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 : 2116230701380 **Name :** VIGNESHWARAN 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 (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
	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

How to Start Designing in Figma

Step 1: Sign Up and Create a New Project

1. Go to figma.com and create an account.
2. Click **"New File"** to open a blank canvas.

Step 2: Create the Frame (Artboard)

1. Select the **Frame Tool (F)** from the left toolbar.
2. Choose a **mobile preset** (e.g., iPhone 13) from the right panel.

Step 3: Design the Login Screen

1. **Background:** Select the frame → Change "Fill" color (e.g., #F0F4F8).
2. **Logo:** Draw a rectangle (R) or use text (T) for "MyApp" at the top center.
3. **Input Fields:**
    - Create two rounded rectangles (R) for username & password.
    - Add placeholder text ("Enter your email", "Enter your password").
4. **Login Button:**
    - Draw a rectangle (R) → Set color (#1E88E5).
    - Add "Login" text inside.

Step 4: Align Elements

1. Select all elements → Click **"Auto Layout" (Shift + A)**.
2. Use alignment tools for proper spacing.

Step 5: Add Interaction (Prototyping)

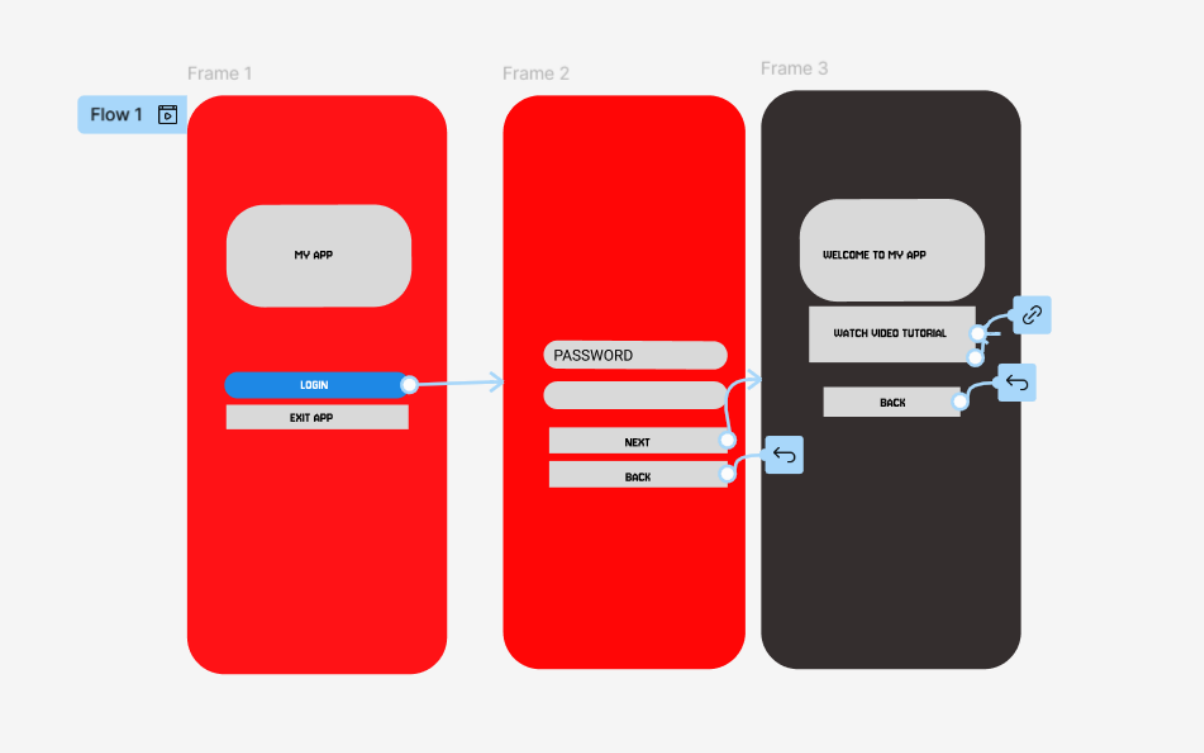
1. Click **"Prototype"** (right panel).
2. Select the "Login" button → Drag the blue dot to a new frame.
3. Set **"On Click" → "Navigate to"** (next screen).
4. Add **"Smart Animate"** for smooth transition.

Step 6: Preview & Share

1. Click **"Play"** (top-right) to preview the prototype.
2. Click **"Share"** to generate a link or invite collaborators.

Step 7: Export Assets

1. Select an element (logo, button) → Click **"Export"** (right panel).
2. Choose format (PNG, JPG, SVG) and download.



Figma UI Setup for Memory Recall Task

A. Home Screen (Instruction Page)

1. **Create Frame** (1024x768px).
2. **Add Instructions** – Heading: "Memory Recall Task", body text with details.
3. **Start Button** – Rectangle (R) with "Start" text, linked to Chunking Phase.

