Okay, I've updated the 5-week Azure-centric internship plan. Each lab now includes 4-6 specific steps to be accomplished and a list of introductory links. I've also added a note about the weekly Teams calls.

A Note on Weekly Video Calls:

This internship plan incorporates three 1-hour open video calls per week via Microsoft Teams. These sessions are intended for:

- **Q&A:** Addressing any questions students have about the topics, labs, or the Capstone Project.
- **Concept Clarification:** Deeper dives into complex topics based on student needs.
- **Lab Guidance:** Providing hints or troubleshooting common issues encountered during labs.
- **Progress Review:** Discussing progress on labs and, later, the Capstone Project.
- **Guest Speakers/Demos:** Optionally, these slots can be used for specific demos or short talks by experienced professionals. Mentors should schedule these calls at convenient times and encourage active participation.

Revised 5-Week Azure-Centric Internship Plan (with Detailed Labs)

Week 1: Introduction to Cloud Computing and Azure Fundamentals

(Unchanged – essential foundation)

Objective: Understand core cloud concepts, navigate the Azure portal, and learn about fundamental Azure services.

Topics:

- **Day 1:** Introduction to Cloud Computing (IaaS, PaaS, SaaS, deployment models, benefits).
- **Day 2:** Introduction to Microsoft Azure (Global infrastructure, Portal tour, ARM concepts, Free Tier/Azure for Students).
- **Day 3:** Azure Core Compute Services (VMs).
  - **Lab 1: Create a B1s Linux VM (free tier). Connect via SSH.**
    - **Introductory Links:**
    - What is an Azure virtual machine?: https://learn.microsoft.com/en-us/azure/virtual-machines/overview[1]
    - Linux virtual machines in Azure: https://learn.microsoft.com/en-us/azure/virtual-machines/linux/overview
    - Quickstart: Create a Linux virtual machine in the Azure portal: https://learn.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal
    - Connect to a Linux VM using SSH: https://learn.microsoft.com/en-us/azure/virtual-machines/linux/ssh-keys-portal (focus on username/password for simplicity if SSH keys are

too advanced for initial lab) or https://learn.microsoft.com/en-us/azure/virtual-machines/linux/connect-ssh?tabs=azurecli

- Steps (4-6):

1. Navigate to the Azure portal and select "Create a resource."
2. Search for and select "Ubuntu Server" (e.g., Ubuntu Server 20.04 LTS or newer). Choose the B1s size (free tier eligible).
3. Configure basic settings: Subscription, Resource Group (create new), VM name, Region, and Administrator account (username and password).
4. Review networking settings (default is fine for now) and create the VM.
5. Once deployed, find the public IP address of the VM.
6. Connect to the Linux VM using an SSH client (like PuTTY on Windows, or terminal on Linux/macOS) with the public IP address and credentials.

- Lab 2: Create a B1s Windows VM (free tier). Connect via RDP. (Deallocate when done).

- Introductory Links:
- Windows virtual machines in Azure: https://learn.microsoft.com/en-us/azure/virtual-machines/windows/overview
- Quickstart: Create a Windows[2] virtual machine in the Azure portal: https://learn.microsoft.com/en-us/azure/virtual-machines/windows/quick-create-portal
- How to connect to and sign on to an Azure virtual machine running Windows: https://learn.microsoft.com/en-us/azure/virtual-machines/windows/connect-logon
- Steps (4-6):

1. In the Azure portal, select "Create a resource."
2. Search for and select "Windows Server" (e.g., Windows Server 2019 Datacenter or newer). Choose the B1s size.
3. Configure basic settings: Subscription, Resource Group, VM name, Region, and Administrator account (username and password). Ensure inbound RDP port (3389) is allowed.
4. Review and create the VM.
5. Once deployed, connect to the Windows VM using Remote Desktop Connection (RDP) with its public IP address and credentials.
6. After exploration, ensure to "Stop" (deallocate) the VM from the Azure portal to conserve free tier hours/credits.

- Day 4: Azure Storage Services (Blob, File, Queue, Table, Disk).

- Lab 3: Create a Storage Account. Manage Blob containers and files.

