GITHUB COPILOT

Intermediate/Advanced Level Projects

TABLE OF CONTENTS

Advanced GitHub Copilot Usages

The project demonstration and exercises will include the below GitHub copilot concepts as needed:

Using GitHub Copilot Effectively

1. Writing Code with Copilot

Generating code snippets

Code suggestions and completions

Editing and refining Copilot-generated code

2. Advanced Copilot Features

Slash commands for code generation and modification

Utilizing context variables to improve suggestions

Engaging with @chat participants for collaborative coding

Mastering Copilot Chat Commands

3. Slash Commands

List of available slash commands

4: Context Menu Commands

Using context menu commands for code navigation

Integrating commands into a workflow

5. Additional Features

- *Voice Support*
  - *Voice Commands*
- *Enhanced Code Suggestions*
  - *Multi-Line Completions*
  - *Smarter Autocompletion*
- *Debugging Assistance*
  - *Real-Time Error Detection*
- *Customize Copilot's Output*
  - *User Preferences*

Project 1: Data Analysis Pipeline with Python and Pandas

Duration: 3 hours 30 mins (with 20 mins break)

Tasks:

1. ***Data Loading and Preprocessing:***
   - *Load a large dataset using Pandas.*
   - *Handle missing values and perform data cleaning.*
   - *Transform data as needed (e.g., normalization, encoding).*
   - *Use Copilot to assist in writing functions for data cleaning (handling missing values, removing duplicates)*
2. ***Exploratory Data Analysis (EDA):***
   - *Perform descriptive statistics.*
   - *Generate visualizations to uncover insights (e.g., histograms, scatter plots).*
3. ***Visualization:***
   - *Use Matplotlib or Seaborn to create visualizations.*
   - *Plot relationships between variables and highlight key findings.*
4. ***Advanced Analysis:***
   - *Perform time series analysis or cohort analysis*
   - *Use Copilot to help write complex data aggregation and transformation operations*
5. ***Summary Report:***
   - *Generate a summary report of the analysis.*
   - *Use Jupyter Notebook to present the findings (optional).*

1. Target Audience:

- ***Junior and Intermediate Data Analysts****: Professionals who have some experience with data analysis but are looking to improve their efficiency and productivity using GitHub Copilot.*

- ***Software Engineers or Developers****: Those who want to integrate data analysis capabilities into their projects.*

- ***Data Science Enthusiasts****: Beginners or intermediate learners looking to build their skills in data wrangling, visualization, and analysis, while leveraging AI tools like GitHub Copilot..*

2. Summary of Agenda and Benefits:

- ***Data Loading and Preprocessing (40 minutes)****:*

  - *Introduction to using Pandas for loading large datasets and preprocessing (missing values, encoding).*

  - *How GitHub Copilot assists in creating efficient data-cleaning functions, saving time in mundane tasks.*

  - ***Benefits****: Increased productivity in cleaning and preparing data by automating repetitive tasks like removing duplicates, filling missing values, etc.*

- ***Exploratory Data Analysis (EDA) (50 minutes)**:*
    - *Conducting descriptive statistics to uncover trends, distributions, and anomalies.*
    - *Use GitHub Copilot to generate quick insights and visualizations like histograms, scatter plots, etc.*
    - ***Benefits**: Reduces the manual effort required for generating visualizations, enabling faster iteration and discovery of patterns in data.*
- ***Visualization (45 minutes)**:*
    - *Utilizing Matplotlib or Seaborn for creating clear and insightful data visualizations.*
    - *Copilot helps write plotting functions quickly, especially when dealing with multiple variables.*
    - ***Benefits**: Users will experience how Copilot can suggest optimized code for visualizations and improve workflow by reducing time spent on repetitive plotting tasks.*
- ***Advanced Analysis (50 minutes)**:*
    - *Performing advanced data aggregation techniques, time series analysis, and cohort analysis.*
    - *GitHub Copilot can assist with complex data operations such as pivoting tables, performing rolling statistics, etc.*
    - ***Benefits**: Learn how Copilot can streamline writing complex operations and handle advanced analysis with less manual coding effort.*
- ***Summary Report (25 minutes)**:*
    - *Generate a final report of the findings using Pandas and Jupyter Notebook.*
    - *Copilot assists in the creation of markdown summaries or sections of the analysis, especially useful for preparing presentations or reports.*
    - ***Benefits**: Automates the creation of structured reports, ensuring that the time spent on documentation is minimized.*

