# INIT CLUB ONBOARDING TASKS

## Task 1 - *Terminal Velocity*

### Description of the Challenge

This task focuses on reducing friction between thought and execution by improving typing speed and terminal usage.

### Sub-problems

- Developing typing discipline
- Learning essential terminal commands

### Easy Level

- Complete a touch-typing course
- Achieve >40 WPM with >95% accuracy

### Advanced Level

- Master command-line navigation and piping
- Use terminal commands regularly for workflow

### Pre-requisites

- Keyboard and terminal access

### Learning Outcomes

- Improved productivity
- Comfort with command-line tools

### Expected Submission

- Screenshot of typing speed profile

### Evaluation Criteria

- Accuracy and speed achieved
- Command familiarity

# Task 2 - *Linux Migration*

## Description of the Challenge

To understand software systems, one must control the environment they run on. This task introduces participants to Linux by setting up a developer-friendly operating system.

### Sub-problems:

- Choosing the correct installation method
- Configuring the Linux environment correctly

#### Easy Level

- Install Linux using Dual Boot **or** WSL
- Verify installation using basic terminal commands

#### Advanced Level

- Use Linux as the primary development environment
- Explore file permissions and running process.

### Pre-requisites

- Personal laptop
- Basic computer usage knowledge

### Learning Outcomes

- Confidence in system-level setup
- Familiarity with Linux environments

### Expected Submission

- Screenshot of neofetch running in the Linux terminal
- GitHub submission link

### Evaluation Criteria

- Successful installation
- Correct configuration
- Valid proof submission

# Task 3 - *Algorithm Tour*

## Description of the Challenge

Participants implement the same algorithms across multiple programming languages to observe how identical logic is expressed, constrained, and reasoned about in different execution environments.

**Sub-problems:**

- Implementing Binary Search
- Implementing Merge Sort
- Comparing language-level differences

## Easy Level

Implement one of the following algorithms in a single programming language:

- Binary Search
- Merge Sort

## Advanced Level

Implement both algorithms in: C/C++, Python, and JavaScript Participants should document observed differences in:

- Control flow and expressiveness
- Memory usage patterns
- Error handling and safety guarantees
- Debugging experience (use of breakpoints and logging is highly recommended.

## Pre-requisites

- Basic programming knowledge

## Learning Outcomes

- Algorithmic thinking
- Cross-language reasoning
- Understanding implementation trade-offs

## Expected Submission

- GitHub Gist containing all implementations
- Brief documentation describing implementation choices and observations

## Evaluation Criteria

- Correctness of implementations,
  Clarity of explanation, Quality of comparison and reasoning

# Task 4 - *The Broken Web App*

## Description of the Challenge

This task introduces debugging by fixing a broken frontend–backend interaction in a web application.

### Sub-problems:

- Inspecting network requests
- Fixing incorrect API communication

### Easy Level

- Identify why the submit button fails
- Fix the communication issue

### Advanced Level

- Debug using Browser DevTools
- Ensure data is correctly sent and store

### Pre-requisites

- Basic HTML/JavaScript knowledge

### Learning Outcomes

- Debugging skills
- Understanding client–server interaction

### Expected Submission

- Pull Request fixing the bug

### Evaluation Criteria

- Correct fix
- Debugging approach

# Task 5 - *Containerization ("It Works on My Machine")*

## Description of the Challenge

This task introduces containerization using Docker to ensure applications run consistently across systems.

## Sub-problems:

- Writing a Dockerfile
- Running multi-service applications

## Easy Level

- Containerize a simple web application

## Advanced Level

- Use docker-compose to run the app with Redis/Database

## Pre-requisites

- Linux terminal familiarity

## Learning Outcomes

- Understanding containers
- Environment consistency

## Expected Submission

- Screenshot of `docker ps` showing running containers

## Evaluation Criteria

- Successful containerization
- Correct service setup

# *Good First Issues* - Bonus task

## Description of the Challenge

Participants engage with real-world open-source projects to understand contribution workflows, collaboration norms, and external codebase standards.

## Easy Level

Explore open-source repositories to understand:

- Project scope and activity
- Contribution guidelines
- Issue labeling and triage practices

## Pre-requisites

- GitHub account
- Basic familiarity with Git workflows

## Learning Outcomes

- Open-source collaboration
- Professional technical communication
- Independent problem selection

## Expected Submission

- Link to a merged or actively reviewed Pull Request
- Brief description of the issue addressed

## Evaluation Criteria

- Relevance and impact of the contribution
- Correctness and integration quality
- Clarity and professionalism of communication