# Programming Languages Course Project Proposal

## Comparing Concurrent Programming Models: Prime Number Finding in Go, Python, and Java

**Student Name:** Jnana Krishnamsetty
**Course:** ITCS 4102/5102
**Team:** Individual Project
**Date:** June 27, 2025

# 1 Executive Summary

This project will compare concurrent programming approaches by implementing a prime number finder in three languages: Go (goroutines), Python (threading vs asyncio), and Java (threads vs CompletableFuture). The implementation will find all prime numbers in a given range using different concurrency models, allowing for direct performance and complexity comparison.

# 2 Project Topic and Scope

## 2.1 Overview

Implement a concurrent prime number finder that:

- Divides a number range into chunks
- Processes chunks concurrently
- Compares different concurrency approaches
- Measures performance and resource usage

## 2.2 Objectives

- Understand different concurrency models
- Implement the same algorithm in three languages
- Compare performance metrics
- Document best practices for each approach
- Provide practical recommendations

## 2.3 Scope

- Simple prime checking algorithm (trial division)
- Range: 1 to 1,000,000
- Concurrent chunk processing
- Performance measurement

- Results aggregation

# 3 Tools and Development Environment

## 3.1 Programming Languages

- **Go 1.22**: Using goroutines and channels
- **Python 3.12**: Comparing threading and multiprocessing
- **Java 21**: Using Thread pools and CompletableFuture

## 3.2 Development Tools

- **IDE**: Visual Studio Code with language extensions
- **Version Control**: GitHub repository
- **Testing**: Built-in testing frameworks for each language

# 4 Libraries and APIs

## 4.1 Topic 1: Language Selection

- **Go**: Native concurrency with goroutines
- **Python**: Threading and multiprocessing libraries
- **Java**: java.util.concurrent package

## 4.2 Topic 2: Development Tools

- **Git/GitHub**: Version control
- **VS Code**: Cross-language development
- **Docker**: Optional containerization

## 4.3 Topic 3: Required Libraries

- **Go**: Standard library only
- **Python**: threading, multiprocessing, time
- **Java**: java.util.concurrent.*

# 5 Main Tasks and Timeline

## 5.1 Week 1: Setup and Basic Implementation (25%)

**Milestone 1: Single-threaded implementations**

- Set up development environment
- Implement basic prime checking in all languages

- Create project structure
- **Deliverable**: Working single-threaded versions

### 5.2 Week 2: Go Concurrent Implementation (20%)

**Milestone 2: Go with goroutines**

- Implement worker pool with goroutines
- Add channels for communication
- Benchmark different worker counts
- **Deliverable**: Concurrent Go implementation

### 5.3 Week 3: Python Implementations (20%)

**Milestone 3: Python threading vs multiprocessing**

- Implement with threading
- Implement with multiprocessing
- Compare GIL impact
- **Deliverable**: Two Python implementations

### 5.4 Week 4: Java Implementation (20%)

**Milestone 4: Java concurrency**

- Implement with ExecutorService
- Try CompletableFuture approach
- Compare thread pool sizes
- **Deliverable**: Concurrent Java implementation

### 5.5 Week 5: Analysis and Report (15%)

**Milestone 5: Final analysis**

- Run comprehensive benchmarks
- Create comparison charts
- Write final report
- **Deliverable**: 5+ page report with analysis

## 6 Deliverables

### 6.1 Final Report (5+ pages)

1. **Introduction** (1 page)

- What is concurrent programming
- Why it matters
- Prime number finding as a test case

2. **Implementation Details** (2 pages)
   - Go: Goroutines and channels
   - Python: GIL and workarounds
   - Java: Thread pools
   - Code snippets showing key differences
3. **Performance Analysis** (1 page)
   - Execution time comparisons
   - CPU and memory usage
   - Scalability with worker count
4. **Developer Experience** (1 page)
   - Ease of implementation
   - Debugging challenges
   - Error handling
5. **Conclusions** (0.5 pages)
   - Best language for CPU-bound tasks
   - Recommendations

### 6.2 Source Code

- Complete implementations
- README with instructions
- Benchmark scripts
- Test cases

## 7 Expected Learning Outcomes

- Deep understanding of concurrency models
- Practical experience with three languages
- Performance analysis skills
- Best practices for concurrent programming

# 8 References


- Donovan, A., & Kernighan, B. (2015). *The Go Programming Language*. Addison-Wesley.
- Goetz, B. et al. (2006). *Java Concurrency in Practice*. Addison-Wesley.
- Beazley, D. (2012). *Understanding the Python GIL*. PyCon Talk.
- Additional resources to be added during research.


**GitHub Repository:** https://github.com/jnanakris/concurrent-prime-finder