IT Management & Audits

Practical Lab Manual

Mobile UX Prototyping Toolkit

Practical P09

Learning Domain

UX Design for FinTech Applications

Course Learning Outcomes

CLO09: Design user experiences for mobile financial applications

Unit

Unit VI: Digital Transformation & UX Design

Time Allocation: 3 hours

Learning Mode: Hands-on (75%) + Theory (25%)

Difficulty Level: Intermediate

Mobile UX Prototyping Toolkit

Practical P09

Quick Reference

Practical Code	P09
Practical Name	Mobile UX Prototyping Toolkit
Slot	T/P-9
Duration	3 hours
CLO Mapping	CLO09
Unit	Unit VI: Digital Transformation & UX Design
Delivery Mode	Hands-on Lab
Target Audience	Intermediate Level
India Integration	HIGH
Screenshot Count	5 Required

Prerequisites

- Basic HTML and CSS knowledge (tags, selectors, properties)
- Understanding of how mobile applications work
- Familiarity with UX design concepts (usability, user flows)
- Web browser with developer tools (Chrome or Firefox recommended)
- Text editor or IDE (VS Code recommended)

Tools Required

Tool	Version	Free	Notes
Web Browser	Latest	✓	Chrome or Firefox
Text Editor / VS Code	Latest	✓	For editing HTML/CSS/JS
HTML/CSS/JavaScript	-	✓	Core web technologies
Chrome DevTools	Built-in	✓	Mobile emulation, inspection
Git	Latest	✓	For cloning the repository

Learning Objectives

- ✓ Understand UX design fundamentals and their application to mobile financial apps
- ✓ Navigate and inspect a multi-screen mobile banking prototype built with HTML/CSS/JS
- ✓ Analyze a design system comprising CSS variables, tokens, and reusable components
- ✓ Evaluate mobile UX against accessibility standards (WCAG 2.1 AA compliance)
- ✓ Customize design system variables to create new visual themes
- ✓ Build a new prototype screen (UPI Payment) integrated into the navigation flow
- ✓ Apply India-specific UX guidelines for financial applications (NPCI, RBI)

What You Will Learn

By the end of this practical, you will:

1. Understand the structure of a mobile-first design system (tokens, components, patterns)
2. Use Chrome DevTools mobile emulation to preview responsive prototypes
3. Inspect and interpret CSS custom properties (design tokens) for colors, typography, and spacing
4. Navigate a 10-screen mobile banking prototype and understand user flow patterns
5. Perform accessibility evaluation including contrast ratios, touch targets, and WCAG compliance
6. Modify design tokens to apply a new visual theme across all prototype screens
7. Create a new UPI Payment screen with QR scanner UI, UPI ID input, and payment confirmation

Real-World Application

Mobile UX design is critical for India's rapidly growing FinTech ecosystem. Applications like **Google Pay**, **PhonePe**, and **Paytm** process billions of UPI transactions monthly, and their success depends heavily on intuitive, accessible user interfaces. The **National Payments Corporation of India (NPCI)** publishes UPI interface guidelines that all payment apps must follow. The **Reserve Bank of India (RBI)** mandates accessibility requirements for banking applications, ensuring digital financial services are usable by all citizens including those with disabilities.

Hands-On Procedure

Part A: Setup

Step 1: Clone the Mobile UX Toolkit Repository

Objective: Download the mobile UX prototyping toolkit and understand its project structure.

Instructions:

1. Open your terminal and navigate to your working directory
2. Clone the mobile-ux-toolkit repository
3. Explore the project structure

Code/Command:

```
# Clone the repository
git clone <repository-url> mobile-ux-toolkit
cd mobile-ux-toolkit

# View the project structure
ls -la
# Expected: prototype/  docs/

# Explore the prototype directory
ls prototype/
# Expected: index.html  screens/  css/  js/

# View all screen files (10 HTML screens)
ls prototype/screens/

# View CSS and JS files
ls prototype/css/
# Expected: design-system.css  components.css
ls prototype/js/
# Expected: prototype.js
```

Clone and Explore the Repository

Expected Output

Project directory contains:
`prototype/index.html` – Entry point for the mobile banking prototype
`prototype/screens/` – 10 HTML screen files (login, dashboard, transfer, etc.)
`prototype/css/design-system.css` – Design tokens and system variables
`prototype/css/components.css` – Reusable UI component styles
`prototype/js/prototype.js` – Navigation logic and interactions
`docs/` – Reference documentation

If you do not have Git installed, download the repository as a ZIP file and extract it manually. The key requirement is having the `prototype/` directory with all its contents.

