RAJALAKSHMI ENGINEERING COLLEGE
AN AUTONOMOUS INSTITUTION
Affiliated to ANNA UNIVERSITY
Rajalakshmi Nagar, Thandalam,
Chennai-602105

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

CS23A34 - USER INTERFACE DESIGN LABORATORY

ACADEMIC YEAR:2024-2025 (EVEN)

INDEX

Reg. No : **Name :**

Branch : **Year/Section :**

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 (Tkinter for 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

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

How to Start Design in FIGMA Tool:

Here's a step-by-step example of how to **use Figma** to create a simple **mobile app login screen**, including basic design and prototyping.

Step 1: Sign Up and Create a New Project

1. Go to figma.com and create an account (if you haven't already).
2. Once logged in, click **"New File"** to start a blank project.
3. You'll see a blank canvas where you can start designing.

Step 2: Create the Frame (Artboard)

1. On the left toolbar, select the **"Frame" tool** (shortcut: F).
2. Choose a mobile preset (e.g., **iPhone 13**) from the right-hand panel.
3. A mobile-sized frame will appear on the canvas, which will act as your app screen.

Step 3: Design the Login Screen

Add a Background Color:

1. Select the frame and go to the right-side panel.
2. Under **"Fill,"** choose a background color (e.g., light blue #E3F2FD).

Insert a Logo:

1. Click the **"Rectangle" tool** (shortcut: R) and draw a placeholder for a logo.
2. Use the **"Text" tool** (shortcut: T) to add your app name, e.g., **"MyApp"**.
3. Adjust font size and color from the right-hand panel.

Add Input Fields:

1. Use the **"Rectangle" tool** to draw two boxes for username and password fields.
2. Add placeholder text inside (e.g., "Enter your email").
3. Apply rounded corners under **"Corner Radius"** in the right panel.

Add a Login Button:

1. Create a button using the **"Rectangle" tool** and set the color to blue (#1E88E5).
2. Use the **"Text" tool** to add the text "Login" inside the button.
3. Group the button and text together by selecting them and pressing Ctrl + G (Windows) or Cmd + G (Mac).

Align Elements:

- Use the alignment tools in the top menu (center everything vertically and horizontally).
- Adjust spacing between elements using the **"Auto Layout"** feature (Shift + A).

Step 4: Prototyping the Interaction

1. Click the **"Prototype" tab** on the right panel.
2. Select the "Login" button and drag the blue dot to a new frame (e.g., a home screen).
3. Set the interaction to **"On Click" → "Navigate to"** the next screen.
4. Choose an animation effect (e.g., "Smart Animate").

Step 5: Preview the Design

1. Click the **"Play" button** in the top-right corner to preview your app prototype.
2. Try clicking on the login button to see the transition to the next screen.

Step 6: Share Your Design

1. Click the **"Share"** button in the top-right corner.
2. You can invite team members via email or generate a shareable link.
3. Adjust permissions (View, Edit, or Comment only).

Step 7: Export Assets

1. Select the elements you want to export (e.g., the logo or button).
2. In the right-hand panel, click **"Export"** and choose a format (PNG, JPG, SVG).
3. Click "Export" to download assets for developers.

Example 2:

Example: Design a Mobile Login Screen in Figma

Step 1: Create a New Figma Project

1. **Sign Up or Log In:** Go to Figma.com and sign in or sign up if you don't already have an account.
2. **Create a New File:** On the Figma homepage, click **"New File"** to start a new project.

Step 2: Set Up the Artboard (Frame)

1. **Select Frame Tool:**
   On the left sidebar, select the **Frame Tool (F)**.
2. **Choose a Mobile Size:**
   In the right panel, you'll see different device options. Choose **iPhone 13** (or any mobile size that fits your needs).
3. **Canvas:**
   This creates a blank mobile screen on your canvas (artboard) where you'll design your login screen.

Step 3: Add Background Color

1. **Select the Frame:**
   Click on the frame you just created (iPhone 13).
2. **Change the Fill Color:**
   On the right panel, under the **Fill** section, click the color box and choose a light background color (for example, #F0F4F8).

Step 4: Add UI Elements

Let's add the key components of a login screen: **Logo, Text Fields, and Button**.

1. Add the Logo:

1. **Create a Logo Placeholder:**
    - Select the **Rectangle Tool (R)** and draw a small square at the top of the frame.
    - You can also use the **Text Tool (T)** to write "MyApp" or upload a logo image if you have one.
2. **Adjust Logo Size and Position:**
    - Place the logo at the top center of the frame.
    - Adjust the size of the logo to make it look balanced on the screen.

---

2. Add the Username Field:

1. **Draw the Rectangle:**
    - Select the **Rectangle Tool (R)** and draw a rounded rectangle below the logo.
2. **Change the Color:**
    - In the right panel, change the fill color to something like #ffffff (white) and adjust the opacity slightly to make it look like an input field.
3. **Add Placeholder Text:**
    - Select the **Text Tool (T)** and add text such as “Enter your email” inside the rectangle.
    - Adjust the font size to something readable (e.g., 16px).

---

3. Add the Password Field:

1. **Duplicate the Username Field:**
    - Select the username field and press Ctrl + D (Windows) or Cmd + D (Mac) to duplicate it below.
2. **Change Placeholder Text:**
    - Update the text to “Enter your password.”

4. Add the Login Button:

1. **Draw the Button:**
    - Select the **Rectangle Tool (R)** and draw a rectangle just below the password field.
2. **Style the Button:**
    - Change the color of the button to #1E88E5 (blue) to make it stand out.
    - Use the **Text Tool (T)** to write “Login” inside the button. Align it centrally.
3. **Adjust Corner Radius:**
    - In the right panel, under **Corner Radius**, adjust the corners to round them for a softer button look (e.g., 20px).

Step 5: Align Elements

1. **Use Auto Layout:**
    - Select all elements on the screen (logo, text fields, and button).
    - Right-click and choose **"Auto Layout"**. This will help align and space the elements evenly.
2. **Adjust Spacing:**
    - Use the **Alignment Tools** in the top bar to align everything horizontally and vertically.

Step 6: Add Interaction (Prototyping)

1. **Go to Prototype Tab:**
   At the top-right, click on the **Prototype** tab.
2. **Link Login Button to Next Screen:**
    - Select the login button and drag the blue circle (right) to a new frame (you can duplicate the frame to create a "home" screen or success screen).
    - Set the interaction to **"On Click" → "Navigate to"** the next screen.
    - Add a transition animation like **Smart Animate** for smooth movement.

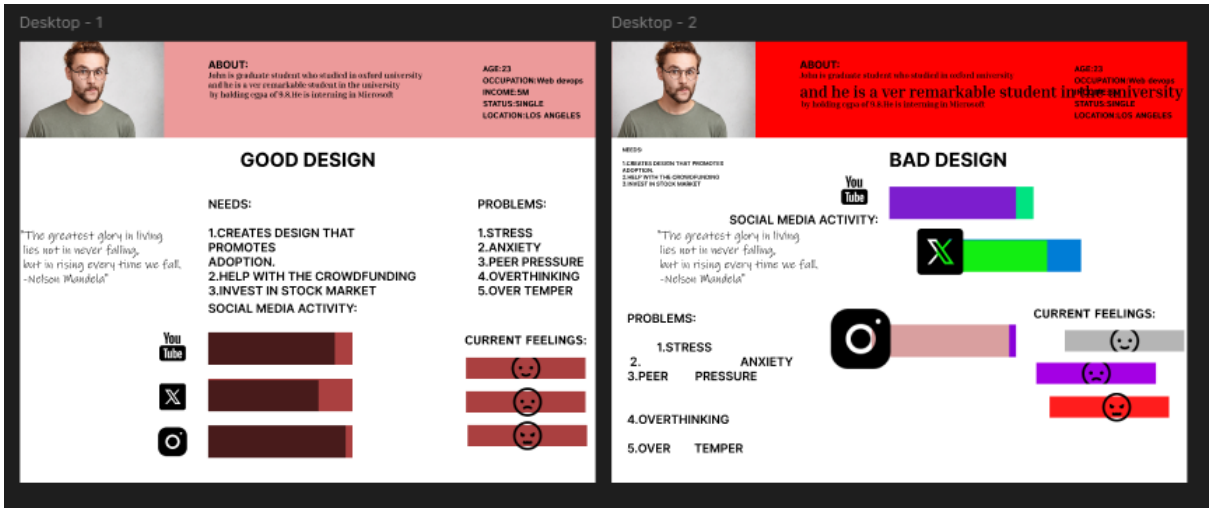
Step 7: Preview Your Design

1. **Preview the Prototype:**
    - Click the **Play Button** in the top-right corner to preview how the design looks.
    - Test the login button to make sure it links to the next screen.

Step 8: Share and Export

1. **Share the Design:**
    - Click **"Share"** in the top-right corner. You can invite team members to view or edit the design by email or share a link.
2. **Export Assets:**
    - Select elements like the logo or button, then click **"Export"** in the bottom-right corner to download them as PNGs, SVGs, or JPGs.

OUTPUT:



RESULT:

Hence we can concule the diffrence betweena Good design and bad design

EX No 2:

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

Figma – How to Setting Up the UI

A. Home Screen (It contains Instruction Page)

Step 1: Create a Frame:

- In Figma, create a new frame (File → New Frame). Set the size to **1024x768px** for a standard desktop view.
- This will be your **Home Screen** where users start the task.

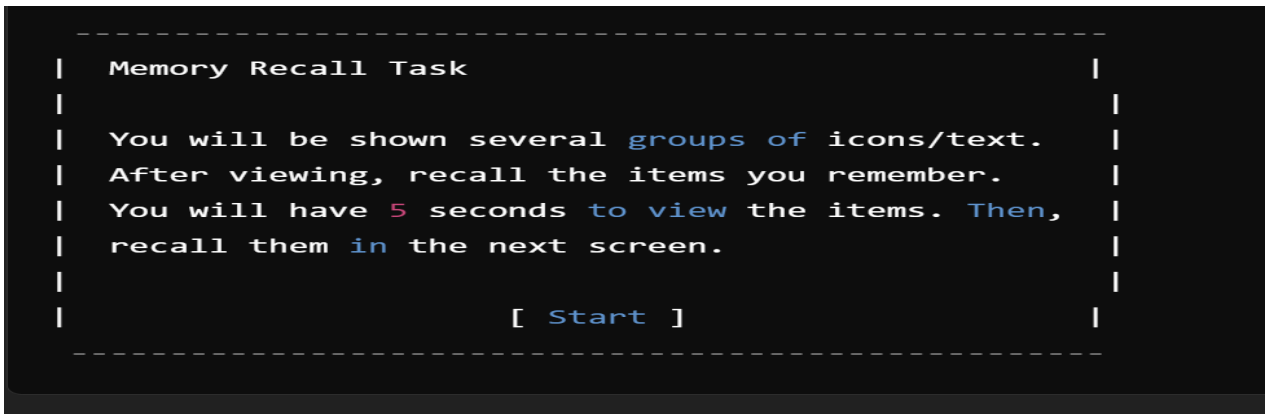
Step 2: Add Instructions:

- Use the **Text Tool (T)** to add a heading like "Memory Recall Task."
- Add a smaller body of text with instructions such as:
  - "You will be shown several groups of icons or text. After viewing, recall the items you remember."
- Use the **Text Tool (T)** to add more detailed instructions like "You will have 5 seconds to view the items. Then, recall them in the next screen."

