**Programming Language Research Reflective Report 2**

Algonquin College

School of Advanced Technology

Computer Programming

**Onur Önel**

Mazin Abou-Seido

23F\_CST8333\_360

Submitted December 06, 2023

A technical report submitted to Algonquin College in partial fulfillment of the requirements for a diploma in Computer Programming

**Learning Insights about C++**

C++ is a high-speed, object-oriented programming language. While it is often described as a "low-level" language, this highlights its capability to work closely with hardware components and perform system-level programming. I started off programming in Java, switching to C++ was a noticeable change. Unlike Java, which compiles from bytecode and runs on the Java Virtual Machine (JVM), C++ directly compiles into machine code for a specific platform. C++ compilation process includes:

* Preprocessing: Before actual compilation, the source code undergoes preprocessing. This step handles directives such as #include and #define, expanding macros and including header files.
* Compilation: The preprocessed source code is then compiled into an object file. At this stage, the compiler checks for syntax errors and translates the code into assembly language.
* Assembly: The assembler takes the assembly code produced in the compilation step and converts it into machine code, resulting in an object file.
* Linking: This is a crucial step in the C++ compilation process. All the object files, possibly from different sources, are linked together along with necessary library files to produce a single executable. The linker resolves references, ensuring that functions or variables referenced from one object file are correctly connected to their definitions in another.
* Execution: The final executable can then be run on the target machine or platform.

C++ is an evolution of the C language, included with exciting features to support high-speed applications. In comparison, it holds similarities with C, but also has similarities of higher-level languages like Java. However, unlike interpreted languages such as Python and JavaScript, C++ is compiled

**Weaknesses of the Language**

The primary drawback of learning C++ as a programming language is its complexity. It can take a very long time to become proficient in C++ if not familiar with another object-oriented programming language.

The usage of pointers is one of the main characteristics that set C++ apart from other object-oriented programming languages. When working on big projects, pointers might consume a lot of system memory, which is not desirable. Another big drawback of pointers is that pointers can be difficult to understand and use correctly, using pointers wrongly can cause a result of complex issues and unexpectedly behaviours. Using an object-oriented programming language such as C++ has a number of security risks due to features like public functions, global variables, and pointers, which is another drawback of the language.

While many people consider having control over memory management to be advantageous, allocating memory manually using pointers can be very time-consuming and easily forgotten when working on code. One major issue of C++ is that it requires manual memory management because it lacks a garbage collector feature that would automatically remove unnecessary data.

Lastly, built-in code threads are not supported by C++. Given that most other languages have this feature, which can make process slower and more complex.

C++ Topics That I Learned and Implemented

**CMake and CMakeLists.txt**

CMake stands out as an extensible, open-source system that builds process across various operating systems and compilers in a manner independent of these environments. Unlike many cross-platform systems, CMake operates the native build environment. Achieves this process with CMakeLists.txt files. These are simple configuration files located in each source directory, generating standard build files. This approach facilitates a more streamlined and adaptable build process.

**Standard Library in C++**

The C++ Standard Library is a critical component, consist of essential classes and functions. These elements are crucial for tasks such as managing data structures and input/output operations. To effectively utilize these operations, it is necessary to include appropriate classes in the project. This inclusion ensures the seamless achievement of project goals, demonstrating the library's utility in practical programming scenarios.

**External Libraries**

* **OpenGL:** This library defines an API for controlling a client application's rendering system. Its careful specification allows for hardware implementation compatibility.
* **GLFW:** An Open Source, multi-platform library for OpenGL, OpenGL ES, and Vulkan. It simplifies creating windows, contexts, surfaces, and managing input and events.
* **Dear ImGui:** A bloat-free GUI library for C++, renowned for its efficiency and portability. It generates optimized vertex buffers for rendering in any 3D-pipeline-enabled application.
* **ImPlot:** A real-time, immediate mode plotting library designed for integration with ImGui. It excels in visualizing program data and creating interactive plots with minimal integration effort

**Headers and Includes**

The use of headers and includes is a best practice in C++ programming for several reasons:

1. They significantly reduce compile time, especially in larger projects.
2. They aid in keeping the code organized, making it easier to locate and modify specific sections.
3. They allow for a clear separation of interface from implementation, enhancing the readability and maintainability of the code.

**MVC Architecture**

* **Model:** Manages the data. Model can be handling CRUD operations, ensuring data integrity and implementing business logic.
* **View:** Usually displays data to the user and collects input other operations.
* **Controller:** Interprets user input, controls Model operations, and updates the View.
* **Data Source/Entity Object:** Represents the data structure, crucial for organizing and maintaining data in the Model.

**SQLITE3**

The transition from file-based data management to a database management system (DBMS) is a significant leap in terms of data handling and efficiency. SQLITE3 was an excellent choice for many projects due to its lightweight, disk-based nature, which does not require a separate server process.

**Database Creation and Schema Design:** The first step is to create a database schema that includes the structure of the existing data stored in Dataset from Travel Records CSV file. This step involves defining tables and columns that correspond to the data fields.

**Installation and Configuration:** The process of downloading and installing SQLITE3, or a comparable database management system, is relatively simple. It generally requires the inclusion and integration of essential library files into the CMakeLists.txt file and the project. For SQLITE3, this means adding files such as sqlite3.c and sqlite3.h, which are crucial for utilizing its database capabilities.

