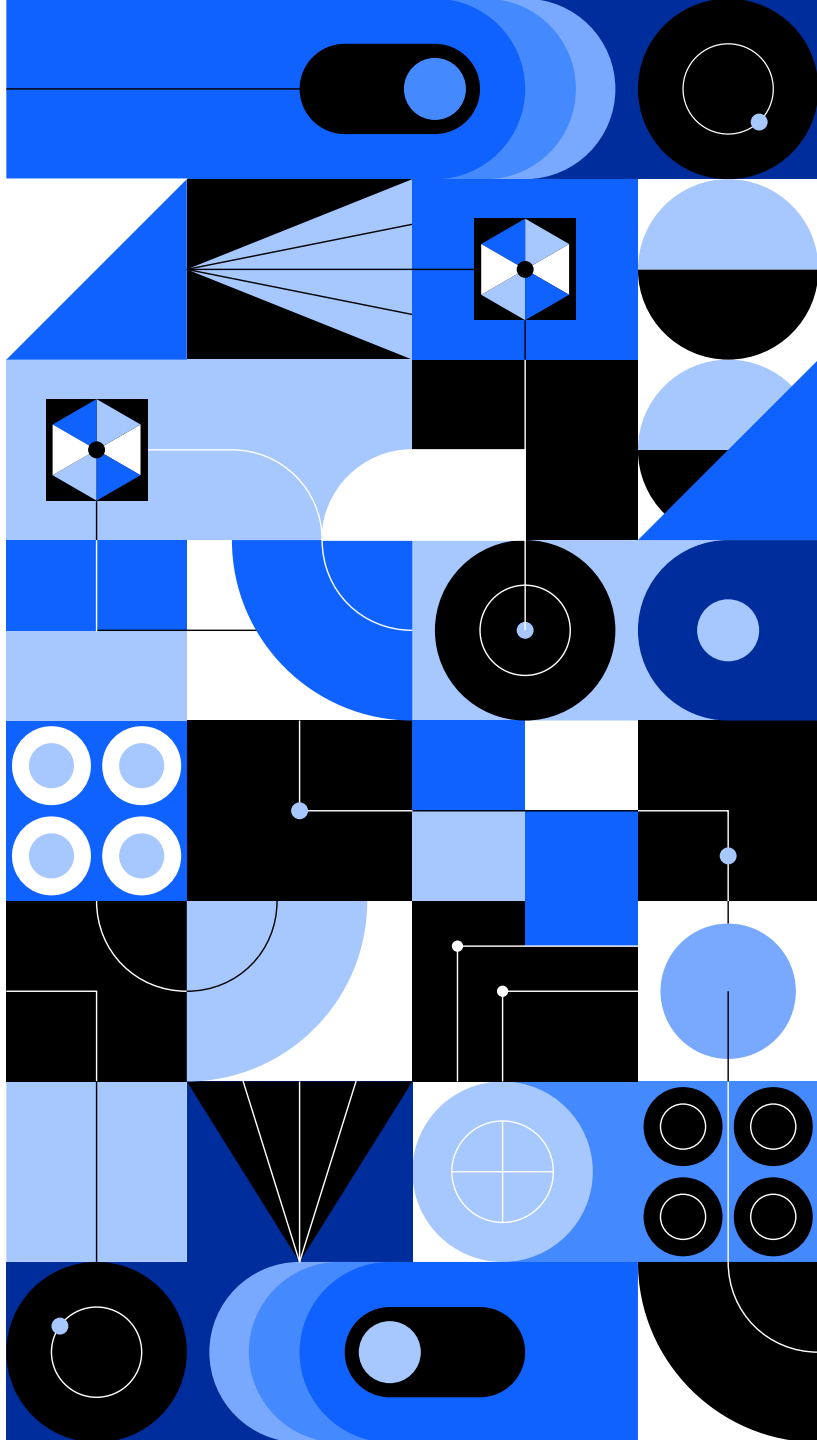
IBM Accelerate Software Developer Track

Wednesdays June 3rd– July 24th, 2024
6:00pm – 8:00pm Eastern Time



• Optional practice test

Wednesday, June 26, 2024

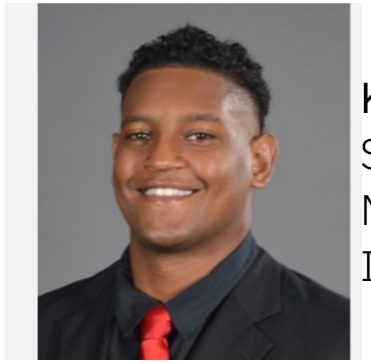
Fundamentals, Client Side versus Server Side, Backend Introduction, and Web Application Security Introduction

Today's speakers:

Sujeily Fonseca
Software Engineer Manager,
MultiCloud SaaS Platform,
IBM Software
SW Track Leader

Daniel Schenker
AI/Linux Toolchain Developer,
IBM Infrastructure

Shidong Shan
Senior Software Engineer,
Security Guardium,
IBM Software



Kidus Woldeyes
Software Engineer,
MultiCloud SaaS Platform,
IBM Software

Session Agenda

1. **Introduction & Fundamental Concepts**
   Front-end versus back-end development.
   Fundamental concepts about backend development.
2. **Client-Side versus Server-Side**
   Overview of what each one entails and the differences between them.
3. **Back-end Introduction**
   Introduction to backend development using Express/Node.
4. **Web Application Security Introduction**
   Introduction to web application security and why it is important.
5. **Lab/Project and GitHub Classroom**
6. **HackerRank Practice Test**
7. **Next Steps & Q/A**

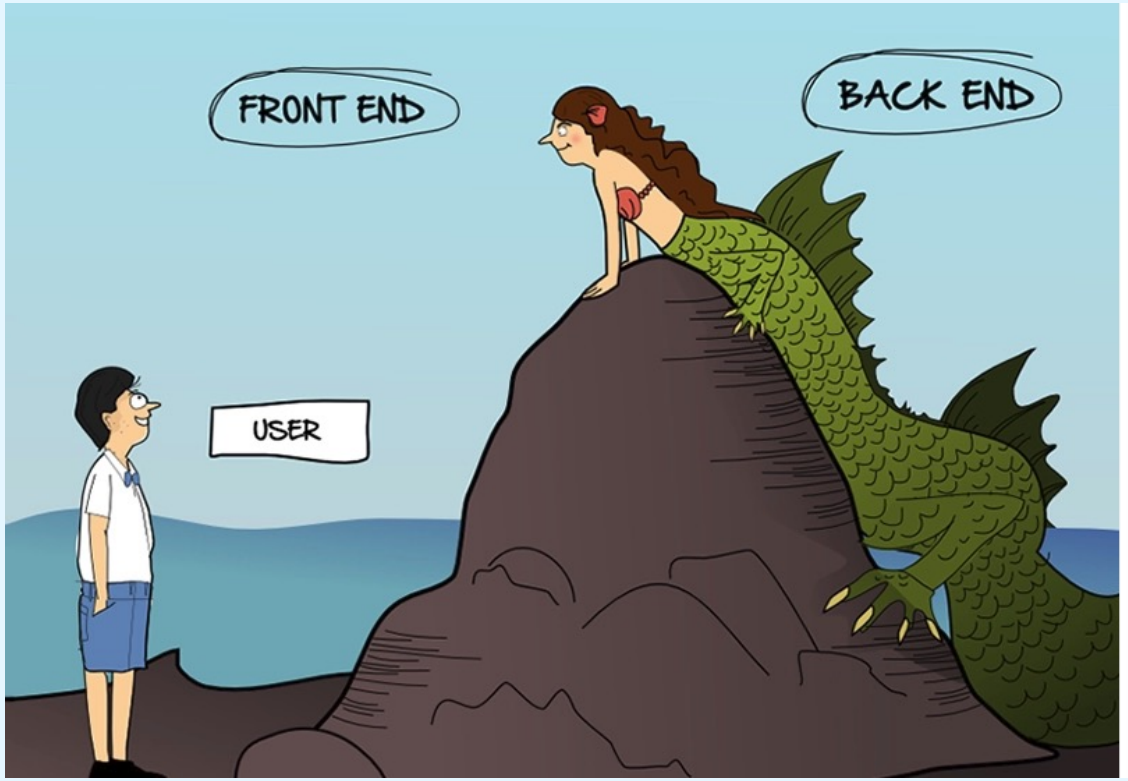
Introduction & Fundamental Concepts

Kidus Woldeyes
Software Engineer,
MultiCloud SaaS Platform

IBM

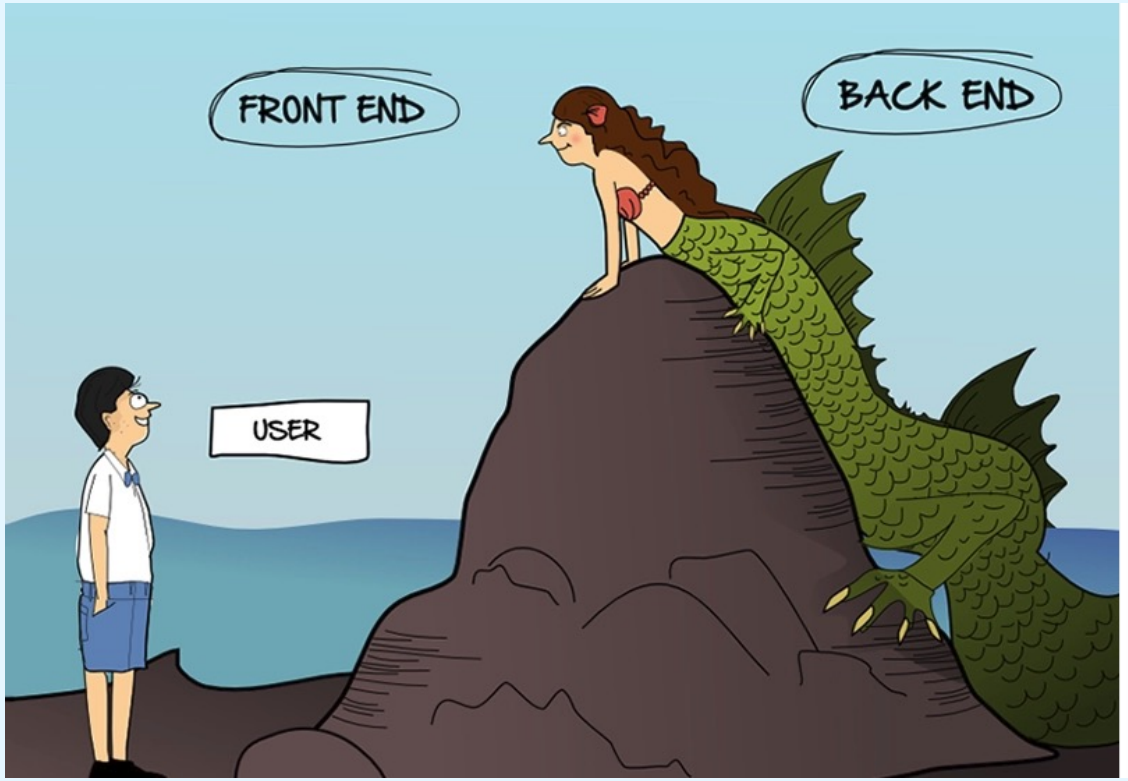
Front-end versus Back-end:
What is the difference?

