To Ponder

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.”

Leslie Lamport
Web Applications: Overview and Architecture

Lecture 1
Road Map in Pictures: Web App

[Image of a web page showing a search for ice cream in Columbus using Yelp, with listings for Cold Stone Creamery and Jeni's Splendid Ice Creams, and a map indicating locations.]
Road Map in Pictures
Road Map in Pictures

Browser ➔ Request ➔ HTTP ➔ Web Server ➔ Response ➔ Browser

- **Rails**
  - Model
  - View
  - Controller
  - Dispatcher
  - Routes

- **HTML**
- **CSS**
- **JavaScript**
- **Ruby**
- **Yelp**
Road Map: Schedule of Topics

- A Language
  - Ruby

- Foundations
  - Version Control, Networking, Regular Expressions

- Static web pages
  - HTML & CSS

- Dynamic web pages
  - JavaScript

- Framework for web applications
  - Rails

- Applied Topics
  - Security, Encodings
Resources

- **Class website**
  - Syllabus (note exam requirement)
  - Handouts, lecture notes, lab assignments
  - Pointers to more resources

- **Piazza**
  - Discussion forum, news, announcements

- **Slack**
  - Group collaboration, messaging, chat

- **Carmen**
  - Grades

- **Face time (not FaceTime™)**
  - Instructor, TA
  - Each other
Technical Content

- **Languages and Technologies**
  - HTTP
  - XML, HTML, CSS, JavaScript
  - Ruby, Ruby on Rails

- **Tools and techniques**
  - Design patterns (MVC)
  - git, linux
  - Regular expressions, unicode, time

- **Advanced topics**
  - Programming languages, networking, cryptography, databases, operating systems
Stability of Content: Concepts

- Conceptual underpinnings will be relevant forever

In this course:
- Single-point of control over change
- Abstraction (vs realization)
- Design patterns
- Regular Expressions (the math part)
- Cryptography (the math part)
- Motivation for version control
- Time-space performance trade-offs
Stability of Content: Technology

- Some technologies have been around a long time, and will likely be relevant for many more years

- Examples in this course:
  - Linux
  - SQL
  - HTTP
  - HTML
  - CSS
  - JavaScript
Stability of Content: Tools

- Some tools come and go
- They are useful for getting things done now, but may not be as relevant or fashionable in 10 years
- Examples in this course
  - Ruby
  - JQuery
  - git
Stability of Content: Framework

- There are many frameworks and libraries for web development
- They come and go so quickly, there is always something new
- Examples:
  - Web frameworks like Rails, Express.js...
  - Ruby gems like Middleman, Nokogiri, Cucumber...
  - JavaScript libraries like Angular, React
  - HTML/CSS libraries like Bootstrap, Baseline, Foundation...
Meta Content: Software Eng.

- Lasting relevance
- Project development in the "real world"
  1. Vague open-ended requirements
  2. Large, complex problems
  3. Teams
Topic 1: Vague Requirements

- Two aspects to engineering:
  - Satisfying the constraints (solving the problem)
  - Optimizing the solution (better, faster, cheaper)

- Must first identify and understand the problem
  - Requirements elicitation

- Recognize tradeoffs
  - Improvement in one aspect at the expense of another
Topic 2: Size and Complexity

- “Programming in the large”
  - Does not all fit in one person’s head or schedule
  - Interfaces, modules, components, classes
- Design
  - Measure twice, cut once
- Process
  - Agile, waterfall, TDD,…
- Documentation
- Testing
Topic 3: Group Work

- Naïve view of CS: Lone wolf hacker
- Reality: large multidisciplinary teams
  - Developers, testers, marketing, HR, management, clients
  - Communication skills are critical
- Many challenges
  - Rely on others
  - Compromises become necessary
  - Personalities
- Many rewards
  - Accomplish more
  - Learn more
In This Course...

- Group work: 4 or 5 people / group
- Multidisciplinary teams
  - I will create cross-cutting technical areas
- Open-ended projects
- Communication skills
  - Presentations to class
Architecture: Desktop App

- User Interface
  - Graphical events (mouse moves, button pushed)

- Application
  - Processing, Calculating

- Data
  - Persistence, Transactions, Triggers
**Client-Server App: 2-Tier**

Where should we cut?

- **User Interface**
  - Ultra-thin client (aka “dumb terminal”) [X11, RDP, character echo from mainframe]
  - UI on client, processing just for display [browser rendering static HTML]
  - Some processing on client [validate form fields before submission]
  - Thick client: connect directly to DB mngr [native look & feel]

- **Data**
  - Cache data on client [responsiveness, less network congestion]
Basic Web App Skeleton: 3-Tier

- User Interface
- Application
- Data

http ➔ HTML, CSS, Javascript

SQL

Languages and Technologies:
- PHP
- Java
- Tomcat
- MySQL
- Microsoft SQL Server
- SQLite
Advantages over Thick Clients

- **Performance**
  - 1 (expensive) network call to app layer results in many calls to data layer
  - Compute-intensive part on faster machine

- **Flexibility**
  - Update app logic without changing client

- **Robustness**
  - Transactions, logging at app level

- **Security**
  - Login, authentication, encryption all better at app level than data level
Web App Skeleton: 4-Tier

User Interface

Presentation Layer

Business Logic

Data
Web App Skeleton: n-Tier...

- User Interface
- Caching
- Presentation Layer
- Workflow
- Business Logic
- Data
Summary

- Technical aspects of course content
  - Many web technologies
- Meta content: Software engineering
  - Vague requirements
  - Large systems
  - Teams
- 2-, 3-, 4-, n-Tier Architectures