Introduction to Web Applications

Today's Journey

- What is a Web App?
- Evolution from local to distributed applications
- Why build web apps?
- Examples of web apps in the wild
- How web apps actually load in browsers

What's a Web App?

Let's build our understanding step by step...

Starting with the simplest case

Local App

An application that runs entirely on your machine

- All computation happens locally
- No network communication needed
- Data stored on local disk
- Processing uses local CPU/memory

Local App Demo

Live Demo: `calculator.py`

```
# Simple calculator - runs entirely on your machine
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
result = num1 + num2
print(f"The sum is: {result}")
```

Browser-Based Local App

A special case: local app running in a web browser

- Still runs locally (no server needed)
- Browser provides the runtime environment
- JavaScript executes in browser
- Data can be stored in browser (localStorage)

Browser-Based Demo: Code

```
<!DOCTYPE html>
<html>
<body>
    <h1>Sum Calculator</h1>
    <input type="number" id="num1" placeholder="First number">
    <input type="number" id="num2" placeholder="Second number">
    <button onclick="calculateSum()">Calculate Sum</button>
    <div id="result"></div>

    <script>
        function calculateSum() {
            const num1 = parseFloat(document.getElementById('num1').value);
            const num2 = parseFloat(document.getElementById('num2').value);
            const sum = num1 + num2;
            document.getElementById('result').textContent =
                `The sum is: ${sum}`;
        }
    </script>
</body>
</html>
```

Browser-Based Demo

Live Demo: `calculator.html`

Opens directly in your browser - no server required!

Activity: Open the calculator.html file on your computer

- What happens when you click Calculate?
- Where is the computation happening?

Think-Pair-Share

Question: Is calculator.html a "web app"?

Distributed App

Application that runs across multiple machines

- Client and server are separate
- Machines communicate over network
- Uses a defined **protocol**
- Enables resource sharing and collaboration

Distributed App Demo

Live Demo: `server.py`

Server running on cloud VM waiting for requests

Activity: Send a calculation request from your terminal:

```
echo "Add 2 3" | nc <ip> <port>
```

What response do you get?

Quick Poll

Who successfully connected to the server and got a result?

- What was different about this experience compared to the local calculator?
- Where did the computation happen this time?

HTTP Distributed App

A distributed app using the HyperText Transfer Protocol (HTTP)

- HTTP = standardized protocol for web communication
- Request/response model
- Client sends HTTP requests
- Server sends HTTP responses

Why HTTP Matters

HTTP is special because:

- **Standardized**: Everyone follows the same rules
- **Text-based**: Human-readable messages
- **Stateless**: Each request is independent
- **Universal**: Supported by all browsers

HTTP Demo

Live Demo: `http_server.py`

Server responds to HTTP GET requests

Activity: Send HTTP request using netcat:

```
printf "GET / HTTP/1.1\r\nHost: <ip>\r\n\r\n" | nc <ip> <port>
```

Notice the HTTP format: method, path, headers

Browser as HTTP Client

Browsers are sophisticated HTTP clients!

- Type URL in address bar
- Browser sends HTTP request
- Server processes and responds
- Browser renders the result

User sees the rendered page!

Browser Client Demo

Activity: Open browser and navigate to:

```
http://<server-ip>:<port>/add?num1=5&num2=3
```

The browser:

1. Constructs HTTP GET request
2. Sends it to the server
3. Receives JSON response
4. Displays the result

Definition: Web App

> **A web application is a software program that:**
>
> - Runs in a web browser
> - Is accessed over the internet or network
> - Is hosted on a server
> - Delivered to users through their browser

This is what we'll be building in CSD 412!

Concept Check

Which of our demos were web apps?

- calculator.py?
- calculator.html (opened from file)?
- server.py with netcat client?
- server.py with browser client?

Discuss with your neighbor

Why Build Web Apps?

Let's explore the advantages...

Advantage 1: No Installation Required

Users don't need to:

- Download installers
- Run setup wizards
- Manage updates
- Deal with compatibility issues

Just open a browser!

- Chrome, Firefox, Safari, Edge
- Works on desktop, laptop, tablet, mobile

Advantage 2: Server-Side Processing

User's machine isn't strained

- Heavy computation happens on server
- Data processing on powerful servers
- Machine learning models run server-side
- Video encoding, image processing, etc.