- Front-end (client-side) development is programming that focuses on building parts of a website or application that users interact with.
- Back-end (server-side) development is programming that focuses on managing data.
- We will discuss client-side and server-side in the next sections.



Front-end versus Back-end:
Why is backend important?

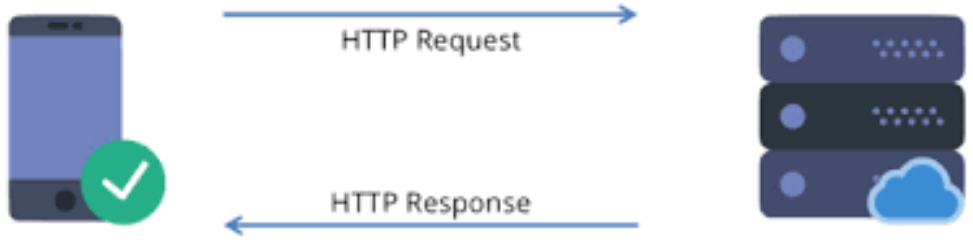
- Backend developers are responsible for managing the server, database, and application logic, providing the front-end with the necessary data to make web applications responsive and functional.
- The backend developer has a vital role in web applications' performance.
- Backend developers handle the business logic.



Fundamental Concepts:
HTTP

What is it?

- HTTP stands for Hypertext Transfer Protocol.
- HTTP is the protocol used for transmitting data over the internet.
- Foundation of data exchange on the web, and it is a client-server protocol, which means requests are initiated by the recipient.



Fundamental Concepts:
HTTP (Cont.)

Request Methods

- **GET**
  - Requests a read-only representation of the specified resource
- **PUT**
  - Replaces all current representations of the target resource with the request payload
- **POST**
  - Submits an entity to the specified resource, often causing a change in state or side effects on the server
- **PATCH**
  - Partial data update
- **DELETE**
  - Deletes the specified resource

Status Codes

102 - Proccesing

200 - Ok

308 - Permanent redirect

429 – Too many requests

500 – Internal server error

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Fundamental Concepts:
APIs

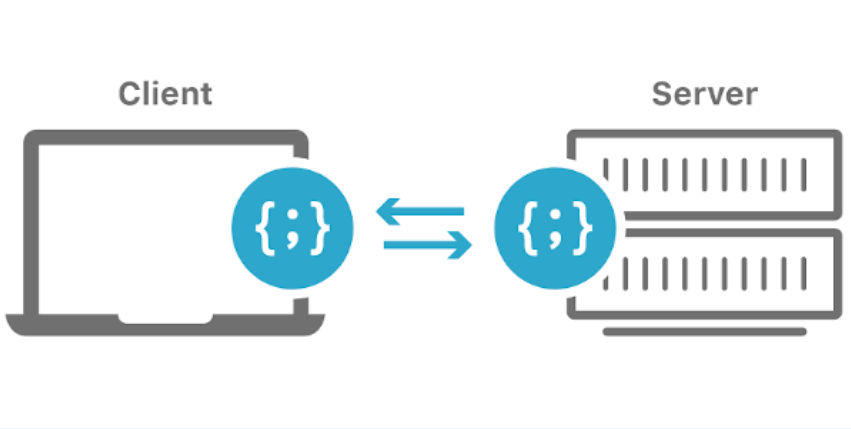
What is it?

- An API, short for Application Programming Interface, is a software-to-software interface.
- APIs provide a secure and standardized way for applications to work with each other.

- **A client application initiates an API call** to retrieve information —also known as a request.
- **After receiving a valid request**, the API makes a call to the external program or web server.
- **The server sends a response** to the API with the requested information.
- **The API transfers the data** to the initial requesting application.

Client-Side versus Server-Side

Kidus Woldeyes
Software Engineer,
MultiCloud SaaS Platform,
IBM Software



IBM

Client-Side versus Server-Side

What is client-side?

- Processing/actions are taken place on the user's (the client's) computer.
- Doesn't need interaction with the server.
- Reduces load on the server (particularly on the processing unit).
- Visible to the users.
- Languages: HTML, CSS, JavaScript (including React and other derivations)

What is server-side?

- Processing/actions are taken place on the web server.
- Requires interaction with the server.
- Allows the server to provide dynamic websites tailored to the user.
- Increases the processing load on server.
- Not visible to the users.
- Languages: PHP, ASP.net, Python, Node, Go, Java

Client-Side versus Server-Side:
Examples

Client-Side

- UI interactions
  - Text
  - Buttons
- Actions performed within the user's browser:
  - Events (onClick, onSubmit)

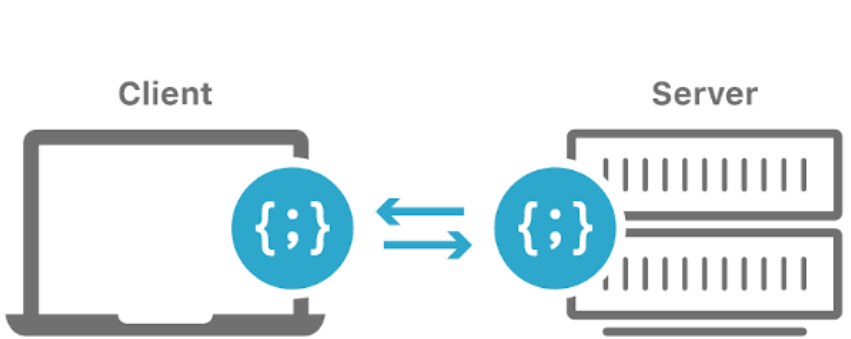
Server-Side

- Databases
- Authentication
- Business logic

Client-Side versus Server-Side:
Why are they separated?

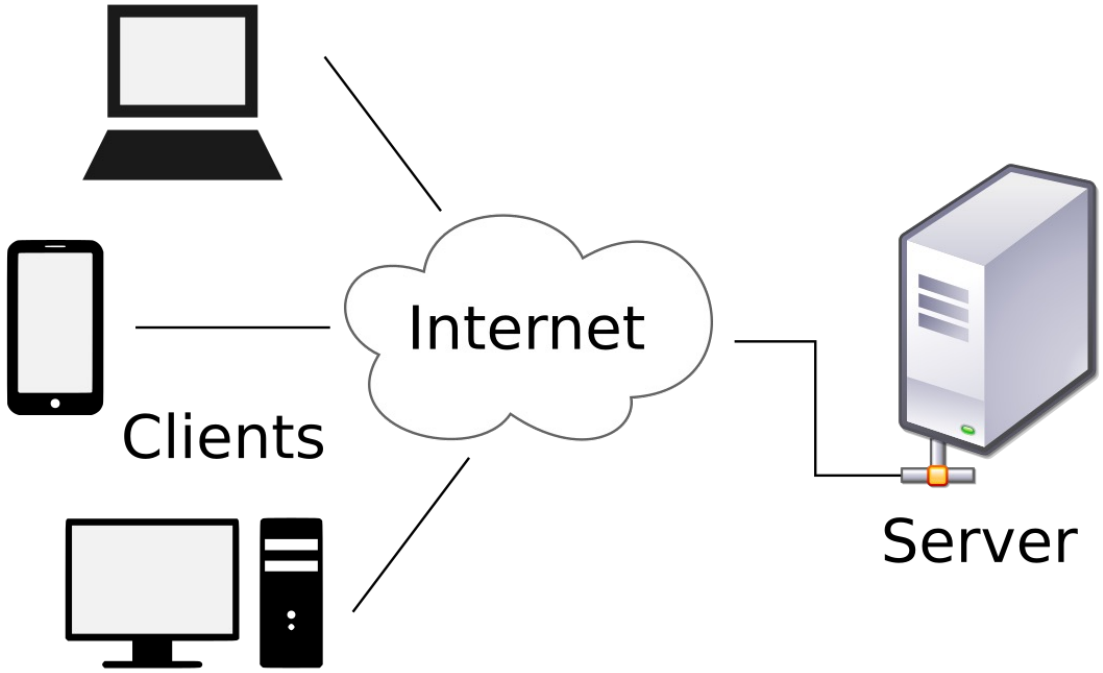
- User experience
- Security
- Control
- Data privacy
- Performance
- Abstraction (modeling)

You can do a lot more with them separate.



Client-Server Architecture

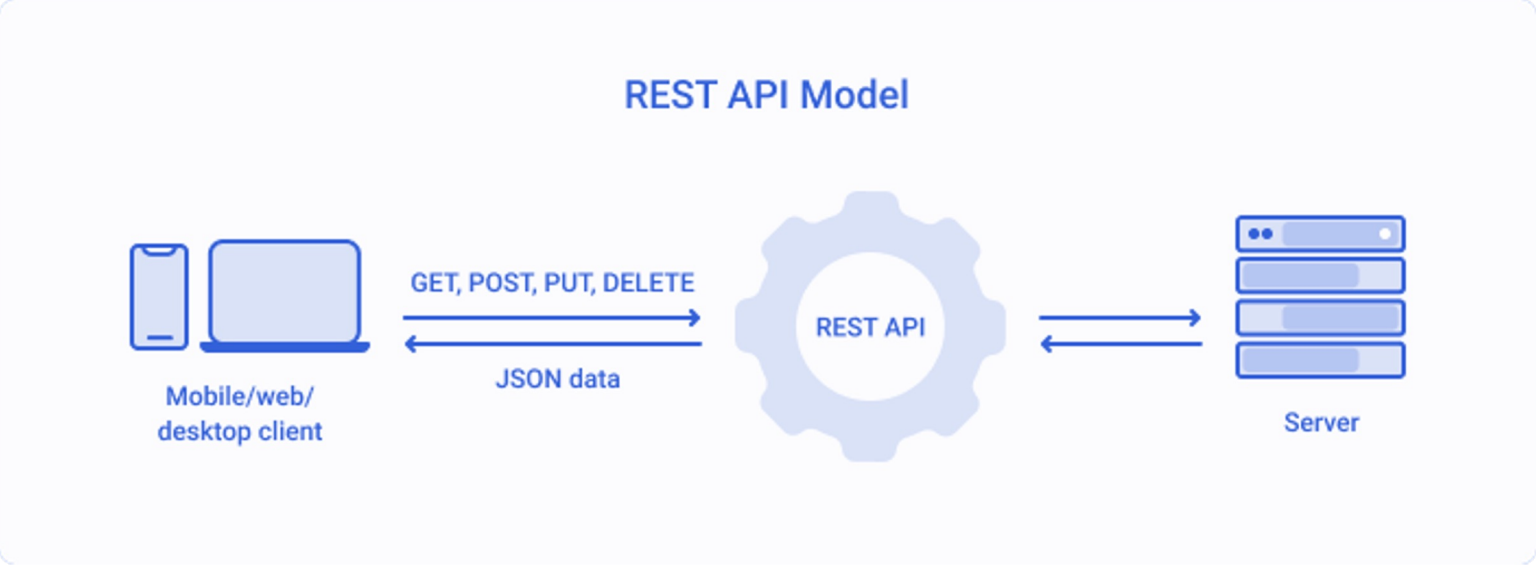
- **Requests:** Sent from the client in order to ask the server for some data.
- **Response:** Sent from the server to the client (reaction of the server to a request of the client).
- **Service:** A specific task that the server provides for the client to use.

