

Foundations of UX and UI

Lesson 1: User Experience Design (UXD)
Module 1: Web Applications Development

Why UX Matters

Ever deleted an app because you couldn't figure it out?

Ever abandoned a shopping cart due to confusing checkout?

That's bad UX in action

Your Real Job as a Developer:

Code is just the tool

Your job is solving human problems

Making people's lives easier



Key Takeaway: Code without good UX is like a Ferrari with square wheels

What is User Experience Design?

UXD is the process of creating products that provide meaningful and relevant experiences to users

Think of UXD as the architect's blueprint for digital experiences

You wouldn't design a house without considering how people will live in it

Key Components of UXD (1/2)

1. User Research – Understanding who your users are and what they need
2. Information Architecture (IA) – Organizing content in a logical, findable way
3. Interaction Design (IxD) – Defining how users interact with your product

Key Components of UXD (2/2)

1. Visual Design – Making it look good (purposefully, not just pretty)
2. Usability Testing – Validating that your design actually works for real people

UX vs UI: The Critical Difference

User Interface (UI) - What you see:

Buttons, colors, fonts, images, animations

Think: The paint, wallpaper, furniture

User Experience (UX) - How it works:

Layout, flow, logic, journey

Think: Floor plan, room arrangement, functionality

UX vs UI: Restaurant Analogy

UI (User Interface):

Menu design, plate presentation, décor, lighting

UX (User Experience):

How easily you get seated

How quickly you order

Waiter attentiveness

Whether bathroom is easy to find



Key Takeaway: You can have beautiful UI with terrible UX, or simple UI with amazing UX

User-Centered Design (UCD) Process

1. Research & Discovery
2. Define & Ideate
3. Prototype & Design
4. Test & Iterate



Key Takeaway: Put users at the center of every design decision, not what YOU think looks cool

Stage 1: Research & Discovery

Learning about your users through:

Interviews

Surveys

Analytics

Observation

Why it matters:

Reveals truth beyond assumptions



Key Takeaway: Research is not optional. Skipping research means designing blind

Stage 2: Define & Ideate

Taking research insights and:

- Brainstorming solutions

- Creating personas

- Mapping user journeys

- Defining problems clearly

Key Principle: 'A problem well-defined is half-solved'



Key Takeaway: Transform messy research data into clear design direction

Stage 3: Prototype & Design

Creating representations of your solution:

Low-fidelity: Sketches, wireframes

Medium-fidelity: Interactive wireframes

High-fidelity: Detailed mockups, prototypes

Why it matters:

Test ideas cheaply before coding

Easier to change a sketch than rewrite code



Key Takeaway: Prototype to learn, not to impress. Fail fast and cheap

Stage 4: Test & Iterate

Testing methods:

Usability Testing: Watch users complete tasks

A/B Testing: Compare two versions statistically

Surveys: Gather satisfaction data

Analytics: Track behavior patterns

The cycle: Design → Test → Learn → Improve → Repeat



Key Takeaway: Your assumptions will be wrong. Testing reveals issues before launch

Nielsen's 10 Usability Heuristics

The 'Ten Commandments' of UX Design

General principles for interaction design that guide good UX decisions

Developed by Jakob Nielsen of Nielsen Norman Group

Let's explore the most important ones...

Heuristic #1: Visibility of System Status

Always keep users informed about what's happening

Examples:

- ✓ Upload progress bars with percentage
- ✓ 'Processing...' indicators
- ✓ 'Item added to cart' confirmations
- ✗ Generic loading spinners with no context



Key Takeaway: Never leave users wondering 'Did that work?'

Heuristic #2: Match System and Real World

Use words and concepts familiar to users

Good examples:

Use 'Send email' not 'SMTP relay configuration'

Use 'Schedule campaign' not 'Campaign scheduler daemon'

Use 'Manage audience' not 'Subscriber database mgmt'



Key Takeaway: Speak your users' language, not technical jargon

Heuristic #3: User Control and Freedom

Users need clearly marked 'emergency exits'

Essential features:

↔ Undo/Redo functionality

X Cancel buttons on long operations

← Back navigation that works

 'Save as draft' options

 Trash/Archive instead of permanent delete

Heuristic #4: Consistency and Standards

Users shouldn't wonder if different words mean the same thing

Internal consistency:

- Same button styles throughout app

- Consistent terminology

- Predictable navigation placement

External consistency:



Key Takeaway: Don't make users learn a new language for your app

Heuristic #5: Error Prevention

Better than good error messages is preventing errors

Prevention strategies:

Constraints: Disable invalid options

Validation: Check input in real-time

Confirmations: 'Are you sure?' for destructive actions

Smart Defaults: Pre-select common choices

Error Prevention Hierarchy

1. Best: Prevent the error from happening
2. Good: Catch it before submission
3. Okay: Show helpful error after submission
4. Bad: Generic 'Error occurred' message