Step 2: Open the Prototype and Enable Mobile Emulation

Objective: Launch the mobile banking prototype in the browser and configure DevTools for mobile-first viewing.
Instructions:

1. Open `prototype/index.html` in Chrome or Firefox
2. Open Chrome DevTools: `Ctrl+Shift+I` (Windows/Linux) or `Cmd+Option+I` (Mac)
3. Click "Toggle Device Toolbar" or press `Ctrl+Shift+M`
4. Select a mobile device preset (e.g., iPhone 12 Pro, Samsung Galaxy S21)
5. Navigate through the prototype screens to get an overview

```
# Open in default browser (Windows)
start prototype/index.html

# Alternative: Use VS Code Live Server extension
# Right-click index.html -> "Open with Live Server"
```

Open Prototype in Browser

Expected Output

The prototype opens showing a mobile banking application.
With mobile emulation enabled:

```
- Device: iPhone 12 Pro (390 x 844)
- Bottom navigation bar: Home, Transfer, Payments, History, More
- Login screen or dashboard displayed as entry point
```

Always test mobile prototypes using DevTools mobile emulation rather than simply resizing the browser window. Emulation accurately simulates mobile viewport dimensions, pixel density, and touch events.

Screenshot 1

What to paste: The prototype `index.html` page displayed in Chrome DevTools mobile emulation mode, showing the device toolbar enabled with a mobile device selected and the prototype rendering within the mobile viewport frame.

Paste your screenshot here

Part B: Exploring the Design System

Step 3: Review the Design System CSS Variables

Objective: Understand the design token architecture in `design-system.css` — color palette, typography scale, spacing system, and breakpoints.
Instructions:

1. Open `prototype/css/design-system.css` in your text editor
2. Identify CSS custom properties in the `:root` selector
3. Categorize tokens: colors, typography, spacing, breakpoints
4. In DevTools Elements panel, inspect the `<html>` element to view computed custom properties

```
# View the design system CSS file
cat prototype/css/design-system.css

# Expected :root variables include:
# --color-primary: #1A73E8;
# --color-secondary: #34A853;
# --color-error: #EA4335;
# --color-background: #FFFFFF;
# --color-text-primary: #202124;
# --font-size-xs: 0.75rem;    /* 12px */
# --font-size-md: 1rem;       /* 16px */
# --font-size-2xl: 2rem;      /* 32px */
# --space-sm: 8px;
# --space-md: 16px;
# --space-lg: 24px;
# --radius-md: 8px;
# --breakpoint-sm: 320px;
```

Design System CSS Structure

Expected Output

The `design-system.css` file contains:
`- 10+ color tokens` defining the application palette
`- 6+ typography size tokens` following a modular scale
`- 6+ spacing tokens` based on an 8px grid system
`- Border radius and breakpoint tokens`
All defined as CSS custom properties on the `:root` pseudo-element.

Design tokens are the foundation of a design system. They create a single source of truth for visual properties. When you change `-color-primary`, every component that references it updates automatically.

Step 4: Review the Component Library

Objective: Examine the reusable UI components defined in `components.css` and inspect them in the browser.

Instructions:

1. Open `prototype/css/components.css` in your text editor
2. Identify component categories: Buttons (primary, secondary, ghost), Cards, Input Fields, Lists, Headers, Bottom Navigation Bar
3. In Chrome DevTools, select an element and inspect its applied styles
4. Note how components reference design tokens via `var(-token-name)`
5. Observe hover, active, and disabled states for interactive elements

```
# View the components CSS file
cat prototype/css/components.css

# Key classes:
# .btn-primary   - Primary action button (min-height: 44px)
# .btn-secondary - Outlined button variant
# .btn-ghost     - Text-only button
# .card          - Content container with shadow
# .input-field   - Form input styling
# .list-item     - List row with icon and text
# .header        - Top app bar
# .bottom-nav    - Bottom tab navigation bar
```

Key Component Classes

Expected Output

Components reference design tokens consistently:
`.btn-primary` uses `var(-color-primary)` for background
`.card` uses `var(-radius-lg)` for border radius
`.input-field` uses `var(-font-size-md)` for text size
All buttons have `min-height: 44px` for touch target compliance.

