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 : 230701396 **Name : Praveenraj**

Branch : CSE **Year/Section : II-D**

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

EXPERIMENT - 01

Aim:

The aim of this experiment is to guide you through the process of designing a simple mobile app login screen using Figma. This includes creating a frame, designing the UI elements, adding interactivity (prototyping), and sharing/exporting the design.

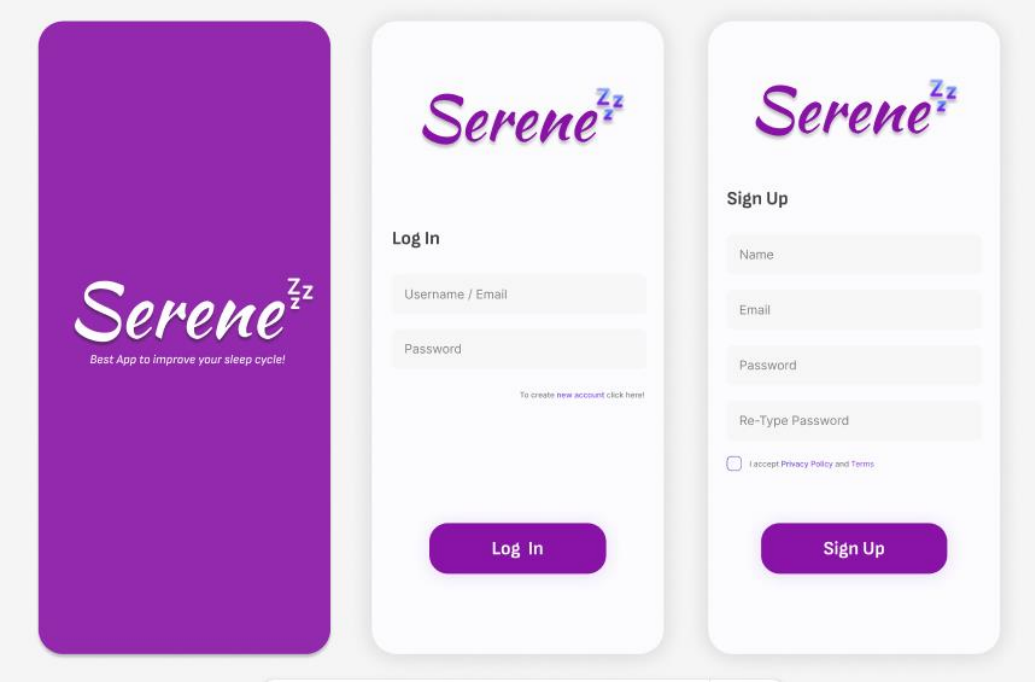
Procedure:

1. **Sign Up and Create a New Project**:
    - Go to figma and create an account if you don't have one.
    - Log in and click on "New File" to start a blank project.
2. **Create the Frame (Artboard)**:
    - Select the **Frame Tool** (shortcut: F).
    - A frame will appear on the canvas, representing your mobile app screen.
3. **Design the Login Screen**:
    - **Add Background Color**:
        - Select the frame and in the right panel, under **Fill**, choose a color (e.g., light blue #E3F2FD).
    - **Insert a Logo**:
        - Use the **Rectangle Tool** (R) to draw a placeholder for a logo.
        - Add text using the **Text Tool** (T) to write your app name (e.g., "MyApp").
        - Adjust the font size and color in the right panel.
    - **Add Input Fields** (Username & Password):
        - Draw two rectangles below the logo for the username and password input fields.
        - Inside each, add placeholder text (e.g., "Enter your email").
        - Apply rounded corners in the **Corner Radius** section of the right panel.
    - **Add a Login Button**:
        - Use the **Rectangle Tool** (R) to create a button and set its color to blue .
        - Use the **Text Tool** (T) to add the text "Login" inside the button.
        - Group the button and text by selecting both and pressing **Ctrl + G** (Windows) or **Cmd + G** (Mac).
4. **Align Elements**:
    - Use the **Auto Layout** feature (Shift + A) to ensure even spacing between elements.
    - Use the **Alignment Tools** in the top menu to center everything vertically and horizontally.
5. **Prototyping**:

- Click on the **Prototype** tab in the right panel.
- Select the **Login button** and drag the blue dot to a new frame (e.g., a home screen).
- Choose an animation effect (e.g., **Smart Animate**).

6. **Preview the Design**:
   - Click the **Play** button in the top-right corner to preview the app prototype.
   - Try clicking the login button to see the transition to the next screen.
7. **Share Your Design**:
   - Click the **Share** button in the top-right corner.
   - You can invite team members via email or generate a shareable link and adjust permissions (View, Edit, or Comment only).
8. **Export Assets**:
   - Select the elements you want to export (e.g., the logo or button).
   - In the right-hand panel, click **Export** and choose a format (PNG, JPG, SVG).
   - Click **Export** to download the assets for developers.

Design:



Result:

Thus, the mobile app login screen was successfully designed in Figma, featuring interactive elements such as a login button and input fields. The design is shareable, and assets were exported for development purposes.

EXPERIMENT - 02

Aim:

The aim of this experiment is to design a user interface (UI) in Figma for evaluating the effect of chunking on user memory. The UI will involve showing groups of visual elements (icons or text), followed by a recall phase where users will attempt to recall the displayed items. The goal is to assess how different chunk sizes and types (icons vs. text) affect memory retention.

Procedure:

1. **Home Screen (Instruction Page)**:
    - Create a frame (1024x768px) to represent the Home Screen.
    - Add instructions to explain the task, including the time limit for viewing the items.
    - Design a "Start" button to transition to the Chunking Phase.
2. **Chunking Phase**:
    - Create a new frame for the Chunking Phase.
    - Display grouped items (either icons or text) in chunks. Optionally, use borders to separate chunks or place them without borders for a different effect.
    - Set the viewing time to 5 seconds using Figma's prototyping tools to transition to the Recall Phase after a 5-second delay.
3. **Recall Phase**:
    - Create a new frame for the Recall Phase where users will recall the displayed items.
    - Provide options for users to recall the items using either multiple-choice (checkboxes/radio buttons) or text input fields.
    - Add a "Submit Recall" button that will move the user to the Result Screen after submission.
4. **Result Screen**:
    - Create a feedback screen that shows how many items the user recalled correctly (e.g., "You recalled 4/5 items correctly!").
    - Evaluate the impact of chunk size (3 vs. 5 items) and chunk type (icons vs. text) on recall.