**Data Migration:** Transitioning the data from csv file to the database is a crucial step. This can be done using import commands provided by SQLITE3, including commands like `.mode csv` and `.import.`

**Database Connection and Query Execution (MVC)**

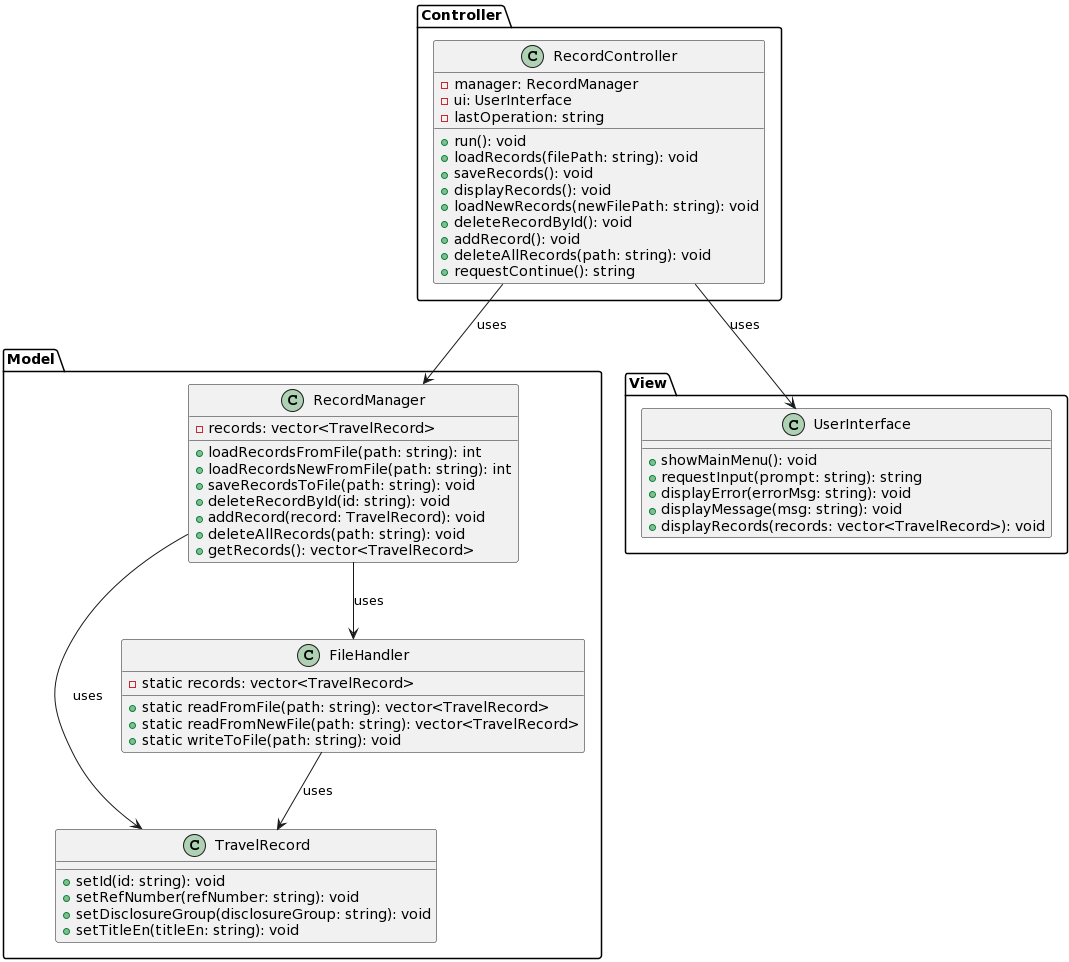
**Model-Layer Database Connection:** In the MVC (Model-View-Controller) framework, establishing a database connection is typically encapsulated within the model layer. This involves utilizing specific APIs or library functions to create and manage connections to the database.

**Controller-Layer Query Execution:** CRUD (Create, Read, Update, Delete) operations are conducted within the controller layer, following the MVC pattern. These operations are carried out using SQL queries which are prepared and executed by the controller, facilitating interaction with the model's data.

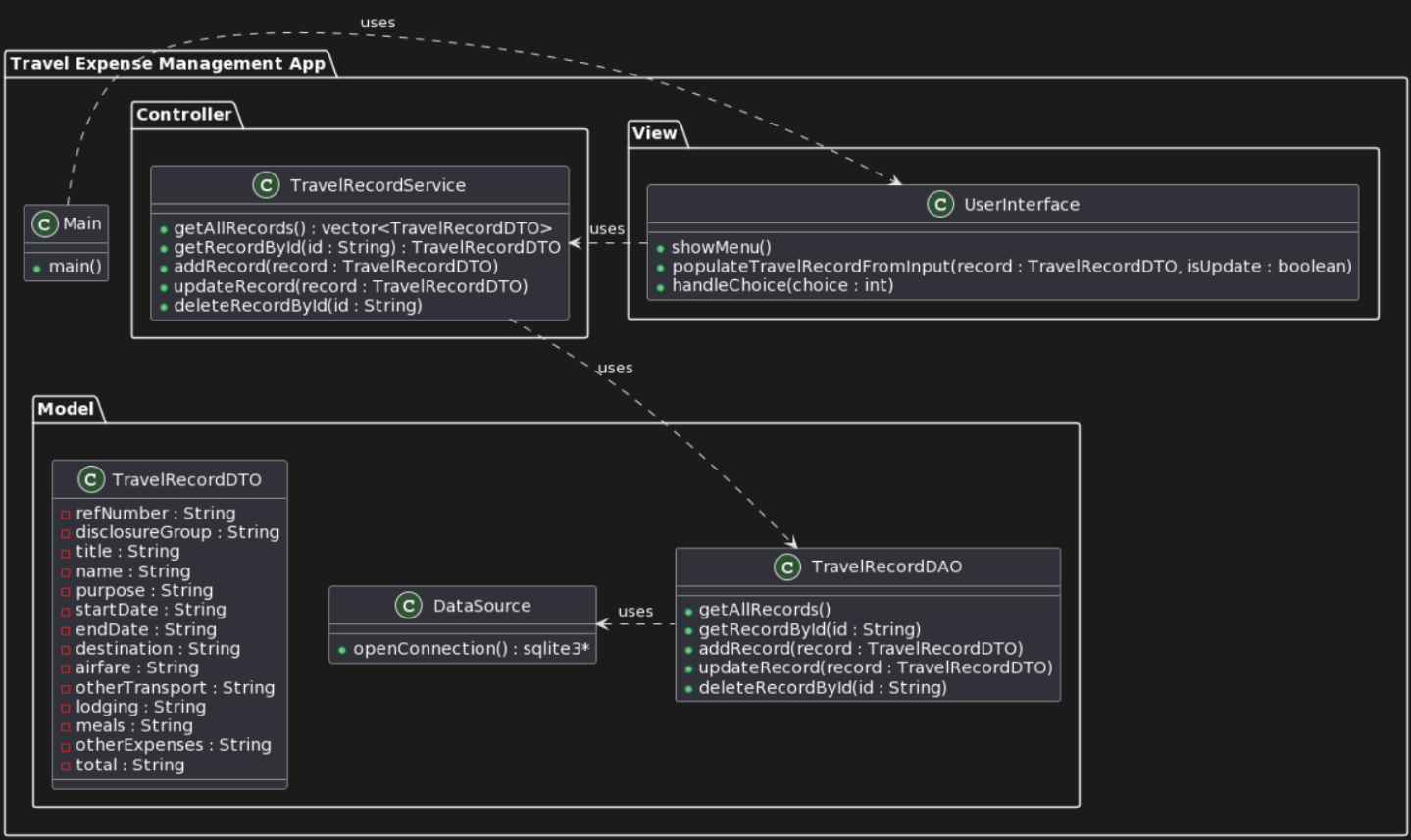
**Model-Layer Error Handling:** The model is responsible for error handling during query execution. This includes verifying the successful execution of SQL commands and managing exceptions or errors that may arise during database operations.