Buttons have a minimum height of 44px. This is a critical accessibility requirement — both Apple's Human Interface Guidelines and WCAG 2.1 specify that touch targets must be at least 44x44 CSS pixels.

Part C: Screen Navigation & Interaction

Step 5: Navigate All 10 Prototype Screens

Objective: Walk through every screen in the prototype, documenting navigation patterns and interaction design.

The 10 Prototype Screens:

#	Screen Name	Key Elements
1	Login / Biometric	Username/password, biometric toggle, “Remember me”
2	Dashboard	Balance card, recent transactions, quick actions
3	Transfer	Beneficiary selection, amount input, confirmation
4	Payments	Bill pay categories, biller search, amount entry
5	History	Transaction list, search bar, date/category filters
6	Profile	User info, settings toggles, security options
7	Notifications	Read/unread states, timestamps, action items
8	Cards Management	Card display, freeze/unfreeze, spending limits
9	Support / Help	FAQ accordion, chat support, call support
10	Onboarding	Welcome slides, feature highlights, get started

Instructions: Navigate all 10 screens. For each screen, note: purpose, key UI elements, navigation entry/exit points. Identify patterns: bottom tab bar, stack navigation, modal overlays.

```
# List all screen HTML files
ls -la prototype/screens/

# Check navigation links across screens
grep -r "href=" prototype/screens/ | head -20

# View navigation logic
cat prototype/js/prototype.js
```

Explore Screen Files

Apply the “3-tap rule”: users should reach any major function within 3 taps from the home screen. Count the taps required for sending money or checking transaction history.

Screenshot 2

What to paste: The Dashboard screen displayed in mobile emulation view, showing the account balance card and recent transactions list with the bottom navigation bar visible.

Paste your screenshot here

Step 6: Evaluate UX Against Accessibility Checklist

Objective: Perform an accessibility audit checking WCAG 2.1 AA compliance.
Accessibility Evaluation Checklist:

Criterion	Standard	Pass?	Notes
Color contrast (normal text)	4.5:1 min	☐	
Color contrast (large text)	3:1 min	☐	
Touch target size	44x44px min	☐	
Minimum font size	12px min	☐	
Images have alt text	Required	☐	
Form inputs have labels	Required	☐	
Focus indicators visible	Required	☐	
ARIA roles where needed	As needed	☐	
Logical heading hierarchy	h1→h2→h3	☐	
No info by color alone	Required	☐	

Instructions: Use DevTools to check contrast ratios (click color swatches in Styles panel). Run Lighthouse accessibility audit: DevTools → Lighthouse → Accessibility → Run. Document issues found.

```
# In Chrome DevTools Console, check touch target sizes:
# document.querySelectorAll('button, a, input').forEach(el => {
#   const rect = el.getBoundingClientRect();
#   if (rect.width < 44 || rect.height < 44) {
#     console.warn('Small target:', el, rect.width, rect.height);
#   }
# });
```

Check Touch Targets via Console

The RBI requires banking applications to meet accessibility standards. India's Rights of Persons with Disabilities Act, 2016 and Digital India accessibility guidelines reinforce these requirements for all financial digital services.

Screenshot 3

What to paste: Chrome DevTools showing the design system CSS variables — either the Elements panel with `:root` styles expanded showing CSS custom properties, or the Styles panel with `design-system.css` variables visible.

Paste your screenshot here

Part D: Customization

Step 7: Customize the Design System Theme

Objective: Modify CSS variables to create a new visual theme for an Indian bank, demonstrating how design tokens enable rapid theming.
Instructions:

1. Open `prototype/css/design-system.css`
2. Change the primary color for an Indian bank theme (e.g., SBI blue: `#1C3F94`, BOB orange: `#F47920`, PNB maroon: `#6B0F1A`)
3. Optionally modify typography and spacing values
4. Save and refresh — observe changes propagate across all screens

```
# Example: SBI-inspired theme modifications
# Edit prototype/css/design-system.css

# Change primary color:
# --color-primary: #1C3F94;      (was #1A73E8)
# --color-primary-dark: #0D2259;
# --color-primary-light: #C8D6F0;
# --color-secondary: #D4A843;

# Quick edit from terminal (Linux/Mac):
sed -i 's/--color-primary:␣#1A73E8/--color-primary:␣#1C3F94/' \
    prototype/css/design-system.css
```

Customize Design Tokens

Expected Output

After modifying tokens and refreshing:
```
- All buttons, links, and active states reflect the new color
- Header and navigation bar update automatically
- Changes apply across all 10 screens without editing components
```
This demonstrates design tokens: one change, universal effect.