Outputs:

Result:

The experiment will allow you to evaluate the effects of chunking on user memory by analyzing recall accuracy across different chunk sizes and types. The feedback screen will display the user's performance, helping assess how varying the number of items per chunk and the format (icons vs. text) influences memory retention.

EXPERIMENT - 03

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:

1. **CLI (Command Line Interface)**:

- Implement a task management system in the terminal where users can add, view, and remove tasks by typing commands.
- The program allows users to choose from a menu with options for adding, viewing, and removing tasks.

Output:

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

2. **VUI (Voice User Interface)**:

- Use the `speech_recognition` library for voice input and `pyttsx3` for voice output.
- The system listens for voice commands to add, view, or remove tasks, and speaks back the results or errors.

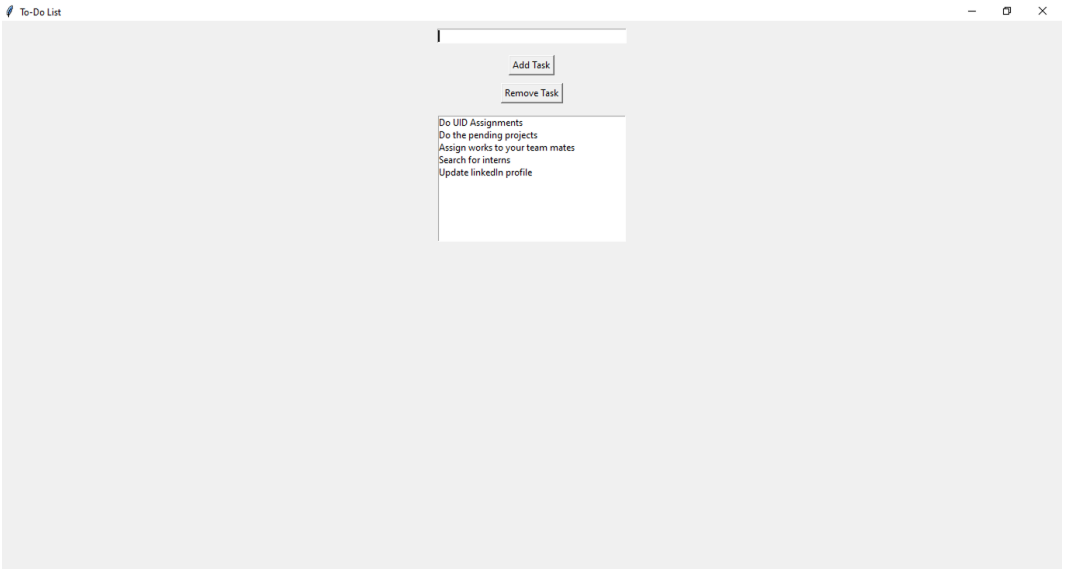
Output:

```
PS C:\Users\DELL\Desktop\PythonUI> & C:/Python313/python.exe c:/Users/DELL/Desktop/PythonUI/todo_list.py
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...
Listening...

* History restored

PS C:\Users\DELL\Desktop\PythonUI>
```

3. **GUI (Graphical User Interface)**:

- Use Tkinter to create a graphical interface with buttons for adding and removing tasks, and a listbox to display tasks.
- Users interact with the GUI by clicking buttons to add or remove tasks.

Output:



Result:

Thus, the CLI, GUI, and VUI allows users to manage tasks through different interaction methods, with user satisfaction varying based on ease of use, speed, and familiarity with the interface is implemented successfully.

EXP NO:3A
TITLE:Create a prototype with familiar and unfamiliar navigation elements. Evaluate ease of use with different user groups using proto.io

AIM:

The aim is to develop a prototype incorporating both familiar and novel navigation

elements and assess usability among diverse user groups using Proto.io.

PROCEDURE:

Step 1: Sign Up & Log In

Go to proto.io

Sign up or log in

Step 2: Create a New Project

Click "Create New Project"

Enter project name (e.g., "Simple App Example")

Select device type (e.g., iPhone X)

Click "Create"

Step 3: Design Home Screen

Add Screen: Click "+" → Select "Blank" → Name it "Home"

Add Elements:

Drag "Header" → Edit text to "Home Screen"

Drag "Button" → Edit text to "Go to Profile"

Add Interaction:

Select button → "Interactions" tab → "+ Add Interaction"

Trigger: Tap/Click, Action: Navigate to Screen → Create "Profile" screen

Step 4: Design Profile Screen

Add Elements:

Drag "Header" → Edit text to "Profile Screen"

Drag "Image" → Upload profile picture

Drag "Text" → Add profile info (e.g., "John Doe, Software Engineer")

Add Back Button:

Drag "Button" → Edit text to "Back to Home"

Add Interaction:

Select button → "Interactions" tab → "+ Add Interaction"

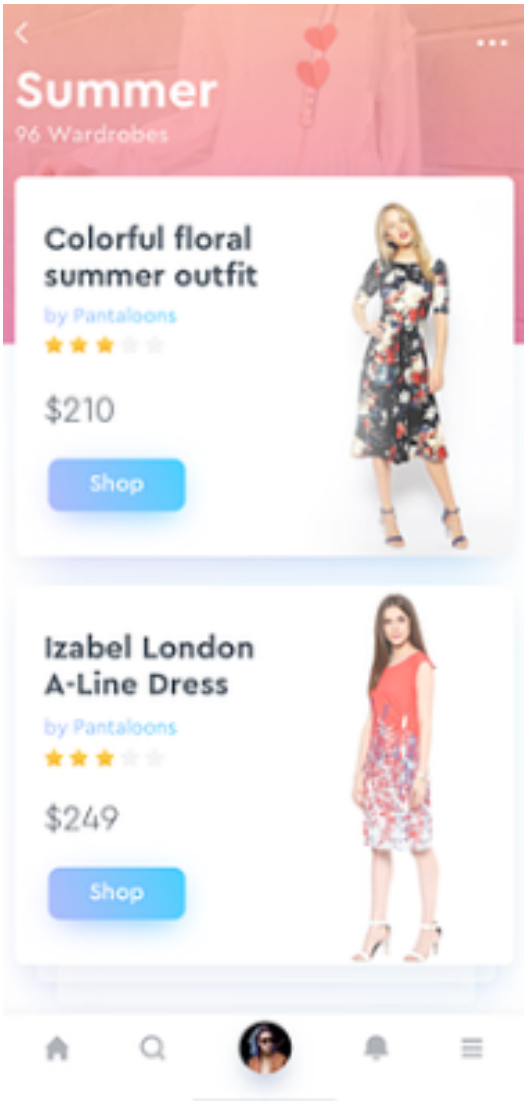
Trigger: Tap/Click, Action: Navigate to Home