User's device just displays results!

Advantage 3: Access Anywhere

True mobility:

- Work from office desktop
- Continue on laptop at coffee shop
- Check status on phone during commute
- Access from tablet at home

Same application, any device with internet + browser

Advantage 4: Always Up-to-Date

No update prompts for users!

- Updates happen on the server
- All users get new version instantly
- Bug fixes roll out immediately
- New features appear automatically

Developer updates once → Everyone benefits

Advantage 5: Easier UI Development

Cross-platform by default:

- HTML/CSS/JS works everywhere
- One codebase for all platforms
- Rich browser APIs available
- Mature frameworks and libraries

Compare to: Native apps requiring separate iOS, Android, Windows, Mac codebases

Think-Pair-Share

Question: Can you think of a situation where a web app would NOT be the best choice?

Web Apps in Your Life

How many web apps have you used **today**?

Examples: Communication

Gmail

- Email client runs entirely in browser
- No email client software to install
- Access from any device

Slack / Discord

- Team communication
- Real-time messaging
- File sharing

Examples: Productivity

Google Docs

- Word processing in browser
- Real-time collaboration
- Automatic saving to cloud

Microsoft 365

- Excel, PowerPoint online
- Share and collaborate
- Access anywhere

Examples: Entertainment

Netflix / YouTube

- Video streaming
- Recommendations based on viewing
- Playback on any device

Spotify

- Music streaming
- Curated playlists
- Cross-device continuity

Examples: Social & Learning

Twitter/X

- Social media platform
- Real-time updates
- Trending topics

Canvas (Your LMS!)

- Course management
- Assignment submission
- Grade checking

Examples: Simple Web Apps

Even our calculator is a web app!

When served from a web server:

- `calculator.html` becomes accessible via URL
- Server delivers HTML/CSS/JavaScript
- Browser executes the code
- User interacts with application

Simple, but demonstrates core web app concepts!

Activity: Web App Scavenger Hunt

Identify 3 applications you use regularly:

1. Is it a web app or native app?
2. How do you know?
3. What are the advantages of its delivery model?

Be ready to share one example!

Loading a Web App

What actually happens when you visit a web app?

Let's trace the journey...

You Type a URL

```
https://www.example.com
```

Browser needs to find the **IP address** of www.example.com

IP addresses are like postal addresses for computers:

- `192.168.1.1`
- `172.217.14.206`

DNS Lookup

Browser asks DNS Server:

"What's the IP address for www.example.com?"

DNS responds:

"That's 93.184.216.34"

DNS = Domain Name System

- Translates human-readable names to IP addresses
- Distributed database across internet

HTTP Request

Browser sends request to server:

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0...
Accept: text/html...
```

Request includes:

- What you want (/)
- Which host (www.example.com)
- What browser you're using

Server Processes Request

Web server:

1. Receives HTTP request
2. Determines what to send back
3. Might query database
4. Might run application logic
5. Generates HTML response

HTTP Response

Server sends back:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1234

<!DOCTYPE html>
<html>
<head>...</head>
<body>...</body>
</html>
```

Browser Rendering

Browser processes the HTML:

1. Parses HTML structure
2. Loads CSS (styling)
3. Loads JavaScript (interactivity)
4. Loads /notes/images and other resources
5. Renders the page on screen

Each CSS file, JS file, image = another HTTP request!

Think About It

A typical web page might make 50+ HTTP requests:

- 1 for HTML
- 5 for CSS files
- 10 for JavaScript files
- 30+ for /notes/images
- Several for fonts, icons, etc.

Each follows the same request/response pattern!

Key Takeaways

Evolution of applications:

Local → Distributed → HTTP → Browser-based

Web apps = HTTP + Browser + Server

Why web apps:

- No installation
- Access anywhere
- Always current
- Cross-platform

Looking Ahead

This course will teach you to build all the pieces:

- **Client-side:** HTML, CSS, JavaScript, React
- **Server-side:** Node.js, Express, databases
- **Communication:** REST APIs, HTTP protocols
- **Deployment:** Making apps available to the world