Step 3: Start Button:

- Create a button at the bottom of the screen. To do this:
  - Draw a **Rectangle** (R) for the button.
  - Use the **Text Tool (T)** to add “Start.”
  - Style the button (color, border radius) to make it stand out.
  - Use **Figma’s Prototyping Tools** (top bar → Prototype) to link this button to the next screen (Chunking Phase).
  - You can also use **interactive components** like hover effects for more realism.

Sample Output of (Step A) Looks like this (Need not to be the same)



B. Chunking Phase (It Display Chunked Items)

Step 1: Create a New Frame:

- Create a new frame for the **Chunking Phase** (the second screen). This frame will display the icons or text.

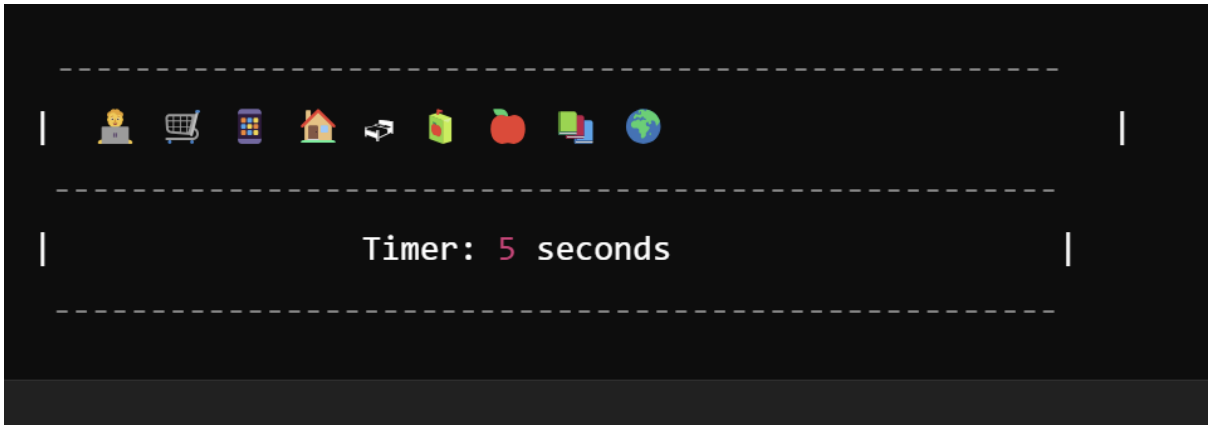
Step 2: Design Chunked Items:

- Use **icons** or **text blocks** that users will have to recall. If you're using text, it could be short phrases or words. If you're using icons, you can either import them from **Figma's resources** or draw simple shapes using Figma's drawing tools.
- **For Chunking with Borders**:
  - Group 3-5 icons or text together in a box (use the **Rectangle Tool (R)**) to visually represent a chunk. You might want to create 3-4 groups.
  - Space these chunks out with some empty space in between them to ensure users can identify each chunk.
- **For Chunking without Borders**:
  - Place the elements next to each other without clear separation. This can be done by not using boxes and just visually mixing the items.

Step 3: Set the Viewing Time:

- **Time Simulation**: Figma does not have true timers, but you can simulate a fixed time by setting the next screen transition after 5 seconds:
  - Select the entire **Frame (Chunking Phase)**.
  - Under the **Prototype** tab, link this frame to the next screen (Recall Phase).
  - Set the interaction to "After Delay" and enter 5000ms (5 seconds).

Sample Output of (Step B) Looks like this (Need not to be the same)



C. Recall Phase

Step 1: Create a New Frame for Recall:

- This is where the user will recall the items they saw in the previous chunking phase.

Step 2: Recall Input (Multiple-choice or Text Input):

- **Option 1: Multiple-Choice**:
  - Create multiple options for the user to select (e.g., 4-5 icons or text options).
  - Use **Checkboxes** or **Radio buttons** to allow users to select what they remember.
  - Add a question at the top: "Select the items you remember seeing."
- **Option 2: Text Input**:
  - Create **Text Input Fields** where users can type what they remember. Create 3-5 input fields depending on how many chunks you showed.
  - This can be done by selecting the **Text Tool (T)**, adding a label ("Item 1", "Item 2"), and setting up input boxes.

Step 3: Submit Button:

- Create a **Submit** button at the bottom using the **Rectangle Tool (R)** and adding text like "Submit Recall."
- Add an interaction to move to the **Feedback Screen** after submission.

Sample Output of the Visual Appearance of the Recall Phase (Step C) in Figma (Text Input Fields):

Sample Output of the Visual Appearance of the Recall Phase (Step C) in Figma (Multiple Choice):