**Data Retrieval and Manipulation in Model:** Retrieving and manipulating data from SQLITE3 is handled in the model layer. This typically involves executing SELECT queries, processing the results, and updating or deleting records as dictated by the application’s logic.

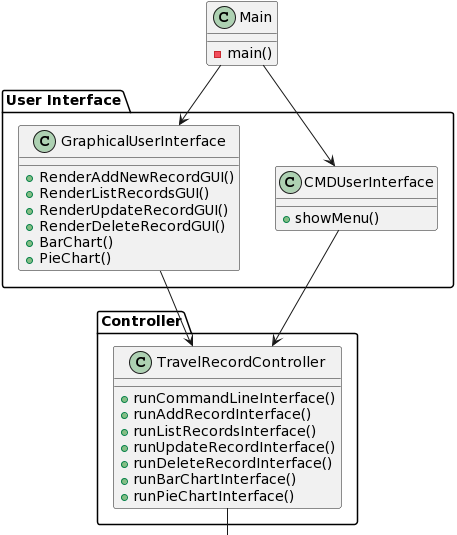
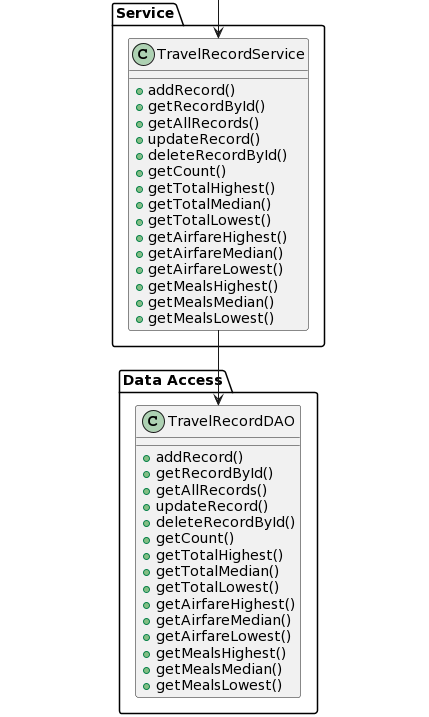
**Integration with MVC Application Logic:** The final aspect involves integrating database operations within the MVC architecture. This ensures that the application's controller and view layers can effectively interact with the model layer, which in turn interacts with the database, allowing the application to perform its intended functions efficiently.