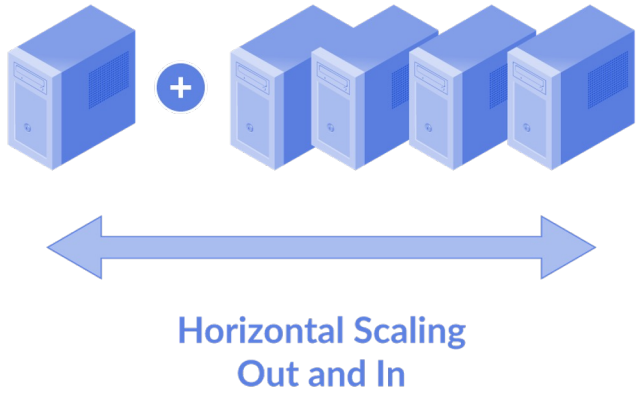


Communication Between Client and Server

RESTful

- REST stands for Representational State Transfer. REST is the set of constraints. RESTful refers to an API adhering to those constraints.
- RESTful communication means the implementation of a web service using HTTP and REST principles.
- REST is based on a stateless protocol.
- REST can scale horizontally.

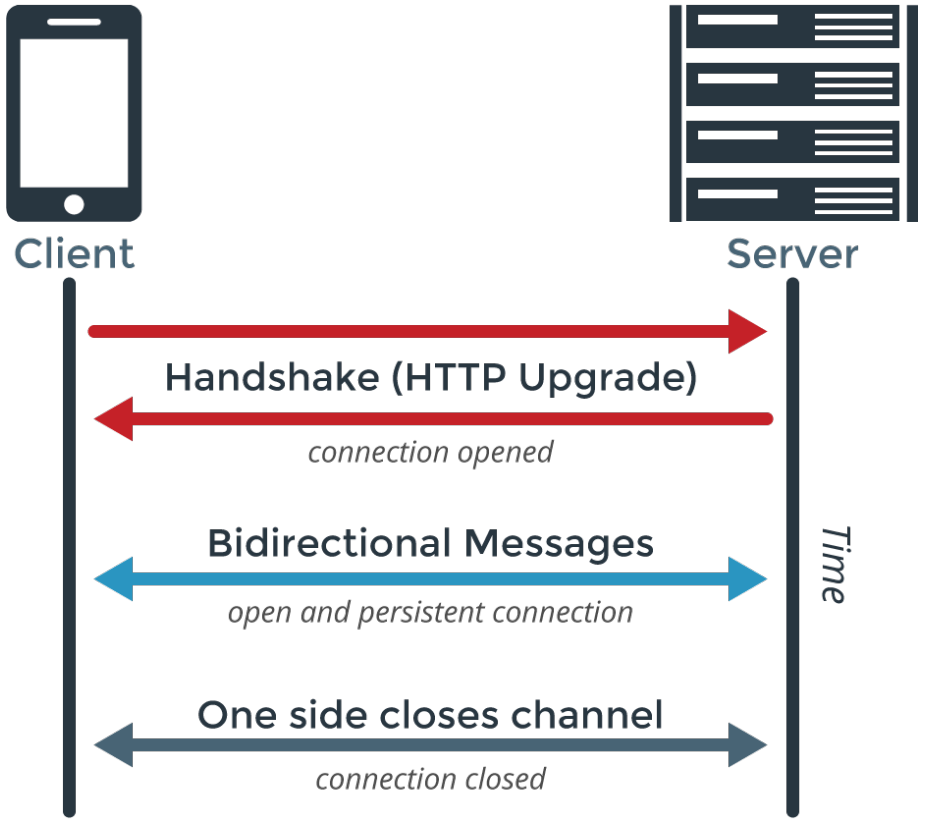


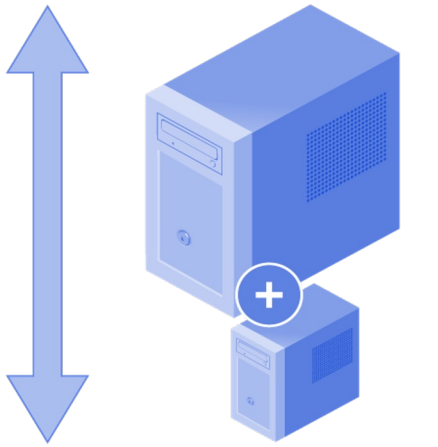


Communication Between Client and Server (Cont.)

WebSocket

- Protocol that makes two-way communication between the user and the server in real-time.
- Alternative for HTTP communication in web applications.
- WebSocket is a stateful protocol.
- WebSocket can scale vertically.

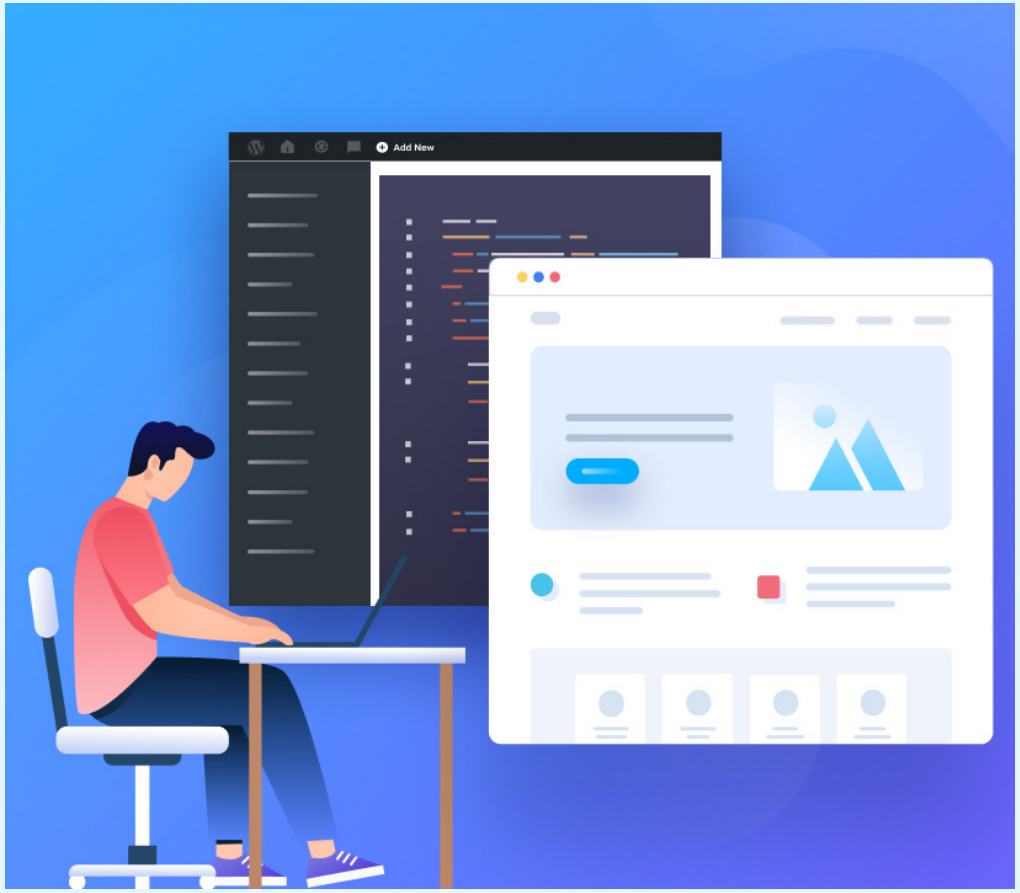




Backend Introduction

Daniel Schenker

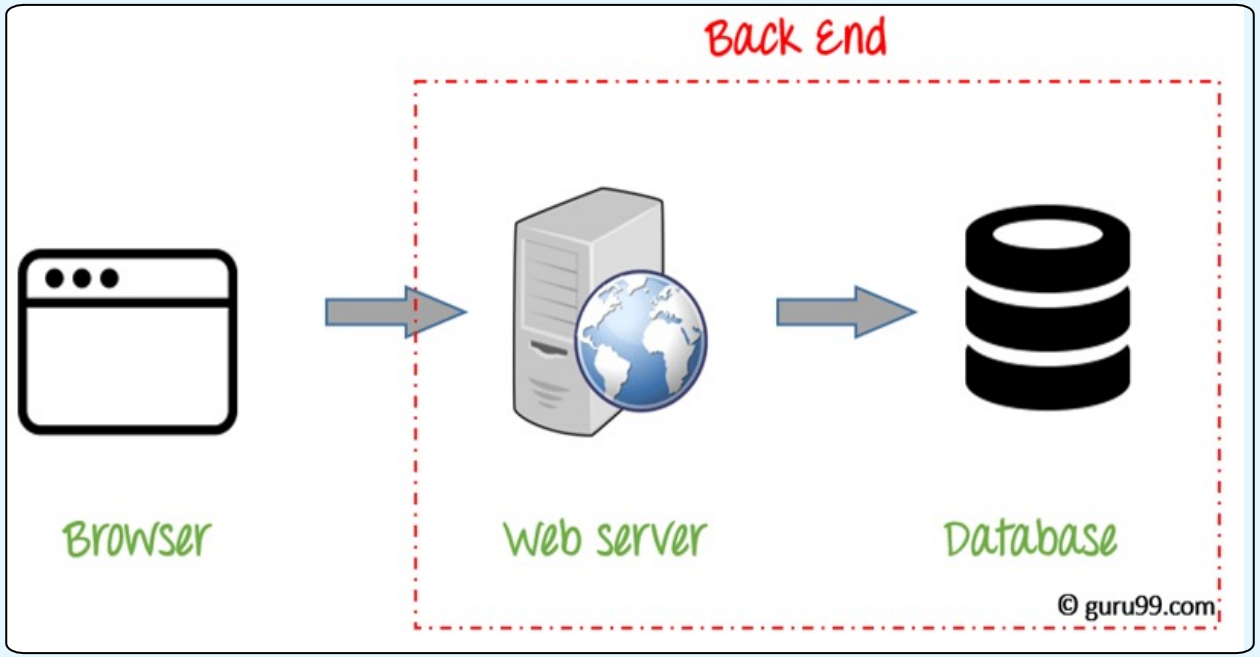
AI/Linux Toolchain Developer,
IBM Infrastructure



