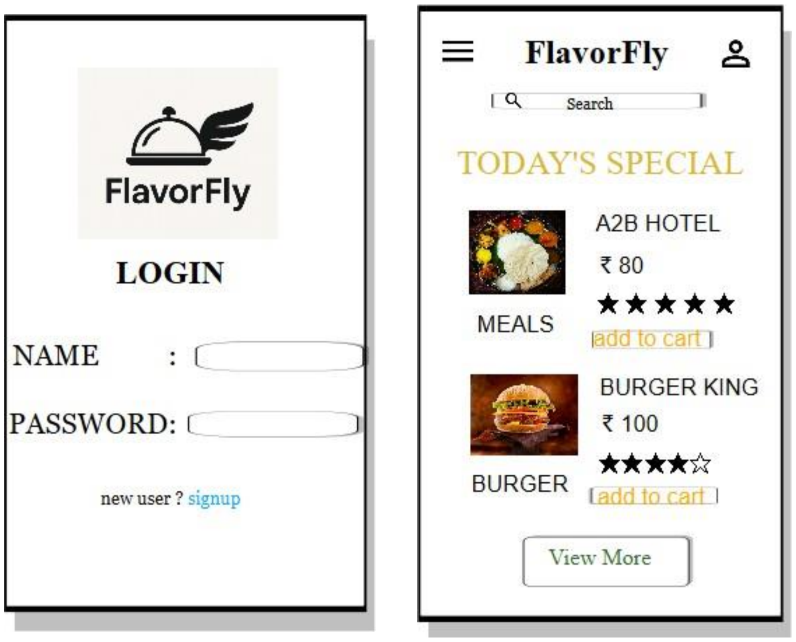
4. **Sketch Each Screen**:
   ○ Use the pen or shape tools to draw basic elements for each screen.
   ○ Focus on major UI components like buttons, text fields, and icons.
5. **Organize the Flow**:
   ○ Arrange the frames in a sequence that represents the user journey.
   ○ Use arrows or lines to show navigation paths between screens.

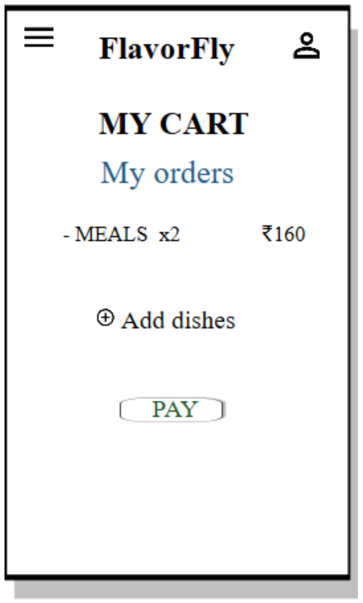
Example Screens for Food Delivery App

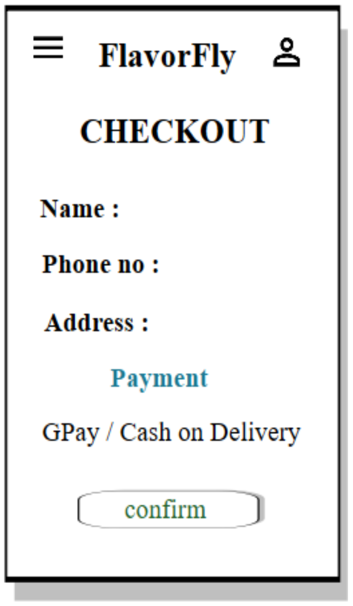
1. **Home Screen**:
   ○ Search bar for finding restaurants
   ○ Categories for different cuisines
2. **Menu Screen**:
   ○ List of food items with images, names, and prices
   ○ Add to Cart buttons
3. **Cart Screen**:
   ○ Items added to the cart with quantity and total price
   ○ Checkout button
4. **Checkout Screen**:
   ○ Delivery address form
   ○ Payment options ○
   Place Order button
5. **Order Confirmation Screen**:
   ○ Order summary
   ○ Estimated delivery time

OUTPUT:


Stylus
Color
Line
Eraser
Backgrounds
Undo
Redo
Pages
Previous
Next
Erase
Board
Web
Documents
Desktop
OpenBoard
Page 1
Page 2
1 / 2
12:01 PM
Document has just been saved...
FlavorFly
LOGIN
NAME :
PASSWORD:
new user ? signup
FlavorFly
Search
TODAY'S SPECIAL
A2B HOTEL
₹ 80
MEALS
add to cart
BURGER KING
₹ 100
BURGER
add to cart
View More
FlavorFly
MY CART
My orders
- MEALS x2 ₹160
Add dishes
PAY


FlavorFly
LOGIN
NAME :
PASSWORD:
new user ? signup
FlavorFly
Search
TODAY'S SPECIAL
A2B HOTEL
₹ 80
★★★★★
MEALS
add to cart
BURGER KING
₹ 100
★★★★☆
BURGER
add to cart
View More





RESULT:

The user flow for a mobile app (e.g., a food delivery app), storyboards has beeen designed using OpenBoard.