3. Participation Requirements:

- ***Basic Knowledge** of Python programming (understanding of variables, loops, and functions).*
- ***Familiarity with Pandas**: Participants should have used Pandas before, at least for basic data loading and manipulation.*
- ***Basic Understanding of Data Analysis**: Should have some experience in handling datasets, performing descriptive statistics, and creating visualizations.*

- ***Environment Setup****: Participants should have Python installed along with Jupyter Notebook and libraries like Pandas, Matplotlib, and Seaborn.*
- ***GitHub Copilot Access****: Participants should have GitHub Copilot enabled in their IDE (such as VSCode) to follow along with the session.*

Project 2: Full-Stack Web Application with React and Node.js

Duration: 3 hours and 30 mins (with 20 mins break)

Tasks:

1. ***Frontend Setup:***
    - *Create a new React project using the Create React App.*
    - *Design a basic user interface with components.*
2. ***Backend Development:***
    - *Set up a Node.js server with Express.*
    - *Create API endpoints for user authentication and resource management.*
3. ***User Authentication:***
    - *Implement user registration and login.*
    - *Use JWT for authentication and authorization.*
4. ***Frontend-Backend Integration:***
    - *Connect React frontend with Node.js backend.*
    - *Implement API calls in React using Axios or Fetch API.*
5. ***Deployment:***
    - *Deploy the full-stack application.*
    - *Ensure the application is running smoothly.*

1. Target Audience:

- ***Frontend and Backend Developers (Beginner to Intermediate)**: Developers with basic knowledge of either frontend or backend development, looking to enhance their skills by building a full-stack application.*

- ***Full-Stack Developers**: Professionals or students aiming to improve their knowledge of integrating React with Node.js, along with deploying web applications.*

- ***Software Engineers transitioning to Web Development**: Those interested in learning the complete workflow of building and deploying full-stack applications.*

2. Summary of Agenda and Benefits:

- ***Frontend Setup (40 minutes)**:*

    - *Introduction to Create React App for quick project setup.*

    - *Create a basic UI with React components such as navigation, forms, and basic styling.*

    - ***Use GitHub Copilot to assist with creating components, JSX structure, and state management**.*

    - ***Benefits**: Speeds up React component creation, reduces time spent on repetitive code and helps developers follow best practices.*

- ***Backend Development (50 minutes)**:*

    - *Set up a Node.js server using Express for handling requests and responses.*
    - *Create simple API endpoints for user authentication and resource management.*
    - ***Use GitHub Copilot to help scaffold Express routes and middleware functions**.*
    - ***Benefits**: Automates the generation of boilerplate code for API endpoints, improving productivity in backend development.*
- ***User Authentication (50 minutes)**:*
    - *Implement user registration and login using JWT for secure authentication.*
    - ***Use Copilot to generate code for JWT token handling, middleware for protected routes, and user validation logic**.*
    - ***Benefits**: Copilot reduces the manual effort required to implement JWT authentication, which often involves repetitive code patterns and security checks.*
- ***Frontend-Backend Integration (40 minutes)**:*
    - *Connect the React frontend to the Node.js backend by making API calls.*
    - *Implement Axios or Fetch for making HTTP requests and updating the front end based on responses.*
    - ***Use Copilot to assist in writing efficient API calls and handling responses in React**.*
    - ***Benefits**: Streamlines the integration process between frontend and backend, saving time when connecting React components to APIs.*
- ***Deployment (30 minutes)**:*
    - *Deploy the full-stack application to a platform like Vercel (optional for frontend) and Heroku/Render (optional for backend).*
    - *Ensure the application runs smoothly, with both the frontend and backend communicating successfully.*
    - ***Use Copilot to write deployment scripts and configure settings**.*
    - ***Benefits**: Helps in setting up deployment scripts, environment variables, and configuration settings, which can be tedious for first-time deployers.*

3. Participation Requirements:

- ***Basic Knowledge of JavaScript**: Understanding JavaScript concepts such as variables, functions, and asynchronous programming.*
- ***Familiarity with React**: Participants should know the basics of React, including components, props, and state management.*

- ***Familiarity with Node.js**: Some exposure to creating a basic server with Express is preferred.*
- ***Tools Setup**:*
  - *Node.js and npm installed.*
  - *VSCode (or another IDE) with GitHub Copilot enabled.*
  - *Create React App, Express.js, Axios, or Fetch installed.*
- ***GitHub Copilot Access**: Participants should have Copilot enabled to benefit from its code-suggestion features during the session.*

Project 3: Machine Learning Model Deployment with Flask

Duration: 3 hours and 30 mins (with 20 mins break)

Tasks:

1. ***Model Training:***
   - *Load and preprocess data.*
   - *Train a machine learning model using Scikit-Learn.*
   - *Save the trained model using joblib or pickle.*
2. ***Flask Application:***
   - *Set up a Flask application.*
   - *Create API endpoints for model predictions.*
   - *Load the saved model and use it for predictions.*
3. ***Testing:***
   - *Test the Flask API locally.*
   - *Ensure the model is making correct predictions.*
4. ***Deployment:***
   - *Deploy the Flask application.*
   - *Test the deployed application to ensure it is working correctly.*

1. Target Audience:

- ***Data Scientists and Machine Learning Engineers****: Professionals looking to learn how to serve machine learning models through APIs.*

- ***Backend Developers****: Those who want to expand their skills by integrating machine learning into backend services using Flask.*

- ***Full-Stack Developers****: Developers interested in incorporating ML-based APIs into their applications.*

2. Summary of Agenda and Benefits:

- ***Model Training (45 minutes)****:*

  - *Load and preprocess a dataset using Scikit-Learn, ensuring the data is cleaned and ready for model training.*

  - *Train a machine learning model (e.g., decision tree, random forest, etc.) using Scikit-Learn.*

  - *Save the trained model for future use using joblib or pickle.*

  - ***Use GitHub Copilot to assist in writing preprocessing functions and model training code****.*

  - ***Benefits****: Copilot speeds up the process of writing common preprocessing functions and model training loops, reducing errors and saving time.*

- ***Flask Application (50 minutes)**:*
    - *Set up a Flask web application that will serve the trained model via an API.*
    - *Create API endpoints that accept user input (e.g., JSON data) and return model predictions.*
    - *Load the previously saved model within the Flask app to make predictions on new data.*
    - ***Use GitHub Copilot to generate the Flask setup and API endpoint structure quickly**.*
    - ***Benefits**: Copilot helps in scaffolding Flask routes and model-loading code, allowing participants to focus on the logic rather than repetitive setup tasks.*
- ***Testing (40 minutes)**:*
    - *Test the Flask API locally by sending POST requests and ensuring that the model makes accurate predictions.*
    - *Use tools like Postman or cURL to simulate API requests and check the API's responses.*
    - ***Use Copilot to write test cases for checking the accuracy and stability of the API predictions**.*
    - ***Benefits**: GitHub Copilot assists in writing test cases, streamlining the testing process, and ensuring coverage for various scenarios.*
- ***Deployment (45 minutes)**:*
    - *Deploy the Flask application on platforms like Heroku or Render.*
    - *Test the deployed API to ensure it is accessible, making correct predictions, and running smoothly in production.*
    - ***Use GitHub Copilot to help with deployment configurations and environment variable setup**.*
    - ***Benefits**: Copilot aids in writing deployment scripts and environment configurations, which can save time and reduce deployment errors.*

3. Participation Requirements:

- ***Basic Knowledge of Python**: Participants should be comfortable with Python programming, including libraries like Pandas and Scikit-Learn.*
- ***Familiarity with Flask**: Basic understanding of creating and running a Flask application.*
- ***Basic Machine Learning Concepts**: Participants should know how machine learning models work and have some experience in training models with Scikit-Learn.*
- ***Tools Setup**:*

    - *Python, Flask, and Scikit-Learn installed.*
    - *Postman or cURL for testing the API locally.*
    - *GitHub Copilot enabled in their IDE (e.g., VSCode).*
- ***Optional****: Accounts on platforms like Heroku or Render for deploying the Flask application.*

Project 4: Automated Testing for Web Application using Selenium

Duration: 3 hours and 30 mins (with 20 mins break)

Tasks:

1. ***Install necessary tools and libraries:***
   - *Java*
   - *Selenium WebDriver*
   - *Browser driver (e.g., ChromeDriver)*
2. ***Writing Basic Test Scripts***
   - *Set up the project structure*
   - *Write a basic test script to open a web page and verify its title*
   - *Implement test cases for critical user flows (e.g., login, navigation)*
3. ***Advanced Test Cases and Browser Compatibility***
   - *Write test scripts for different browsers (e.g., Firefox, Safari)*
   - *Implement test cases for more complex user interactions (e.g., form submissions, AJAX requests)*
4. ***Integrating Selenium Tests with CI/CD***
   - *Set up a CI/CD pipeline to run Selenium tests automatically*
   - *Use GitHub Actions for Integration*

1. Target Audience:

- ***QA Engineers and Test Automation Developers**: Professionals familiar with Java, looking to learn or enhance their skills in automated testing using Selenium.*

- ***Java Developers**: Developers who want to integrate test automation into their workflow to ensure the quality of web applications.*