- Introductory Links:
- Introduction to Azure Blob storage: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction[3]

- Create a storage account: https://learn.microsoft.com/en-us/azure/storage/common/storage-account-create?tabs=azure-portal
- Quickstart: Upload,[4] download, and list blobs with the Azure portal: https://learn.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal[5]
- Azure Storage Explorer: https://azure.microsoft.com/en-us/products/storage/storage-explorer/[6]
- Steps (4-6):

1. In the Azure portal, search for and select "Storage accounts," then click "Create."
2. Configure the storage account: Subscription, Resource Group, Storage account name (globally unique), Region, Performance (Standard), Redundancy (LRS for cost-effectiveness/free tier).
3. Create the storage account. Once deployed, navigate to it.
4. Go to "Containers" under "Data storage" and create a new container with a chosen name and access level (e.g., Private).
5. Upload a sample file (e.g., a text file or image) to the container through the Azure portal.
6. Download the file back to your local machine. Optionally, explore with Azure Storage Explorer.

- Day 5: Azure Networking Basics (VNets, Subnets, NSGs).
  - Lab 4 (Conceptual/Light Practical): Review VNet/NSG for created VMs.
- Introductory Links:
- What is Azure Virtual Network?: https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview
- Network security groups: https://learn.microsoft.com/en-us/azure/virtual-network/network-security-groups-overview
- View network topology: https://learn.microsoft.com/en-us/azure/network-watcher/view-network-topology (May be more advanced, but good for understanding)
- Steps (4-5):

1. Navigate to one of the Virtual Machines created earlier (e.g., the Linux VM).
2. In the VM's overview page, locate and click on the "Virtual network/subnet" link to explore the VNet settings.
3. Go to the "Networking" settings of the VM to view the attached Network Security Group (NSG) and its inbound/outbound rules (e.g., the RDP or SSH rule).
4. Click on the NSG name to explore its properties, including default rules and how to add new rules (conceptual understanding, no changes needed unless guided).
5. Discuss with peers/mentor how these settings control network traffic to and from the VM.

- Week 1 recap.

Learning Resources: Microsoft Learn (AZ-900 paths), Azure documentation.

Week 2: Web Application Development and Deployment on Azure

Objective: Learn to deploy and manage web applications using Azure App Service and explore basic database options.

Topics & Labs:

- Day 1: Introduction to Azure App Service (Web Apps, App Service Plans).
- Day 2: Deploying a Simple Web App.
  - Lab 5: Create a simple "Hello World" web app locally (Node.js/Python).
    - Introductory Links (Choose based on language):
    - Node.js: https://nodejs.org/en/docs/guides/getting-started-guide/
    - Python (Flask): https://flask.palletsprojects.com/en/2.3.x/quickstart/ or Python (Django): https://docs.djangoproject.com/en/4.2/intro/tutorial01/
    - Setting up your local environment for App Service development: https://learn.microsoft.com/en-us/azure/app-service/configure-local-development-environment?tabs=powershell (General guidance)
    - Steps (4-6):

1. Ensure you have the chosen language runtime (Node.js or Python) installed on your local machine.
2. Create a new project directory for your web app.
3. Write a minimal "Hello World" application (e.g., a simple Express app for Node.js or a Flask app for Python that returns "Hello World!" on the root path).
4. Include necessary files (e.g., package.json for Node.js, requirements.txt for Python).
5. Test the application locally by running it and accessing it in your web browser (e.g., http://localhost:3000).
6. Initialize a local Git repository in the project directory (git init, git add ., git commit -m "Initial commit").

- Lab 6: Deploy to Azure App Service (Free tier) via Git or ZIP deploy.
  - Introductory Links:
  - Azure App Service overview: https://learn.microsoft.com/en-us/azure/app-service/overview
  - Quickstart: Deploy an ASP.NET web app (adapt for Node.js/Python): https://learn.microsoft.com/en-us/azure/app-service/quickstart-dotnetcore?tabs=net60&pivots=development-environment-vs (Example for .NET, find language-specific quickstarts below)
  - Deploy Python app to App Service: https://learn.microsoft.com/en-us/azure/app-service/quickstart-python?tabs=flask%2Cwindows%2Cazure-cli&pivots=python-framework-flask

- Deploy Node.js app to App Service: https://learn.microsoft.com/en-us/azure/app-service/quickstart-nodejs?pivots=platform-linux
- Deployment options for Azure App Service: https://learn.microsoft.com/en-us/azure/app-service/deploy-options
- Steps (4-6):

1. In the Azure portal, create a new "Web App" resource.
2. Configure the Web App: Subscription, Resource Group, Name (globally unique), choose "Code" for Publish, select the correct Runtime stack (Node.js or Python version), Operating System (Linux recommended for these runtimes), and Region.
3. Select an App Service Plan: Create a new one, choose the "Free F1" SKU (or the lowest cost available under Azure for Students credit).
4. After creation, navigate to the "Deployment Center" of your App Service.
5. Choose a deployment source (e.g., "Local Git" or "GitHub" if code is pushed there, or use "ZIP deploy" via Azure CLI or Kudu). Follow the instructions to connect your local Git repo or upload the ZIP.
6. Browse to your deployed web app's URL (<your-app-name>.azurewebsites.net) to verify the deployment.