****

**Project Version 1**



**Project Version 2**

****

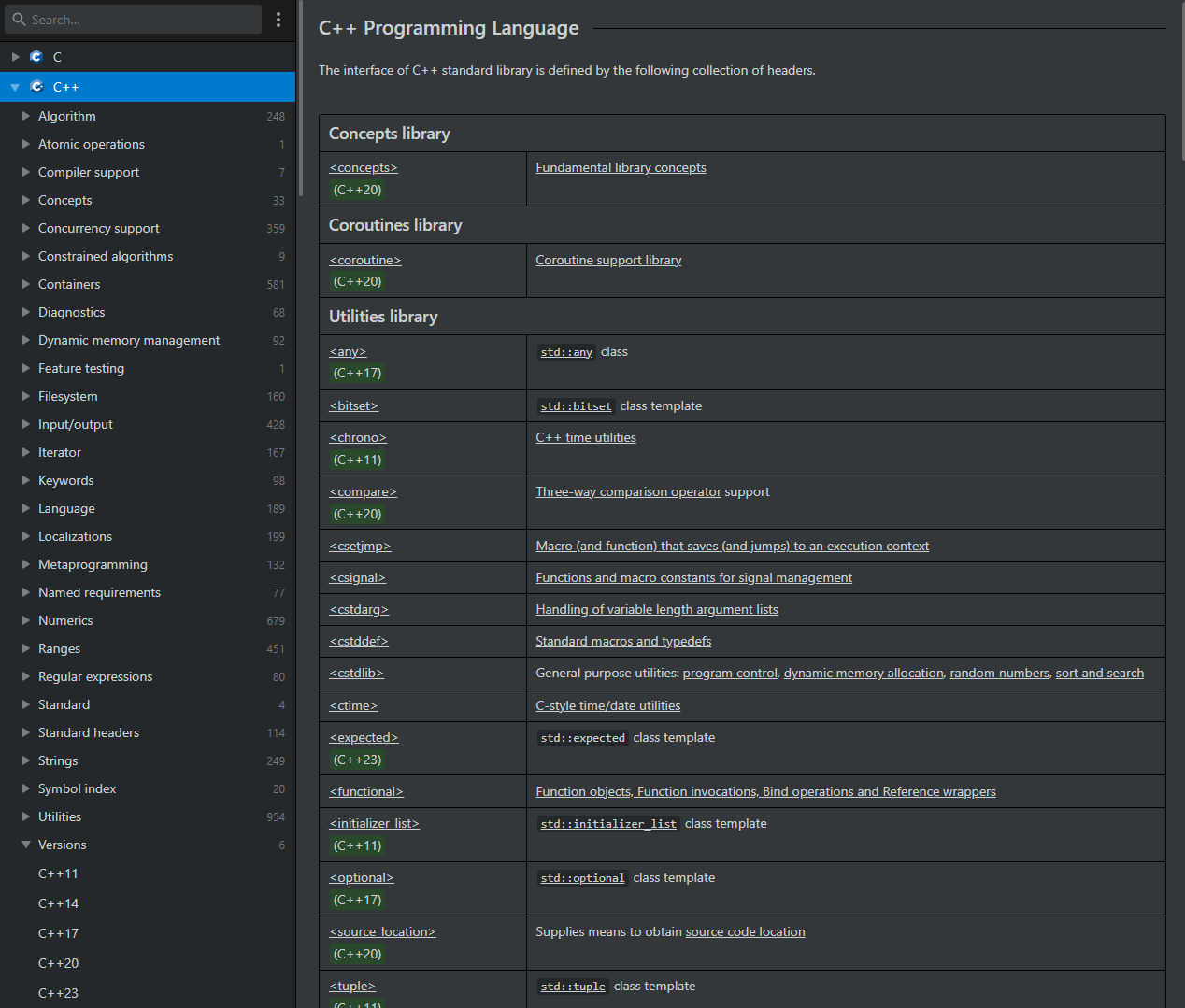
**Project Version 3**

**Evaluating My C++ Learning Resources**

"DevDocs" is a user-friendly platform for C++ documentation. It's easy to navigate and brings a lot of insights about C++ topics. The search works fast and the site lets you search at other programming languages as well. However, for detailed C++ information, checking the official documentation might be a better idea.