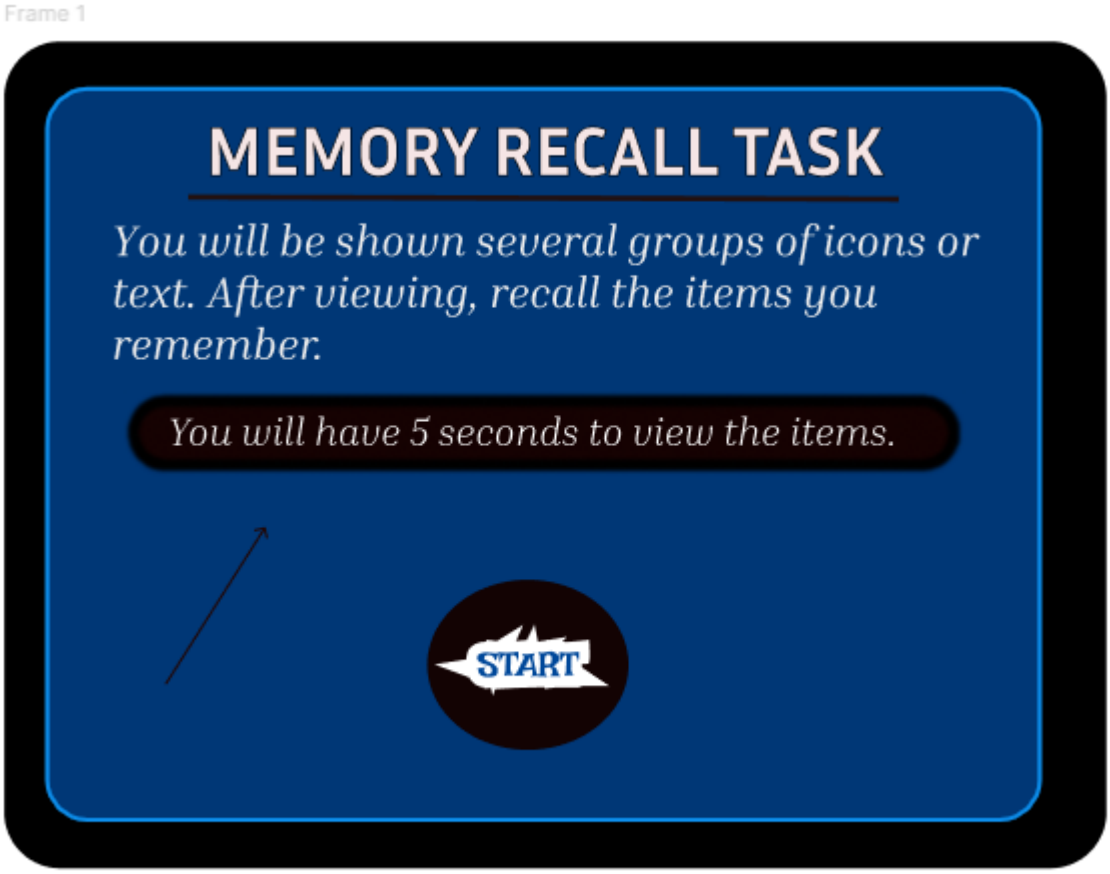
D. Result Screen

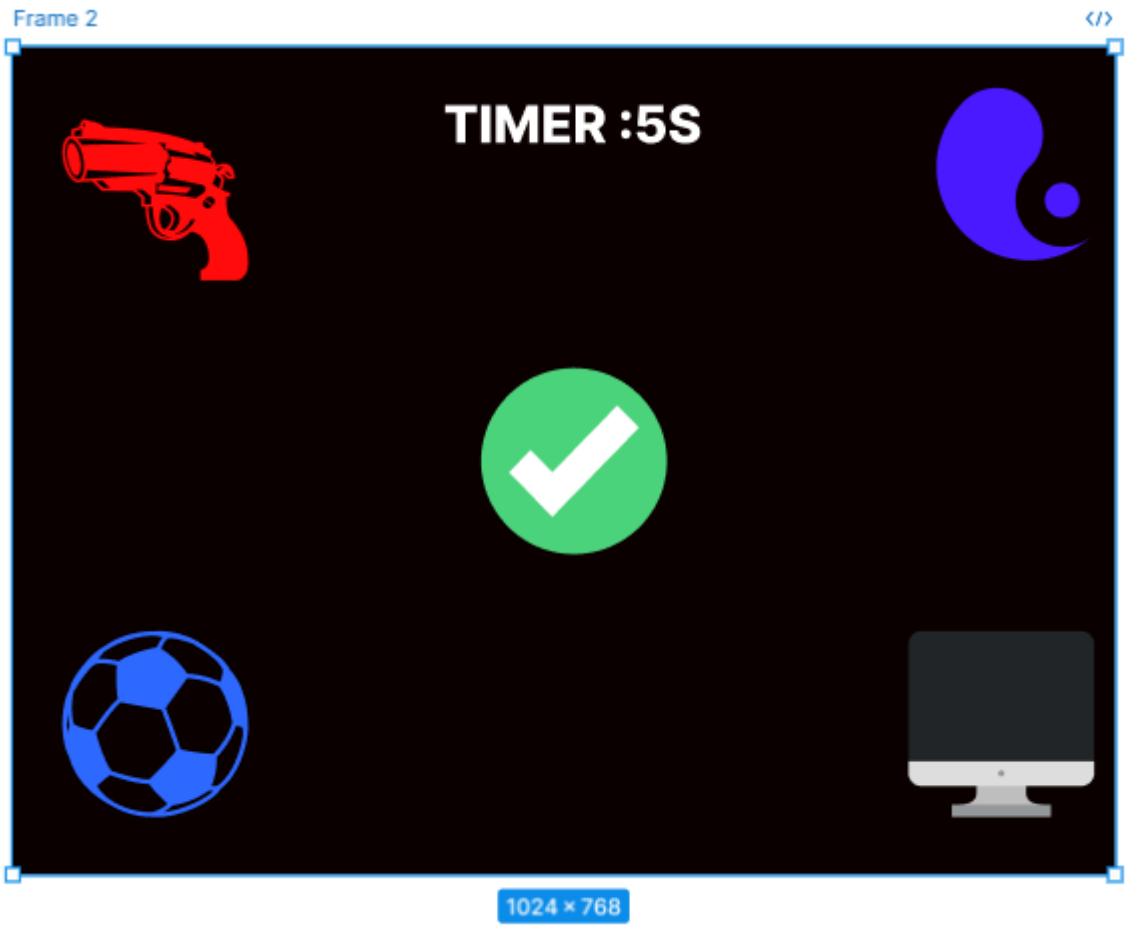
1. **Create a Feedback Screen**:
    - After the user submits their recall, provide feedback.
    - Add text like: “You recalled 4/5 items correctly!” or “Good job, you remembered 3 out of 5 items.”
2. **Analyze**:
    - For your experiment, you can vary the **chunk size** (3 vs. 5 items per chunk) and the **chunk type** (icons vs. text) across different test sessions to evaluate their impact on recall.

Sample Output of Visual Appearance of the Result Screen in Figma:

```
 ------------------------------------------------
|  You recalled 4/5 items correctly!             |
|                                                |
|                    [ Next ]                    |
 ------------------------------------------------
```

OUTPUT:

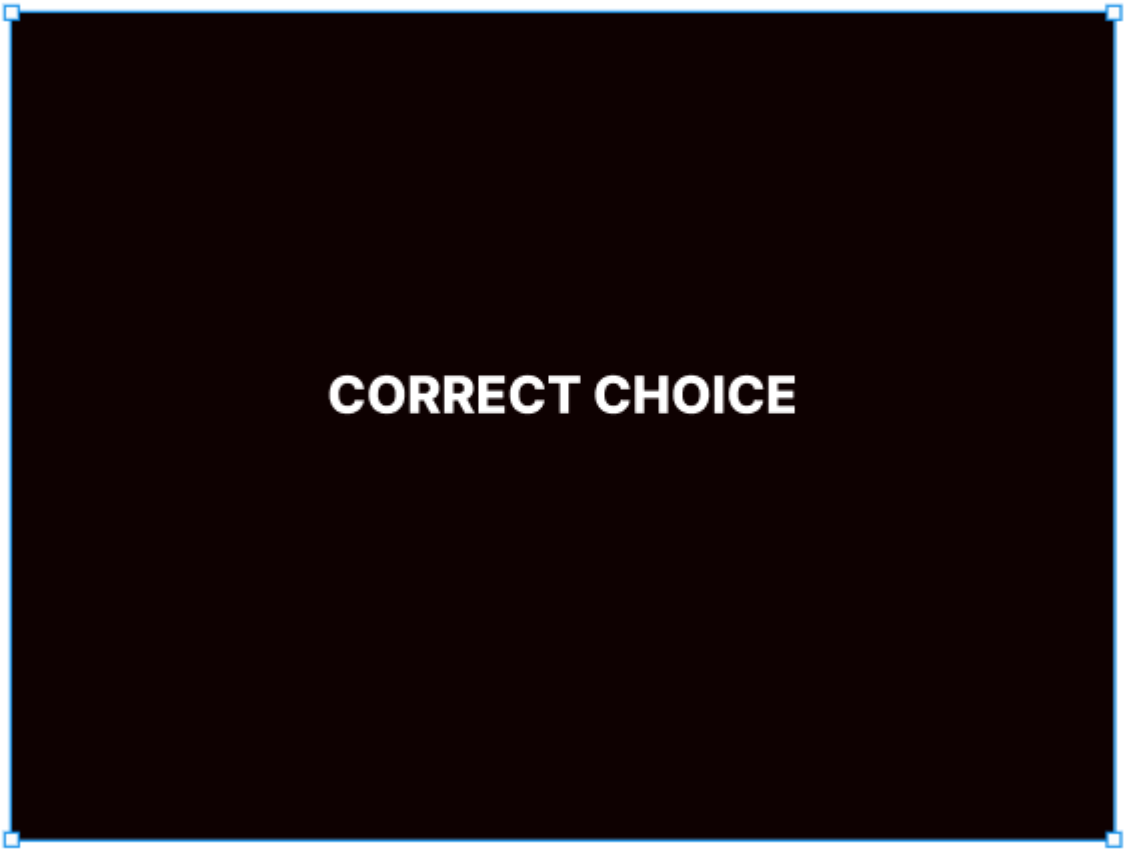



Frame 2
TIMER :5S
1024 × 768

Frame 3

1024 × 768

Frame 4



1024 × 768

RESULT:

Hence we implemented the effect of chunking on user memory.

Excercise 3 **Date:**

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:

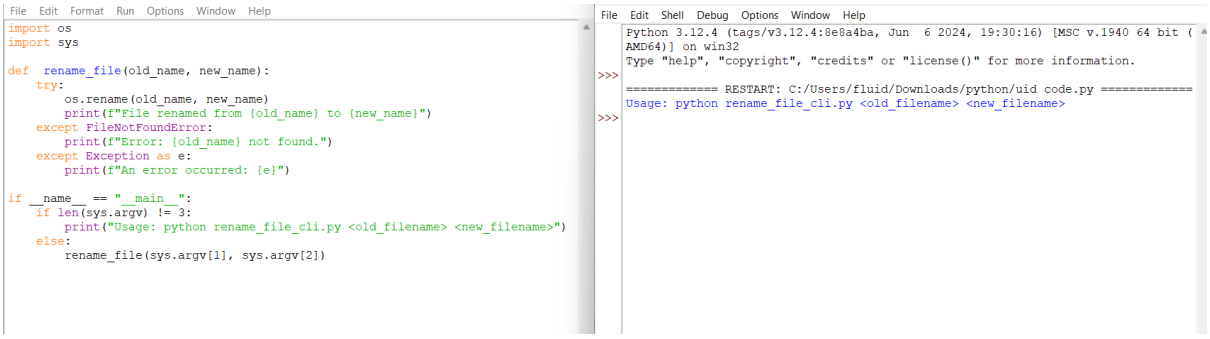
i) CLI (Command Line Interface)

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:

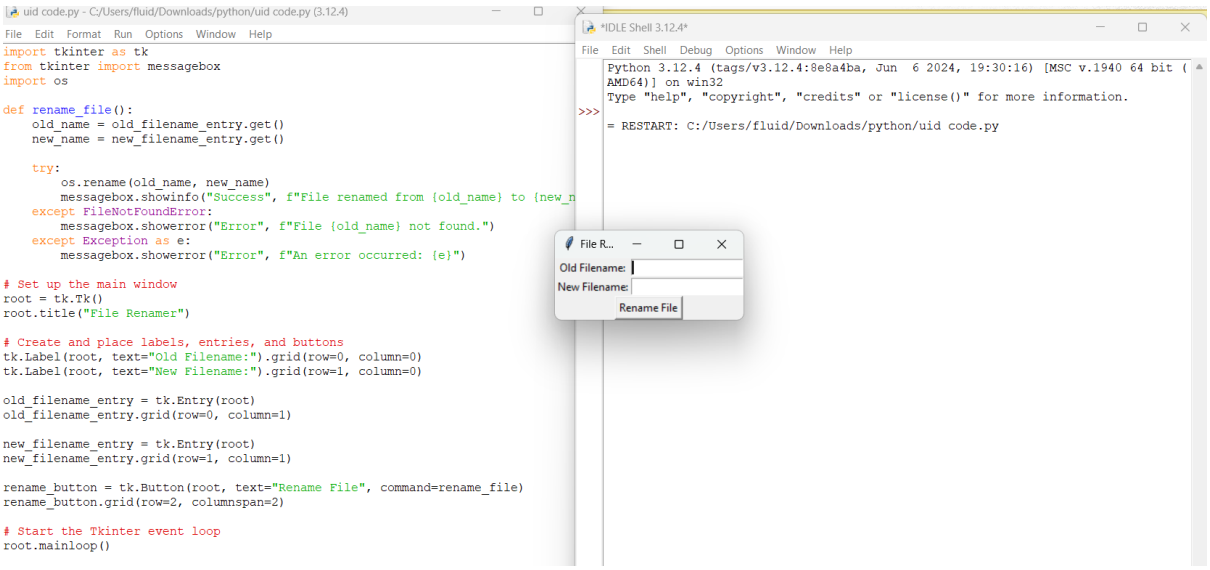


ii) GUI (Graphical User Interface)

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:



iii) VUI (Voice User Interface)

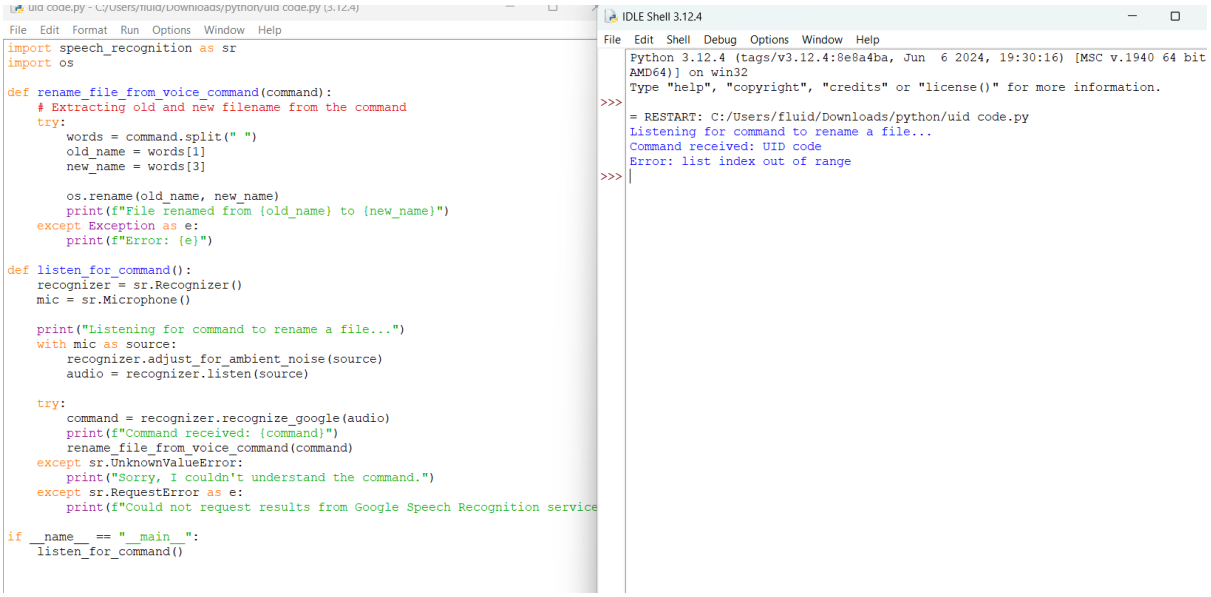
speech_recognition library for voice input and the pyttsx3 library for text-to-speech output. Make sure you have these libraries installed (pip install SpeechRecognition pyttsx3).