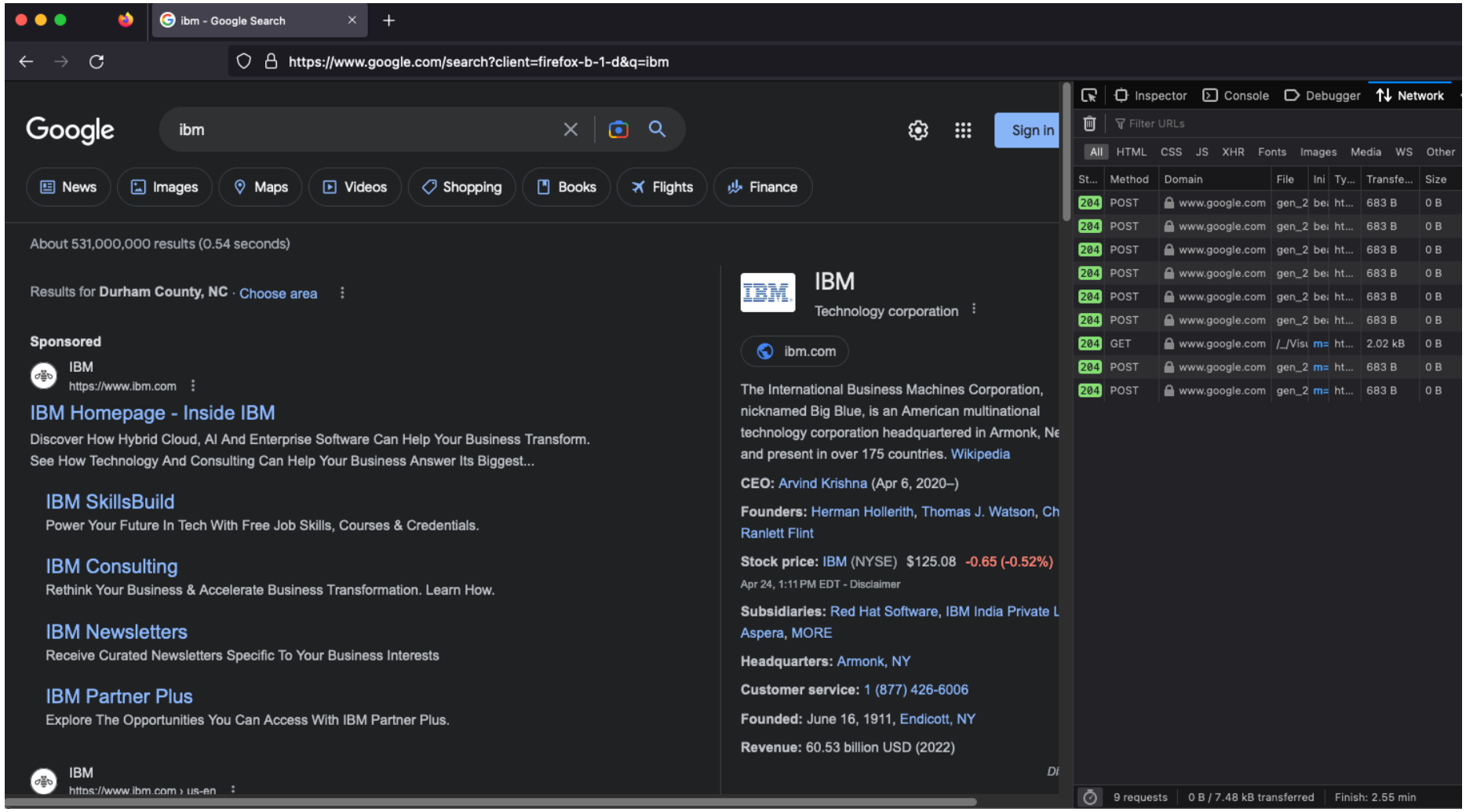
IBM

Backend Development

- **Back-end Development** refers to server-side development. It focuses on accessing and updating databases, implementing business logic, and server architecture.
- It contains behind-the-scene activities that occur when performing any action on a website.
- It can be an account login or making a purchase from an online store.
- Code written by back-end developers helps applications communicate with database information.

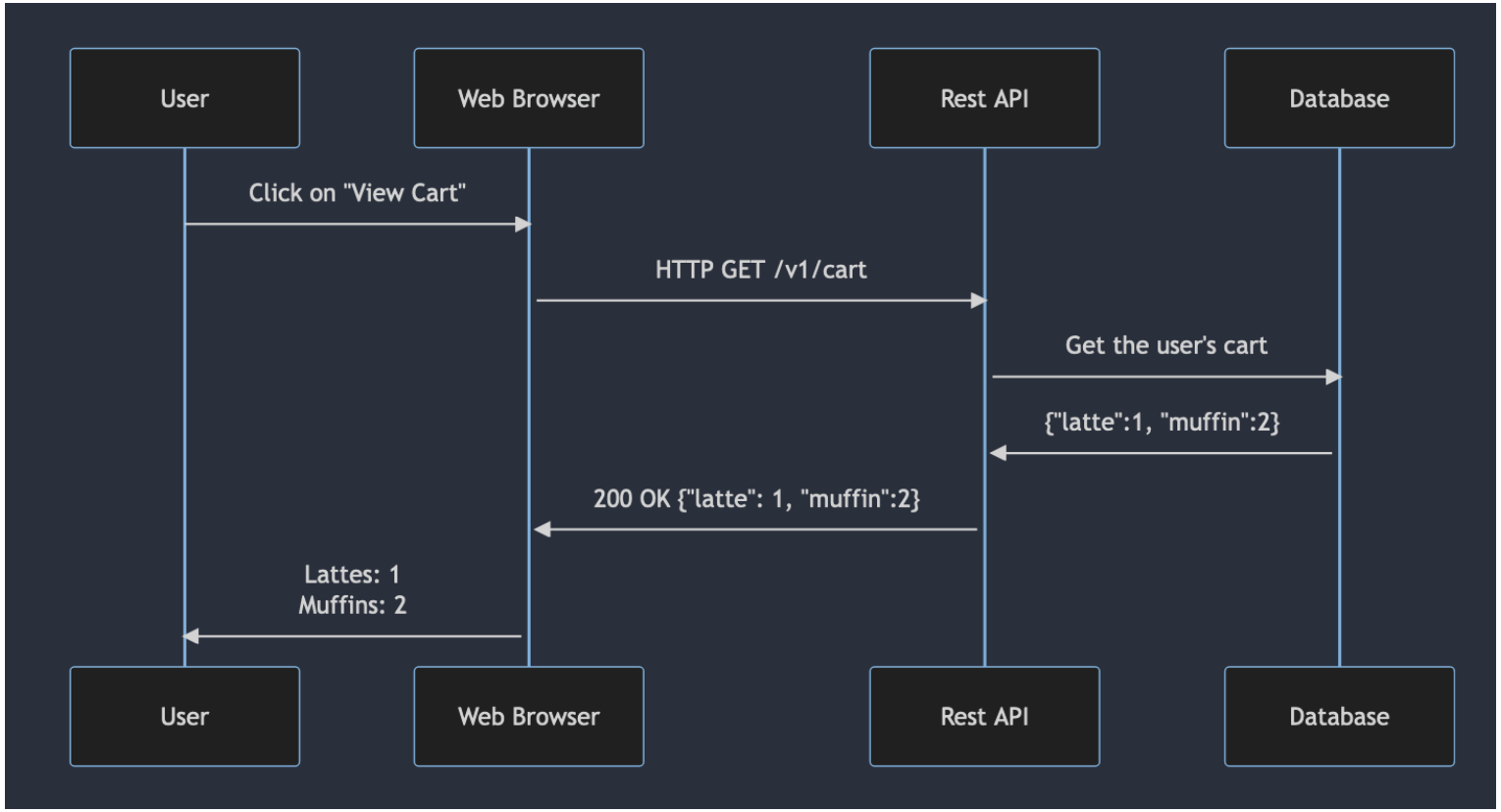


Backend Development:
Example



Rest API:
Deep-Dive

- Connects the front and back end through HTTP requests.
- Can query the database for what it needs.
- Sends response back.



What is Express and how does it work?

Express is a web application framework for **Node.js that** allows you to spin up robust **API and Web Servers** in an easy way.

Creating a Server

```
const express = require('express' 4.17.1 )
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
```