<https://devdocs.io/cpp/>

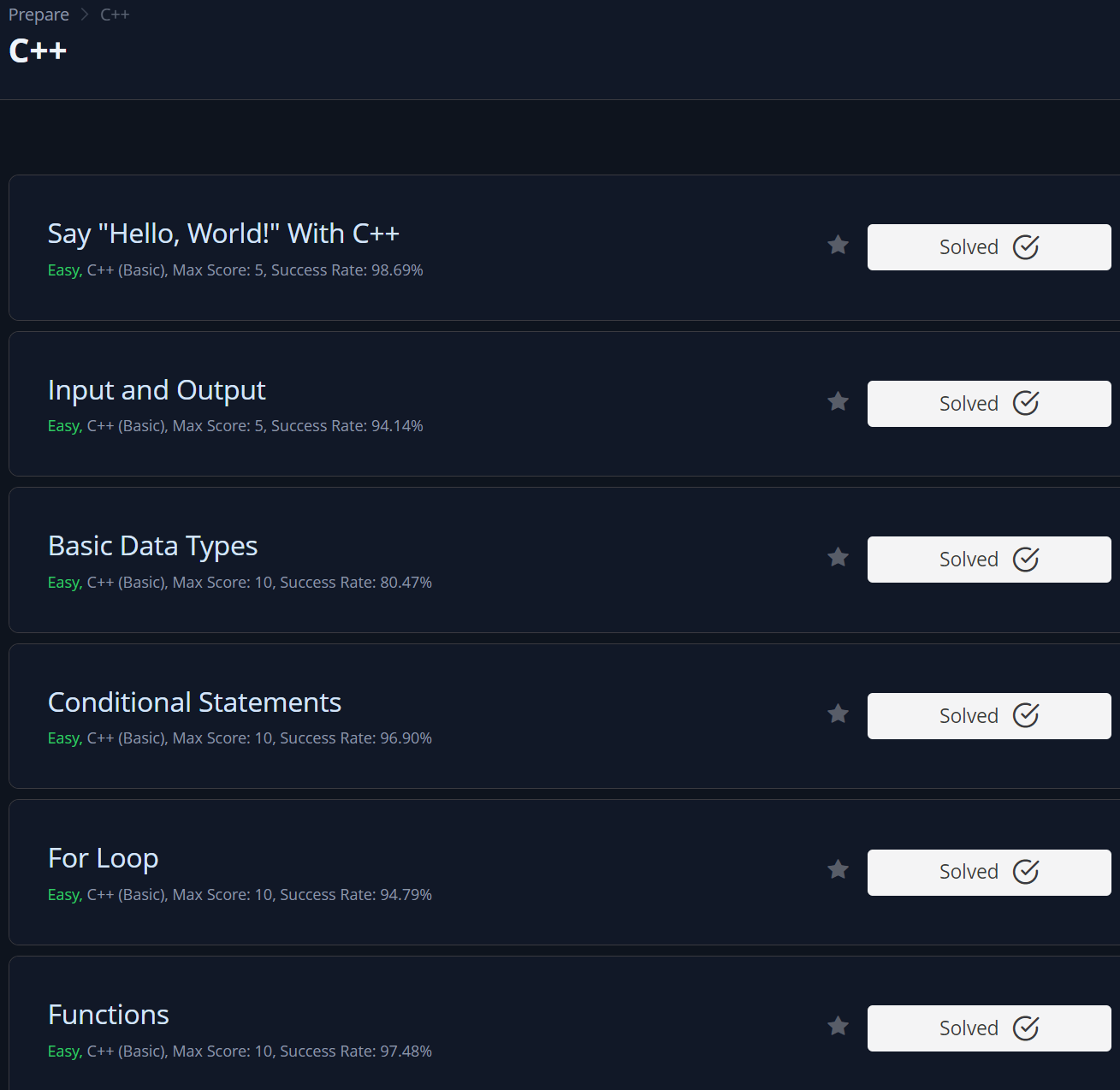
**(Documentation)**



“HackerRank” is a widely recognized platform for coders who want to improve their skills through challenges and contests across various domains. The site's dashboard is designed to provide a personalized experience, guiding users through practice problems, competitive programming challenges, and even job opportunities. With its intuitive interface, the dashboard highlights users' progress and upcoming contests, allowing for seamless navigation. One of the platform's standout features is its emphasis on real-world problems, bridging the gap between coding theory and its practical application.

<https://www.hackerrank.com/dashboard>

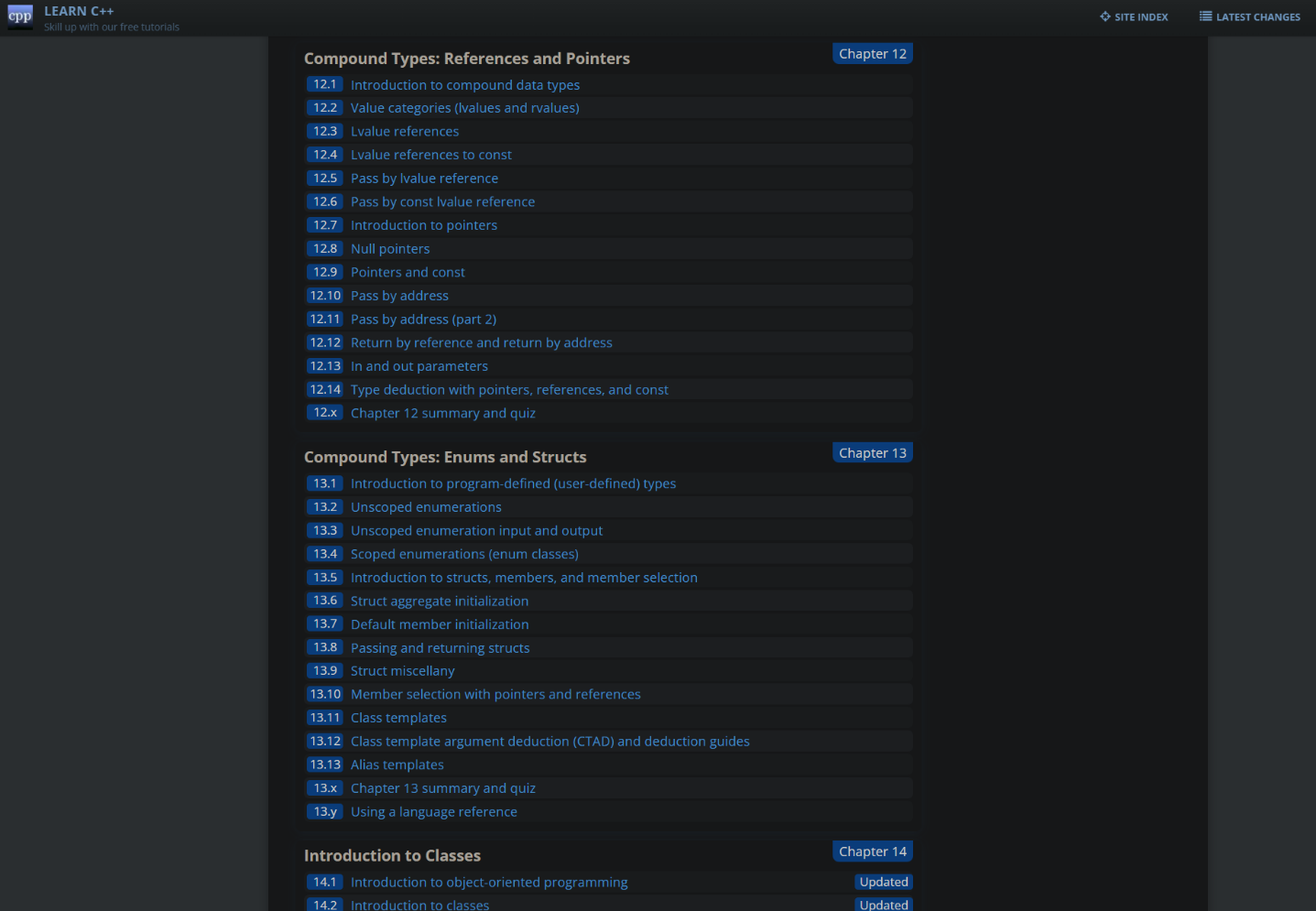
**(Practice Tool)**



“LearnCpp.com” is a dedicated online resource for master the C++ programming language. This website provides comprehensive tutorials ranging from the basics to more advanced topics, making it suitable for beginners as well as experienced programmers looking to delve deep in C++. The structured layout of LearnCpp.com ensures that users can follow a logical progression in their learning journey. Each tutorial is detailed, with examples and exercises to reinforce understanding.