- Day 3: Managing and Configuring Web Apps.
  - Lab 7: Explore App Service settings, basic logs.
- Introductory Links:
- Configure Azure App Service apps: https://learn.microsoft.com/en-us/azure/app-service/configure-common?tabs=portal
- App Service logging: https://learn.microsoft.com/en-us/azure/app-service/troubleshoot-diagnostic-logs
- Kudu service overview for App Service: https://github.com/projectkudu/kudu/wiki (Advanced, but good to know it exists for diagnostics)
- Steps (4-6):

1. Navigate to your deployed Web App in the Azure portal.
2. Explore "Configuration" > "Application settings": Add a sample application setting (e.g., APP_ENVIRONMENT with value Development).
3. Explore "Configuration" > "General settings": Review platform settings, startup command (if needed for your app).
4. Enable "App Service logs" (Application Logging (Filesystem), Web server logging). Set retention period and quota if desired.
5. Access the "Log stream" to view real-time logs after making a few requests to your web app.
6. Optionally, explore the Kudu console (<your-app-name>.scm.azurewebsites.net) to view deployed files and logs.

- Day 4: Introduction to Azure Databases (Azure SQL, Cosmos DB - Free Tiers).

- Lab 8: Create an Azure SQL Database (serverless). Basic table creation.
  - Introductory Links:[7]
  - What is Azure SQL Database?: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview
  - Serverless compute tier for Azure SQL Database: https://learn.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview (Free grant details are important)
  - Quickstart: Create a single database - Azure SQL Database: https://learn.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?tabs=azure-portal
  - Azure Data Studio: https://learn.microsoft.com/en-us/sql/azure-data-studio/download-azure-data-studio
  - Steps (4-6):

1. In the Azure portal, search for and create an "Azure SQL Database."
2. Configure the database: Subscription, Resource Group, Database name. Create a new Server: provide a unique server name, admin login, password, and region.
3. Choose "Configure database" under Compute + storage. Select "Serverless" as the compute tier. Keep default vCores and data max size within free grant limits if specified or reasonable for learning.
4. Configure networking: Under "Set server firewall," click "Add current client IP address" and enable "Allow Azure services and resources to access this server."
5. Create the database. Once deployed, connect to it using Azure Data Studio (or SQL Server Management Studio) with the server admin credentials.
6. Using the query editor in Azure Data Studio, create a simple table (e.g., CREATE TABLE MyGuests (id INT PRIMARY KEY, firstname VARCHAR(30), lastname VARCHAR(30));).

- Day 5: Connecting Web App to a Database.
  - Lab 9 (Stretch Goal/Guided): Modify web app to connect to Azure SQL Database.
  - Introductory Links (Choose based on language):
  - Python (Flask with pyodbc): https://learn.microsoft.com/en-us/sql/connect/python/pyodbc/step-3-proof-of-concept-connecting-to-sql-with-pyodbc
  - Node.js (Tedious): https://learn.microsoft.com/en-us/sql/connect/node-js/step-3-proof-of-concept-connecting-to-sql-with-node-js
  - Connection strings for Azure SQL Database: https://learn.microsoft.com/en-us/azure/azure-sql/database/connect-query-portal?tabs=windows#c-ado-net (Shows format, adapt for language)
  - Use environment variables for secrets in App Service: https://learn.microsoft.com/en-us/azure/app-service/configure-common?tabs=portal#configure-app-settings
  - Steps (4-6):

1. Install the necessary database driver/library in your local web app project (e.g., pyodbc for Python, tedious for Node.js). Add it to requirements.txt or package.json.

2. Modify your web app code to include logic for connecting to the Azure SQL Database using the connection string. (Get the connection string from the Azure portal for your SQL database).
3. Implement a simple functionality, like inserting a new record into the table created in Lab 8 or reading records from it and displaying them.
4. Store database credentials securely using environment variables (locally for testing, and as Application Settings in Azure App Service). Do NOT hardcode credentials in your source code.
5. Test the database connectivity locally.
6. Re-deploy your updated web app to Azure App Service. Configure the database connection string as an Application Setting in the App Service configuration. Test the deployed app.