- ***DevOps Engineers**: Those interested in incorporating automated tests in a CI/CD pipeline with Java and Selenium.*

2. Summary of Agenda and Benefits:

- ***Install Necessary Tools and Libraries (40 minutes)**:*

  - *Install Java, Selenium WebDriver, and a browser driver like Chrome Driver.*

  - *Set up the working environment, including Maven for dependency management.*

  - ***Use GitHub Copilot to help scaffold the project structure and Selenium setup code**.*

  - ***Benefits**: Copilot will assist in setting up project dependencies, avoiding manual setup errors, and ensuring proper configuration of the Java environment for Selenium.*

- ***Writing Basic Test Scripts (50 minutes)**:*

  - *Set up the project structure with proper package organization.*

    - *Write a basic Selenium test script in Java to open a web page, verify its title, and implement test cases for critical user flows (e.g., login and navigation).*
    - ***Use Copilot to assist in generating Selenium test scripts, simplifying the process of writing verification logic****.*
    - ***Benefits****: Copilot speeds up the process of writing basic test scripts by auto-suggesting common methods and assertions, reducing development time.*

- ***Advanced Test Cases and Browser Compatibility (50 minutes)****:*
    - *Write test scripts for different browsers (e.g., Firefox, Safari) to ensure browser compatibility.*
    - *Implement test cases for more complex user interactions such as form submissions, AJAX requests, and dynamic content.*
    - ***Use Copilot to help generate browser-specific test scripts and complex interactions with Selenium WebDriver****.*
    - ***Benefits****: Copilot will help reduce the manual effort required to write cross-browser test cases and handle complex interactions.*

- ***Integrating Selenium Tests with CI/CD (45 minutes)****:*
    - *Set up a CI/CD pipeline with GitHub Actions to automatically run Selenium tests after code changes.*
    - *Ensure proper integration with the Selenium WebDriver in the CI environment.*
    - ***Use Copilot to generate GitHub Actions configuration files for running Selenium tests****.*
    - ***Benefits****: Copilot can help in setting up the pipeline with minimal effort, reducing errors and ensuring that tests are continuously executed as part of the development lifecycle.*

3. Participation Requirements:

- ***Java Knowledge****: Participants should be comfortable with Java programming, including object-oriented concepts.*
- ***Familiarity with Web Applications****: Understanding of basic web navigation and user interactions.*
- ***Tools Setup****:*
    - *Java installed along with Maven for dependency management.*
    - *Selenium WebDriver and browser drivers (ChromeDriver, GeckoDriver for Firefox, etc.).*
    - *GitHub Copilot enabled in their IDE (e.g., IntelliJ).*

- ***Optional**: Familiarity with GitHub Actions for CI/CD integration.*

Project 5: Serverless ETL Pipeline

Duration: 3 hours and 30 mins (with 20 mins break)

Tasks:

1. ***Data Source Setup:***
   - *Set up a sample data source (e.g., S3 bucket with CSV files)*
   - *Create a trigger for new data arrivals*
2. ***ETL Function:***
   - *Develop a serverless function (e.g., AWS Lambda, Azure Functions) for data transformation*
   - *Use Copilot to assist in writing data parsing and transformation logic*
3. ***Data Storage:***
   - *Set up a data warehouse or database for transformed data*
   - *Implement efficient data insertion or upsert operations*
4. ***Orchestration:***
   - *Create a workflow to coordinate the ETL process (e.g., AWS Step Functions, Azure Logic Apps)*
   - *Implement error handling and retries*
5. ***Monitoring and Logging:***
   - *Set up CloudWatch/Grafana or equivalent for monitoring the ETL pipeline*
   - *Implement detailed logging for each step of the process*

1. Target Audience:

- ***Data Engineers and Cloud Developers****: Professionals looking to build serverless ETL pipelines using cloud services and Prisma ORM.*

- ***Full-Stack Developers****: Developers who want to integrate database management using Prisma ORM in their data processing workflows.*

2. Summary of Agenda and Benefits:

- ***Data Source Setup (30 minutes)****:*

  - *Set up a sample data source (e.g., an S3 bucket with CSV files).*

  - *Create a trigger for new data arrivals using an event-driven approach.*

  - ***Use Copilot to assist in setting up the S3 event trigger code****.*

  - ***Benefits****: Automated event detection and trigger setup for processing new data.*

- ***ETL Function (50 minutes)****:*

  - *Develop a serverless function (e.g., AWS Lambda) to handle data extraction, transformation, and loading (ETL).*