<https://www.learncpp.com/>

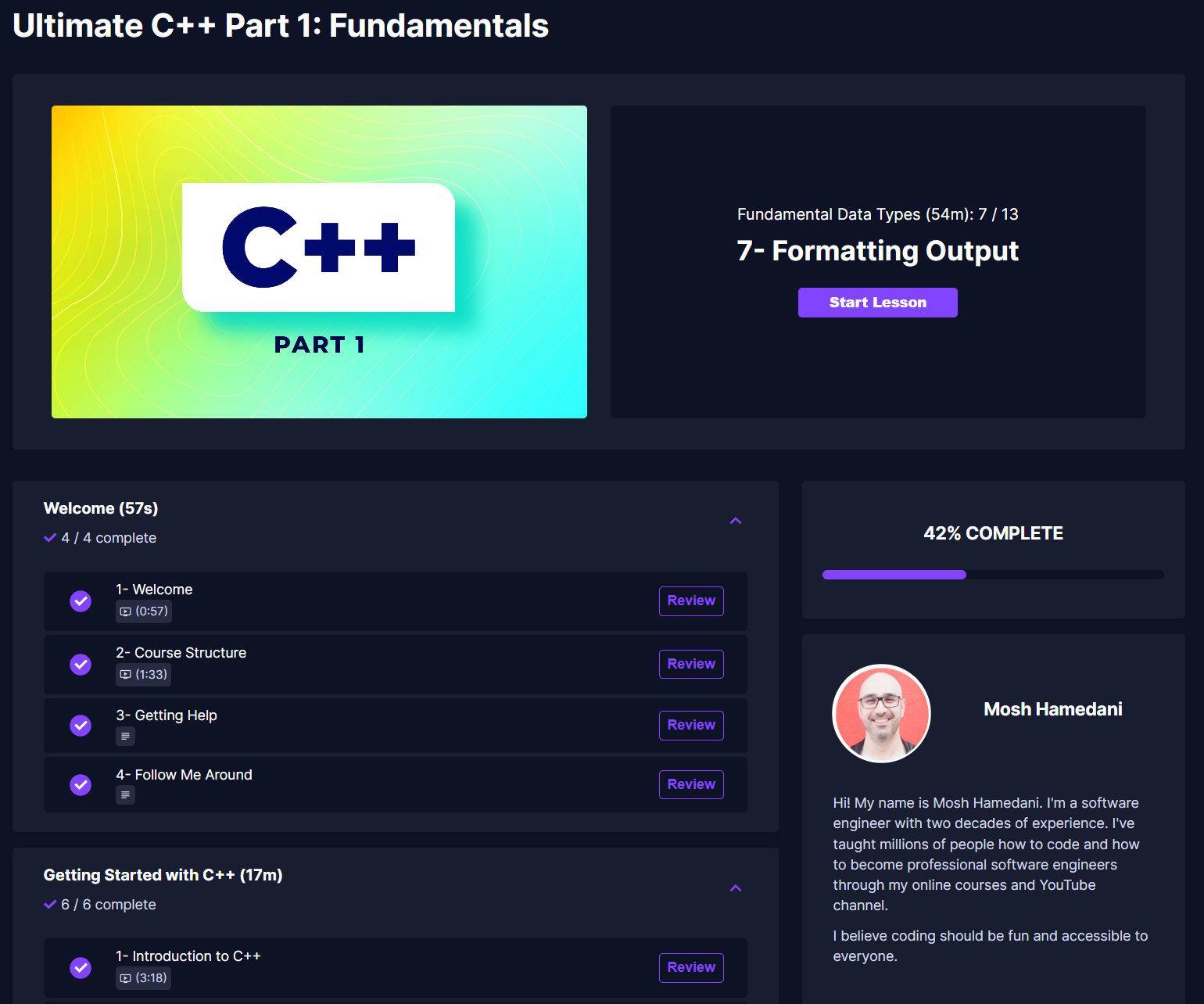
**(Tutorials)**



"Code with Mosh" is a popular online learning platform published by Mosh Hamedani, a software engineer and educator. The platform is known for offering a range of high-quality courses covering various programming languages, frameworks, and development tools. Mosh's teaching approach is pragmatic, focusing on best practices and real-world application. His courses often cater to both beginners and experienced developers, breaking down complex topics into easily digestible segments.