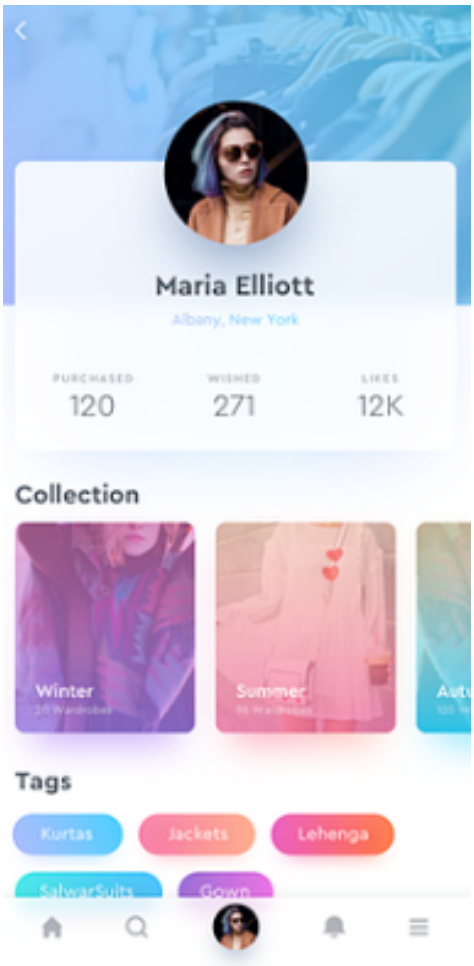
Step 5: Preview & Share

Preview: Click "Preview" to test navigation

Share: Click "Share" → Copy link → Share for feedback

OUTPUTS:





RESULT:

Successfully created an interactive prototype with a Home and Profile screen, implemented navigation using buttons, tested interactions using proto.io.

Link: https://pr.to/2GSLJI/

Excercise 3b

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.

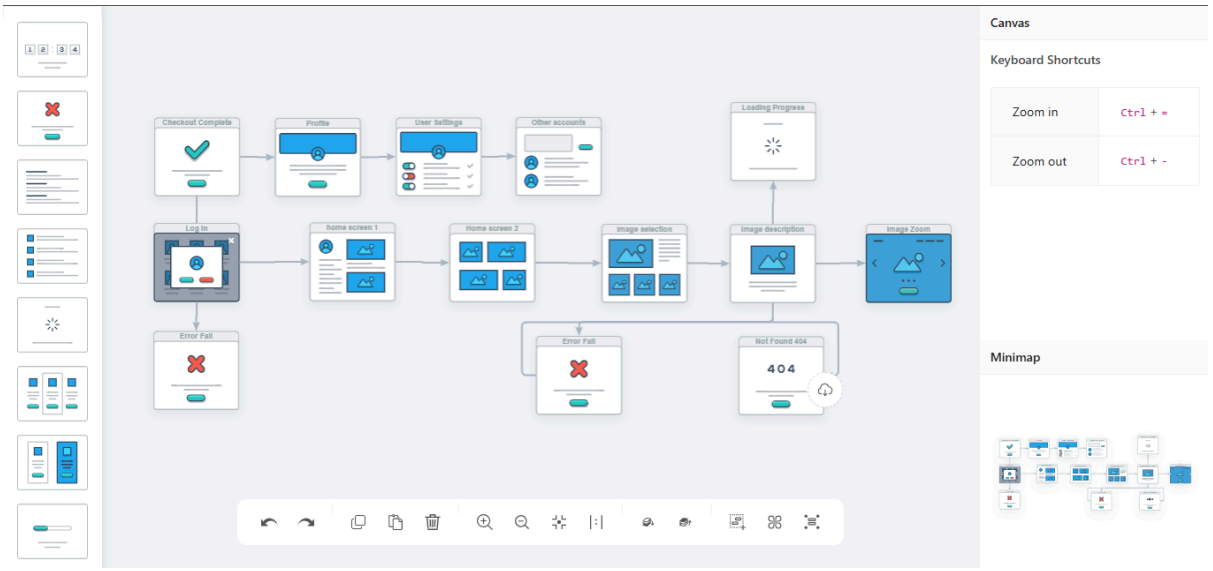
Step 5: Conduct Testing

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:



RESULT:

Thus, the prototype with both well-known and new navigation elements and measure user-friendliness across different user groups using Wireflow is designed.