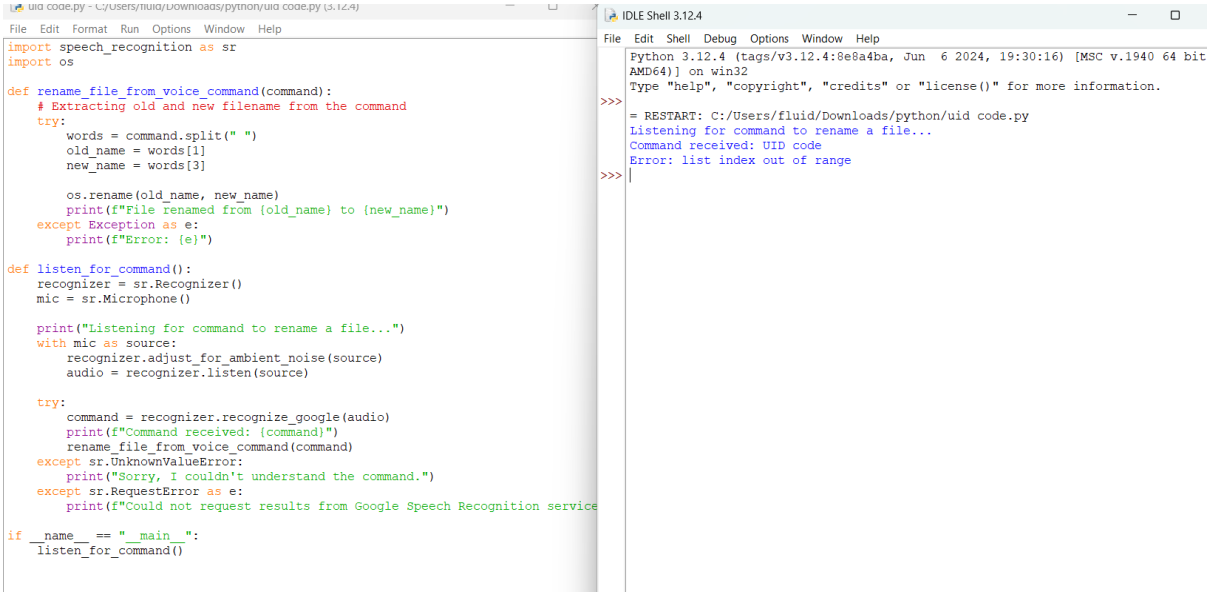
```
import speech_recognition as sr
import pyttsx3

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:





```
File  Edit  Format  Run  Options  Window  Help
import speech_recognition as sr
import os

def rename_file_from_voice_command(command):
    # Extracting old and new filename from the command
    try:
        words = command.split(" ")
        old_name = words[1]
        new_name = words[3]

        os.rename(old_name, new_name)
        print(f"File renamed from {old_name} to {new_name}")
    except Exception as e:
        print(f"Error: {e}")

def listen_for_command():
    recognizer = sr.Recognizer()
    mic = sr.Microphone()

    print("Listening for command to rename a file...")
    with mic as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        command = recognizer.recognize_google(audio)
        print(f"Command received: {command}")
        rename_file_from_voice_command(command)
    except sr.UnknownValueError:
        print("Sorry, I couldn't understand the command.")
    except sr.RequestError as e:
        print(f"Could not request results from Google Speech Recognition service