<https://members.codewithmosh.com/courses>

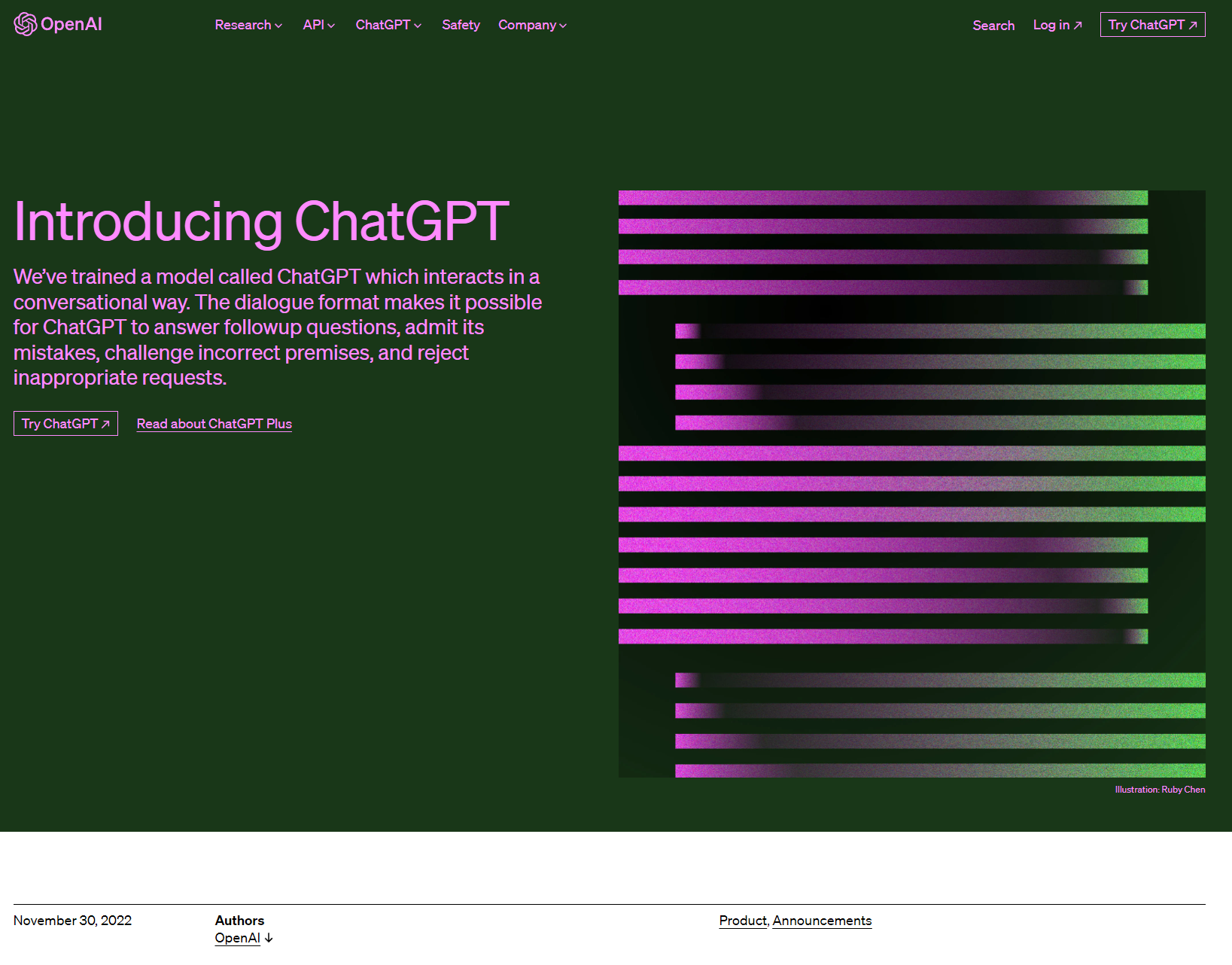
**(Practices with Videos)**



“ChatGPT”, developed by OpenAI, is an advanced conversational AI model designed to understand and respond to user queries. In the context of learning C++, ChatGPT serves as an invaluable tool. It offers instant clarifications on C++ concepts, assists with coding challenges, provides explanations for error messages, and even suggests best practices. Instead of scrolling through forums or reading lengthy documentation, learners can interact directly with ChatGPT for real-time assistance. This personalized and interactive approach not only makes the learning process more efficient but also fosters a deeper understanding of the language.