B. Chunking Phase (Display Chunked Items)

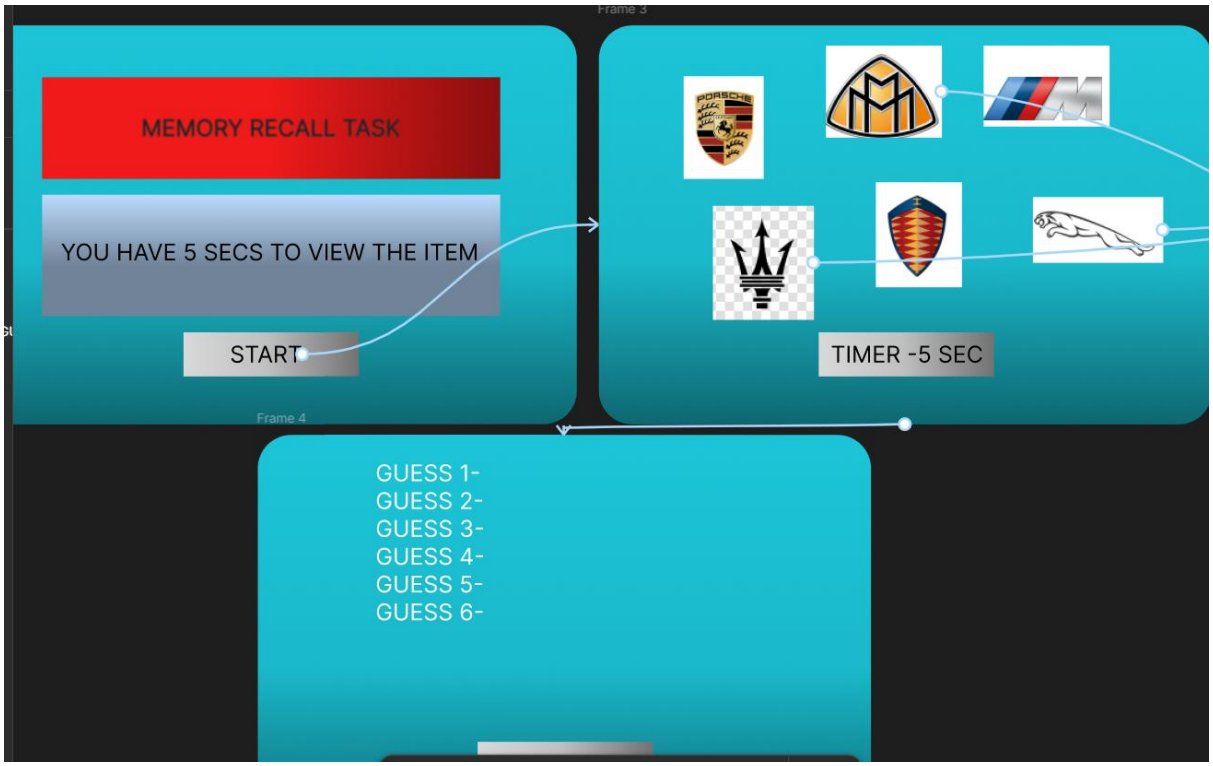
1. **Create New Frame** for chunked icons or text.
2. **Design Chunks** – Group 3-5 items with/without borders.
3. **Set Viewing Time** – Auto-transition after 5000ms to Recall Phase.

C. Recall Phase

1. **Create Recall Frame** for user input.
2. **Input Options:**
    - **Multiple-Choice:** Select from given options.
    - **Text Input:** Type recalled items.
3. **Submit Button** – Navigates to Result Screen.

D. Result Screen

1. **Feedback Message** – Display recall accuracy (e.g., “4/5 items correct”).
2. **Experiment Analysis** – Vary chunk size and type to measure recall impact.


Frame 3
MEMORY RECALL TASK
YOU HAVE 5 SECS TO VIEW THE ITEM
START
TIMER -5 SEC
Frame 4
GUESS 1-
GUESS 2-
GUESS 3-
GUESS 4-
GUESS 5-
GUESS 6-

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:

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

```
gopal died
Task 'gopal died' added.

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice:
3
Enter task number to remove:
gopal died
Please enter a valid number.

Options: 1. Add Task  2. View Tasks  3. Remove Task  4. Exit
Enter your choice:
2
Your tasks:
1. gopal died

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

Session Killed due to Timeout.
Press Enter to exit terminal
```

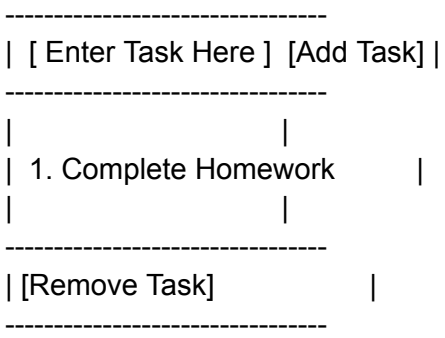
ii) GUI (Graphical User Interface)

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:

Listening...

Listening...

Listening...

Listening...

Listening...

Listening...

Listening...

RESULT:

the above python programs has been executed succesfully

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN
Year/Branch/Section : II/CSE/D
Register No. :
230701380**Semester :** IV
Academic Year: 2024-25

Ex. No. : 4a

Register No. : 230701380 **Name :** G VIGNESHWARAN

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:

