

A THREE-COURSE SEQUENCE FOR API-BASED WEB DEVELOPMENT AND DEPLOYMENT

Jeremy Shafer

Assistant Professor of Management Information Systems
jeremy@temple.edu | community.mis.temple.edu/jshafer

David Schuff


Professor of Management Information Systems
schuff@temple.edu | [@dschuff](https://twitter.com/dschuff) | community.mis.temple.edu/dschuff



The task

- Teaching API-based web development
- When it has become very, very complicated
 - React/Node/jQuery/Angular/Axios/Express
- And little space in overall undergrad curriculum
- And other subjects to teach in the major (Cybersecurity, UX)
- Build an application, but not programmers
- DevOps mentality, but not operations





The Environment: Fox School of Business and MIS Department

- Fox School of Business
 - Nine academic departments
 - Largest b-school in Philadelphia
 - 6,500+ students
- MIS Department
 - ~385 majors
 - ~270 minors (2)

Why APIs?

- Modern application development depends on
 - Consuming resources
 - Simplify development
 - Making resources available
 - Internal and external interoperability
 - Data access layers
 - SaaS/API economy
 - Business models built around web-based services
 - It's about architecture and development



Our approach

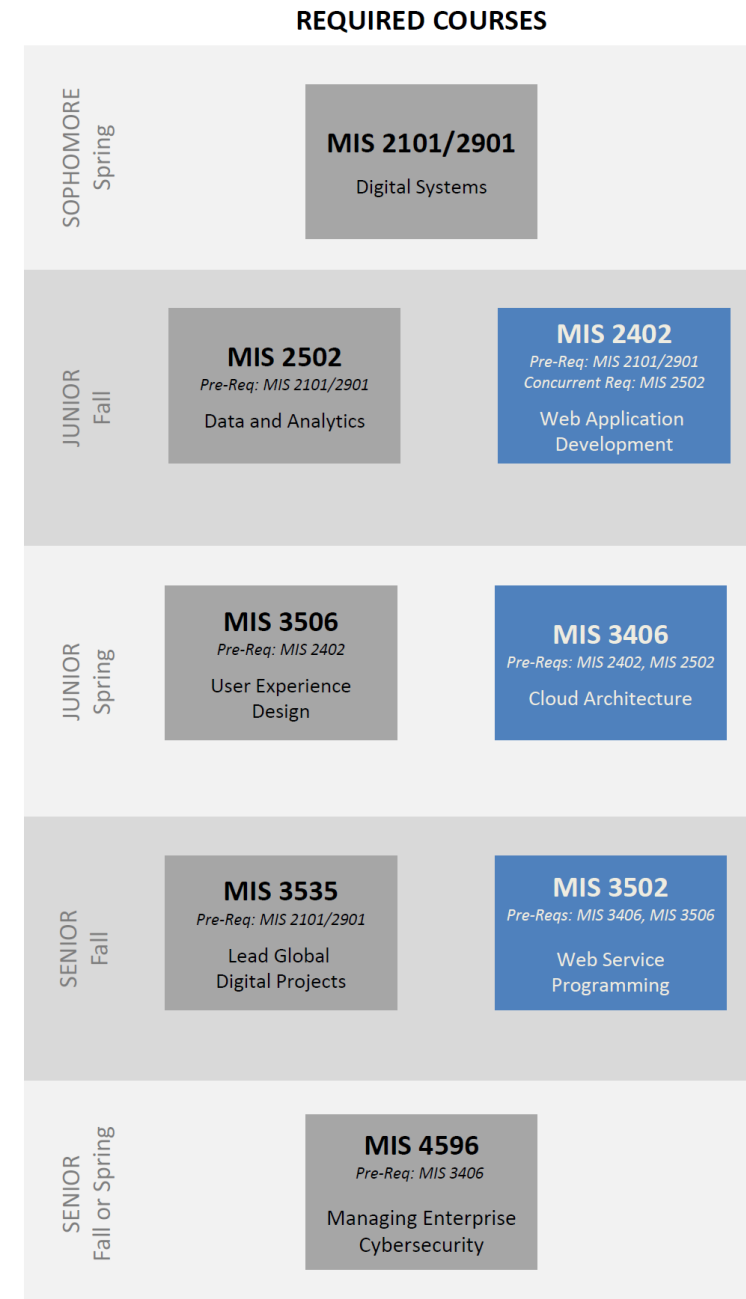
Three course sequence

JavaScript as the language

- One language used across multiple courses
- Ubiquitous in industry

Five unifying principles

1. **Keep it real!** *real tools and products*
2. **Narrow scope, high expectations.** *do a few things well*
3. **Focus on APIs.** *consume and create*
4. **Use frameworks.** *simplify development*
5. **Don't start from scratch.** *too big a lift otherwise*