When creating themes for Indian banks, consider cultural color associations. Blue conveys trust (SBI, HDFC). Green suggests growth. Saffron connects to national identity. Always verify WCAG-compliant contrast ratios.

Step 8: Create a New UPI Payment Screen

Objective: Build a “UPI Payment” screen with QR scanner UI, UPI ID input, amount field, and confirmation, then integrate it into the navigation.

Required UI Elements: Header with back button, QR Code scanner placeholder, “OR” divider, UPI ID input (`name@bankhandle`), Amount field with INR symbol, Remarks field, “Pay Now” button, Bottom navigation bar.

```html
<!-- prototype/screens/upi-payment.html -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
  <title>UPI Payment</title>
  <link rel="stylesheet" href="../css/design-system.css">
  <link rel="stylesheet" href="../css/components.css">
</head>
<body>
  <header class="header">
    <button class="btn-ghost"
            aria-label="Go back">&#8592; Back</button>
    <h1>UPI Payment</h1>
  </header>
  <main class="screen-content">
    <div class="card qr-scanner-area"
         role="button" aria-label="Scan QR Code">
      <p>Tap to Scan QR Code</p>
    </div>
    <div class="divider"><span>OR enter UPI ID</span></div>
    <div class="form-group">
      <label for="upi-id">UPI ID</label>
      <input type="text" id="upi-id" class="input-field"
             placeholder="name@bankhandle">
    </div>
    <div class="form-group">
      <label for="amount">Amount (INR)</label>
      <input type="number" id="amount" class="input-field"
             placeholder="0.00">
    </div>
    <div class="form-group">
      <label for="remarks">Remarks (optional)</label>
      <input type="text" id="remarks" class="input-field"
             placeholder="Add a note">
    </div>
    <button class="btn-primary full-width">Pay Now</button>
  </main>
  <nav class="bottom-nav" aria-label="Main navigation">
    <a href="dashboard.html" class="nav-item">Home</a>
    <a href="transfer.html" class="nav-item">Transfer</a>
    <a href="upi-payment.html"
       class="nav-item active">UPI</a>
    <a href="history.html" class="nav-item">History</a>
    <a href="profile.html" class="nav-item">More</a>
  </nav>
  <script src="../js/prototype.js"></script>
</body>
</html>
```

Create UPI Payment Screen

Expected Output

The UPI Payment screen renders in mobile emulation showing:

```
- Header with "UPI Payment" title and back navigation
- QR code scanner placeholder area
- UPI ID input with placeholder "name@bankhandle"
- Amount field with INR currency context
- "Pay Now" primary action button (full width)
- Bottom navigation with UPI tab active
```

Follow NPCI's UPI interface guidelines: the UPI ID format is `name@psp` (e.g., `john@okaxis`). Common PSP handles: `@okaxis` (Google Pay), `@ybl` (PhonePe), `@paytm` (Paytm).

Screenshot 4

What to paste: Accessibility audit results — either Chrome Lighthouse accessibility score and findings, or your completed manual accessibility checklist with pass/fail markings.

Paste your screenshot here

Screenshot 5

What to paste: Your customized prototype screen in mobile emulation — either the new UPI Payment screen from Step 8 or the prototype with your modified design system theme from Step 7.

Paste your screenshot here

Conceptual Background

UX Design Fundamentals

User Experience (UX) design encompasses the entire journey a user takes when interacting with a product. Two foundational concepts are **mental models** (users' preconceived expectations based on prior experience) and **affordances** (visual cues suggesting how an element can be used).