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

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

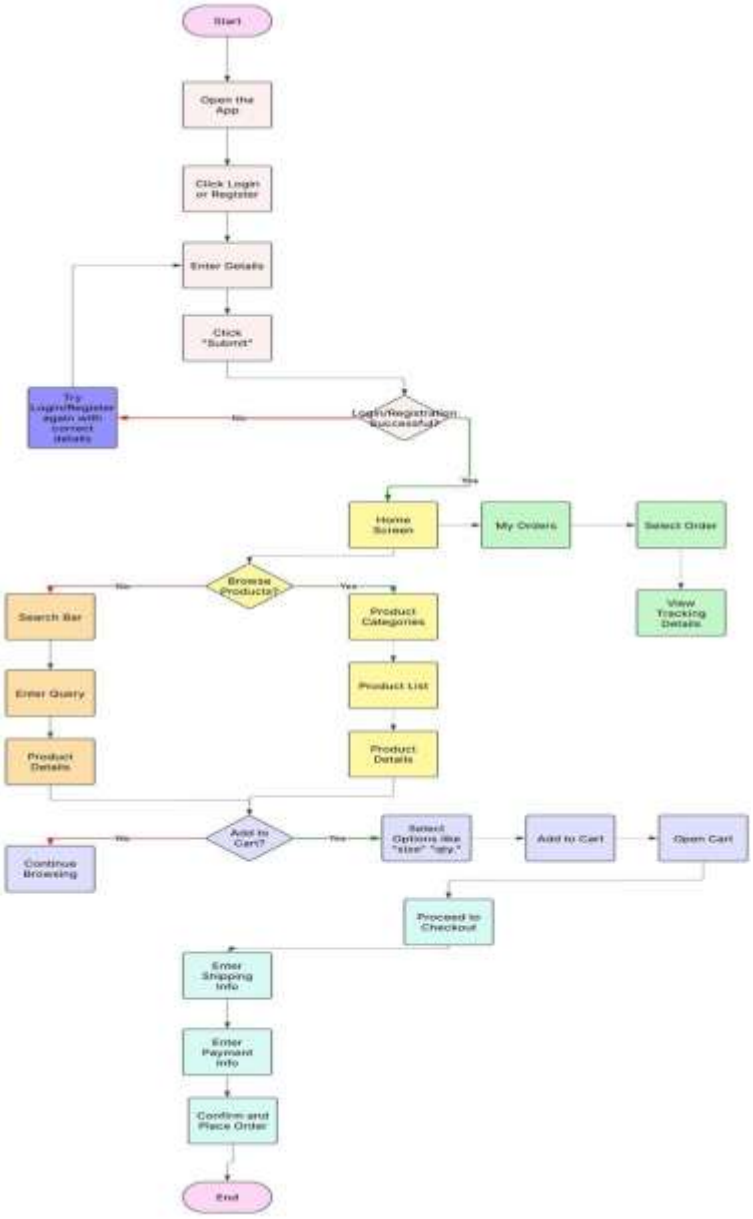
Add to Cart Flow
● Steps:
○ View product details
○ Select options (size, color, quantity)
○ Add product to cart

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:



RESULT:

Task analysis for an online shopping app has been conducted to document user flows and corresponding wireframes using Lucidchart has been successfully created.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 4b

Register No. : 230701380 **Name : G VIGNESHWARAN**

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:


Landing Page
Browse Collection
Featured Items
Item Details
Shopping Bag
Payment
Purchase Complete
Order Tracking

RESULT:

Task analysis for an online shopping app has been conducted to document user flows and corresponding wireframes using dia has been successfully created.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 5a

Register No. : 230701380 **Name : G VIGNESHWARAN**

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 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.

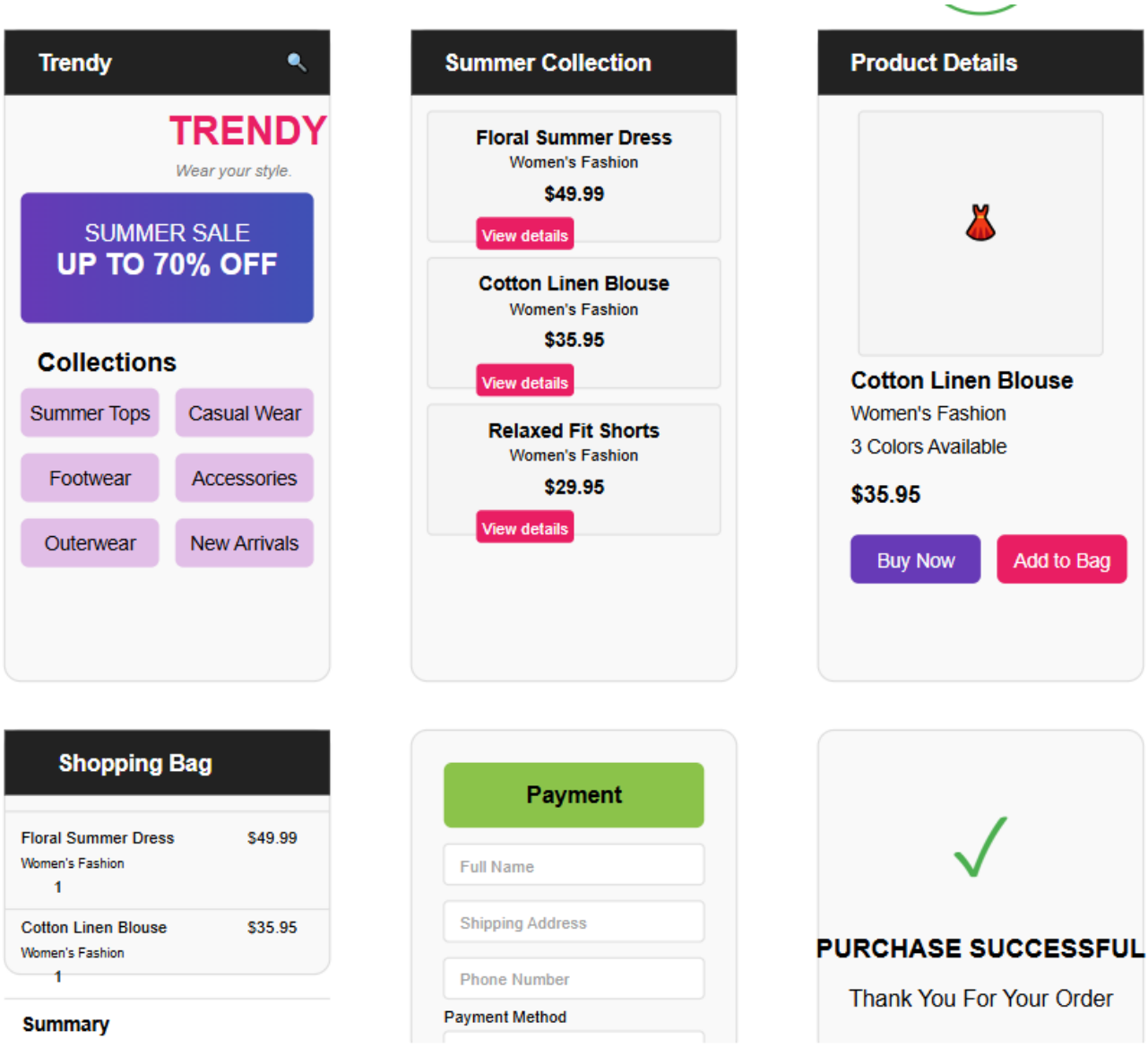
○ Export the prototype as an HTML file or share it via Axure Cloud.