Web application development (MIS2402)

Objectives

- Apply basic programming principles
- Develop critical thinking, problem-solving skills
- Make API calls using web protocols

Skills

- JavaScript and HTML
- Debugging tools and techniques in VSCode
- Bootstrap and jQuery

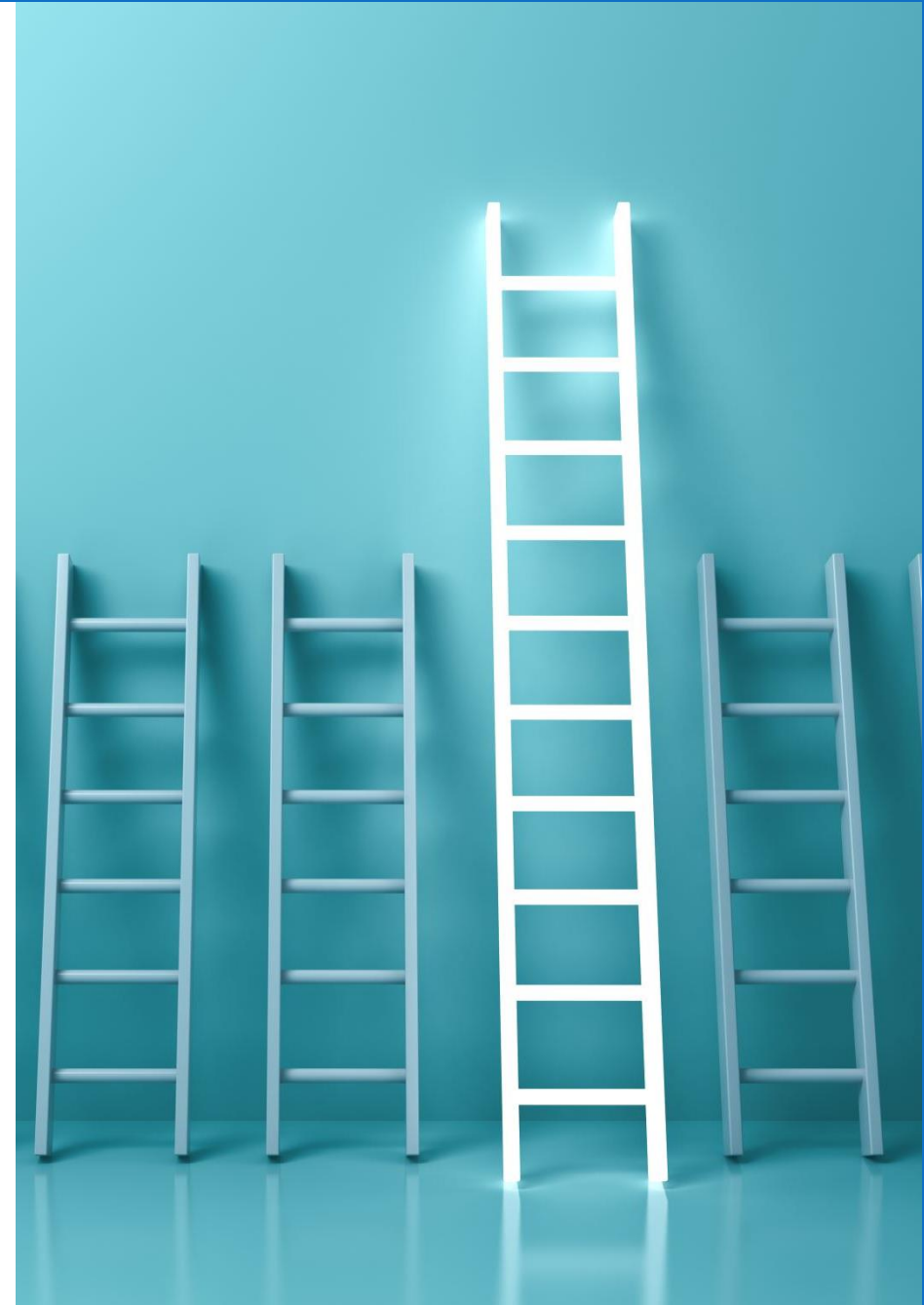
Cloud architecture (MIS3406)

Objectives

- Create a scalable, robust cloud-based application hosting environment
- Create an API – data access layer
- Host the API on the cloud environment