if __name__ == "__main__":
    listen_for_command()
```

```
IDLE Shell 3.12.4
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/fluid/Downloads/python/uid code.py
Listening for command to rename a file...
Command received: UID code
Error: list index out of range
>>>
```

RESULT:

Hence we implemented and developed 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.

Excercise 3b **Date:**

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:

Tool link: https://wireflow.co/

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

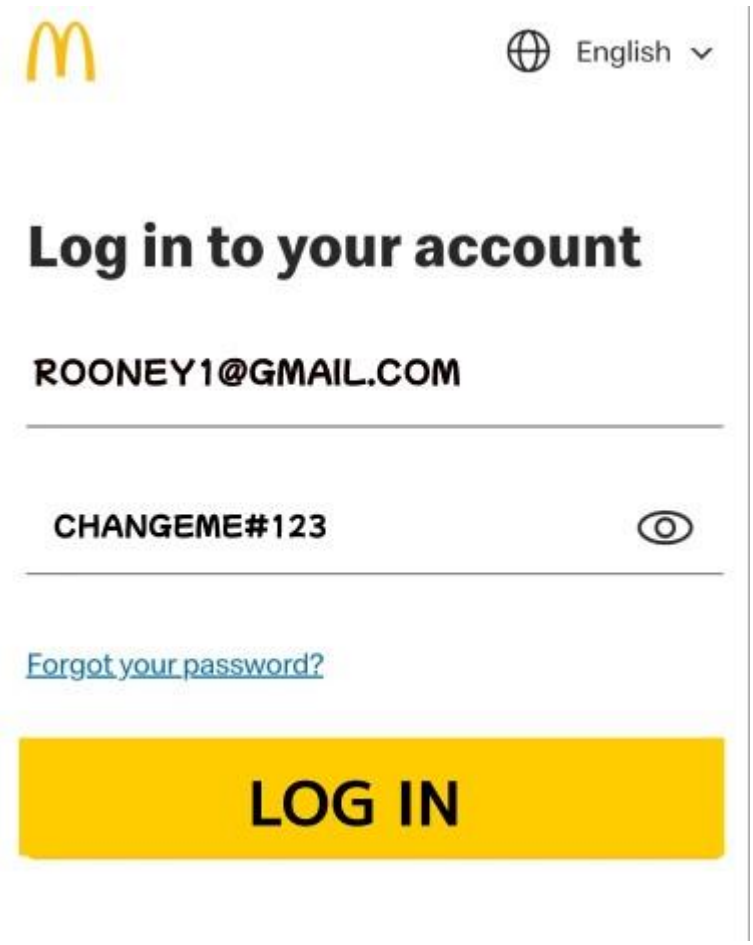
1. Share the Prototype:
    - Invite users to interact with your prototype via a shareable link from Wireflow.
2. Test Sessions:
    - Ask users to complete tasks using both types of navigation. Observe their interactions and collect feedback.
3. Collect Feedback:
    - Utilize Wireflow's feedback features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

FAMILIAR DISPLAY:



This is a **familiar** MC DONALDS login screen. It asks the user to enter their **EMAIL** to log in. The screen features the **MC DONALDS logo**, a **"log in" button** in brown. This is likely used for ordering food or accessing a customer account

UNFAMILIAR DISPLAY:

The burger image represents an **unfamiliar interactive display** where touching any part of it **navigates to another page** instead of behaving like a regular static image. This approach is similar to **unfamiliar navigation methods**, such as **Infinite Scrolling without visible indicators**,where traditional user expectations are challenged. It is likely part of a **digital ad, clickable banner, or food app UI**, designed to engage users and seamlessly direct them to a **menu, ordering page, or promotional offer** without conventional navigation elements.

OFFER PAGE:

This image is a **MC DONALDS offer page**, showcasing limited-time deals on popular menu items. It features three promotions: a **MC SAVER FOR ₹89** at **70% off**, the **BUY 1 GET 1 FREE ONLY ON NOV 11 2025**, and **BURGER MEAL (WITH FRIES) AT 99** with **50% off**. Each offer is highlighted with bold prices and **"ORDER NOW"** buttons, making it easy for customers to place an order. The fiery background and engaging visuals enhance the appeal of the deals.

RESULT:

It’s implemented that a prototype with both well-known and new navigation elements and measure user-friendliness across different user groups using Wireflow.

Excercise 4a **Date:**

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.

Tool Link: **https://www.lucidchart.com/pages/**

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

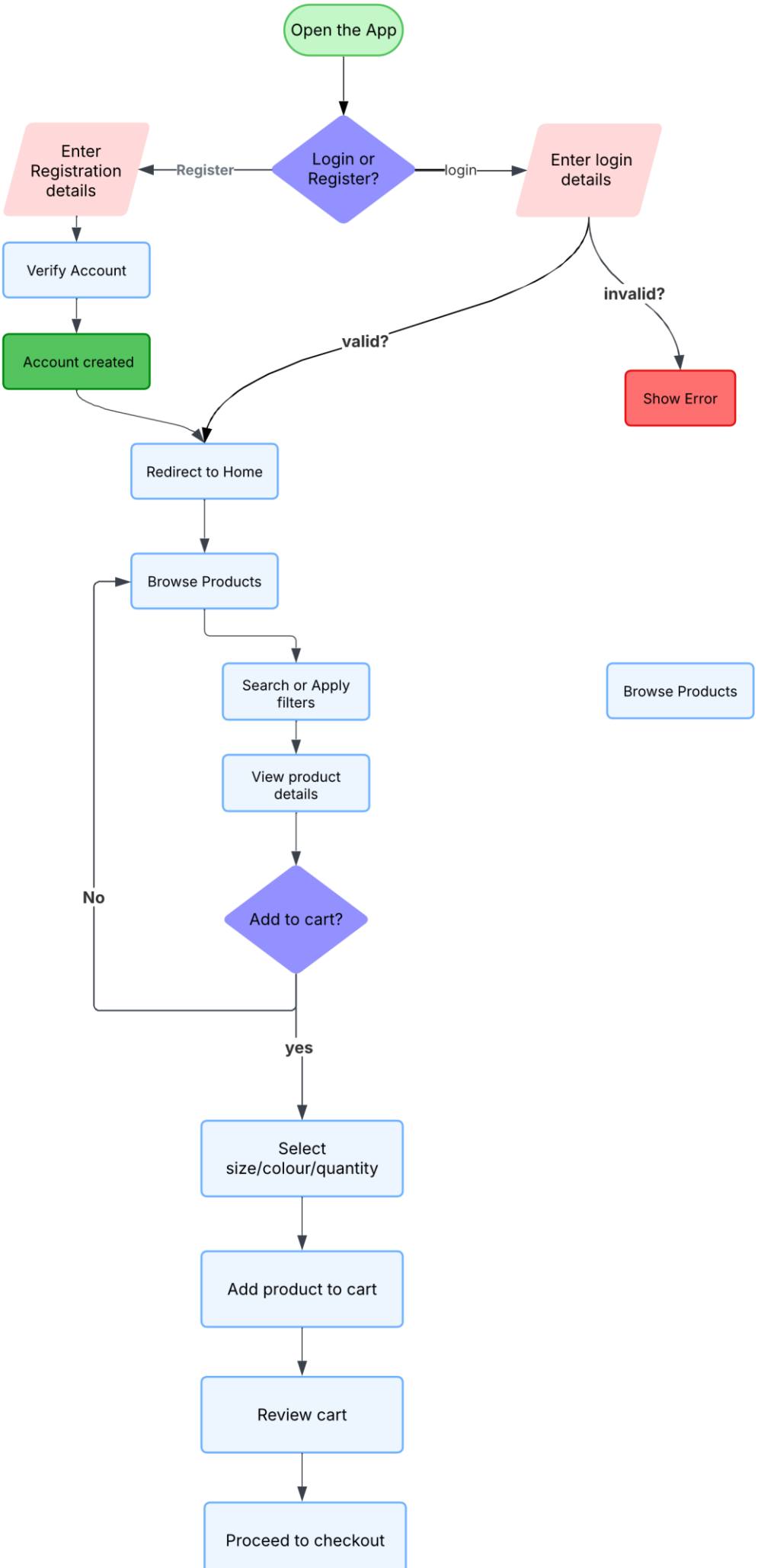
Checkout Flow

- 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:


Open the App
Login or Register?
Register
login
Enter Registration details
Enter login details
Verify Account
Account created
invalid?
valid?
Show Error
Redirect to Home
Browse Products
Search or Apply filters
View product details
Add to cart?
No
yes
Select size/colour/quantity
Add product to cart
Review cart
Proceed to checkout