How do we access the backend?

```
app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

Web Application Security Introduction

Shidong Shan
Senior Software Engineer,
Security Guardium,
IBM Software

What is Web Application Security?

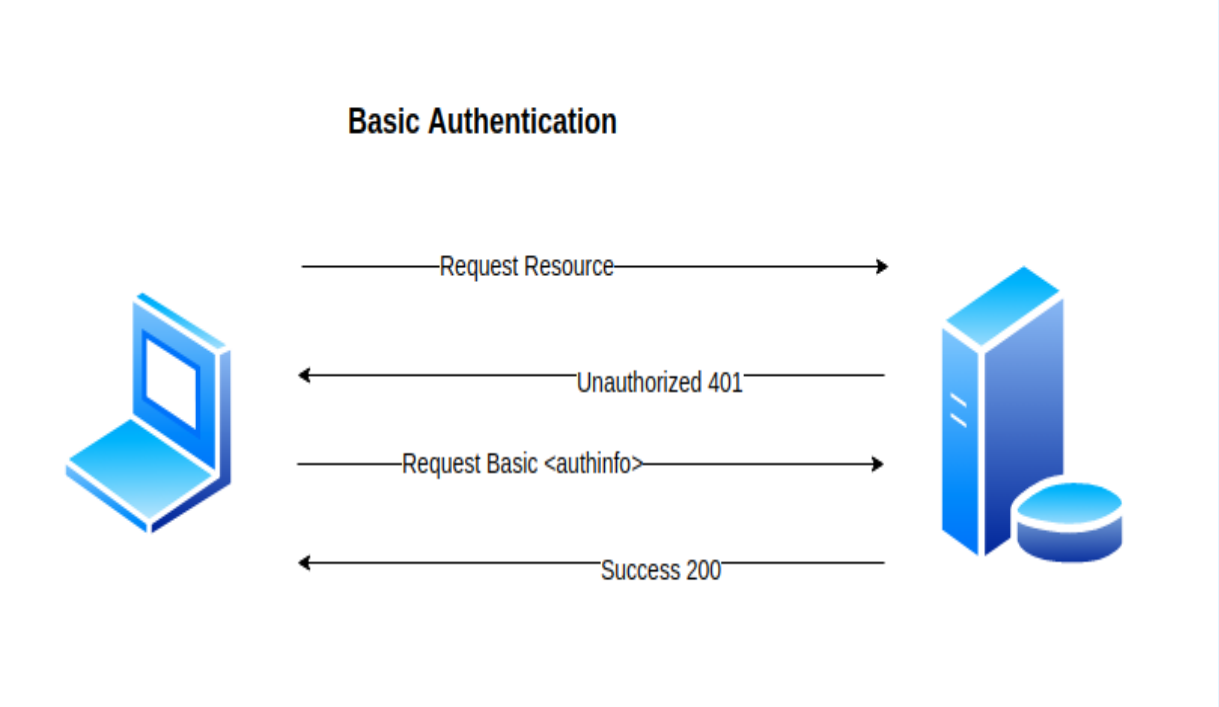
- Web Application Security (Web AppSec) is the practice of defending websites, web applications, and web services against malicious cyber-attacks, such as SQL injection, cross-site scripting, or other forms of potential threats

- Include a collection of security controls engineered into a Web application to protect its assets from potentially malicious agents

- Leverage secure development practices and implementing security measures throughout the software-development life cycles

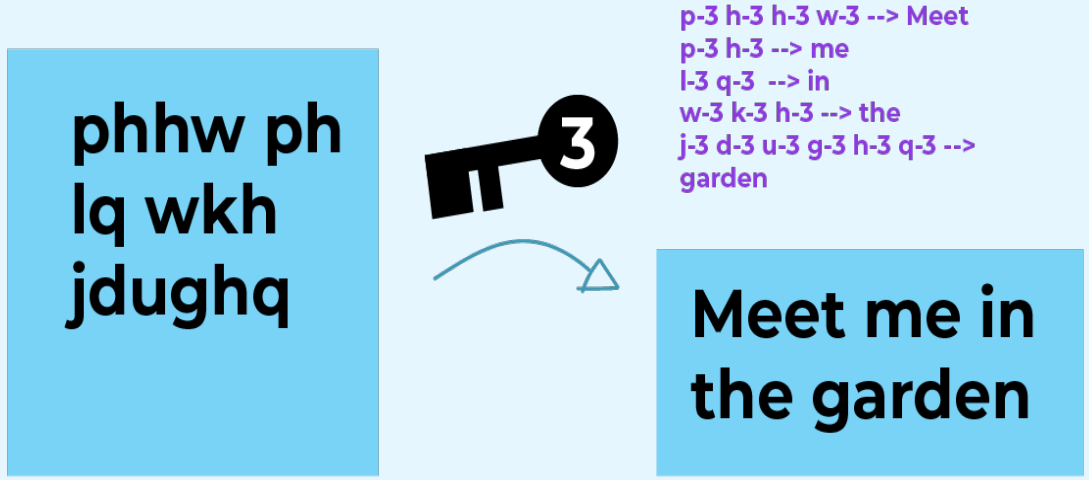
Authentication and Authorization

- Authentication validates that users are whom they claim to be.
- Common authentication methods:
  - Username and Passwords
  - One-time passcode
  - Authentication apps
  - Biometrics
  - multi-factor authentication (MFA)
- Authorization grants the users permissions to access what resources
  - Access control
  - Client privileges
- Authorization must always follow authentication.
- A user must first prove their genuine identities before the server can grant the user access to the requested resources.



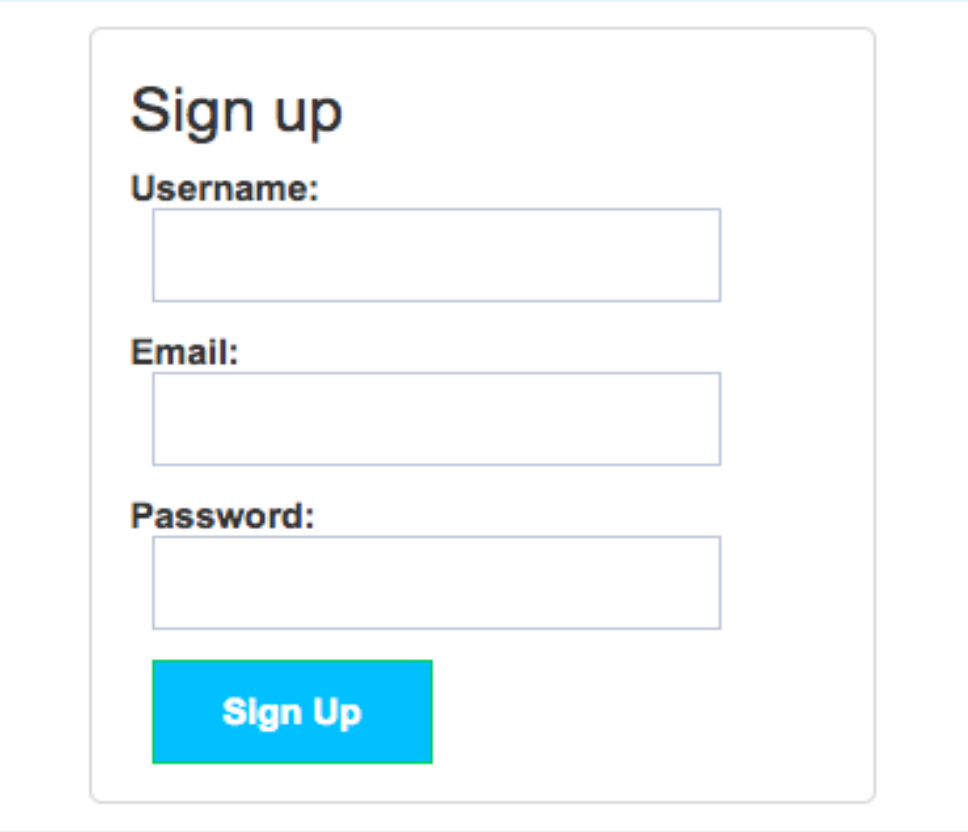
Encryption and Decryption

- To protect sensitive data in transit and at rest from unauthorized use