Excercise 4a

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

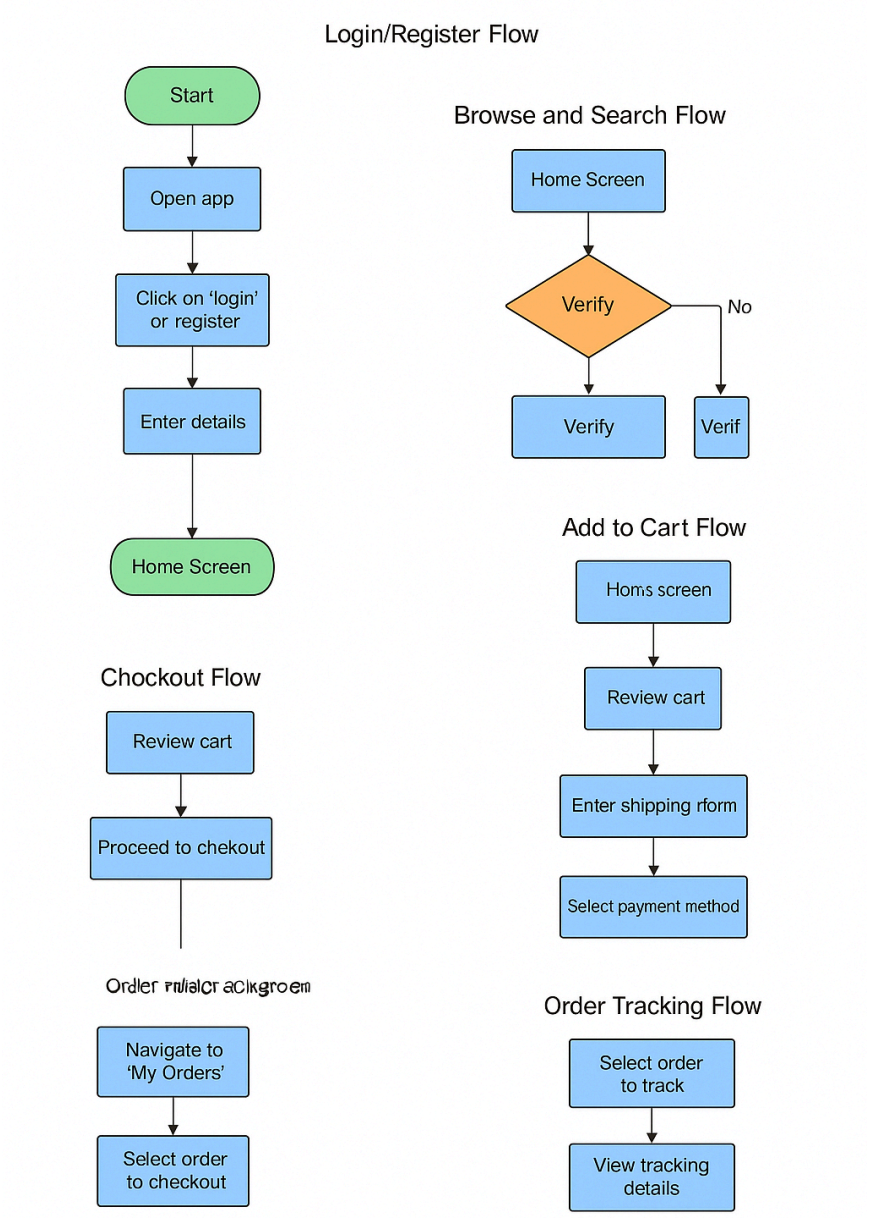
Checkout Flow

- 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:



RESULT:

Thus, the the main tasks within an online shopping app is documented using LucidChart.

Excercise 4b

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

AIM:

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

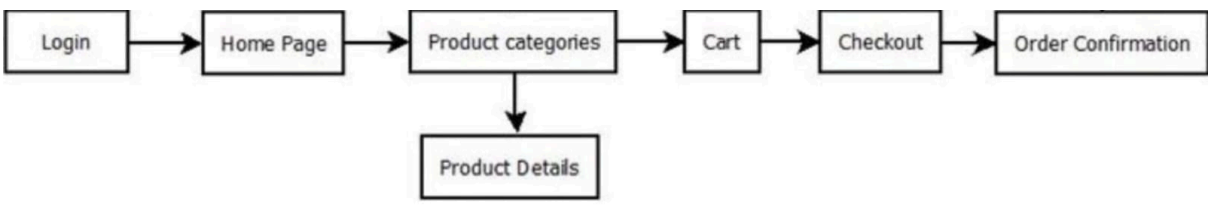
PROCEDURE:

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

1. Install Dia:
    - Download Dia from the official website (http://dia-installer.de/)
    - Install Dia on your computer
    - Open Dia:
    - Launch the Dia application.
2. Create New Diagram:
    - Go to File -> New Diagram.
    - Select Flowchart as the diagram type.
3. Add Shapes:
    - Use the shape tools (rectangles, ellipses, etc.) to create wireframes for each screen.
        - For example:
            - Home Page: Rectangle
            - Product Categories: Rectangle
            - Product Listings: Rectangle
            - Product Details: Rectangle
            - Cart: Rectangle

        - Checkout: Rectangle
        - Order Confirmation: Rectangle
        - Order History: Rectangle
4. Connect Shapes:
    - Use the line tool to connect shapes, representing the user flows.
        - For example:
            - Home Page -> Product Categories
            - Product Categories -> Product Listings
            - Product Listings -> Product Details
            - Product Details -> Cart
            - Cart -> Checkout
            - Checkout -> Order Confirmation
            - Order Confirmation -> Order History
5. Label Shapes:
    - Double-click on each shape to add labels.
        - For example:
            - Label the rectangle as "Home Page", "Categories", "Product Listings", "Product Details", "Cart", "Checkout", "Order Confirmation", "Order History".
6. Save the Diagram:
    - Go to File -> Save As.
    - Save the diagram with a meaningful name, such as "Online Shopping App User Flows".

OUTPUT:



RESULT:

thus, the task analysis for an app, such as online shopping, document user flows, and create corresponding wireframes using Dia is performed.