- Week 2 recap.

Learning Resources: Microsoft Learn (Azure development paths), Azure service documentation.

Week 3: Exploring AI with Microsoft Copilot & Azure AI

Objective: Understand Microsoft Copilot capabilities (especially GitHub Copilot) and get introduced to Azure AI.

Topics & Labs:

- Day 1: Introduction to AI, Generative AI, and the Copilot Ecosystem. Ethical considerations.
- Day 2: Getting Started with GitHub Copilot (Setup, code completion/suggestions).

- Lab 10: Install and configure GitHub Copilot in a supported IDE (e.g., VS Code). Use it to write simple functions or scripts.

- Introductory Links:
- GitHub Copilot documentation: https://docs.github.com/en/copilot
- Getting started with GitHub Copilot in VS Code: https://docs.github.com/en/copilot/getting-started-with-github-copilot?tool=vscode
- GitHub Student Developer Pack (for potential free access): https://education.github.com/pack
- Steps (4-6):

1. Ensure you have an active GitHub Copilot subscription (e.g., via trial or GitHub Student Developer Pack).
2. Install a supported IDE like Visual Studio Code.
3. In VS Code, search for and install the GitHub Copilot extension from the Marketplace.
4. Sign in to GitHub through VS Code to authorize the Copilot extension.
5. Open a new file (e.g., a Python or JavaScript file). Start writing a comment describing a simple function (e.g., "# function to add two numbers") or begin typing function signature, and observe Copilot's suggestions.

6. Experiment with accepting, rejecting, or cycling through suggestions to complete a few simple functions or code snippets.

- Day 3: Advanced GitHub Copilot (Chat, best practices) & Introduction to Azure AI Platform (Azure OpenAI, AI Search, AI Vision, AI Language).

- Lab 11: Use GitHub Copilot Chat to explain a piece of existing code, generate unit tests for a simple function, and refactor a small code snippet.

- Introductory Links:

- Using GitHub Copilot Chat in your IDE: https://docs.github.com/en/copilot/github-copilot-chat/using-github-copilot-chat-in-your-ide

- GitHub Copilot Chat examples: (Often found within product announcements or community blogs)

- Prompt engineering for GitHub Copilot: https://github.blog/2023-06-20-prompt-engineering-guide-generative-ai-github-copilot/

- Steps (4-6):

1. Ensure GitHub Copilot Chat is enabled and accessible in your IDE (e.g., VS Code, it's often part of the main Copilot extension now).
2. Open an existing code file (or the one from Lab 10). Select a function or code block.
3. Use Copilot Chat to ask it to "/explain" the selected code.
4. Select a simple function you wrote. Ask Copilot Chat to "/tests" to generate unit tests for it. Review and understand the generated tests.
5. Identify a small piece of code that could be improved (e.g., made more concise or readable). Ask Copilot Chat to refactor it or suggest improvements (e.g., "How can I make this more efficient?").
6. Experiment with different Copilot Chat commands like /fix for potential bugs or asking general programming questions related to your code.

- Day 4: Building with Azure AI - Azure OpenAI Service & Azure AI Studio (Prompt engineering basics).

- Lab 12 (Conceptual/Guided Exploration): Explore Azure AI Studio, Azure OpenAI Service playground, or other Azure AI services (e.g., Language) with free tiers.

- Introductory Links:

- What is Azure AI Studio?: https://learn.microsoft.com/en-us/azure/ai-studio/what-is-ai-studio

- Azure OpenAI Service: https://azure.microsoft.com/en-us/products/ai-services/openai-service/

- Quickstart: Get started using Azure OpenAI Service (Chat playground): https://learn.microsoft.com/en-us/azure/ai-services/openai/chatgpt-quickstart?tabs=command-line&pivots=rest-api (Requires access to Azure OpenAI)

- What is Azure AI Language?: https://learn.microsoft.com/en-us/azure/ai-services/language-service/overview (Good alternative if Azure OpenAI access is pending)

- Steps (4-6):