Nielsen's 10 Usability Heuristics

1. **Visibility of System Status:** Timely feedback (loading spinners during payments)
2. **Match Between System and Real World:** Familiar language ("Send Money" not "Initiate Fund Transfer")
3. **User Control and Freedom:** Undo/cancel options for pending transactions
4. **Consistency and Standards:** Follow platform conventions (bottom navigation)
5. **Error Prevention:** Confirmation before irreversible actions (large transfers)
6. **Recognition Rather Than Recall:** Show recent payees instead of requiring memorization
7. **Flexibility and Efficiency:** QR scan for speed, manual UPI entry as fallback
8. **Aesthetic and Minimalist Design:** Remove unnecessary visual clutter
9. **Help Users Recover from Errors:** Clear messages ("Invalid UPI ID format")
10. **Help and Documentation:** Searchable, task-focused help content

Mobile-First Design and Design Systems

Mobile-first design means designing for the smallest screen first. Key principles: **Content Priority** (limited space forces prioritization), **Touch-First Interactions** (larger tap targets, minimal typing), **Progressive Disclosure** (essential info first, details on demand), and **One-Handed Operation** (frequent controls within thumb reach).

A **design system** has four layers: **Design Tokens** (primitive values: colors, fonts, spacing), **Components** (reusable blocks: buttons, cards, inputs), **Patterns** (component combinations: login forms, payment flows), and **Templates** (page layouts: dashboard, settings).

Financial App UX and Accessibility

Financial UX requires: **Trust Indicators** (logos, security badges), **Error Prevention** (confirmation before irreversible transfers), **Clear Feedback** (unambiguous success/pending/failed status), and **Session Security** (auto-lock, biometric re-auth).

WCAG 2.1 Level AA requires: **Perceivable** (4.5:1 contrast), **Operable** (44px touch targets), **Understandable** (clear error messages), **Robust** (semantic HTML, ARIA labels). Key ARIA attributes: `aria-label`, `aria-live`, `role`, `aria-hidden`.

India Context: UPI and Digital Financial UX

NPCI UPI Guidelines: Standard UPI ID format (`username@psp`), Bharat QR interoperability, transaction limits up to INR 1,00,000, mandatory confirmation before PIN entry. **RBI Requirements:** Multi-language support, screen reader compatibility, simple navigation for seniors, biometric alternatives. **Digital India:** WCAG 2.0 AA compliance, assistive technology support, mobile-responsive baseline.

User Personas: Priya (28, urban, daily UPI), Ramesh (55, business owner, QR payments, Hindi), Suresh (65, retired, large text, accessibility). **Real-World UX:** Google Pay (minimalist, person-centric), PhonePe (feature-rich, regional languages), Paytm (super-app, mini-apps). All share: bottom navigation, biometric auth, recent contacts, prominent QR scanning.

Assessment & Deliverables

Assessment Questions

Q1. List and briefly explain any four of Nielsen's 10 usability heuristics with mobile banking examples.

Q2. What are design tokens and why are they important? How do CSS custom properties enable token management?

Q3. Explain the difference between a component and a pattern in a design system. Give two examples of each.

Q4. What is the minimum touch target size per WCAG 2.1 and why? What problems occur when targets are too small?

Q5. Describe three UX best practices specific to financial applications and why each matters for monetary transactions.

Q6. What is progressive disclosure? Give an example from the dashboard screen.

Q7. Explain the NPCI UPI ID format and PSP handles. Why is QR scanning important as an alternative to manual entry?

Q8. Compare Google Pay and PhonePe UX approaches. Identify one strength and one weakness of each.

Deliverables Checklist

Item	**Description**	**Type**	**Status**
Screenshot 1	Prototype in mobile emulation view	Paste	☐
Screenshot 2	Dashboard with balance and transactions	Paste	☐
Screenshot 3	Design system CSS variables in DevTools	Paste	☐
Screenshot 4	Accessibility audit results	Paste	☐
Screenshot 5	Customized screen (UPI or theme)	Paste	☐
HTML File	UPI Payment screen code	Paste	☐
CSS Changes	Design token modifications	Text	☐
Accessibility	Completed checklist with findings	Text	☐

Verification Checklist

- ☐ Repository cloned and prototype directory verified
- ☐ Prototype opened with DevTools mobile emulation enabled
- ☐ Design system CSS variables reviewed and categorized
- ☐ Component library inspected in DevTools
- ☐ All 10 prototype screens navigated and documented
- ☐ Accessibility evaluation completed (contrast, touch targets, ARIA)
- ☐ Design system customized with new theme
- ☐ New UPI Payment screen created with all required elements
- ☐ UPI Payment screen linked into prototype navigation
- ☐ All 5 screenshots captured and pasted
- ☐ Assessment questions answered