  - Data in transit, or data in motion, is data actively moving from one location to another
  - Data at rest is data that is not actively moving such as data stored on a hard drive, laptop, flash drive, or archived/stored in some other way.
- Encryption converts the data into a secret code and makes it unreadable
  - To protect data in motion, encrypt sensitive data prior to moving and/or use encrypted connections (HTTPS, SSL, TLS, FTPS, etc) to protect the contents of data in transit.
  - To protect data at rest, encrypt sensitive files prior to storing them and/or choose to encrypt the storage drive itself.
- Decryption is the reverse of encryption.
- Decryption converts the encrypted data back to its original (readable) format. Requires a matching decryption key



Introduction to Frontend and Backend Validation

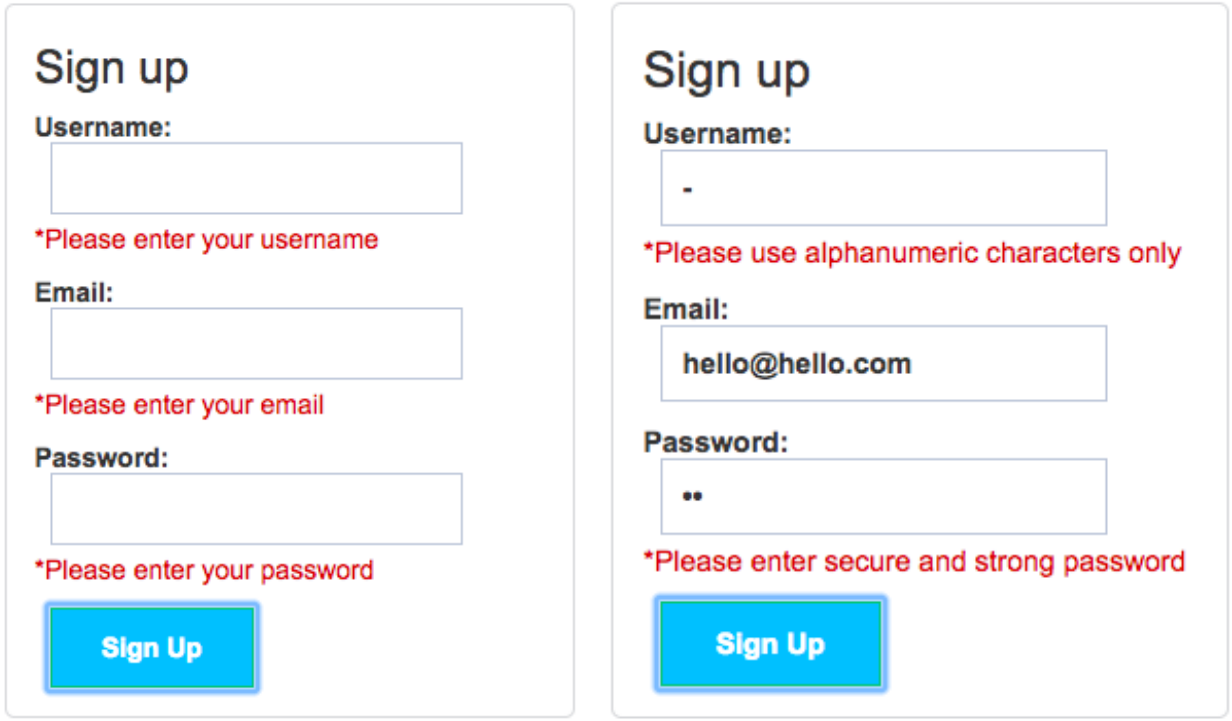
- Validations are used to ensure that only valid data is saved into your database
- Client-side (web-browser) validation is a UI/UX design feature for a better user experience
- Server-side validations are integral components of security and data handling
- All user input data must be validated to protect web applications from malicious attacks on the server resources and databases.



Validation on the Client-Side

What can you do?

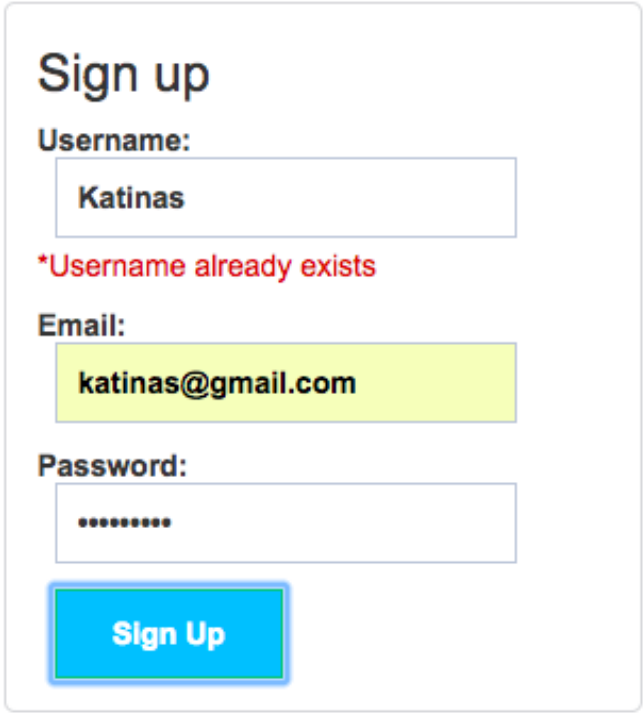
- Showcase custom error messages.
- Can create validations for each user field.