Skills

- AWS (EC2, RDS, Elastic Beanstalk)
- IP-based Networking
- Node.js and Express



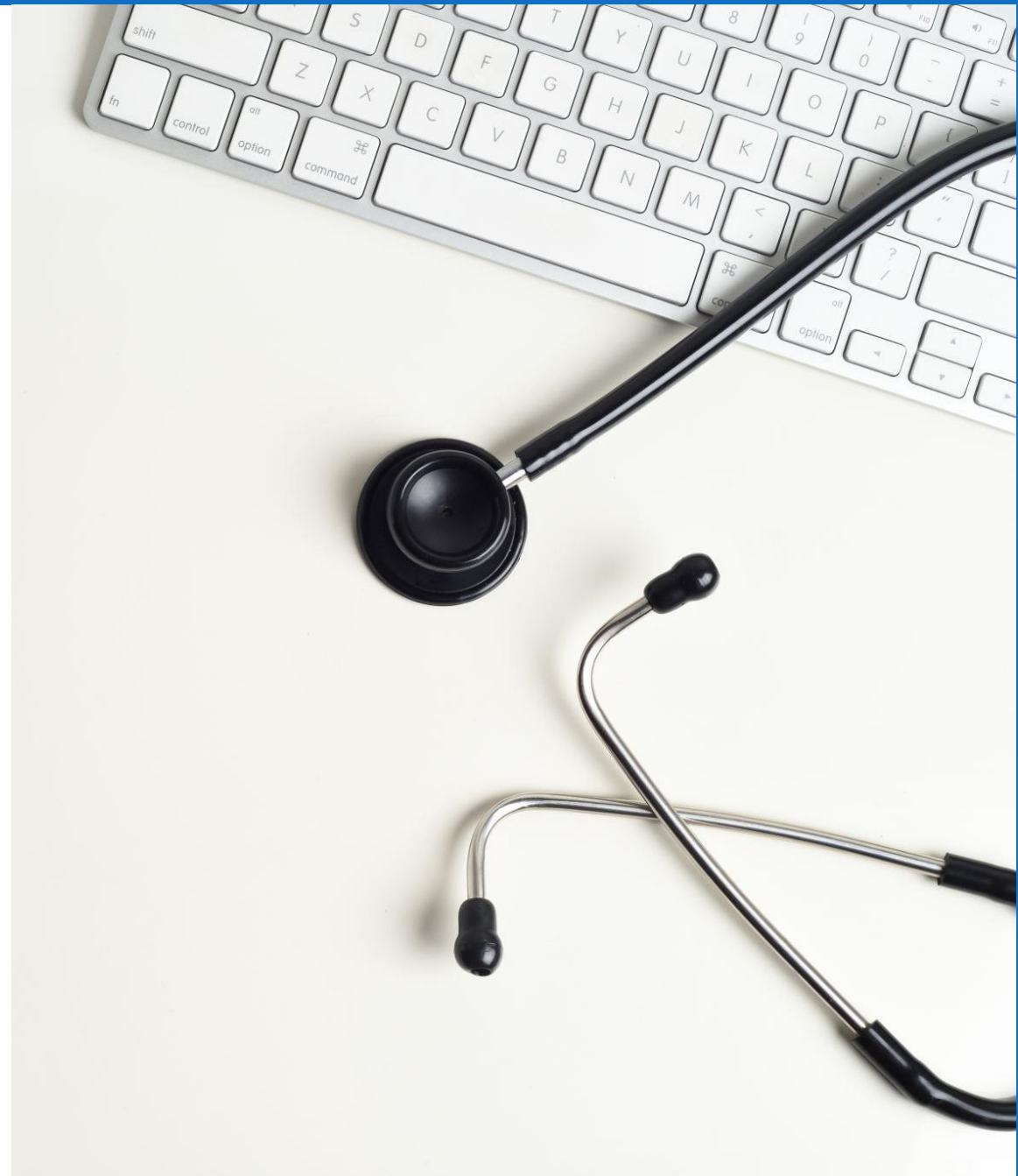
Web service programming (MIS3502)

Objectives

- Develop mobile-ready, web-based, API-driven applications
- Develop in line with RESTful conventions
- Design and implement a business solution

Skills

- All prior: AWS, JavaScript, Node.js, Bootstrap, jQuery, etc.
- Develop on Linux platform



Sample assignment: US State Codes API (Web Application Development)

<https://tinyurl.com/a16USAstates>

USA States (Ajax, jQuery and JSON)

Using external data

In this assignment you will write JavaScript that retrieves JSON data from an API endpoint, found here:

<http://misdemo.temple.edu/states>

It is important to note that there is nothing particularly magical about the misdemo server here. The endpoint URL could be a machine anywhere on the internet!