2. User Training and Support:

○ Conduct training sessions to familiarize users with the new interface.

○ Provide documentation and support for any issues.

OUTPUT:



RESULT:

Demonstration of the lifecycle stages of UI design via the RAD model and development of a small interactive interface employing Axure RP has been successfully completed.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 6

Register No. : 230701380 **Name : G VIGNESHWARAN**

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:

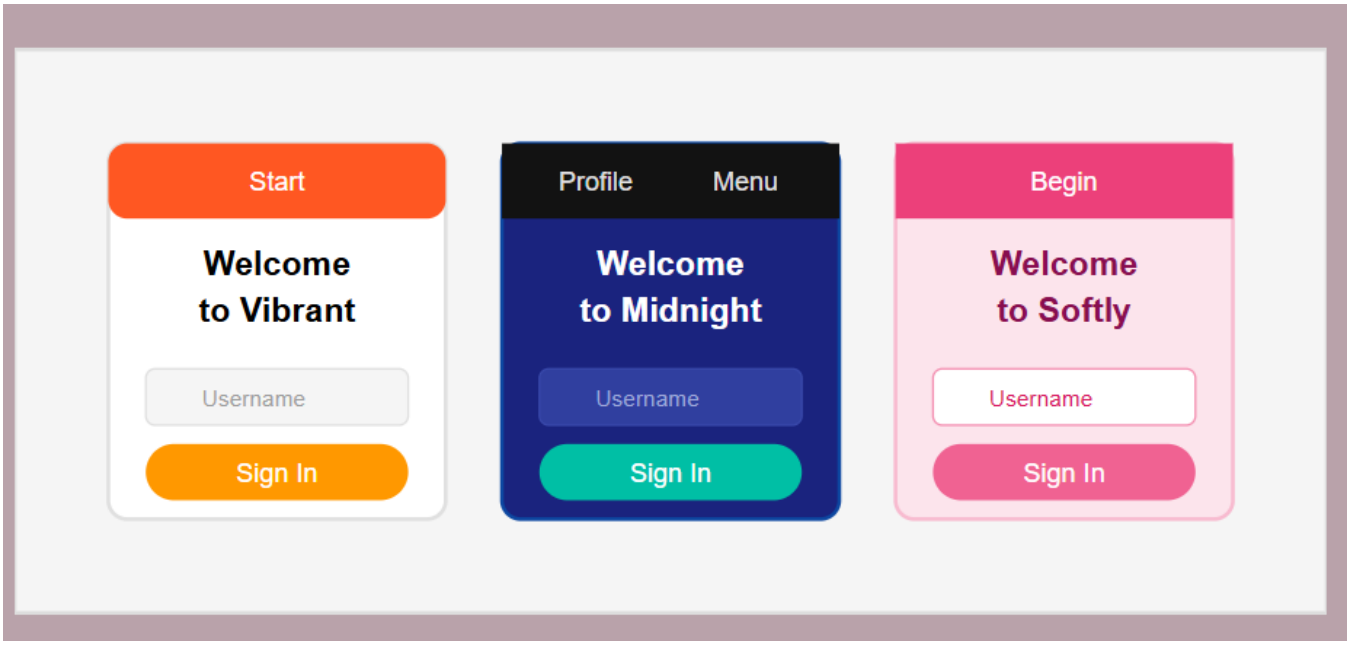
○ Based on the feedback, make necessary adjustments to the layout and color scheme.

○ 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:

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.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 7a

Register No. : 230701380 **Name : G VIGNESHWARAN**

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.

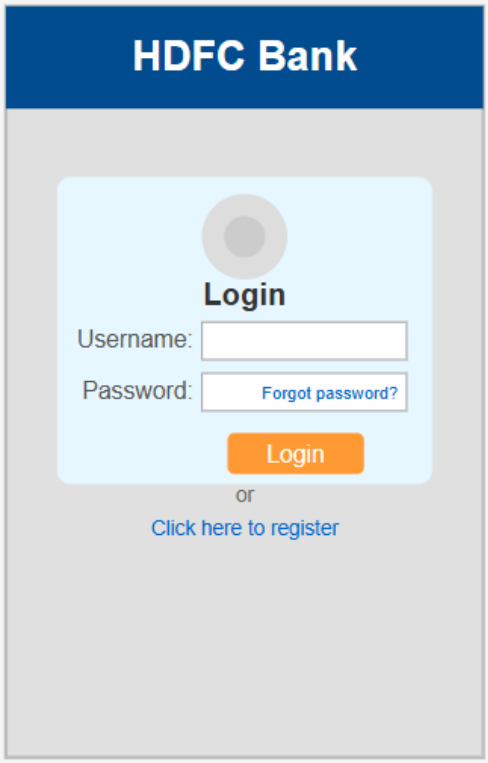
7. Add Annotations:

○ Include annotations to explain the functionality of different elements.