Ex. No. : 9 **Date : 15.04.2025**

Design input forms that validate data (e.g., email, phone number) and display error messages using HTML/CSS, JavaScript (with Validator.js)

AIM:

The aim is to design input forms that validate data, such as email and phone number, and display error messages using HTML/CSS and JavaScript with Validator.js.

PROCEDURE:

Step 1: Setting Up the HTML Form

Start by creating an HTML form with input fields for the email and phone number.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Form Validation</title>
   <link rel="stylesheet" href="style.css">
</head>
```

```
<body>
    <div class="container">
        <form id="myForm">
            <label for="email">Email:</label>
            <input type="email" id="email" name="email" required>
            <span id="emailError" class="error"></span>

            <label for="phone">Phone Number:</label>
            <input type="text" id="phone" name="phone" required>
            <span id="phoneError" class="error"></span>

            <button type="submit">Submit</button>
        </form>
    </div>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/validator/13.6.0/validator.min.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Step 2: Styling the Form with CSS

Next, add some basic styling to make the form look nice.

css

```
/* style.css */
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.container {
    background-color: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-bottom: 5px;
}

input {
    margin-bottom: 10px;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 3px;
}
```

```
button {
    padding: 10px;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 3px;
    cursor: pointer;
}

button:hover {
    background-color: #218838;
}

.error {
    color: red;
    font-size: 0.875em;
}
```

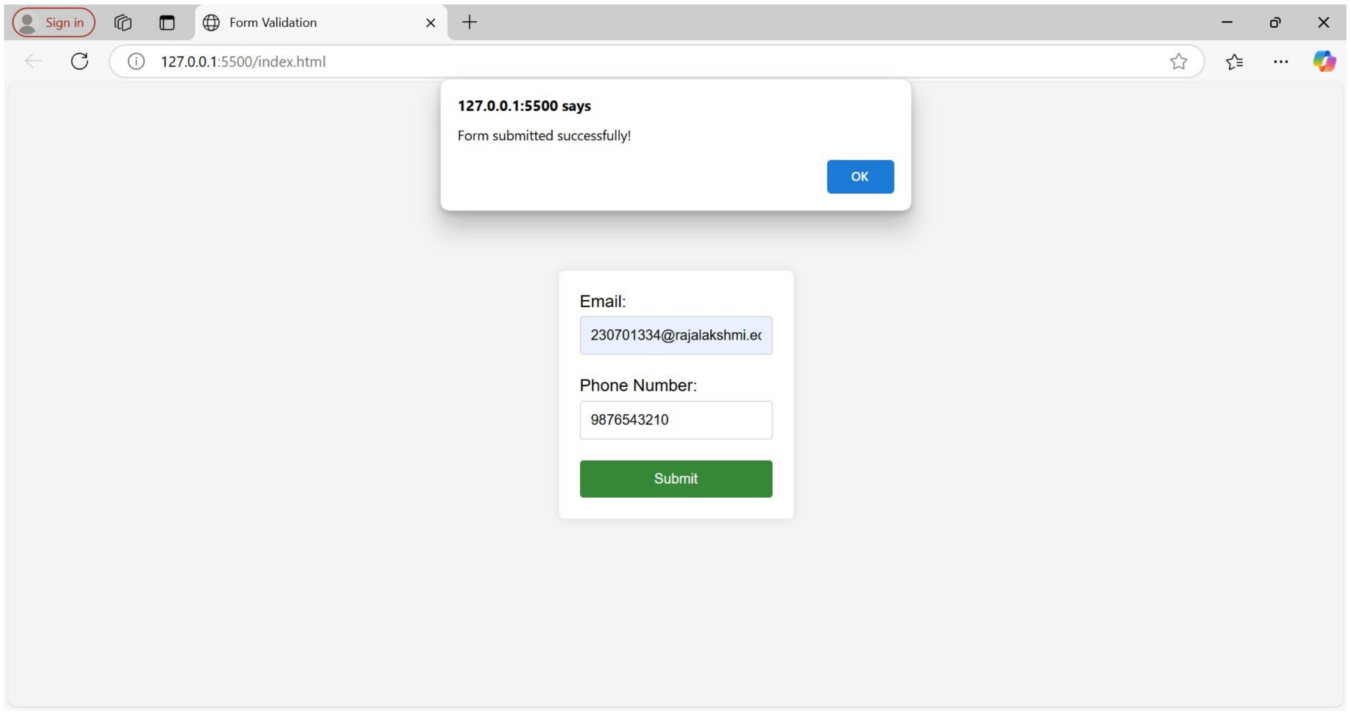
□ Step 3: Adding JavaScript for Validation