1. Option A (If Azure OpenAI access is available via credits/subscription): a. Navigate to the Azure portal and create an Azure OpenAI resource (if not already provisioned and access is granted). b. Deploy a model (e.g., a GPT-3.5-Turbo model). c. Explore the "Chat playground" in Azure AI Studio or directly with the Azure OpenAI resource. d. Experiment with different system messages and user prompts to understand how the model responds. Try tasks like summarization, translation, or code explanation.
2. Option B (Explore Azure AI Language as an alternative): a. In the Azure portal, create an Azure AI Language resource (this has a free tier). b. Navigate to Language Studio (https://aka.ms/languageStudio) and connect your Azure resource. c. Try out pre-built features like "Sentiment Analysis" or "Key Phrase Extraction" by inputting sample text. d. Observe the JSON output and understand the insights provided by the service.
3. Understand the concept of "prompt engineering" – how crafting good prompts influences AI output.
4. Discuss potential use cases for these AI services in applications.

- Day 5: Introduction to Microsoft Copilot Studio (Building custom copilots).
  - Lab 13 (Guided Exploration): Explore Microsoft Copilot Studio (trial/dev plan) to create a simple FAQ bot.
    - Introductory Links:
    - Microsoft Copilot Studio overview: https://learn.microsoft.com/en-us/microsoft-copilot-studio/overview
    - Quickstart: Create and publish a Copilot Studio copilot: https://learn.microsoft.com/en-us/microsoft-copilot-studio/quickstart-copilot
    - Sign up for a Copilot Studio trial: https://learn.microsoft.com/en-us/microsoft-copilot-studio/sign-up-individual
    - Steps (4-6):

1. Sign up for a Microsoft Copilot Studio trial or use a developer environment if available.
2. Navigate to the Copilot Studio web interface and create a new copilot.
3. Define a few "Topics" for your copilot. Start with simple FAQ-style topics (e.g., "What are your hours?", "Where are you located?").
4. For each topic, define trigger phrases (what users might say) and create simple message responses in the authoring canvas.
5. Test your copilot using the built-in test pane. Try different trigger phrases.
6. Explore how to add slightly more complex logic, like a question with multiple choice options, if time permits. (Publishing is optional for this lab).

  - Week 3 recap.

Learning Resources: GitHub Copilot Documentation, Microsoft Learn (Generative AI, AI-900, Azure OpenAI, Copilot Studio paths).

Week 4: Azure Essentials[8] & Capstone Project Kick-off

Objective: Cover Azure security and monitoring basics, provide a condensed DevOps overview, and dedicate the latter half of the week to initiating the Capstone Project.

Topics & Labs:

- Day[9] 1: Azure Security Fundamentals
  - Shared responsibility model, Microsoft Defender for Cloud (free tier features), Identity and Access Management (IAM)/RBAC, Network Security Groups (NSGs) review.
  - Lab 14: Review Microsoft Defender for Cloud recommendations. Explore RBAC roles.
    - Introductory Links:
    - What is Microsoft Defender for Cloud?: https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-cloud-introduction
    - What is Azure role-based access control (Azure RBAC)?: https://learn.microsoft.com/en-us/azure/role-based-access-control/overview
    - Tutorial: Improve your security posture with Microsoft Defender for Cloud: https://learn.microsoft.com/en-us/azure/defender-for-cloud/tutorial-security-posture
    - Steps (4-6):

1. Navigate to "Microsoft Defender for Cloud" in the Azure portal.
2. Review your current "Secure score" and explore the recommendations provided for your Azure resources.
3. Click on a few recommendations to understand the suggested actions (no need to implement all, just understand).
4. Navigate to a Resource Group you created earlier. Go to "Access control (IAM)."
5. View the current role assignments. Click "Add" > "Add role assignment" to see the list of available roles (e.g., Reader, Contributor, Owner). Understand their purpose (do not assign a new role unless guided).
6. Discuss the principle of least privilege.

- Day 2: Azure Monitoring Basics & Condensed DevOps Overview
  - Azure Monitoring: Azure Monitor (metrics, activity logs), Application Insights (for App Service and Functions - free tier limits), creating basic alerts.
    - Lab 15: Explore Azure Monitor metrics. View the Activity Log. Set up a simple alert.
    - Introductory Links:
    - Azure Monitor overview: https://learn.microsoft.com/en-us/azure/azure-monitor/overview
    - Metrics in Azure Monitor: https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/data-platform-metrics
    - Azure Activity Log: https://learn.microsoft.com/en-us/azure/azure-monitor/essentials/activity-log