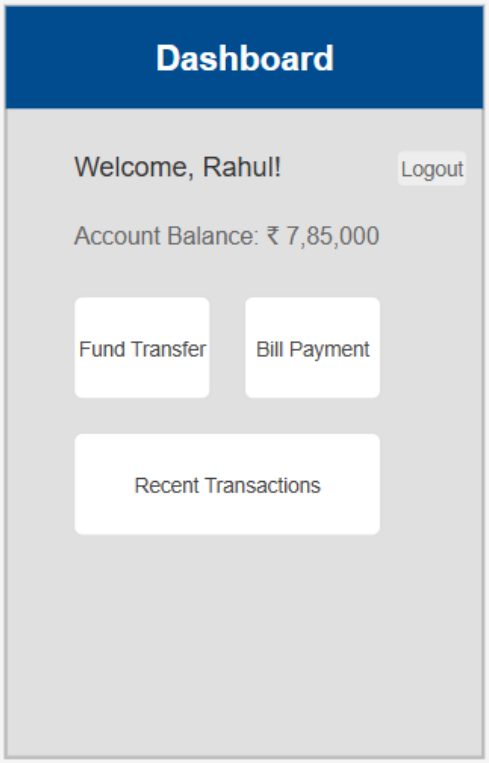
8. Export Your Wireframes:

○ Once satisfied with your digital wireframes, export them in your preferred format (e.g., PNG, PDF).

OUTPUT:



Transfer Money Page



Transaction History

RESULT:

A low-fidelity paper prototype for a banking app has been formed and convert into digital wireframes using Pencil Project.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 7b

Register No. : 230701380 Name :G V I G N E S H W A R A N

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:

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):

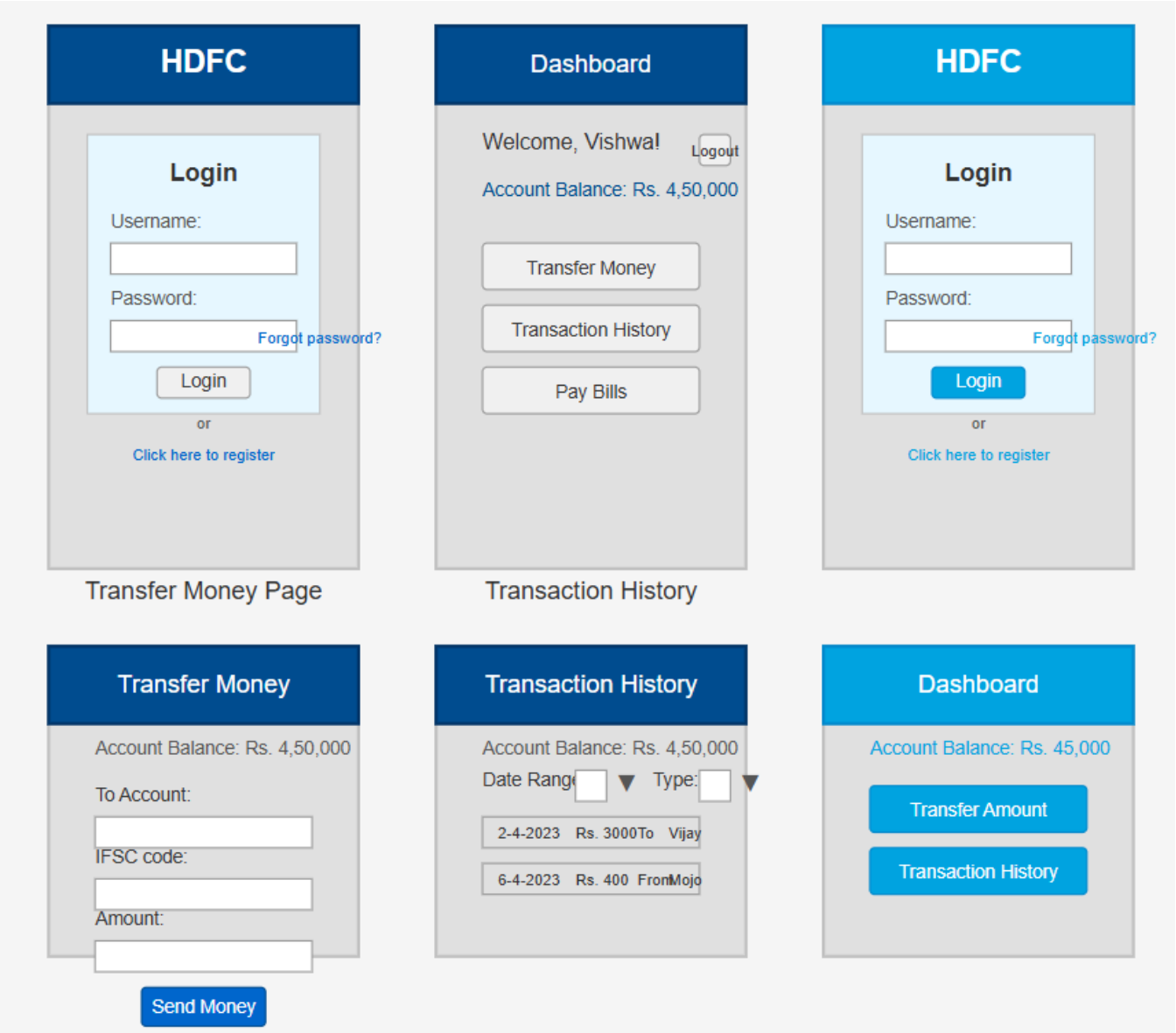
○ If you want to show navigation flows, you can add arrows or other indicators to demonstrate how users will move between screens.