  - *Parse and transform incoming data using Lambda.*
  - ***Use Copilot to write parsing and transformation logic****.*
  - ***Benefits****: Copilot assists in writing optimized ETL logic, improving productivity and reducing manual errors.*
- ***Data Storage with Prisma ORM (40 minutes)****:*
  - *Set up a database using Prisma ORM (SQLite, PostgreSQL, MySQL, etc.) for storing the transformed data.*
  - *Use Prisma's schema definitions to structure your data models and manage database operations like Upsert.*
  - ***Use Copilot to assist in writing Prisma queries for efficient data insertion/upserts****.*
  - ***Benefits****: Prisma simplifies complex database interactions, ensuring seamless and efficient data management.*
- ***Orchestration (45 minutes)****:*
  - *Use AWS Step Functions to orchestrate the ETL process.*
  - *Implement retries and error handling for robust data processing.*
  - ***Use Copilot to streamline the orchestration setup****.*
  - ***Benefits****: Reduces development time for orchestration and increases pipeline reliability.*
- ***Monitoring and Logging (30 minutes)****:*
  - *Set up Grafana or similar tools for monitoring the ETL pipeline.*
  - *Implement detailed logging to track the flow of data through the pipeline.*
  - ***Use Copilot to write logging and monitoring code****.*
  - ***Benefits****: Enhanced observability and troubleshooting for ETL operations.*

3. Participation Requirements:

- ***Familiarity with Cloud Services****: Basic knowledge of AWS services like Lambda, S3, and Step Functions.*
- ***Basic Prisma ORM Knowledge****: Understanding of using Prisma for database interactions.*
- ***Tools Setup****:*
  - *AWS account, Prisma ORM setup, PostgreSQL/MySQL database ready.*
  - *Grafana or other monitoring tools set up for pipeline health visualization.*

    - *GitHub Copilot enabled in their IDE (e.g., VSCode).*
- ***Optional**: Experience with SQL or database schema design.*

Project 6: Developing Microservices Architecture with Docker and K8s

Duration: 3 hours and 30 mins (with 20 mins break)

Tasks:

1. ***Microservices Design:***
   - *Design a simple microservices architecture (e.g., user service, product service, order service)*
   - *Create basic implementations of each service using a preferred language/framework*
2. ***Dockerization:***
   - *Write Dockerfiles for each microservice*
   - *Use Copilot to assist in creating docker-compose.yml for local development*
3. ***Kubernetes Deployment:***
   - *Create Kubernetes deployment YAML files for each service*
   - *Implement service and ingress resources*
4. ***Service Communication:***
   - *Implement inter-service communication (e.g., using REST or gRPC)*
   - *Use Copilot to help write client code for service-to-service calls*
5. ***Monitoring and Logging:***
   - *Integrate a basic monitoring solution (e.g., Prometheus)*
   - *Implement centralized logging (e.g., using ELK stack or Cloud-native solutions)*

1. Target Audience:

- **Backend and Full-Stack Developers**: Those looking to learn microservices with a modern, high-performance framework.

- **DevOps Engineers**: Interested in managing containerized microservices with Docker and Kubernetes.

- **JavaScript Developers**: Those familiar with JavaScript who want to use it in a high-performance backend setup.

2. Summary of Agenda and Benefits:

- **Microservices Design (45 minutes)**:

  - Design a simple architecture with services (user, product, order) using Fastify in JavaScript.

  - **Benefits**: Fastify offers fast routing and JSON schema support for efficient API handling, and Copilot assists in generating boilerplate service code.

- **Dockerization (45 minutes)**:

  - Write Dockerfiles for each service.

  - Use docker-compose.yml for local orchestration.

    - **Benefits**: Fastify’s lightweight footprint makes it ideal for containerization, and Copilot simplifies Docker configuration.

- **Kubernetes Deployment (1 hour)**:
    - Create Kubernetes YAML files for deploying each Fastify service.
    - Configure services and ingress resources.
    - **Benefits**: Copilot aids in the setting up of Kubernetes resources, and Fastify is resource-efficient, enhancing scalability.
- **Service Communication (45 minutes)**:
    - Implement inter-service RESTful communication.
    - **Benefits**: Fastify’s async capabilities and Copilot’s code generation make setting up service-to-service calls smoother.
- **Monitoring and Logging (45 minutes)**:
    - Integrate Prometheus and use a centralized logging solution.
    - **Benefits**: Fastify’s plugin ecosystem supports flexible logging and monitoring.

3. Participation Requirements:

- **JavaScript Knowledge**: Familiarity with JavaScript, including async/await.
- **Setup Requirements**:
    - Docker, Kubernetes CLI, and GitHub Copilot.
    - Fastify and Node.js installed.