# Weekly Project Diary

# Capstone Phase 1

Project Title: Development of an Online C/C++ Compiler

Submitted by: Adam

Supervisor: Prof. Ravneet Kaur

## Week 1

### Synopsis:

Initiated the Online C/C++ Compiler project by finalizing the problem statement and understanding the challenges faced in C/C++ environment setups. Outlined key features such as runtime input support, terminal output, and responsive design.

### Research Paper Work:

Reviewed literature on web-based compilers and existing tools like OnlineGDB and Replit. Explored technical papers discussing secure code execution, compiler architecture, and real-time feedback in learning environments.

### Literature Review:

Studied case studies and articles highlighting the use of online compilers in education, and their impact on reducing the entry barrier for programming students.

### Code Work:

Set up the basic project structure using the MERN stack. Initialized React and Node.js environments with basic routing setup.

### PPT Work:

Prepared the initial project proposal presentation, including the problem definition, objectives, and a rough system architecture.

### Conclusion:

Successfully set the foundation for the project. Next week will focus on frontend UI components and deeper research on sandboxed execution.

## Week 2

### Synopsis:

Focused on frontend development and enhanced understanding of real-time code execution logic for C/C++.

### Research Paper Work:

Explored system-level compilation with g++, and studied existing methods to securely execute compiled code on the server.

### Literature Review:

Looked into containerization and sandboxing techniques for executing unsafe code securely.

### Code Work:

Integrated Monaco Editor into the React frontend. Customized theme and added syntax highlighting for C/C++.

### PPT Work:

Updated the system flow diagrams and frontend mockups for the user interface.

### Conclusion:

Frontend coding environment was successfully initialized. Next focus: backend setup for code compilation and output rendering.

## Week 3

### Synopsis:

Began implementation of backend code execution using Node.js and child processes.

### Research Paper Work:

Studied secure usage of Node.js child\_process and strategies to prevent server crashes from malicious inputs.

### Literature Review:

Reviewed case studies on compiler sandboxing using Docker and chroot environments.

### Code Work:

Built an API to accept C/C++ code, write it to a temporary file, compile it using g++, and return the output.

### PPT Work:

Added backend structure and flow to the presentation with sample compilation test cases.

### Conclusion:

Achieved successful compilation and basic output. Next: runtime input and error handling implementation.

## Week 4

### Synopsis:

Focused on supporting runtime inputs and debugging error reporting.

### Research Paper Work:

Studied solutions for piping input to compiled binaries and capturing real-time output.

### Literature Review:

Read developer blogs and forum discussions on handling dynamic I/O for C/C++ programs on web servers.

### Code Work:

Added runtime input support. Built error handling to capture compiler and runtime errors and return meaningful messages.

### PPT Work:

Demonstrated input/output flow with a UI interaction mockup.

### Conclusion:

User input feature is now functional. Improved error feedback loop is helping with debugging. Next step: integrate Xterm.js for interactive output.

## Week 5

### Synopsis:

Enhanced the UI for terminal output and began testing real-time output updates.

### Research Paper Work:

Investigated how to create a responsive CLI experience using web technologies.

### Literature Review:

Analyzed UI/UX improvements in educational code platforms to improve learning engagement.

### Code Work:

Integrated Xterm.js into the frontend for terminal-like experience. Connected Xterm with backend output.

### PPT Work:

Included screenshots of new UI and output formatting.

### Conclusion:

Frontend now mimics a real terminal. Ready for testing and optimization.

## Week 6

### Synopsis:

Security testing and sandboxing implementation took center stage this week.

### Research Paper Work:

Focused on system hardening methods, including process isolation, user permissions, and Docker-based sandboxing.

### Literature Review:

Studied how online judge platforms isolate execution environments.

### Code Work:

Implemented basic security checks and isolated compilation inside a restricted environment.

### PPT Work:

Added execution flow highlighting security layers and sandbox boundaries.

### Conclusion:

Sandboxed execution working for most test cases. More testing planned for edge cases and potential abuse inputs.

## Week 7

### Synopsis:

Worked on UI polishing and improved performance for I/O heavy programs.

### Research Paper Work:

Explored stream handling for large outputs and concurrency management.

### Literature Review:

Reviewed load balancing strategies and time/memory limits from platforms like HackerRank and Codeforces.

### Code Work:

Optimized response handling and adjusted system to support timeouts and resource caps.

### PPT Work:

Presented side-by-side comparison of optimized vs non-optimized execution time.

### Conclusion:

Performance improvements achieved. UI is smoother and more responsive. Next step: test usability and finalize core logic.

## Week 8

### Synopsis:

This week centered around real-time feedback improvements and frontend responsiveness.

### Research Paper Work:

Explored notification patterns and interaction feedback design.

### Literature Review:

Analyzed modern frontend practices to improve usability for beginner programmers.

### Code Work:

Implemented output animations and live feedback for successful/failed compilations.

### PPT Work:

Updated project scope and deliverables to reflect feature enhancements.

### Conclusion:

Compiler now feels more interactive and beginner-friendly. Project nearing minimum viable version.

## Week 9

### Synopsis:

User testing and feedback collection was initiated to refine the tool.

### Research Paper Work:

Studied usability testing methods for educational tools.

### Literature Review:

Reviewed projects focused on interactive programming education and their user engagement techniques.

### Code Work:

Fixed minor bugs based on feedback and improved messaging for error codes and runtime crashes.

### PPT Work:

Prepared demo walkthrough and test case slides.

### Conclusion:

Feedback helped improve user clarity and error handling. Project in the final stages of refinement.

## Week 10

### Synopsis:

Finalized the Online Compiler and prepared documentation and project report.

### Research Paper Work:

Wrapped up final referencing and cited key technologies and security practices used.

### Literature Review:

Compiled summaries of relevant tools, papers, and platforms used for benchmarking.

### Code Work:

Final code clean-up, added comments and documentation. Prepared deployment version.

### PPT Work:

Created final presentation slides, added architecture, features, and demo results.

### Conclusion:

Project development is complete. Tool is ready for demonstration and submission.