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:


HDFC
Login
Username:
Password:
Forgot password?
Login
or
Click here to register
Transfer Money Page
Dashboard
Welcome, Vishwa!
Logout
Account Balance: Rs. 4,50,000
Transfer Money
Transaction History
Pay Bills
Transaction History
HDFC
Login
Username:
Password:
Forgot password?
Login
or
Click here to register
Transfer Money
Account Balance: Rs. 4,50,000
To Account:
IFSC code:
Amount:
Send Money
Transaction History
Account Balance: Rs. 4,50,000
Date Rang
Type:
2-4-2023 Rs. 3000To Vijay
6-4-2023 Rs. 400 FromMojo
Dashboard
Account Balance: Rs. 45,000
Transfer Amount
Transaction History

RESULT:

A low-fidelity paper prototype for a banking app has been formed and convert into digital wireframes using Inkscape.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 8a

Register No. : 230701380 **Name : G VIGNESHWARAN**

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.

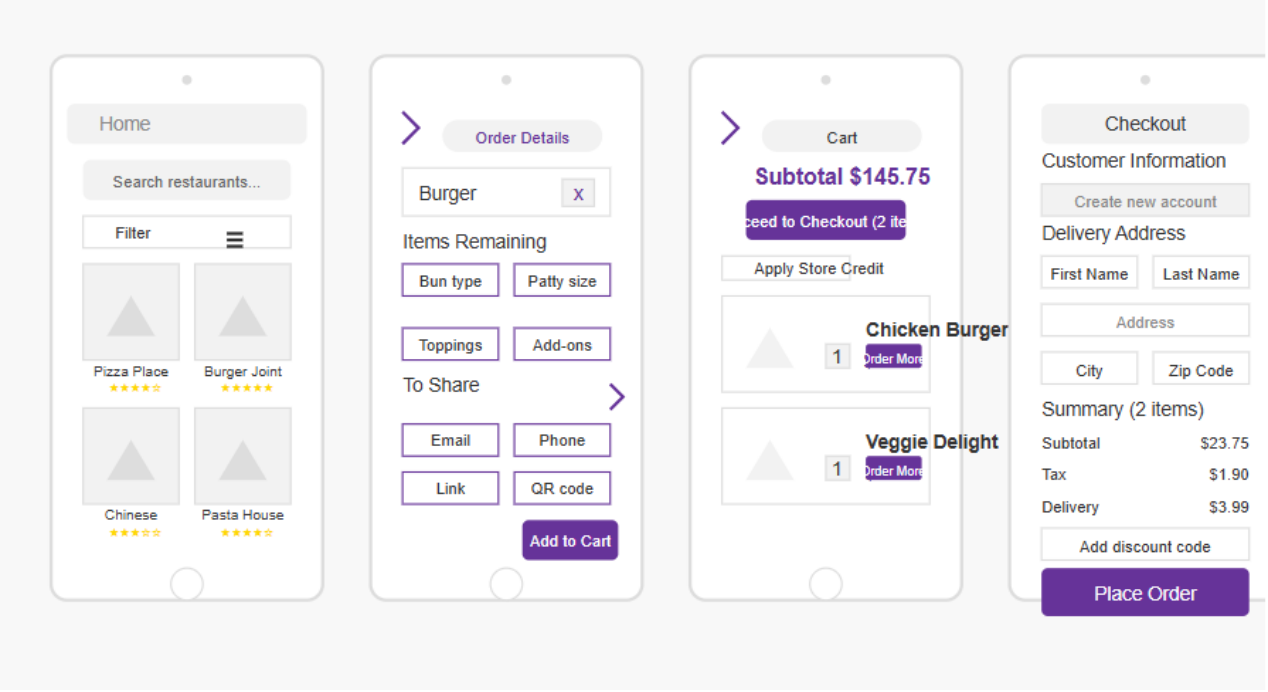
● Place Order Button: Finalize the order.

Order Confirmation Screen

● Order Summary: Shows the order details.

● Estimated Delivery Time: Provides an estimated delivery time.

OUTPUT:


Home
Search restaurants...
Filter
Pizza Place
Burger Joint
Chinese
Pasta House
Order Details
Burger
X
Items Remaining
Bun type
Patty size
Toppings
Add-ons
To Share
Email
Phone
Link
QR code
Add to Cart
Cart
Subtotal $145.75
ceed to Checkout (2 ite
Apply Store Credit
Chicken Burger
1
Order More
Veggie Delight
1
Order More
Checkout
Customer Information
Create new account
Delivery Address
First Name
Last Name
Address
City
Zip Code
Summary (2 items)
Subtotal $23.75
Tax $1.90
Delivery $3.99
Add discount code
Place Order

RESULT:

Storyboards representing the user flow for a mobile food delivery app, using Balsamiq has been successfully created.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 8b