Browse Products

RESULT:

Hence we implemented the steps a user takes to complete the main tasks within an online shopping app.

Excercise 4b **Date:**

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.

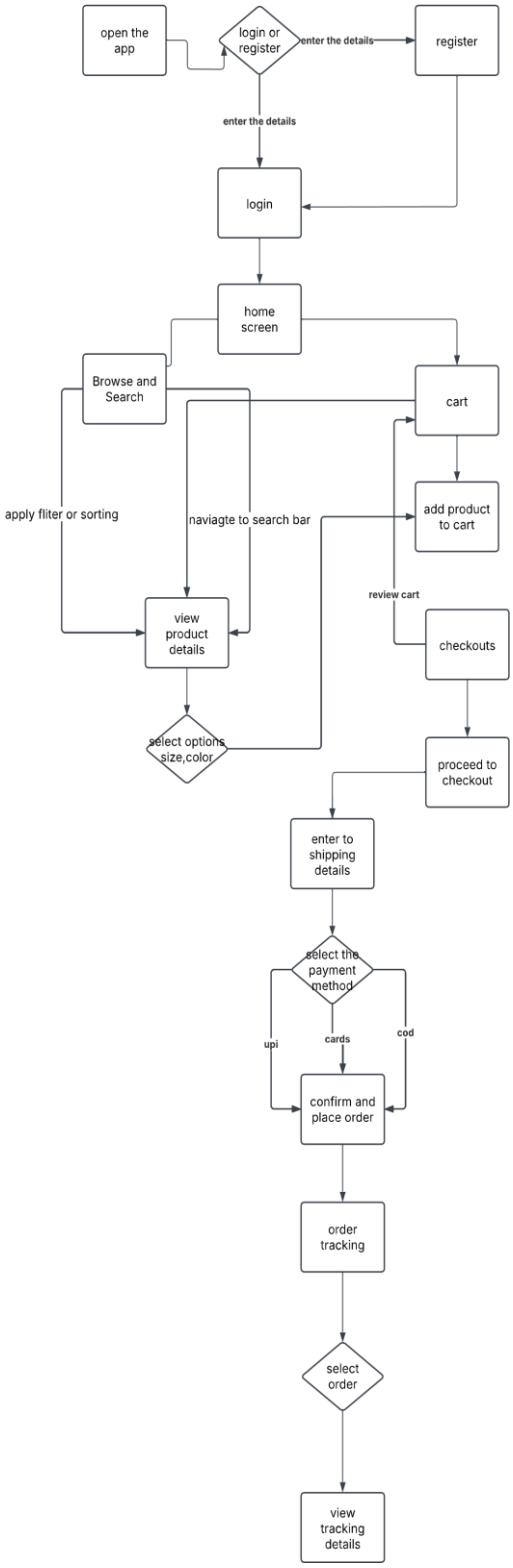
PROCEDURE:

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:

Hence we implemented task analysis for an app, such as online shopping, document user flows, and create corresponding wireframes using Dia.

Excercise 5a **Date:**

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:

Tool Link: https://www.axure.com/

Simulating the Lifecycle Stages for UI Design Using the RAD Model

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
        - Order History
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

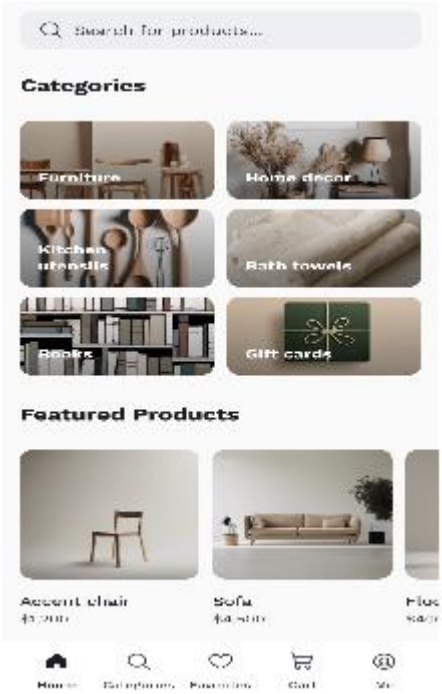
1. **Develop Interactive Prototypes:**
    - Convert wireframes into interactive prototypes by adding interactions and transitions.
    - Use dynamic panels to create interactive elements (e.g., carousels, pop-ups).
2. **Test and Iterate:**
    - Preview the prototype using the Preview button.
    -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

Hence implemented the lifecycle stages of UI design via the RAD model and develop a small interactive interface employing Axure RP.

Excercise 5b **Date:**

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.

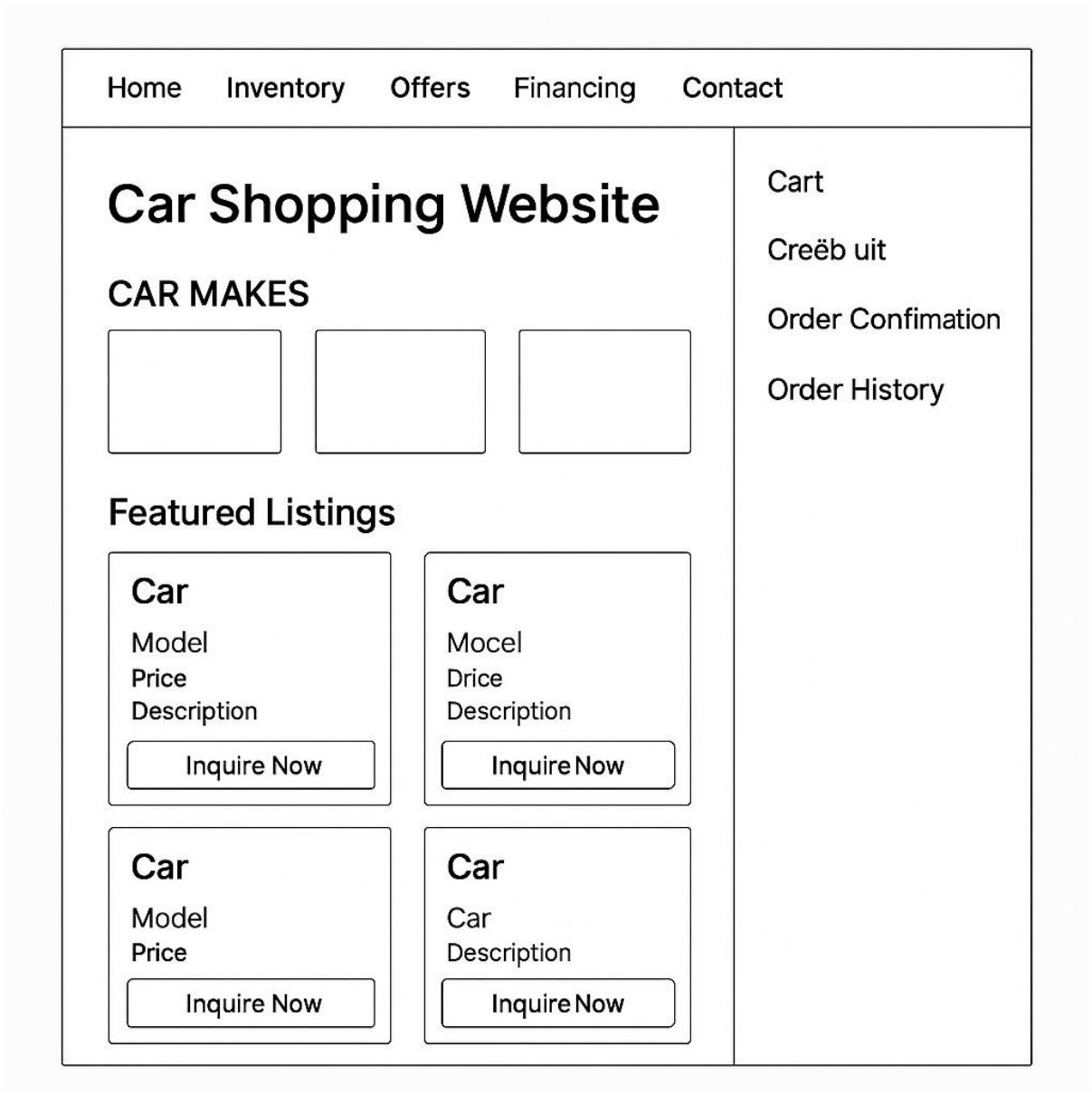
Output in OpenProj:

- 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:



RESULT:

The implementation of the lifecycle stages of UI design using the RAD model and design a small interactive interface with OpenProj has been done successfully

Excercise 6 **Date:**

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 -> 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:

RESULT:

Hence we implemented that using different app layouts and color schemes and evaluate user feedback on aesthetics and usability using GIMP.

Excercise 7a **Date:**

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.

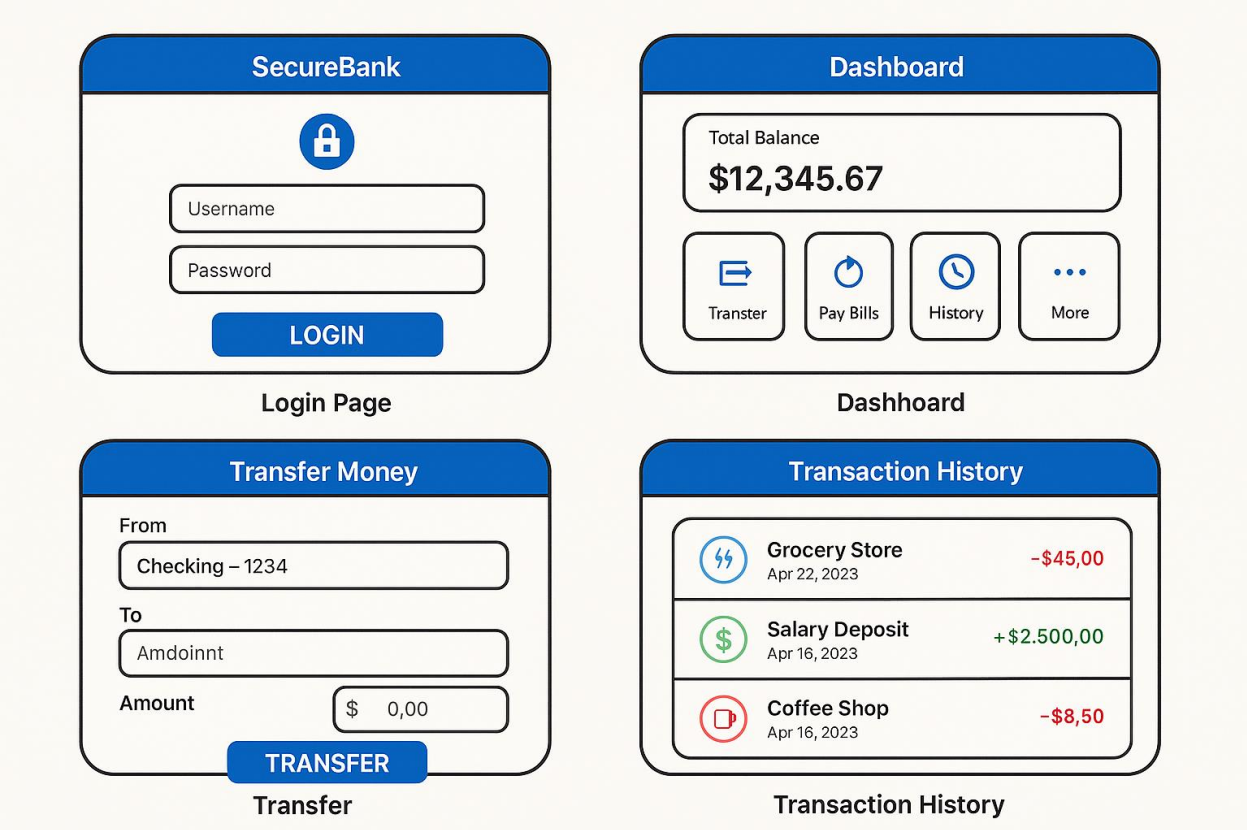
Step 2: Convert Paper Prototypes to Digital Wireframes Using Pencil Project

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

Excercise 7b **Date:**

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

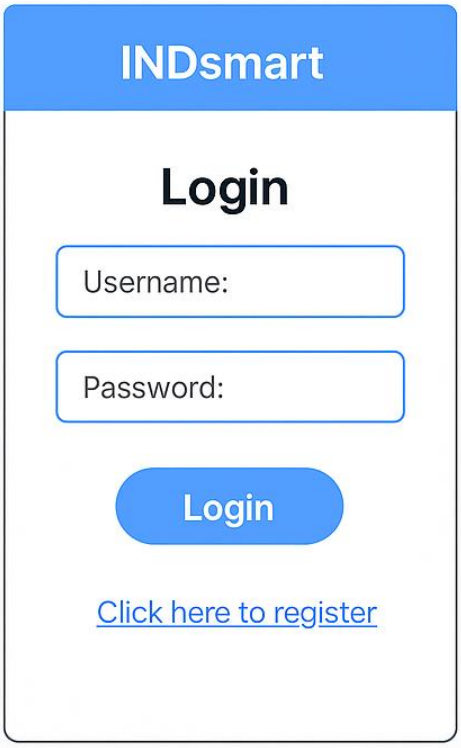
1. Identify Core Features:
    - Determine the essential features of the banking app (e.g., login, dashboard, account management, transfers).
2. Sketch Basic Layouts:
    - Use plain paper and pencils to sketch the main screens.
    - Focus on the primary elements like buttons, navigation menus, and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

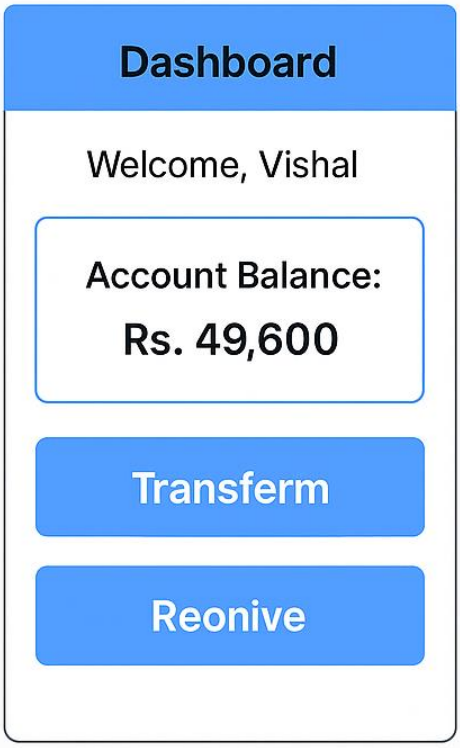
Step 2: Convert Paper Prototypes to Digital Wireframes Using Inkscape

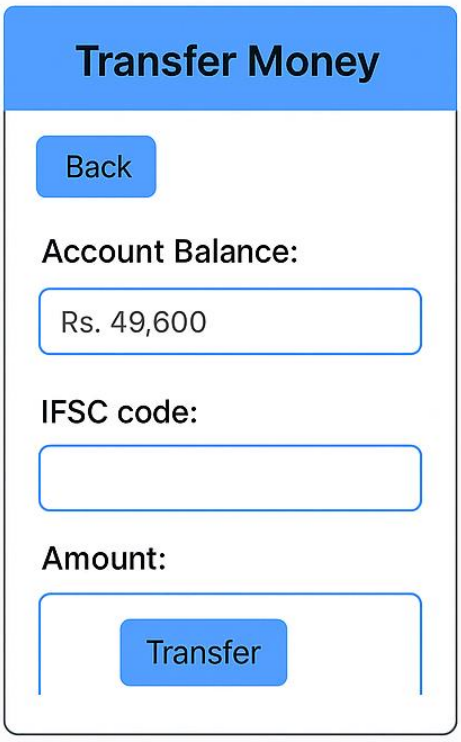
1. Install Inkscape:
    - Download and install Inkscape from the official website.
2. Create a New Document:

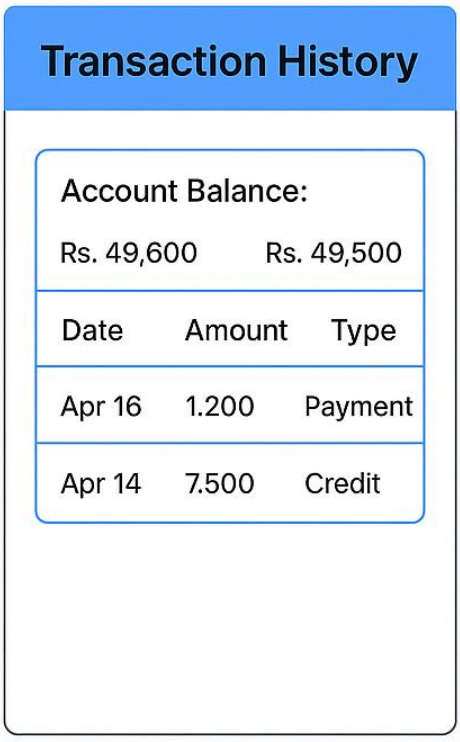
    - Open Inkscape and create a new document by clicking on File > New.
3. Set Up the Document:
    - Set the dimensions and grid for your design. Go to File > Document Properties to adjust the size.
    - Enable the grid by going to View > Page Grid.
4. Draw Basic Shapes:
    - Use the rectangle and ellipse 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:


INDsmart
Login
Username:
Password:
Login
Click here to register


Dashboard
Welcome, Vishal
Account Balance:
Rs. 49,600
Transferm
Reonive


Transfer Money
Back
Account Balance:
Rs. 49,600
IFSC code:
Amount:
Transfer


Transaction History
Account Balance:
Rs. 49,600
Rs. 49,500
Date
Amount
Type
Apr 16
1.200
Payment
Apr 14
7.500
Credit

RESULT:

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

Excercise 8a **Date:**

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 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.

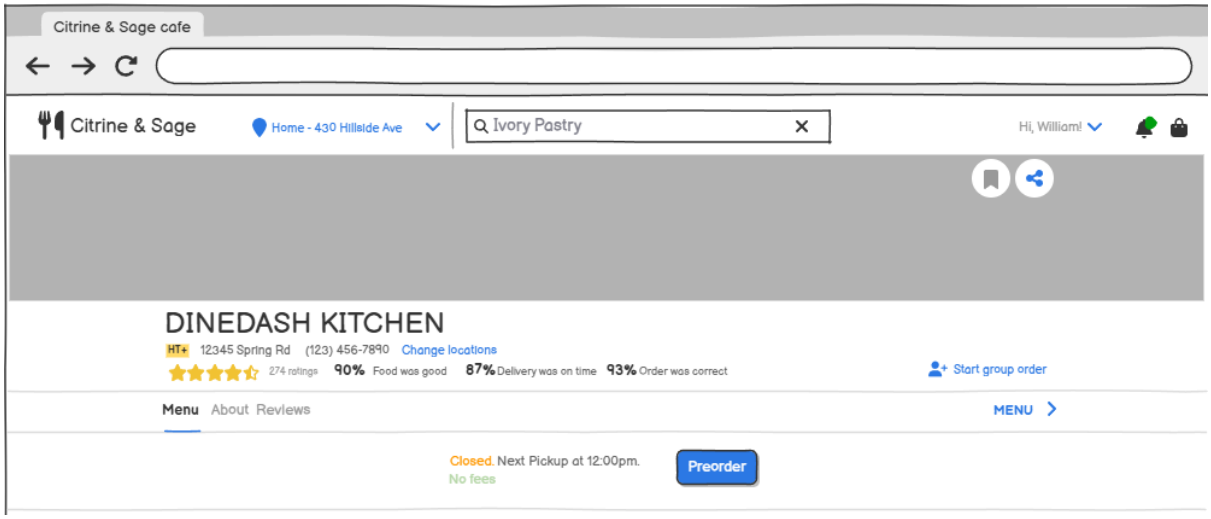
● Place Order Button: Finalize the order.

Order Confirmation Screen

● Order Summary: Shows the order details.

● Estimated Delivery Time: Provides an estimated delivery time.

OUTPUT:



Items Matching "desserts"

Tiramisu
Coffee-flavored layered dessert (Italy)
%16.50+

Crème Brûlée
Rich custard with caramelized sugar topping (France)
%16.50+

Gelato
Italian-style ice cream, denser and creamier
%16.50+

Eclairs
Choux pastry filled with cream and glazed (France)
%16.50+

Panna Cotta
Creamy, molded dessert, often with fruit sauce (Italy)
%16.50+

Strudel
Thin pastry with a fruit filling, often apple (Austria)
%16.50+

Show 2 more

Popular Items

Macarons – A French Delight!
Macarons are delicate, colorful, almond-based cookies that are filled with a flavorful filling like buttercream, ganache, or jam. These elegant treats have become synonymous with French luxury desserts.

To Share

Food Item
Brief description of what this food item is.
%16.50+

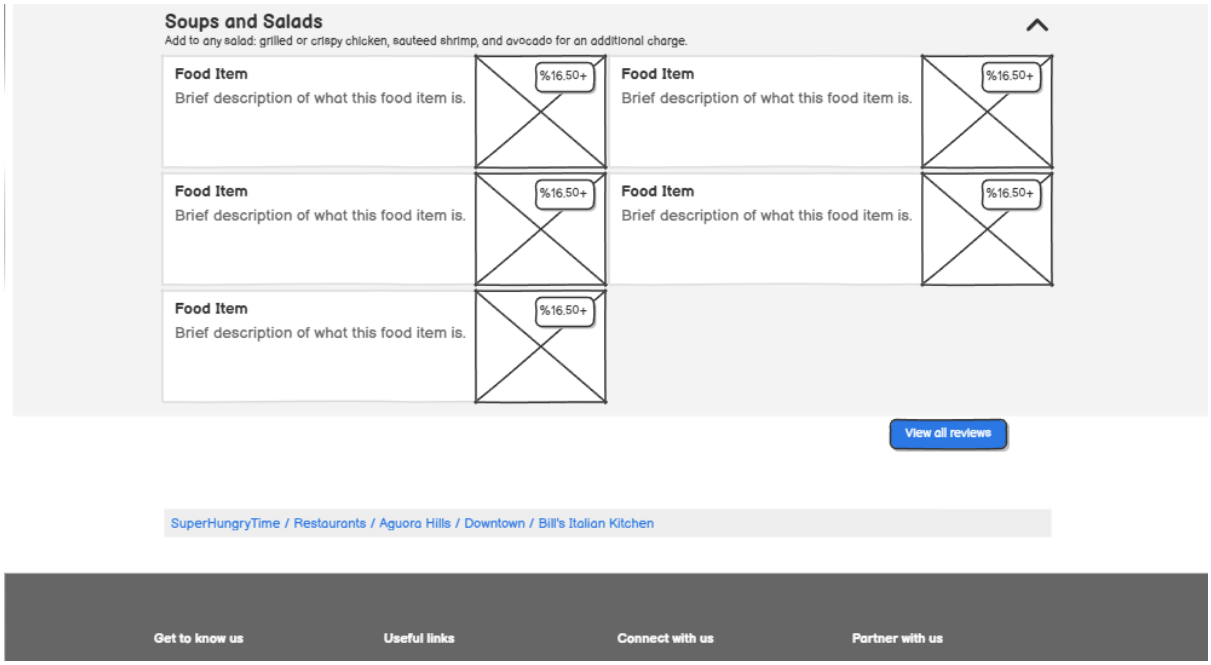
Food Item
Brief description of what this food item is.
%16.50+

Food Item
Brief description of what this food item is.
%16.50+

Food Item
Brief description of what this food item is.
%16.50+

Food Item
Brief description of what this food item is.
%16.50+

Food Item
Brief description of what this food item is.
%16.50+



RESULT:

Represented the user flow for a mobile app (e.g., food delivery app) using Balsamiq

Excercise 8b **Date:**

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, adding 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.

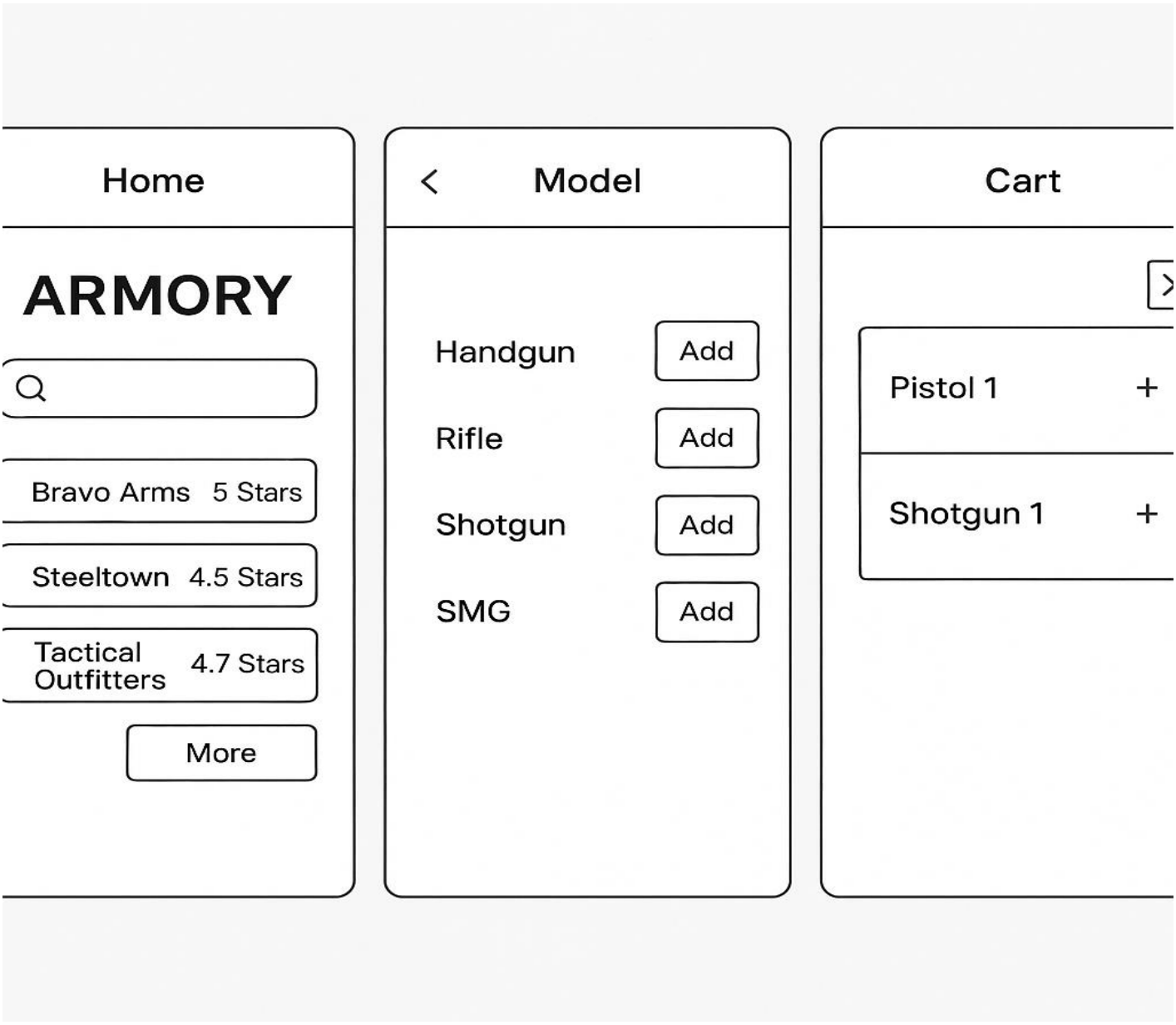
- Focus on major UI components like buttons, text fields, and icons.

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:



RESULT:

Hence we made a user flow for a gun delivery app and the storyboards has been designed using OpenBoard.

Excercise 9 **Date:**

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>Validated Form</title>
  <link rel="stylesheet" href="styles.css" />
</head>
<body>
  <div class="container">
    <h2>Sign Up</h2>
    <form id="myForm">
      <label for="email">Email:</label>
      <input type="text" id="email" name="email" />
      <span class="error" id="emailError"></span>
```