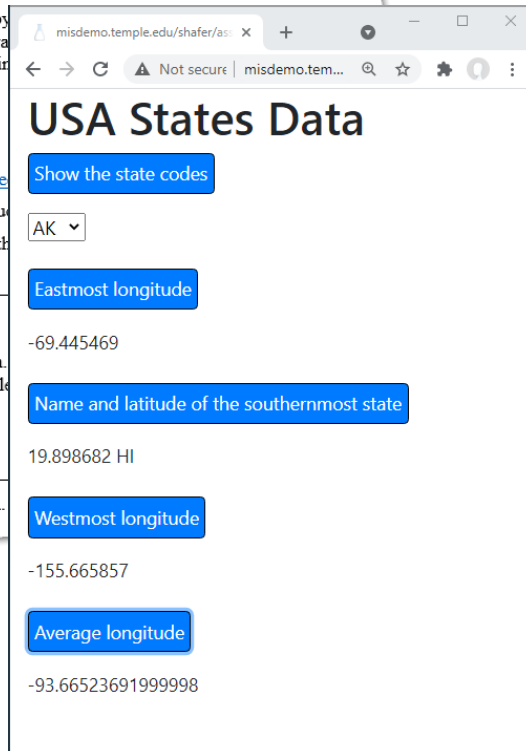
ADVISORY! There's quite a lot in the start file that can be copy own" questions. This is not recommended. Students are encouraged in each question. This is the best way to learn the syntax and reir

Getting started (Together as a class)

1. Visit the API endpoint by typing <http://misdemo.temple.edu/states>
2. Retrieve assignment16_states.zip provided by your instructor
3. Extract the code into your mis2402workspace and open the file
4. Take note of the basic structure of the getJSON method.

```
$getJSON("",function(result){
  // the variable named result has
  // block level scope inside the callback function.
  // it is a good practice to display result in console
  console.log(result);
});
```

5. Review the code found in the displayStatecodes function. Does the "goes here???" string?



USA States Data

Show the state codes

AK

Eastmost longitude

-69.445469

Name and latitude of the southernmost state

19.898682 HI

Westmost longitude

-155.665857

Average longitude

-93.66523691999998

1. Using VSCode and other "real" resources
2. Narrow Scope / High Expectations
3. API Focused
4. Using frameworks
5. Don't start from scratch

Start file: <https://tinyurl.com/6a23nuzm>

Solution: <https://tinyurl.com/bd56pnww>

Cloud Architecture Sample Assignment: Deploy TollCalculator to AWS

The screenshot shows a web browser window with the URL `ec2-34-232-52-69.compute-1.amazonaws.com/TollCalculator.html`. The page title is "Pennsylvania Turnpike Prototype Toll Calculator". It features a form with two dropdown menus for selecting interchanges (one set to "57 - Pittsburgh" and the other to "326 - Valley Forge") and a radio button for payment method (set to "Cash"). A "Get toll" button is at the bottom. A modal dialog box is open, displaying a message: "ec2-34-232-52-69.compute-1.amazonaws.com says The toll from 57 to 326 paying with cash is \$39.30" with an "OK" button.

Below the browser window is an AWS Cloud Architecture diagram. It shows a VPC (TUA12345 VPC - 10.0.0.0/16) with two Availability Zones. In Availability Zone #1, there is an Elastic Load Balancer (listeners on ports 80 and 8080) connected to a TUA12345 AZ1 Public Subnet containing Web Server 1 and App Server 1 (based on load). In Availability Zone #2, there is a TUA12345 AZ2 Public Subnet containing Web Server 2 and App Server 2 (based on load). A TUA12345 AZ1 Private Subnet contains an RDS Live Database. A Route53 icon is also present in the diagram.

Full project instructions:

<https://tinyurl.com/46wvemyf>

Web Service Programming Sample Assignment: Rock- paper-scissors

The screenshot shows a web browser window with the URL `https://ec2-3-213-197-125.comput...`. The page title is "MIS3502 Template". The page content includes a "Make a choice!" heading and three buttons: "Rock", "Paper", and "Scissors". Below these buttons is a "Shoot" button. A text box displays the game results: "You chose: rock", "Computer chose: rock", and "Tie game. Go again!".

Overlaid on the browser window is a JSON response viewer showing the following data:

```
0: "Issue a GET against ./shoot with a 'choice' response. The array response has an element outcome (win,loss,tie), and an element [1] th (0=loss,0.5=tie,1=win). Valid options for th 'rock', 'paper', 'scissors'."
1: "Issue a GET against ./history with a 'user to text response. The text summarizes history losses against the computer"
2: "This API created by Jeremy Shafer."
```

Assignment Instructions: <https://tinyurl.com/a05rps>

Start file: <https://tinyurl.com/52dy6h8g>

Solution: <https://tinyurl.com/mmsvfe9d>



Challenge:

Web frameworks are exceedingly difficult

Templates, examples, starter files

Challenge:

Web is a “black box”

Tutorials, drilling, forced interaction

Challenge:

Empowering students – they need to believe!

In-class exercises are practice for assignments

Thanks!

- Ongoing process of refinement
- The API is integral
 - Important on its own
 - Enables complex applications