Register No. : 230701380 Name : G VIGNESHWARAN

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

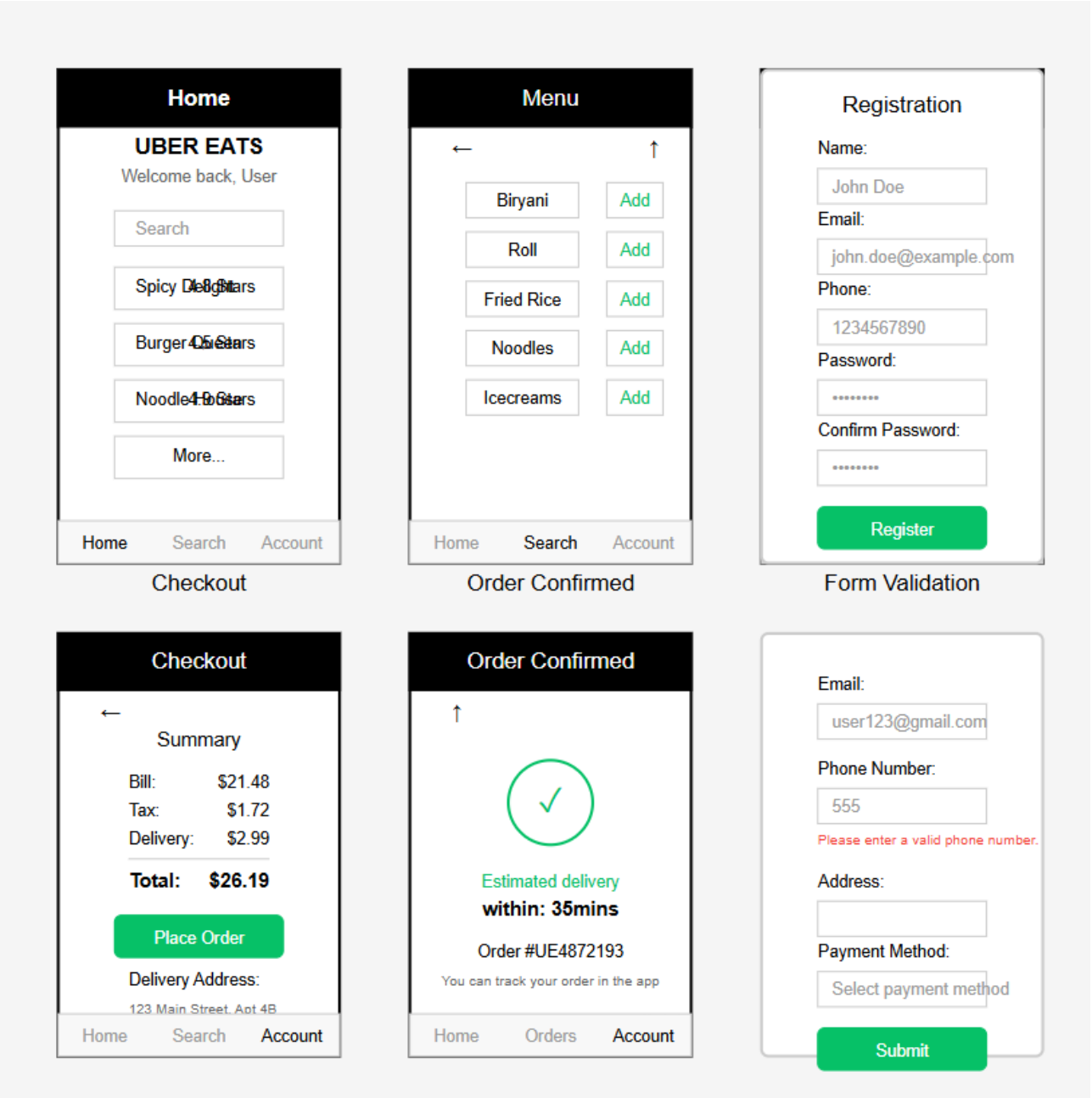
○ Place Order button

5. Order Confirmation Screen:

○ Order summary

○ Estimated delivery time

OUTPUT:



RESULT:

Storyboards representing the user flow for a mobile food delivery app, using OpenBoard has been successfully created.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 9

Register No. : 230701380 **Name : G VIGNESHWARAN**

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.

```
<!-- index.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"></scri
pt>
  <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.

```
/* 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; }
```

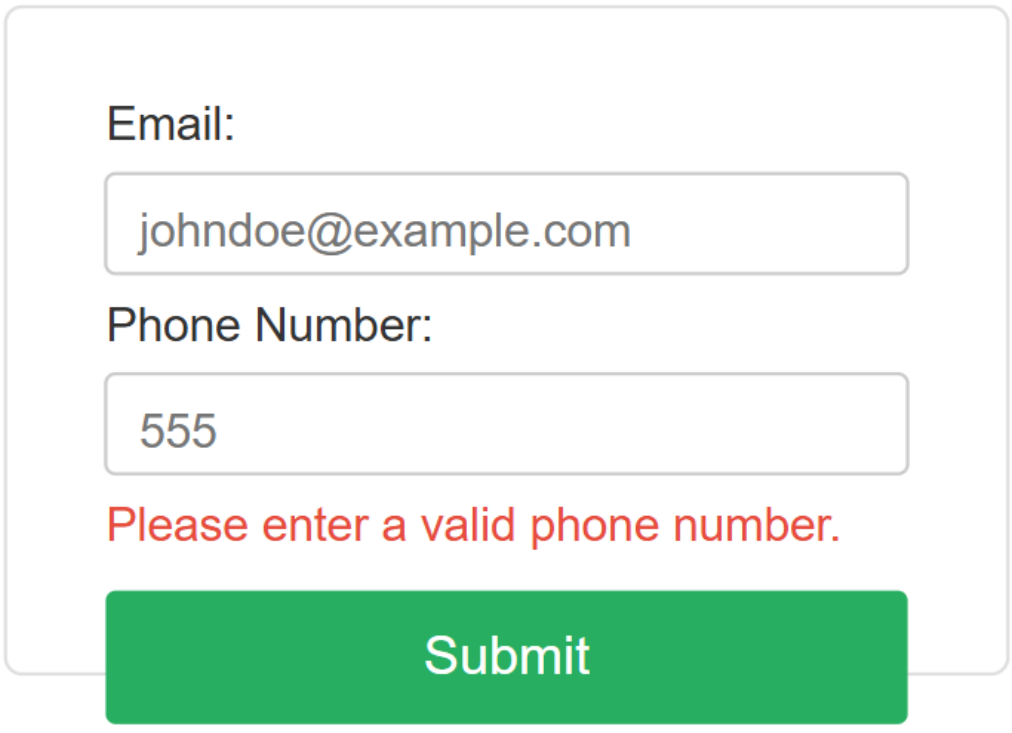
Step 3: Adding JavaScript for Validation Finally, add JavaScript to validate the input fields using Validator.js and display error messages.