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

    <button type="submit">Submit</button>
   </form>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/validator@13.9.0/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:

```
body {
  font-family: Arial, sans-serif;
  background: #f0f2f5;
  display: flex;
  justify-content: center;
  padding-top: 50px;
}

.container {
  background: white;
```

```
  padding: 20px 30px;
  border-radius: 8px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  width: 300px;
}

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

label {
  margin-top: 10px;
}

input {
  padding: 8px;
  margin-top: 5px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.error {
  color: red;
  font-size: 0.9em;
  margin-top: 5px;
}
```

```
button {
  margin-top: 20px;
  padding: 10px;
  background-color: #4caf50;
  border: none;
  color: white;
  font-weight: bold;
  border-radius: 4px;
  cursor: pointer;
}

button:hover {
  background-color: #45a049;
}
```

Step 3: Adding JavaScript for Validation

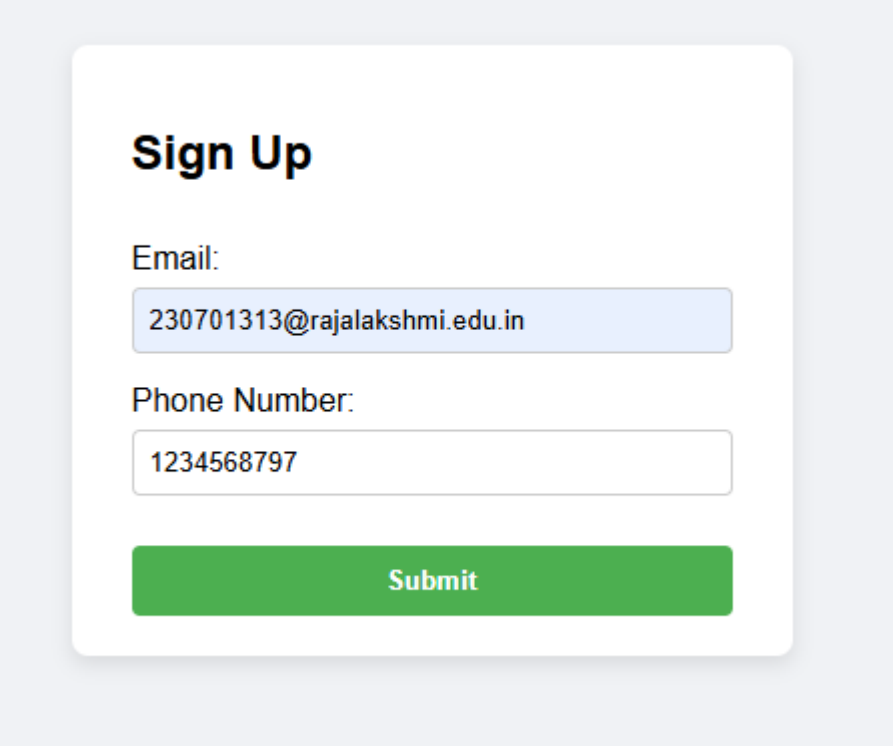
Finally, add JavaScript to validate the input fields using Validator.js and display error messages.

JAVASCRIPT:

```
document.getElementById('myForm').addEventListener('submit', function (e) {
  e.preventDefault();

  const email = document.getElementById('email').value.trim();
  const phone = document.getElementById('phone').value.trim();

  let isValid = true;
```

```
  // Reset errors
  document.getElementById('emailError').textContent = '';
  document.getElementById('phoneError').textContent = '';

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

  // Validate phone (assumes US phone format, adjust as needed)
  if (!validator.isMobilePhone(phone, 'en-US')) {
    document.getElementById('phoneError').textContent = 'Please enter a valid US phone number.';
    isValid = false;
  }

  if (isValid) {
    alert('Form submitted successfully!');
    // Here you can submit the form or send data via AJAX
  }
});
```

OUTPUT:



RESULT:

Hence the validator has been implemented by using javascript,html,css

Excercise 10 **Date:**

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 Dashboard</title>
 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
 <style>
  body {
   font-family: Arial, sans-serif;
   padding: 40px;
   background-color: #f4f4f4;
   display: flex;
   flex-direction: column;
   align-items: center;
  }
  h2 {
```

```
    margin-bottom: 20px;
   }
   .chart-container {
    width: 600px;
    margin-bottom: 40px;
    background: white;
    padding: 20px;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
   }
  </style>
</head>
<body>

  <h2>Inventory Management Dashboard</h2>

  <div class="chart-container">
   <canvas id="barChart"></canvas>
  </div>

  <div class="chart-container">
   <canvas id="pieChart"></canvas>
  </div>

  <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:

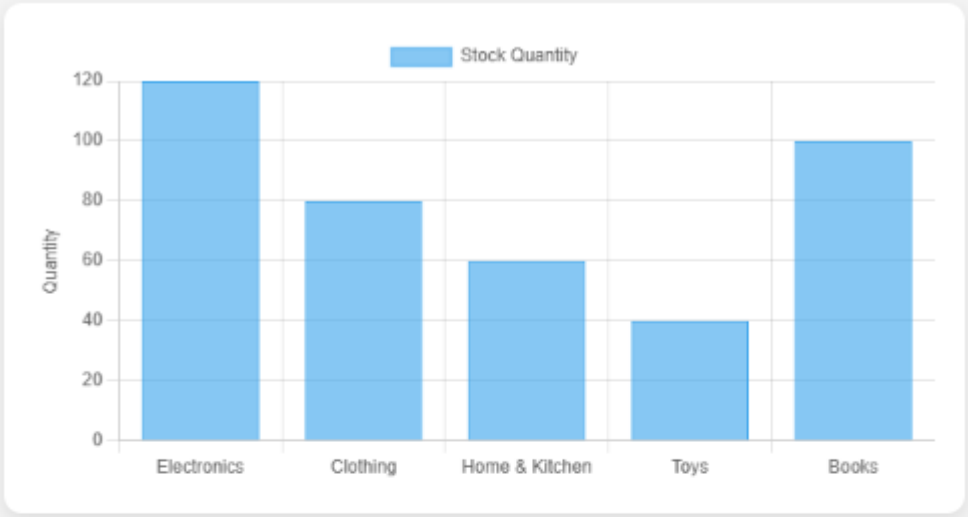
```
// Sample inventory data
const inventoryData = [
  { category: 'Electronics', quantity: 120 },
  { category: 'Clothing', quantity: 80 },
  { category: 'Home & Kitchen', quantity: 60 },
  { category: 'Toys', quantity: 40 },
  { category: 'Books', quantity: 100 },
];

// Extract labels and quantities
const labels = inventoryData.map(item => item.category);
const quantities = inventoryData.map(item => item.quantity);

// ===== Bar Chart =====
const barCtx = document.getElementById('barChart').getContext('2d');
const barChart = new Chart(barCtx, {
  type: 'bar',
  data: {
    labels: labels,
    datasets: [{
      label: 'Stock Quantity',
```

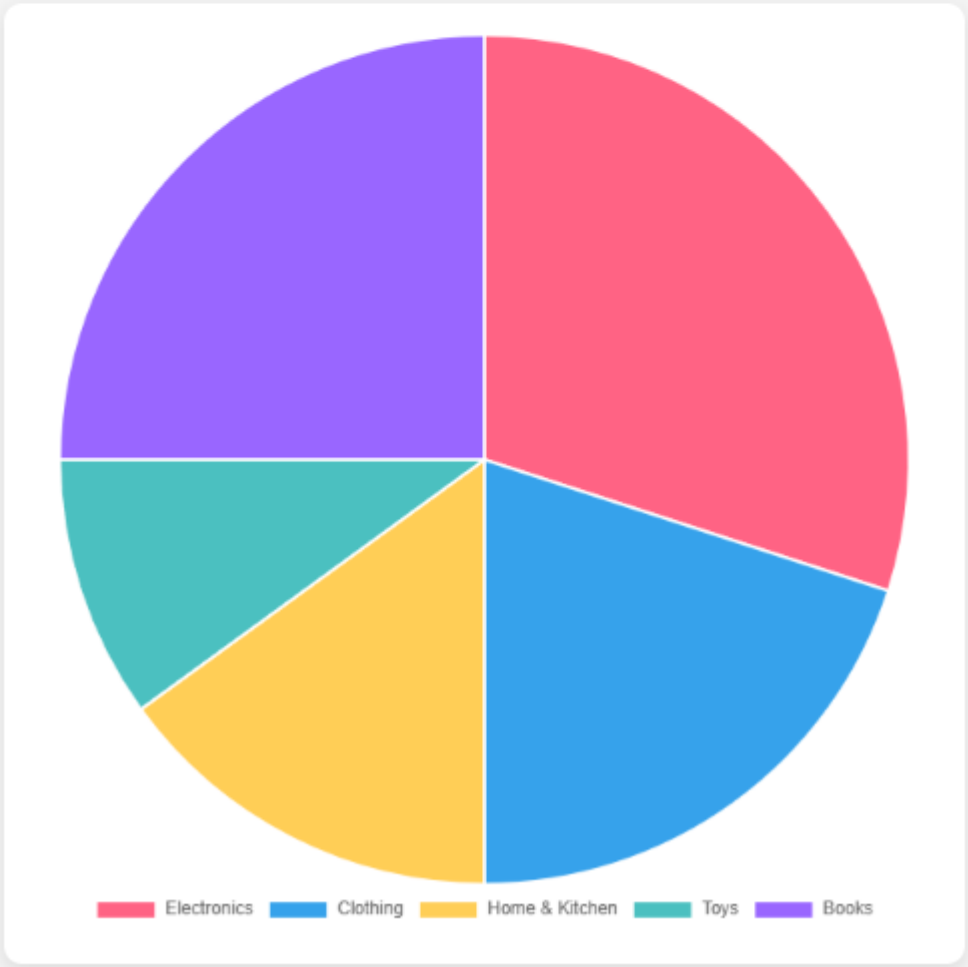
```
      data: quantities,
      backgroundColor: 'rgba(54, 162, 235, 0.6)',
      borderColor: 'rgba(54, 162, 235, 1)',
      borderWidth: 1,
    }]
  },
  options: {
    responsive: true,
    scales: {
      y: {
        beginAtZero: true,
        title: { display: true, text: 'Quantity' }
      }
    }
  }
});

// ===== Pie Chart =====
const pieCtx = document.getElementById('pieChart').getContext('2d');
const pieChart = new Chart(pieCtx, {
  type: 'pie',
  data: {
    labels: labels,
    datasets: [{
      data: quantities,
      backgroundColor: [
        '#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0', '#9966FF'
```

```
    ]
   }]
  },
  options: {
   responsive: true,
   plugins: {
    legend: {
     position: 'bottom'
    }
   }
  }
});
```

OUTPUT:

Inventory Management Dashboard





RESULT:

Data visualization (e.g., pie charts, bar graphs) for an inventory management system using javascript has been implemented