- A regular expression (regex) is a sequence of characters that specifies a match pattern in input text.
- Can use different regex and make it as relaxed or strict as required.
- Check for correct format and throw different errors if invalid.

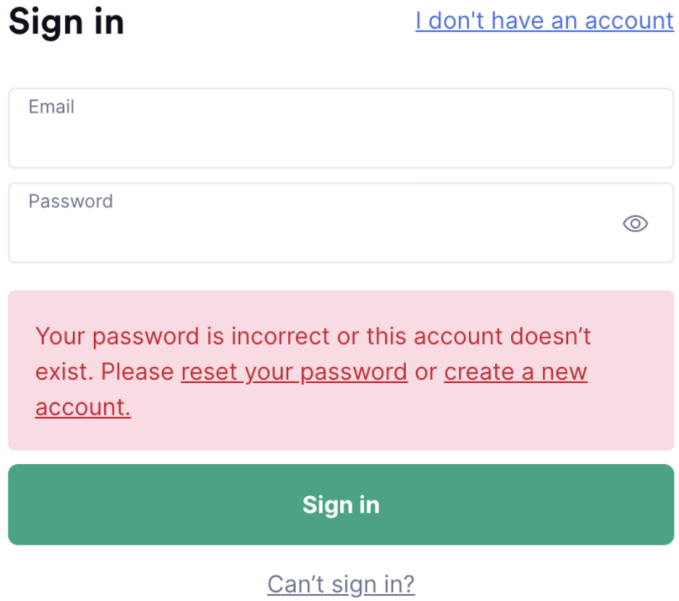


Validation on the Server-Side

What can you do?

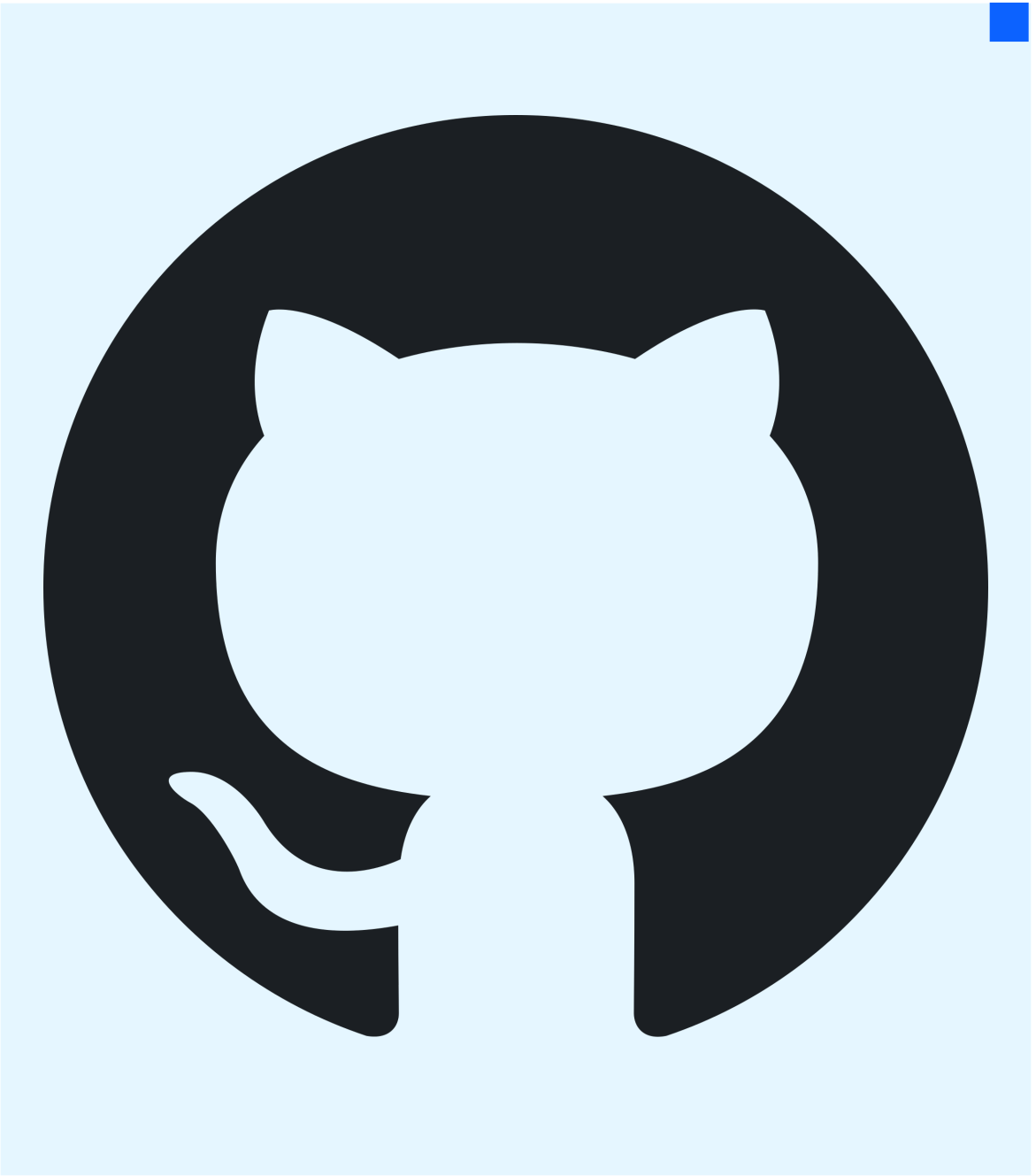
- Validate user input against existing data.
- Example scenarios:
  - If the username/email already exists
  - If the password is correct





Project/Lab and GitHub Classroom

Sujeily Fonseca
Software Engineer Manager,
MultiCloud SaaS Platform,
IBM Software
SW Track Leader



To-do-list Lab:
Review

- Submissions for Lab/Project 4 will close July 8, 2024, 11:59 PM ET
    - We will pull the code from your **master/main** branch at this time to provide feedback.
    - Make sure you merge any branches before then.
- Use the Slack channel and office hours for any questions.

To-do-list Lab:
Instructions

Feature Requirements:

- Create and initialize an Express application (Back-end)
- Create a connection between the Front-end and Back-end
- Create a JSON file to represent a database
- Create a **POST** request to submit data to a JSON file

Implementation Requirements:

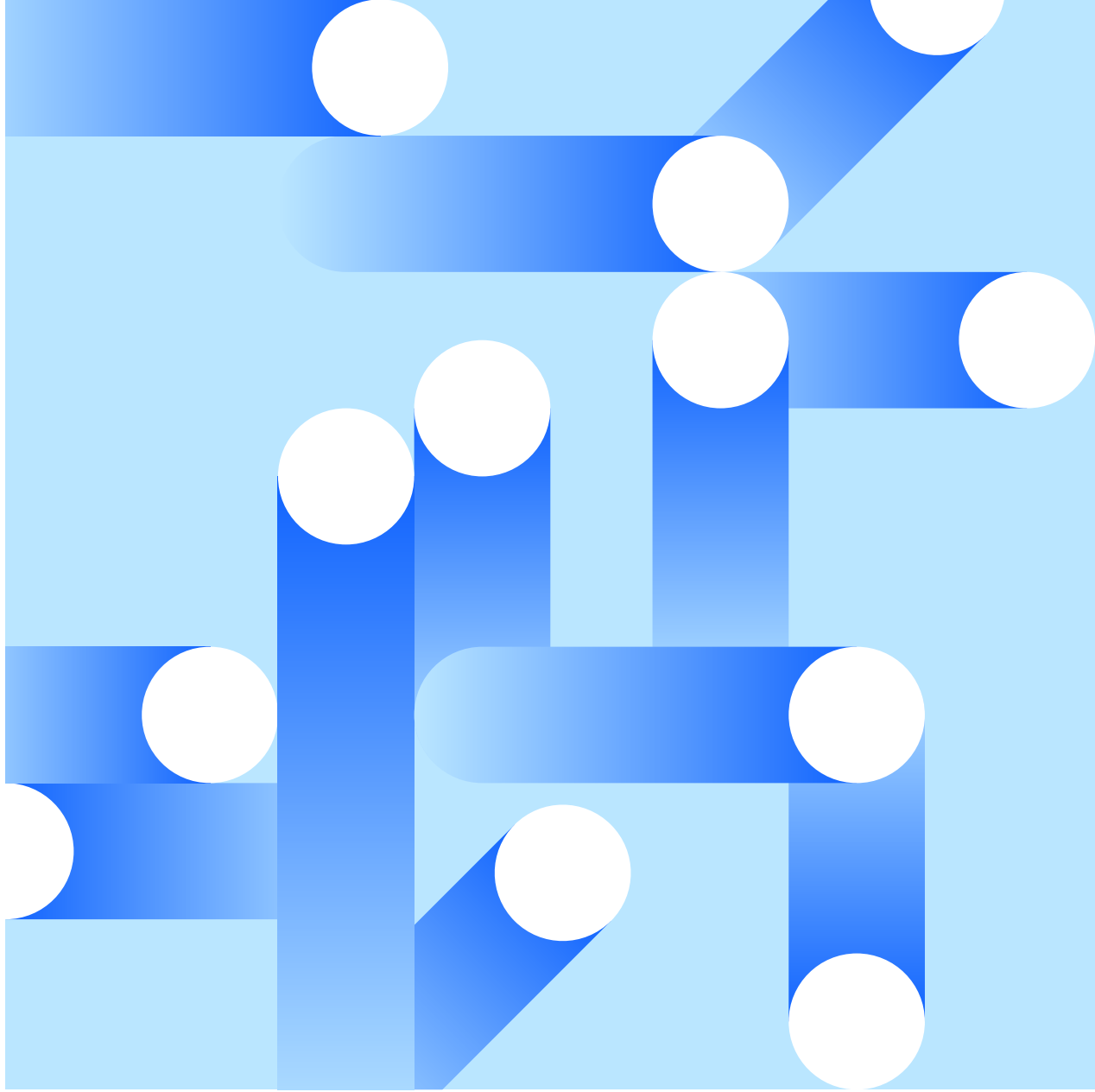
- Use Express w/in the Backend component
- Use Axios w/in the Front-end component (To-do List Application)

Stretch Assignment (Optional Challenge):

- Implement support to use a real database, such as Mongo DB, instead of a local file