```
// script.js

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

  e.preventDefault();
```

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

const emailError = document.getElementById('emailError');
const 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 valid, log the input values
```

```
  if (validator.isEmail(email) && validator.isMobilePhone(phone, 'any')) {
    console.log('Email:', email);
    console.log('Phone:', phone);
  }
});
```

OUTPUT:



RESULT:

Input forms that validate data, such as email and phone number, and display error messages using HTML/CSS and JavaScript with Validator.js has been designed.

RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI
ENGINEERING COLLEGE

CS23A34
USER INTERFACE AND DESIGN LAB

Laboratory Observation NoteBook

Name : G VIGNESHWARAN

Year/Branch/Section : II/CSE/D

Register No. : 230701380

Semester : IV

Academic Year: 2024-25

Ex. No. : 10

Register No. : 230701380 **Name : G VIGNESHWARAN**

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.

```
<!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;
      display: block;
    }
  </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

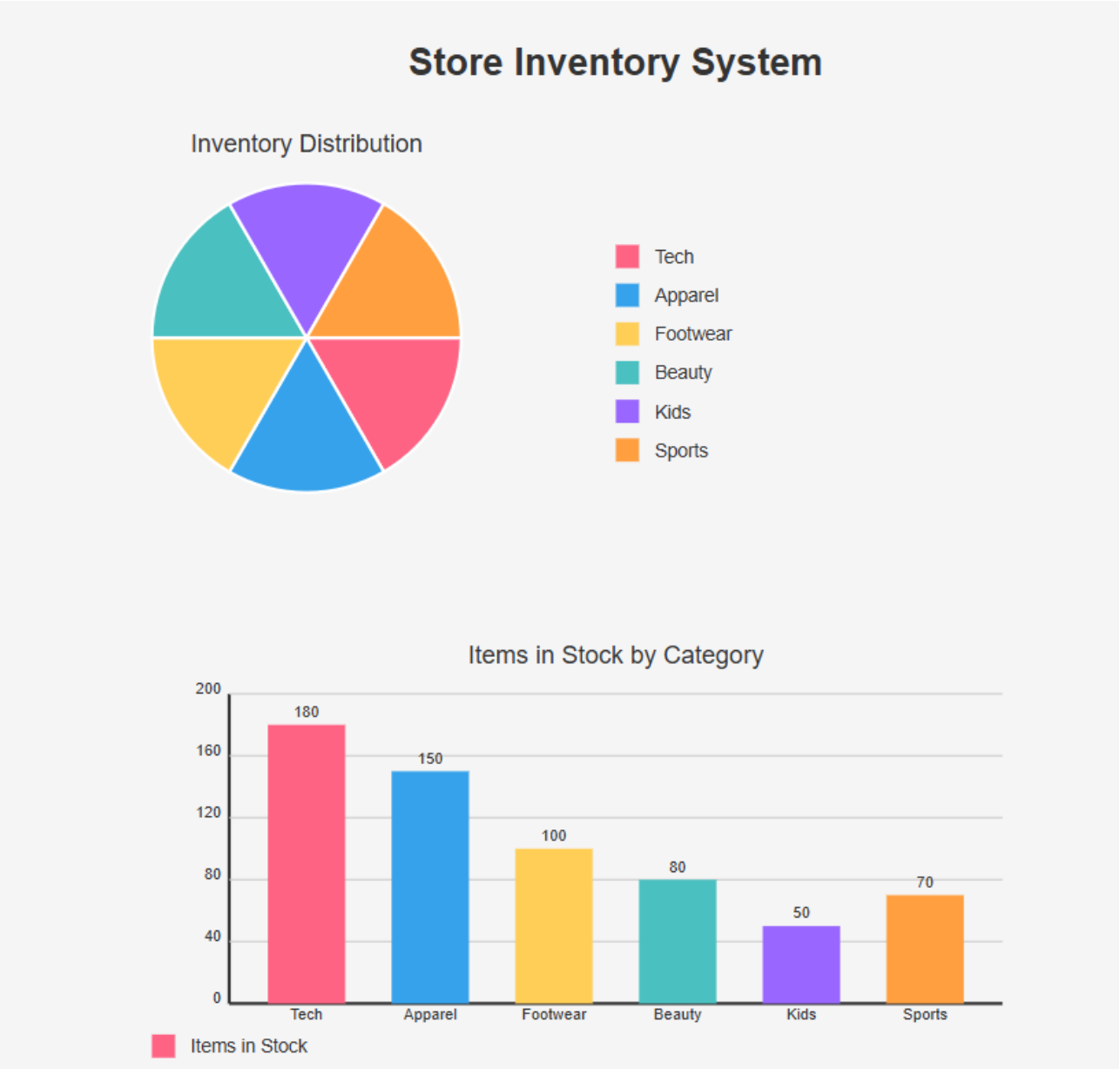
Next, create a JavaScript file (script.js) to handle the data visualization logic.

```
// 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,
  plugins: {
      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,
  plugins: {
      title: {
        display: true,
        text: 'Items in Stock by Category'
      }
    },
    scales: {
      y: {
        beginAtZero: true
      }
    }
  }
});
```

OUTPUT:



RESULT:

Data visualizations, such as pie charts and bar graphs, for an inventory management system using JavaScript has been successfully created.