# 🧠 Final Project – Code Analysis System as Part of wit push

**Backend Development – Final Project**

🎉 Congratulations! You’ve been hired by a startup company called **CodeGuard**!  
Your mission is to develop a code analysis system that integrates with the wit push command to ensure high-quality code is maintained across all commits.

## 🧱 System Overview

### 🎯 Project Goal:

Build a backend system that automatically analyzes Python files every time the user runs wit push. The system will detect common code quality issues and return visual graphs with insights.

This simulates a basic form of internal **CI (Continuous Integration)** focused on code quality.

### 📦 Technologies:

* **Language**: Python
* **Server**: FastAPI
* **Code Analysis**: ast (Abstract Syntax Tree)
* **Visualization**: matplotlib

## 🛠️ Project Requirements

### 📂 System Components:

1. **wit**  
   A simplified version control system that includes the following commands:  
   init, add, commit, log, and **push**.
2. **FastAPI Server**  
   A backend server that listens to wit push requests and performs:
   * Code analysis using ast
   * Issue detection
   * Graph generation (as images)

## 🌐 API Endpoints

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /analyze | POST | Accepts files and returns graphs |
| /alerts | POST | Accepts files and returns issue warnings |

## 🔍 Code Quality Checks (using AST)

The server should perform the following checks for each pushed file:

* **Function Length**:  
  Warn if a function is longer than 20 lines.
* **File Length**:  
  Warn if the entire file is longer than 200 lines.
* **Unused Variables**:  
  Warn if a variable is assigned but never used.
* **Missing Docstrings**:  
  Warn if a function has no documentation string.

## 📊 Visual Graphs (Matplotlib)

The server should generate and return the following graphs based on the code analysis:

1. **Histogram** – Distribution of function lengths
2. **Pie Chart** – Number of issues per issue type
3. **Bar Chart** – Number of issues per file
4. **BONUS**! consider implementing a line graph to track the number of issues over time.

Graphs should be returned as PNG files (or links) after the analysis.

## 🗂️ Documentation Requirements

Include a README.md file with:

* Overview of the project
* Installation and execution instructions
* Folder structure of the project
* Explanation of each endpoint

## 🚀 Bonus Challenge (Optional)

As an extra feature, impress your team lead by identifying variables written in **non-English** letters (e.g., Hebrew variable names like מספר instead of number), and return a warning about them.

## 📅 Project Submission

* All code should be uploaded to a **separate GitHub repository** dedicated to this final project.
* **GitHub username for submission**: HadassaAvimorNew