GitHub Classroom:

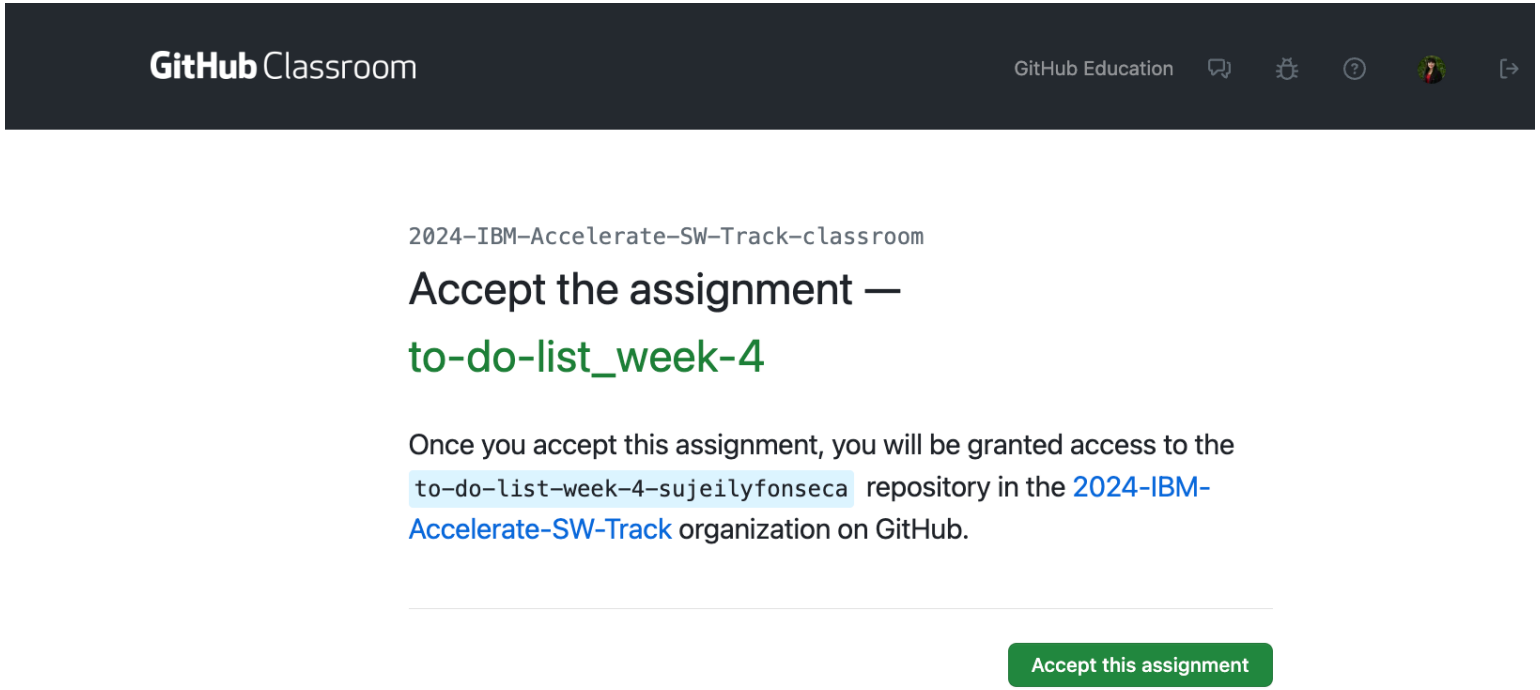
Let’s walk through the process together!



IBM

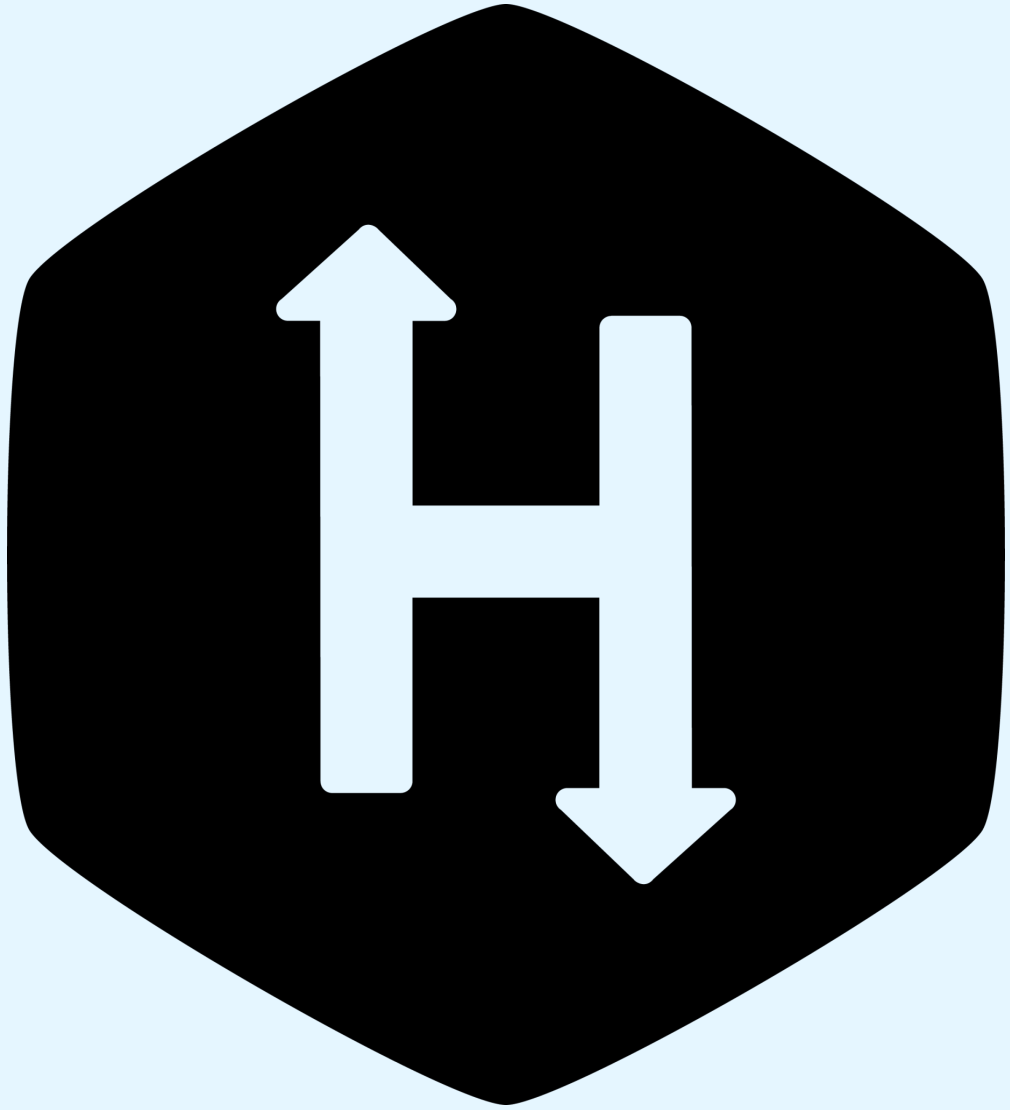
GitHub Classroom:

Let’s walk through the process together!



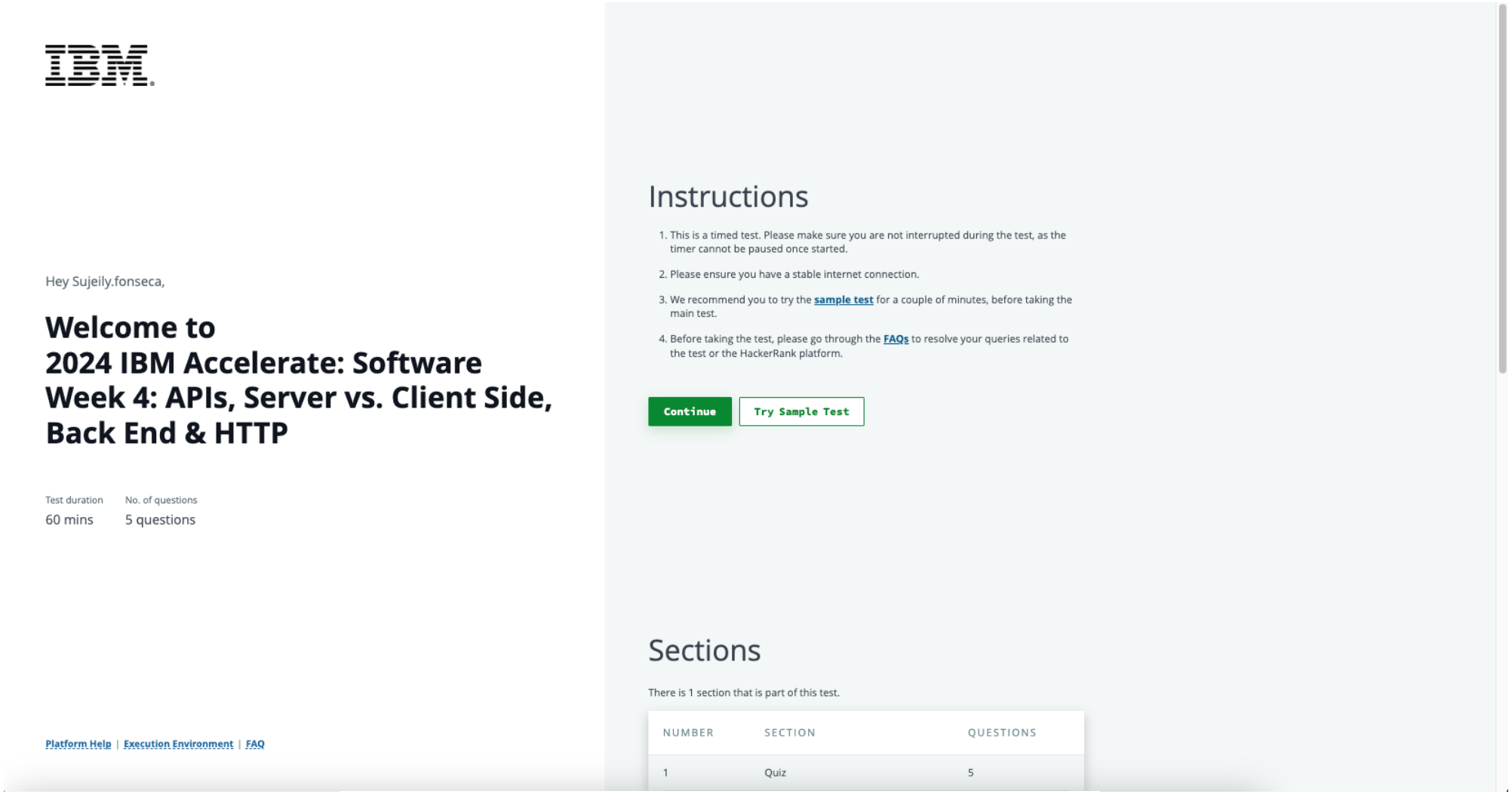
- Short link: https://ibm.biz/accelerate-sw-week4
- Direct link: https://classroom.github.com/a/SmdVjiJ4

HackerRank Practice Test



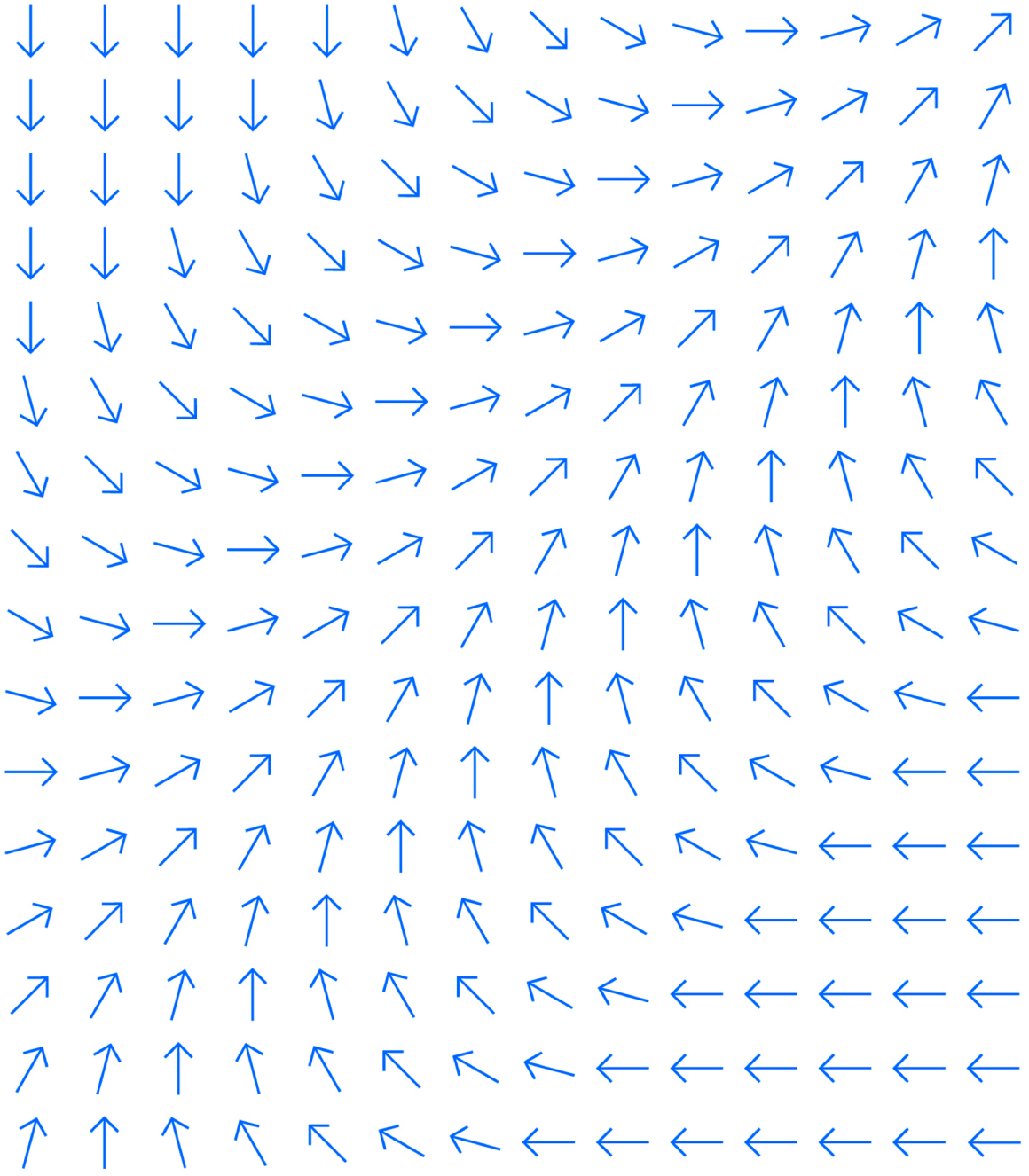
IBM

HackerRank Practice Test



https://hr.gs/accelerate-sw-week4

Next Steps



Next Steps

1. Finish your lab (https://ibm.biz/accelerate-sw-week4) by Monday.
2. If you need further help, ask in the slack channel (#ibm-accelerate-2024-software) or join the office hours (Thursday at 6:00 PM ET).
3. Complete the practice test (https://hr.gs/accelerate-sw-week4).
4. Check out the solutions for the lab and HackerRank test once available.
5. Slides and video recordings of the session are available in Week 4 folder.
6. Watch for Week 5 materials to start to arrive by Monday.
7. Network and connect with your peers today and after today's session.

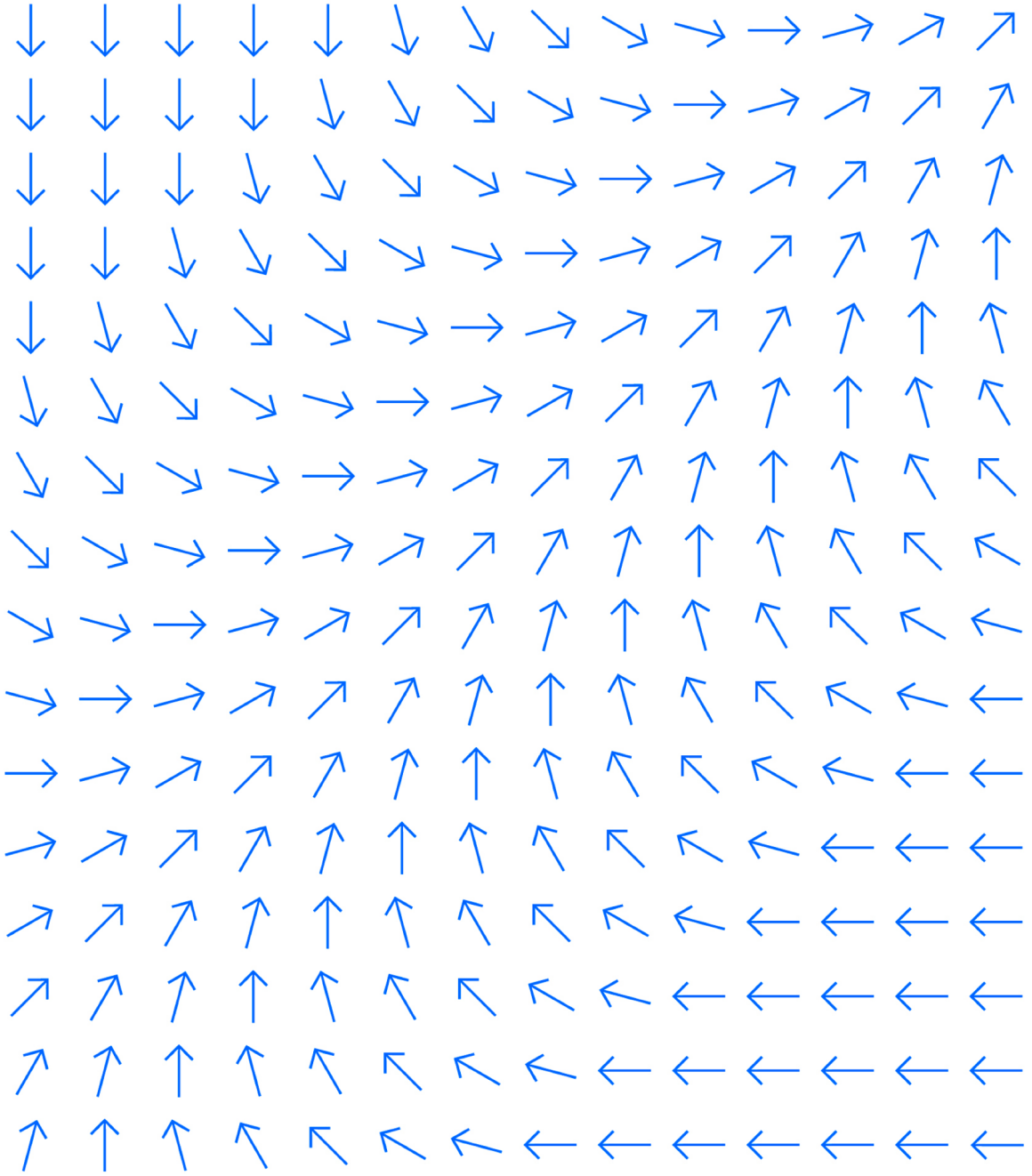
Q&A

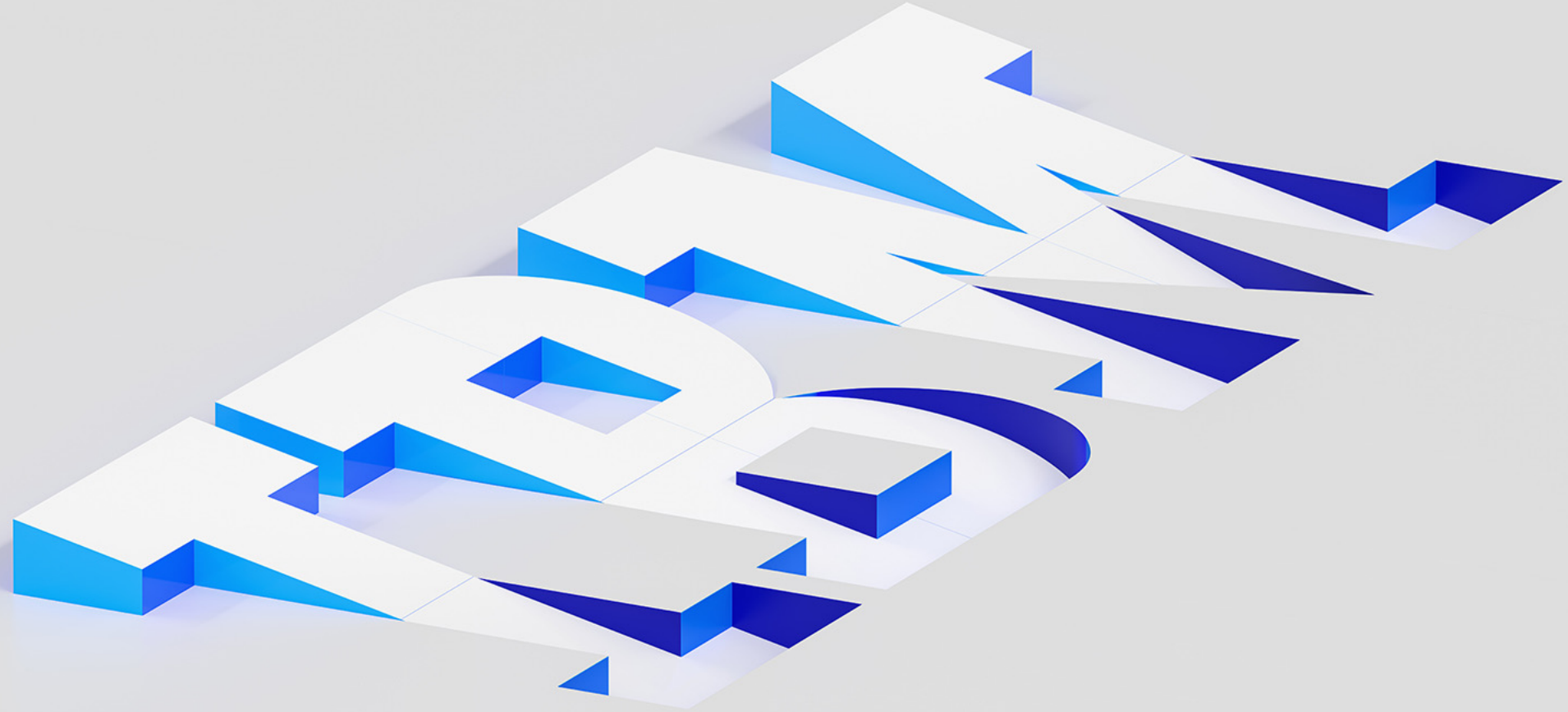
Sujeily Fonseca
Software Engineer Manager,
MultiCloud SaaS Platform,
IBM Software,
SW Track Leader

Daniel Schenker
AI/Linux Toolchain Developer,
IBM Software

Shidong Shan
Senior Software Engineer,
Security Guardium,
IBM Software

Kidus Woldeyes
Software Engineer,
MultiCloud SaaS Platform,
IBM Software





Thank You