Finally, add JavaScript to validate the input fields using Validator.js and display error messages.

javascript

```
□ /* script.js */
document.getElementById('myForm').addEventListener('submit', function (e) {
    e.preventDefault();

    let email = document.getElementById('email').value;
    let phone = document.getElementById('phone').value;

    let emailError = document.getElementById('emailError');
    let phoneError = document.getElementById('phoneError');

    // Clear previous error messages
    emailError.textContent = '';
    phoneError.textContent = '';

    // Validate email
    if (!validator.isEmail(email)) {
        emailError.textContent = 'Please enter a valid email address.';
    }

    // Validate phone number
    if (!validator.isMobilePhone(phone, 'any')) {
        phoneError.textContent = 'Please enter a valid phone number.';
    }

    // If no errors, submit the form (for demonstration purposes, we'll just log the values)
    if (validator.isEmail(email) && validator.isMobilePhone(phone, 'any')) {
        console.log('Email:', email);
        console.log('Phone:', phone);
    }
});
□
```

OUTPUT:

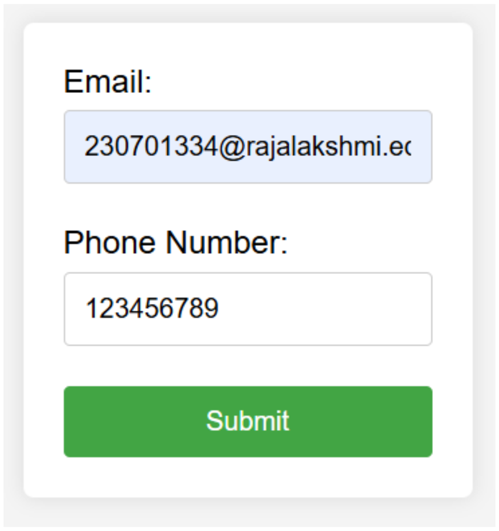


127.0.0.1:5500 says

Form submitted successfully!

Email: 230701334@rajalakshmi.ec

Phone Number: 9876543210



Email: 230701334@rajalakshmi.ec

Phone Number: 123456789

RESULT:

Hence we designed the input forms that validate data, such as email and phone number, and display error messages using HTML/CSS and JavaScript with Validator.js

Ex. No. :10 **Date : 15.04.2025**

Create a data visualization (e.g., pie charts, bar graphs) for an inventory management system using javascript

AIM:

The aim is to create data visualizations, such as pie charts and bar graphs, for an inventory management system using JavaScript.

PROCEDURE:

Step 1: Set Up Your HTML File

First, create an HTML file to hold your canvas for the chart and include Chart.js.

```
html
□<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Inventory Management Visualization</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         text-align: center;
         margin: 50px;
```

```
        }
        canvas {
            margin: 20px auto;
        }
    </style>
</head>
<body>
    <h1>Inventory Management System</h1>
    <canvas id="pieChart" width="400" height="400"></canvas>
    <canvas id="barChart" width="400" height="400"></canvas>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

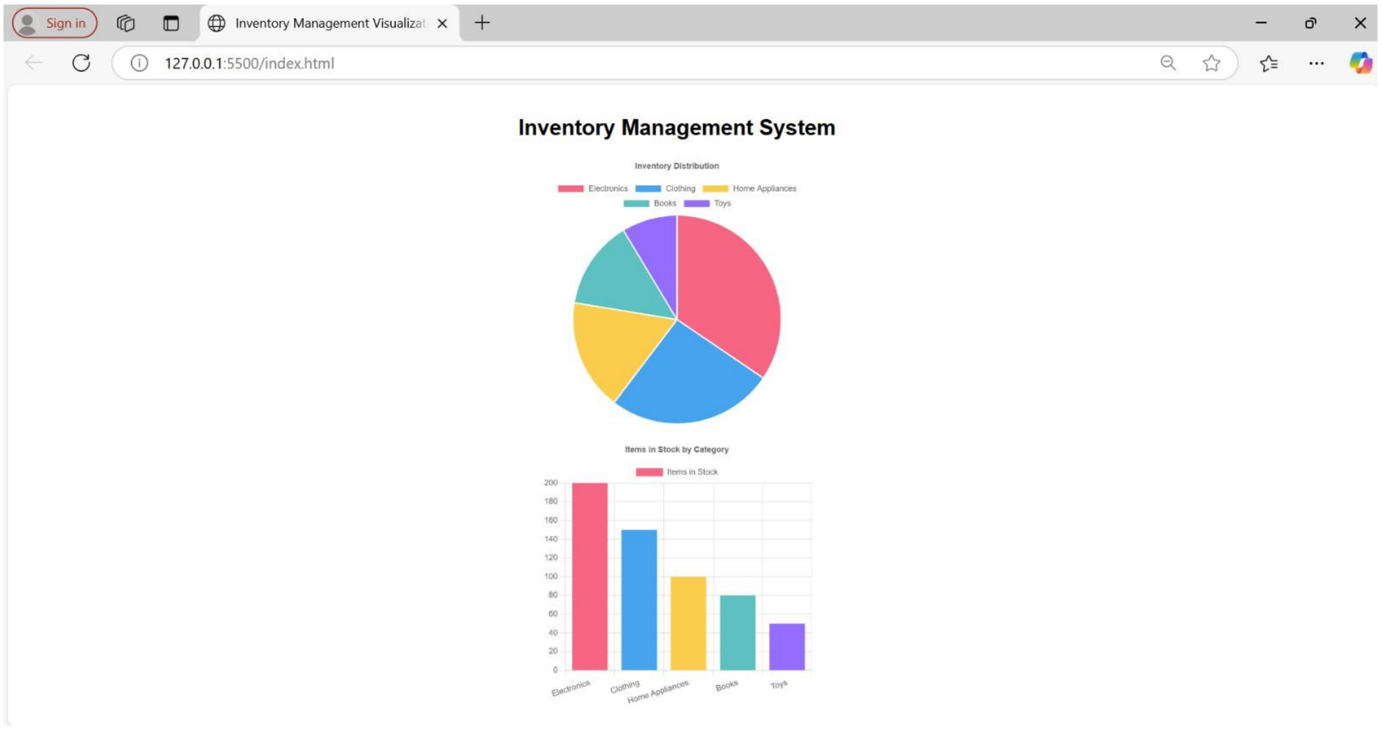
□ Step 2: Create the JavaScript File for Charts

Next, create a JavaScript file (script.js) to handle the data visualization logic.

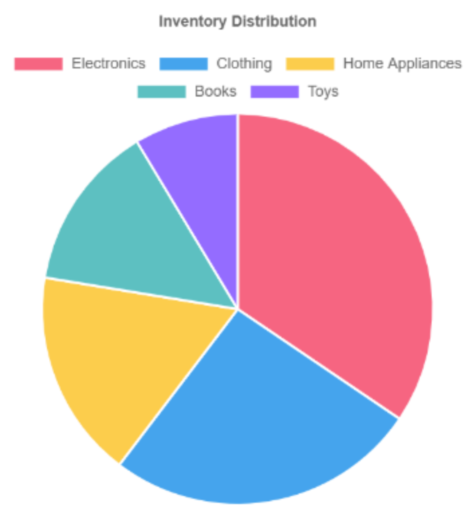
javascript

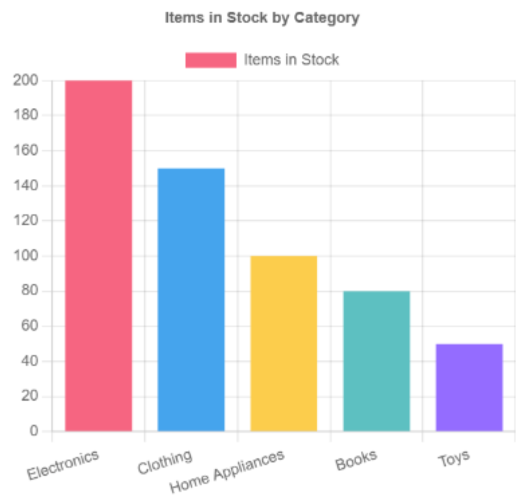
```
□// script.js


// Data for the inventory
const inventoryData = {
    labels: ['Electronics', 'Clothing', 'Home Appliances', 'Books', 'Toys'],
    datasets: [
        {
            label: 'Items in Stock',
            data: [200, 150, 100, 80, 50],
            backgroundColor: [
                '#FF6384',
                '#36A2EB',
                '#FFCE56',
                '#4BC0C0',
                '#9966FF'
            ],
        }
    ]
};

// Creating the Pie Chart
const ctxPie = document.getElementById('pieChart').getContext('2d');
const pieChart = new Chart(ctxPie, {
    type: 'pie',
    data: inventoryData,
    options: {
        responsive: true,
        title: {
            display: true,
            text: 'Inventory Distribution'
        }
    }
});

// Creating the Bar Chart
const ctxBar = document.getElementById('barChart').getContext('2d');
const barChart = new Chart(ctxBar, {
    type: 'bar',
    data: inventoryData,
    options: {
        responsive: true,
        title: {
            display: true,
            text: 'Items in Stock by Category'
        },
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
    }
});
□
```

OUTPUT:



Inventory Management System





RESULT:

Hence we have created data visualizations, such as pie charts and bar graphs, for an inventory management system using JavaScript.