Excercise 5a

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

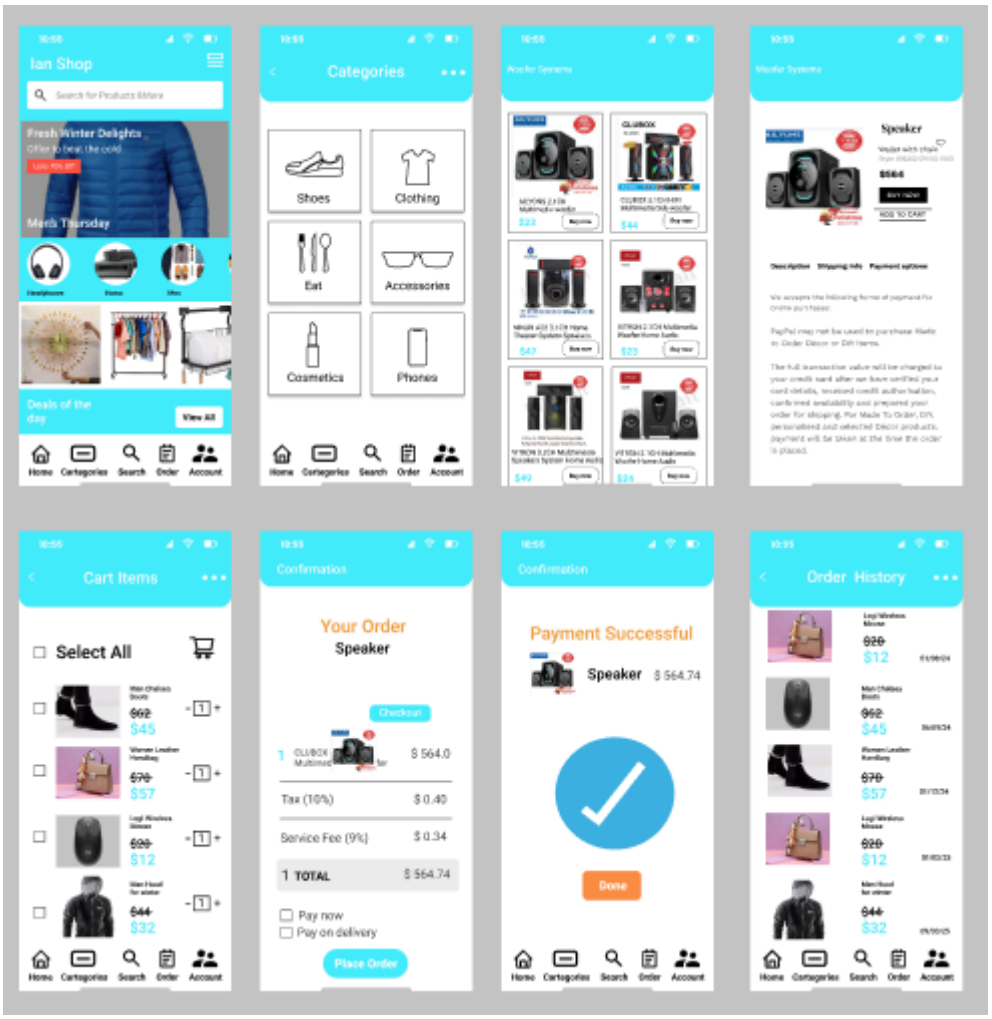
1. **Develop Interactive Prototypes:**
    - Convert wireframes into interactive prototypes by adding interactions and transitions.
    - Use dynamic panels to create interactive elements (e.g., carousels, pop-ups).
2. **Test and Iterate:**
    - Preview the prototype using the Preview button.
    -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

Thus, the lifecycle stages of UI design via the RAD model and develope a small interactive interface employing Axure RP is implemented.

Excercise 5b

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.

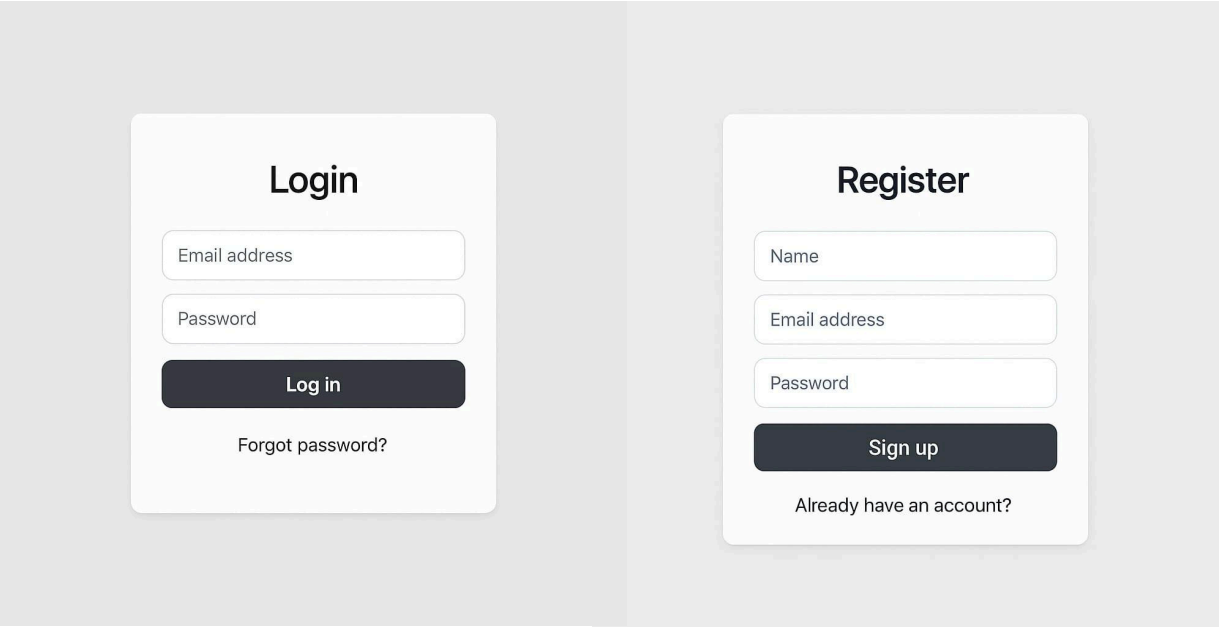
Output in OpenProj:

- 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:

Thus, a small interactive interface using OpenProj was developed successfully.

Excercise 6

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).