- Create alerts for Azure services: https://learn.microsoft.com/en-us/azure/azure-monitor/alerts/alerts-create-metric-alert-rule-portal
- Steps (4-6):

1. Navigate to "Monitor" in the Azure portal.
2. Select "Metrics." Choose a resource (e.g., one of your VMs or your App Service). Select a metric to view (e.g., "Percentage CPU" for a VM, or "Requests" for an App Service).
3. Explore the Activity Log: View recent activities in your subscription. Filter[10] by resource group or operation.
4. Go to "Alerts" and create a new alert rule.
5. Define the alert condition (e.g., if Percentage CPU on a VM is greater than 80% for 5 minutes).
6. Configure an action group (e.g., to send an email notification to your address – a new action group will need to be created). Create and observe the alert (you might need to intentionally stress the VM CPU to trigger it, or set a low threshold for testing).

- Lab 16: If Application Insights was integrated, explore basic telemetry.
- Introductory Links:
- What is Application Insights?: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-insights-overview
- Application Map in Application Insights: https://learn.microsoft.com/en-us/azure/azure-monitor/app/app-map
- Live Metrics: Monitor app performance in real time: https://learn.microsoft.com/en-us/azure/azure-monitor/app/live-metrics
- Steps (4-6):

1. If Application Insights was enabled for your App Service (often done by default or with a toggle during creation), navigate to the Application Insights resource associated with your app.
2. Explore the "Application map" to visualize the components of your application and their interactions.
3. View "Live Metrics" while sending some requests to your web app to see real-time telemetry.
4. Look into "Failures" to see if any exceptions or failed requests have been logged.
5. Explore "Performance" to understand response times for different operations.
6. Try writing a simple Kusto Query Language (KQL) query in "Logs" to retrieve specific traces or requests (e.g., requests | take 10).

- DevOps Concepts & Azure Repos Overview (Condensed):
- What is DevOps?: Core principles.
- Version Control with Git: Brief overview.
- Introduction to Azure Repos for source code management.

- Introductory Links:
- What is DevOps?: https://azure.microsoft.com/en-us/solutions/devops/
- What is Azure Repos?: https://learn.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos
- Git documentation: https://git-scm.com/doc
- Brief Demo/Self-Study (No dedicated lab steps, focus on concepts for Capstone):
- Show how to create a new project in Azure DevOps.
- Demonstrate initializing an Azure Repo or importing an existing Git repository.
- Explain the concept of branching, committing, and pushing changes.
- Briefly introduce Azure Pipelines conceptually as a way to automate builds and deployments (covered more in self-study).
- Day 3: Capstone Project - Phase 1: Ideation, Planning & Team Formation
  - Introduction, brainstorming, team formation, project planning, proposal outline.
- Day 4: Capstone Project - Phase 2: Design & Initial Setup
  - Detailed design, setting up dev environment, provisioning initial Azure services, setting up Git repos.
- Day 5: Capstone Project - Phase 3: Early Development & Prototyping
  - Focused development, building prototypes, troubleshooting, mentor check-in.

Learning Resources: Azure Security & Monitor documentation, Azure DevOps basics on Microsoft Learn.

Week 5: Capstone Project Deep Dive & Showcase

Objective: Dedicate the entire week for students to develop, refine, test, document, and present their Capstone Projects.

(No specific new labs this week, as the focus is entirely on the Capstone Project development, drawing from all previously learned concepts and labs.)

Introductory Links for Capstone Development (General):

- Revisit links from previous weeks based on the Azure services chosen for the project.
- Effective presentations: https://www.microsoft.com/en-us/microsoft-365-life-hacks/presentation-skills/effective-presentation-skills
- Writing good documentation: (Many community guides available, e.g., Google "technical documentation best practices")
- Azure Architecture Center (for design patterns and best practices): https://learn.microsoft.com/en-us/azure/architecture/

Daily Focus:

- Day 1: Capstone Project - Phase 4: Core Development & Feature Implementation
- Day 2: Capstone Project - Phase 5: Testing, Debugging & Iteration
- Day 3: Capstone Project - Phase 6: Finalizing Development, Documentation & Presentation Prep
- Day 4: Capstone Project - Phase 7: Project Showcase & Presentations (Part 1)
- Day 5: Capstone Project - Phase 8: Project Showcase (Part 2), Internship Wrap-up & Future Learning

---

This detailed plan should provide a solid framework for your 5-week internship. Remember to adjust the complexity of the labs based on the students' progress and leverage the weekly Teams calls effectively.