Grading Rubric

Criteria	**Description**	**Points**	**Score**
Setup & Navigation	Repo cloned, emulation enabled	10	___/10
Design System Analysis	CSS variables reviewed, categorized	15	___/15
Component Inspection	Components identified in DevTools	10	___/10
Screen Navigation	All 10 screens explored, documented	10	___/10
Accessibility Audit	WCAG checklist completed, issues noted	15	___/15
Theme Customization	Tokens modified, theme applied globally	10	___/10
UPI Screen Creation	New screen with all required elements	15	___/15
Assessment Questions	All 8 questions answered correctly	10	___/10
Documentation	Screenshots and explanations complete	5	___/5
	TOTAL	**100**	___/100

Appendix A: Nielsen's 10 Heuristics Reference

#	Heuristic	Description
1	Visibility of System Status	Keep users informed through timely feedback
2	Match System & Real World	Use familiar language and conventions
3	User Control & Freedom	Provide undo/redo and easy exit from unwanted states
4	Consistency & Standards	Follow platform conventions consistently
5	Error Prevention	Eliminate error-prone conditions or require confirmation
6	Recognition over Recall	Make options visible; minimize memory load
7	Flexibility & Efficiency	Provide accelerators for expert users
8	Aesthetic & Minimalist	Show only relevant information
9	Error Recovery	Plain language errors with suggested solutions
10	Help & Documentation	Searchable, task-focused help as last resort

Appendix B: Accessibility Quick Reference

Criterion	Level	Standard	How to Test
Text Contrast (Normal)	AA	4.5:1	DevTools color picker
Text Contrast (Large)	AA	3:1	Text 18pt+ or 14pt bold
Touch Target Size	AA	44x44px	Inspect element dimensions
Keyboard Navigation	A	All interactive	Tab through elements
Form Labels	A	All inputs	Check `<label>`/`aria-label`
Alt Text	A	All images	Check `alt` attributes
Heading Hierarchy	A	Logical order	Verify h1, h2, h3 sequence

Appendix C: CSS Design Token Reference

Token	Default	Usage
`-color-primary`	`#1A73E8`	Actions, active states, links
`-color-secondary`	`#34A853`	Success, positive amounts
`-color-error`	`#EA4335`	Errors, negative amounts
`-color-warning`	`#FBBC04`	Warnings, pending states
`-color-text-primary`	`#202124`	Body text, headings
`-font-size-md`	1rem (16px)	Body text, inputs
`-font-size-2xl`	2rem (32px)	Hero values (balance)
`-space-sm/md/lg`	8/16/24px	Padding, gaps, sections

Appendix D: Mobile Screen Sizes (India Market)

Device	CSS Resolution	DPR	India Share
iPhone 12/13	390 x 844	3x	High
Samsung Galaxy A series	360 x 800	2x	Very High
Xiaomi Redmi series	360 x 780	2x–3x	Very High

Budget Android devices dominate in India. Always test at 360px width as the baseline.

Appendix E: Troubleshooting Guide

Solutions:

1. Ensure the `<meta name="viewport">` tag is present in `<head>`
2. Clear cache and hard reload: `Ctrl+Shift+R`
3. Try a different device preset in DevTools device toolbar

Solutions:

1. Hard refresh: `Ctrl+Shift+R` (bypasses cache)
2. Check for typos in CSS property names (case-sensitive)
3. Use VS Code Live Server for automatic reload on save

Solutions:

1. Verify file saved as `prototype/screens/upi-payment.html`
2. Update `<a href>` links in other screen files' bottom navigation
3. Update `prototype.js` if it manages navigation programmatically
4. Check browser console for 404 errors on navigation clicks

Appendix F: Resources

→ Nielsen Norman Group: `https://www.nngroup.com`

→ Material Design: `https://m3.material.io`

→ WCAG 2.1: `https://www.w3.org/TR/WCAG21/`

→ NPCI UPI: `https://www.npci.org.in/what-we-do/upi/product-overview`

→ Apple HIG: `https://developer.apple.com/design/human-interface-guidelines`

Tools Used in This Practical

Tool	Purpose	Cost
Chrome DevTools	Mobile emulation, inspection, accessibility audit	Free
VS Code	Code editing with Live Server extension	Free
Lighthouse	Automated accessibility and performance auditing	Free
Git	Repository cloning and version control	Free
HTML/CSS/JS	Core technologies for prototype implementation	Free

—END OF LAB MANUAL—

Document Version: 1.0

IT Management & Audits – Practical Lab Series