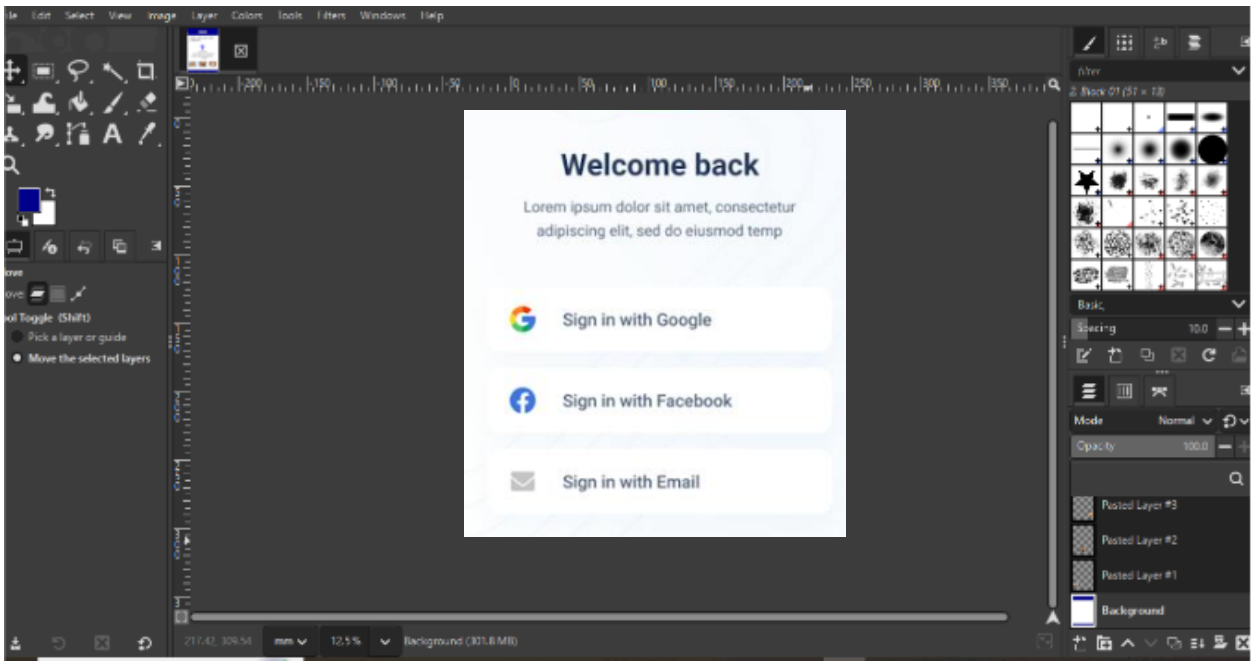
Step 5: Collect User Feedback

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:

Thus, ifferent app layouts and color schemes and evaluate user feedback on aesthetics and usability using GIMP is designed.

Excercise 7a

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.

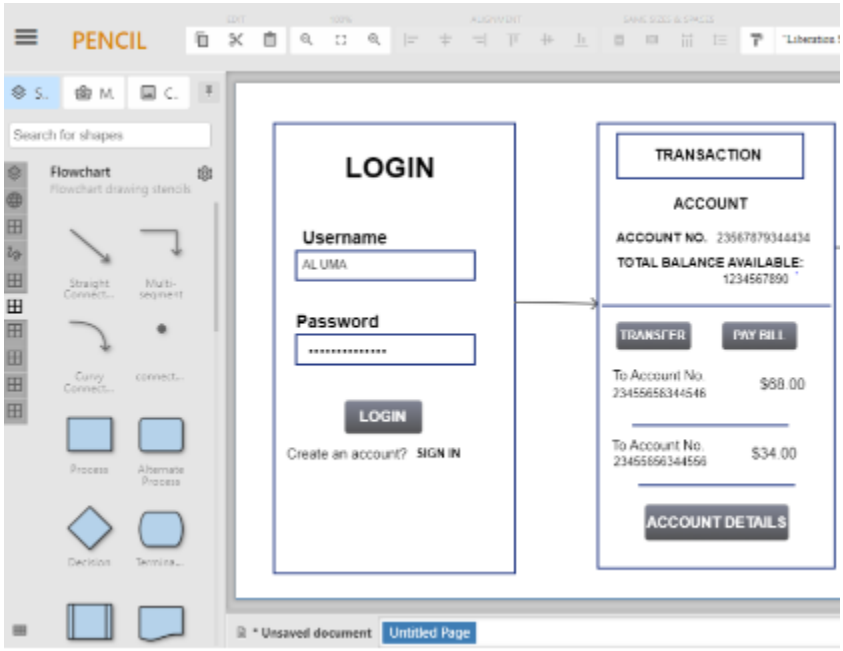
Step 2: Convert Paper Prototypes to Digital Wireframes Using Pencil Project

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:

Thus, low-fidelity paper prototypes for a banking app and convert them into digital wireframes with Pencil Project is successfully designed.

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:

RESULT:

Thus, low-fidelity paper prototypes for a banking app and digitize them into wireframes using Inkscape is designed.

Excercise 8a

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

AIM:

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

PROCEDURE:

Tool Link: https://balsamiq.com/

Step 1: Define the User Flow

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

Step 2: Create Storyboards Using Balsamiq

1. **Install Balsamiq**:
    - Download and install Balsamiq from the https://balsamiq.com/ website.
2. **Create a New Project**:
    - Open Balsamiq and create a new project.
3. **Add Wireframe Screens**:

    - Use the “+” button to add new wireframe screens for each key screen in your app.
4. **Design Each Screen**:
    - Use Balsamiq's components to design the UI for each screen.
    - Include basic elements like buttons, text fields, and images.
5. **Organize the Flow**:
    - Arrange the screens in the order users will navigate through them.
    - Connect the screens with arrows to represent user actions.

Example Screens for Food Delivery App

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

Example Output

Here's how the wireframes might look:

Home Screen

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

Menu Screen

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

Cart Screen

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

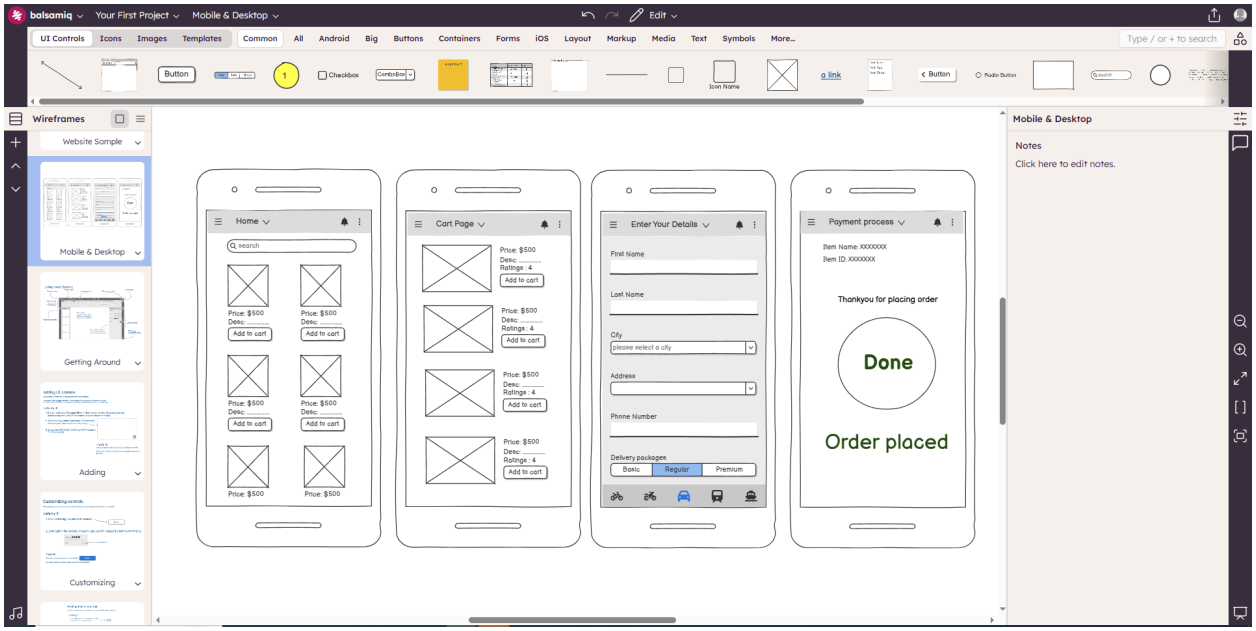
Checkout Screen

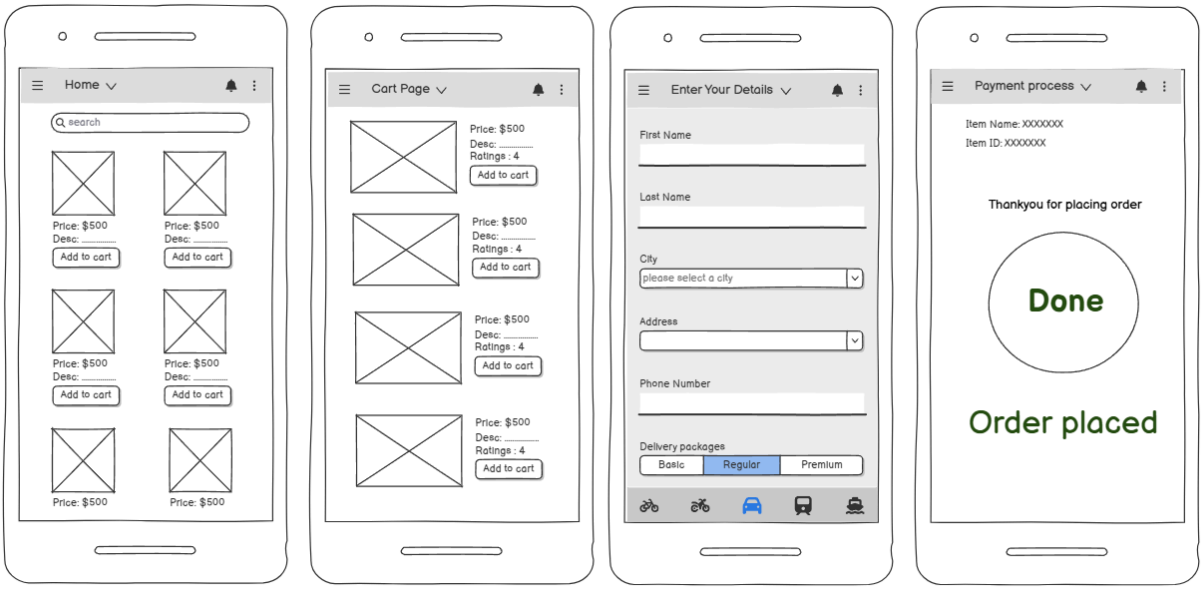
- **Delivery Address Form**: Users enter their delivery address.
- **Payment Options**: Choose between different payment methods.
- **Place Order Button**: Finalize the order.

Order Confirmation Screen

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

OUTPUT:





RESULT:

Thus, user flow for a mobile app, such as a food delivery app, using Balsamiq is designed successfully.

Excercise 8b

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).

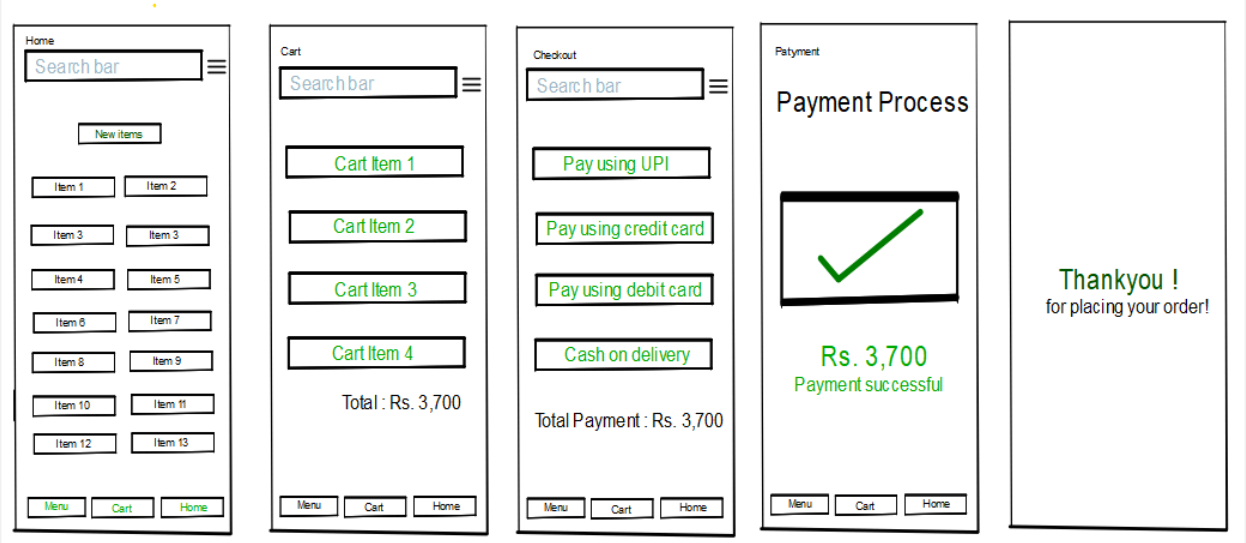
Step 2: Create Storyboards Using OpenBoard

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:

Thus, user flow for a mobile app is designed using OpenBoard.