Heuristic #6: Recognition Over Recall

Make objects, actions, and options visible

Don't make users memorize information:

- ✓ Show recent searches
- ✓ Display breadcrumbs (You are here)
- ✓ Auto-complete suggestions
- ✗ Require memorizing commands



Key Takeaway: Recognition is easier than recall - reduce memory burden

Heuristic #7: Flexibility and Efficiency

Accelerate interaction for experts while remaining accessible

Techniques:

-  Keyboard shortcuts
-  Favorites/Bookmarks
-  Recently used items
-  Customizable interfaces

Serve novices AND experts simultaneously

Heuristic #8: Aesthetic and Minimalist Design

Design shouldn't contain irrelevant information

Design philosophy:

Every element competes for attention

More content = harder to find what matters

Simple ≠ Simplistic

Remove, reduce, hide (in that order)



Key Takeaway: Perfection: when there's nothing left to take away

Heuristic #9: Help Users Recover from Errors

Error messages should be helpful, not cryptic

Good error messages include:

- What happened (in plain language)

- Why it happened (if known)

- How to fix it (concrete steps)

- Positive tone (helpful, not blaming)

Heuristic #10: Help and Documentation

Even with great design, users may need help

Best practices:

-  Contextual help: Right where users need it
-  Searchable: Easy to find answers
-  Task-oriented: 'How do I...' format
-  Concise: Get to the point quickly

Information Architecture (IA)

The practice of organizing and structuring content in a logical and intuitive way

The Library Analogy:

Don't throw books randomly on shelves

Group by category, then by topic

Then alphabetically

That's IA

Core IA Principles (1/2)

1. Organization

How content is grouped and categorized

Hierarchical, sequential, or matrix

2. Labeling

What you call things

Navigation items, buttons, sections

Core IA Principles (2/2)

3. Navigation

How users move through information

Global, local, contextual, breadcrumbs

4. Search

How users find specific content

Search functionality, filters, faceted search

Accessibility in UX Design

Designing products everyone can use, including people with disabilities

Why it matters:

Legal: Required by law (ADA, WCAG)

Business: 1 in 4 adults has a disability

Ethical: Right thing to do

Universal: Benefits everyone



Key Takeaway: Curb cuts help wheelchairs AND strollers AND delivery people

The Four POUR Principles (1/2)

1. Perceivable

Information must be presentable in ways users can perceive

Alt text, captions, sufficient color contrast

2. Operable

Interface components must be operable

Keyboard navigation, sufficient time, no seizures

The Four POUR Principles (2/2)

3. Understandable

Information and operation must be understandable

Readable text, predictable navigation, clear errors

4. Robust

Content must work with assistive technologies

Valid HTML, proper ARIA labels, screen readers

User Personas

Fictional characters representing different user types

Typical components:

Demographics: Age, occupation, location

Goals: What they want to achieve

Frustrations: Pain points and challenges

Tech Savviness: Comfort with technology

Behaviors: Usage patterns

Mobile-First Design

Designing for mobile devices first, then scaling up

Why mobile-first:

- 60%+ of web traffic is mobile

- Constraints breed creativity

- Forces prioritization of essential features

- Progressive enhancement approach

Mobile Design Considerations

Touch targets: Minimum 44x44 pixels

Screen real estate: Prioritize ruthlessly

Performance: Slow networks, limited battery

Context: Users on-the-go, distracted

One-handed use: Keep important actions in thumb reach

UX Metrics That Matter

Task Success Metrics:

Task success rate, time on task, error rate

Business Metrics:

Conversion rate, revenue per user, CLV

Satisfaction Metrics:

Net Promoter Score (NPS), CSAT, SUS

Common UX Mistakes to Avoid (1/2)

- X Designing for yourself, not users
- X Skipping research phase
- X Not testing with real users
- X Ignoring accessibility
- X Copying competitors blindly
- X Jumping straight to hi-fi designs

Common UX Mistakes to Avoid (2/2)

- X Inconsistent UI patterns
- X Poor error handling
- X No loading states
- X Unclear navigation
- X Too many form fields
- X Not measuring results

Final Takeaways (1/2)

1. You are not your user

Research before designing

Test your assumptions

Listen to feedback

2. Simplicity is hard

Easy to add features

Hard to remove them

Final Takeaways (2/2)

3. Design for everyone

Accessibility isn't optional

Inclusive design benefits all

4. Iterate, iterate, iterate

First design won't be perfect

Continuous improvement

Data-driven decisions

Your UX Journey Starts Now

As a Developer:

- Consider UX in every line of code

- Ask 'Why?' before 'How?'

- Put yourself in users' shoes

Remember:

- Your goal isn't to build what you think users want

- It's to discover what they need and build that

Questions?

Thank you for your attention!