Excercise 9

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

AIM:

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

PROCEDURE:

Step 1: Setting Up the HTML Form

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

html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <form id="myForm">
            <label for="email">Email:</label>
```

```
            <input type="email" id="email" name="email" required>
            <span id="emailError" class="error"></span>

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

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

Step 2: Styling the Form with CSS

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

css

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

.container {
```

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

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

label {
    margin-bottom: 5px;
}

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

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

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

.error {
    color: red;
```

```
    font-size: 0.875em;
}
```

Step 3: Adding JavaScript for Validation

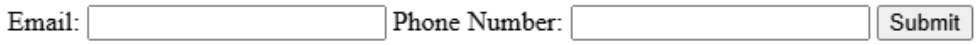
Finally, add JavaScript to validate the input fields using Validator.js and display error messages.

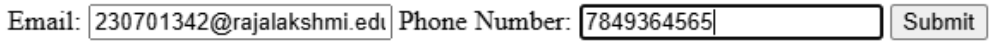
javascript

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

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

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

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

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

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

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

```
        console.log('Phone:', phone);
    }
});
```

OUTPUT:





RESULT:

Thus, the output UI is implemented successfully.

Excercise 10

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

AIM:

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

PROCEDURE:

Step 1: Set Up Your HTML File

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

```
html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Inventory Management Visualization</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 50px;
        }
        canvas {
            margin: 20px auto;
```

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

Step 2: Create the JavaScript File for Charts

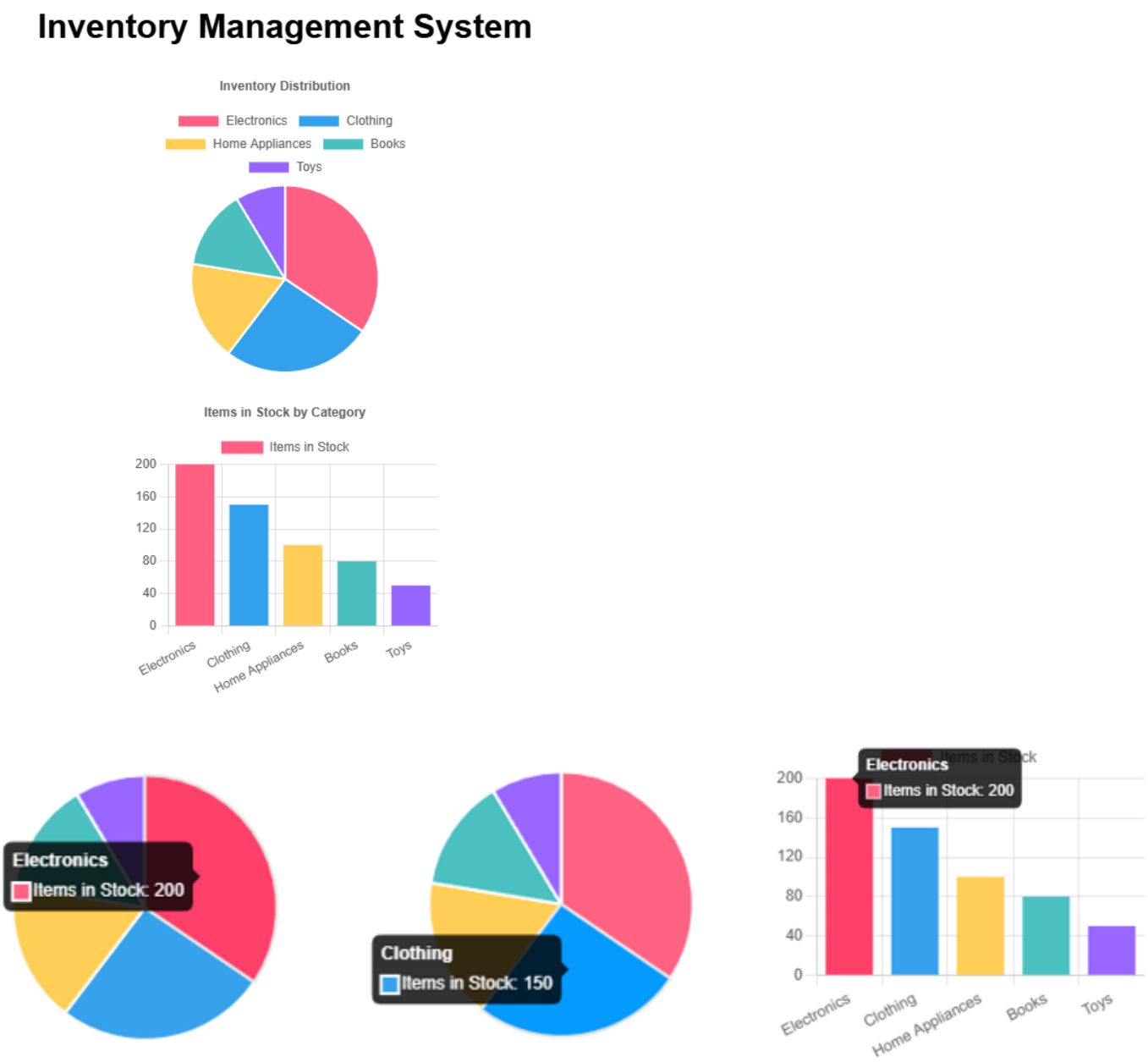
Next, create a JavaScript file (script.js) to handle the data visualization logic.

javascript

```
// script.js

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

```
};

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

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

OUTPUT:



RESULT:

Thus, data visualizations, such as pie charts and bar graphs, for an inventory management system using JavaScript is executed successfully.