<https://openai.com/blog/chatgpt>

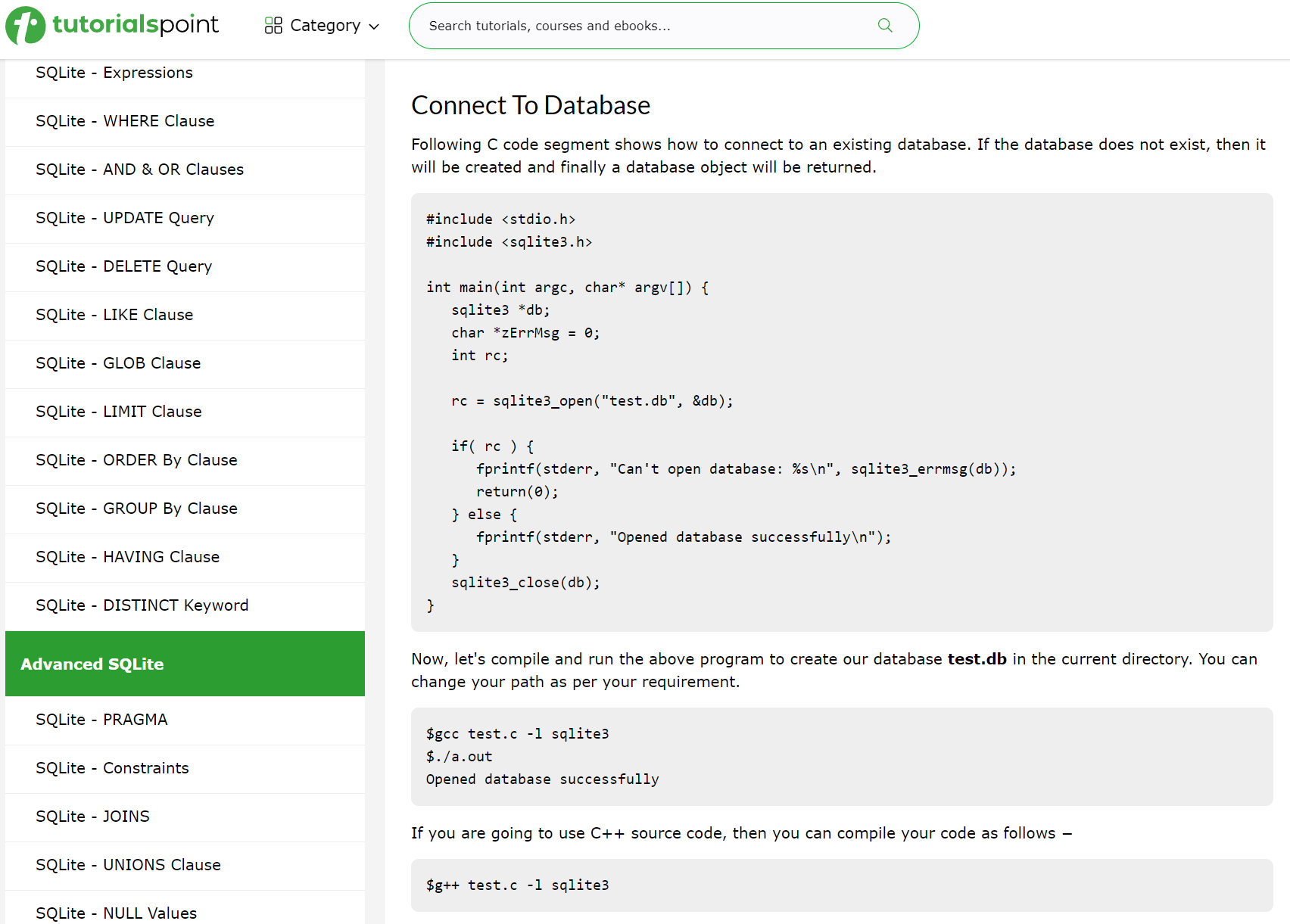
**(Artificial Intelligence)**



The webpage from TutorialsPoint provides a comprehensive guide on using SQLite in C/C++ programs. It covers installation of SQLite library, various SQLite interface routines, and detailed examples of creating, updating, and managing a database using C/C++ code. The examples include operations like opening a database, creating tables, inserting, selecting, updating, and deleting records. Each operation is explained with C code snippets, demonstrating how to execute SQL commands and handle data in SQLite using C/C++

https://www.tutorialspoint.com/sqlite/sqlite\_c\_cpp.htm

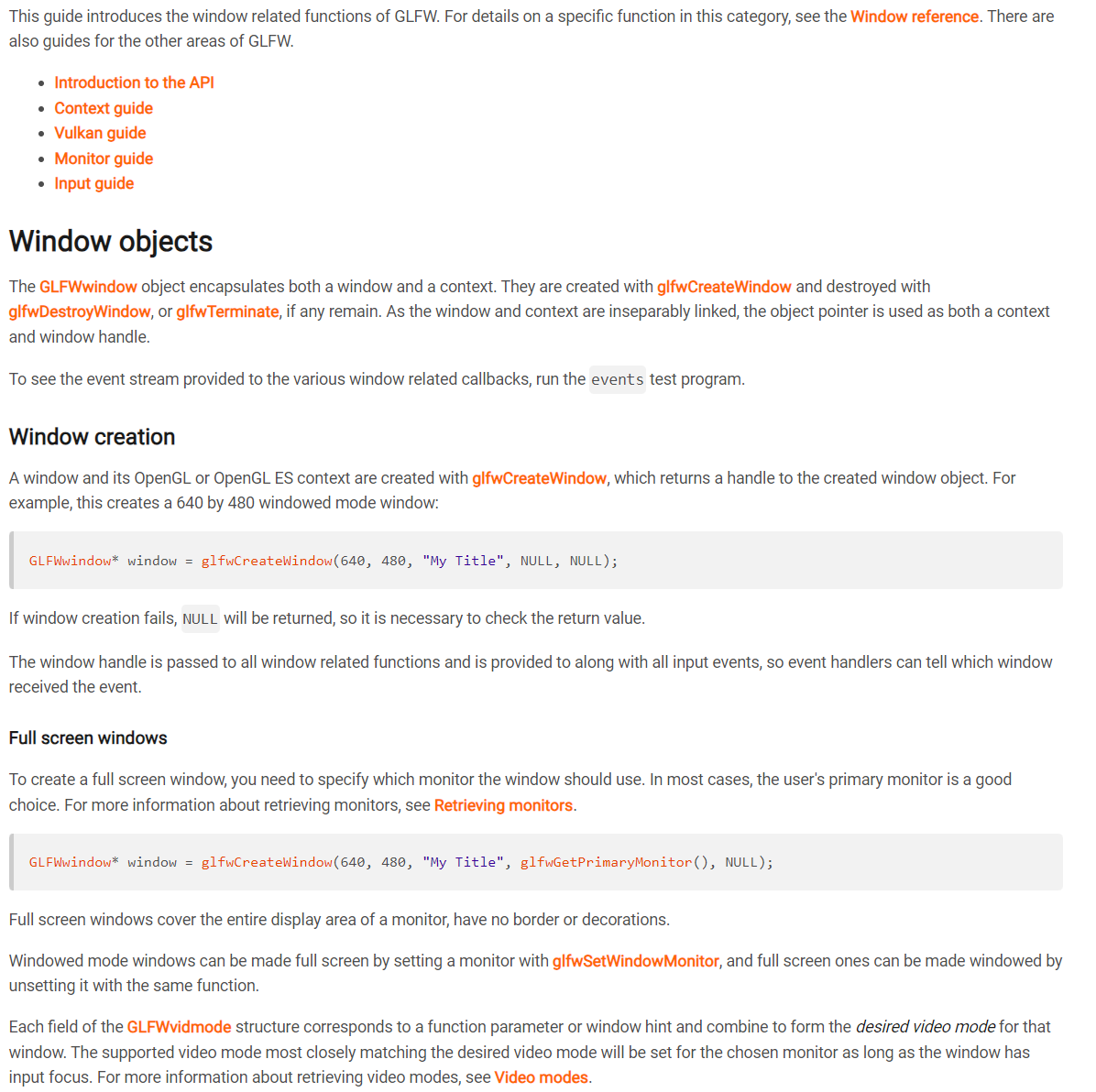
**(Documentation)**

****

The GLFW documentation provides a detailed guide on using GLFW, an open-source library for OpenGL, OpenGL ES, and Vulkan development. It includes instructions for installation, creating windows, handling events, and managing input. The documentation is structured into various sections, such as window, context, and input guides, offering comprehensive information for both beginners and advanced users. It also covers topics like API initialization, error handling, and transitioning from GLFW 2 to 3, along with frequently asked questions and standards conformance. This resource is essential for developers working with GLFW for graphics application development

https://www.glfw.org/docs/latest/

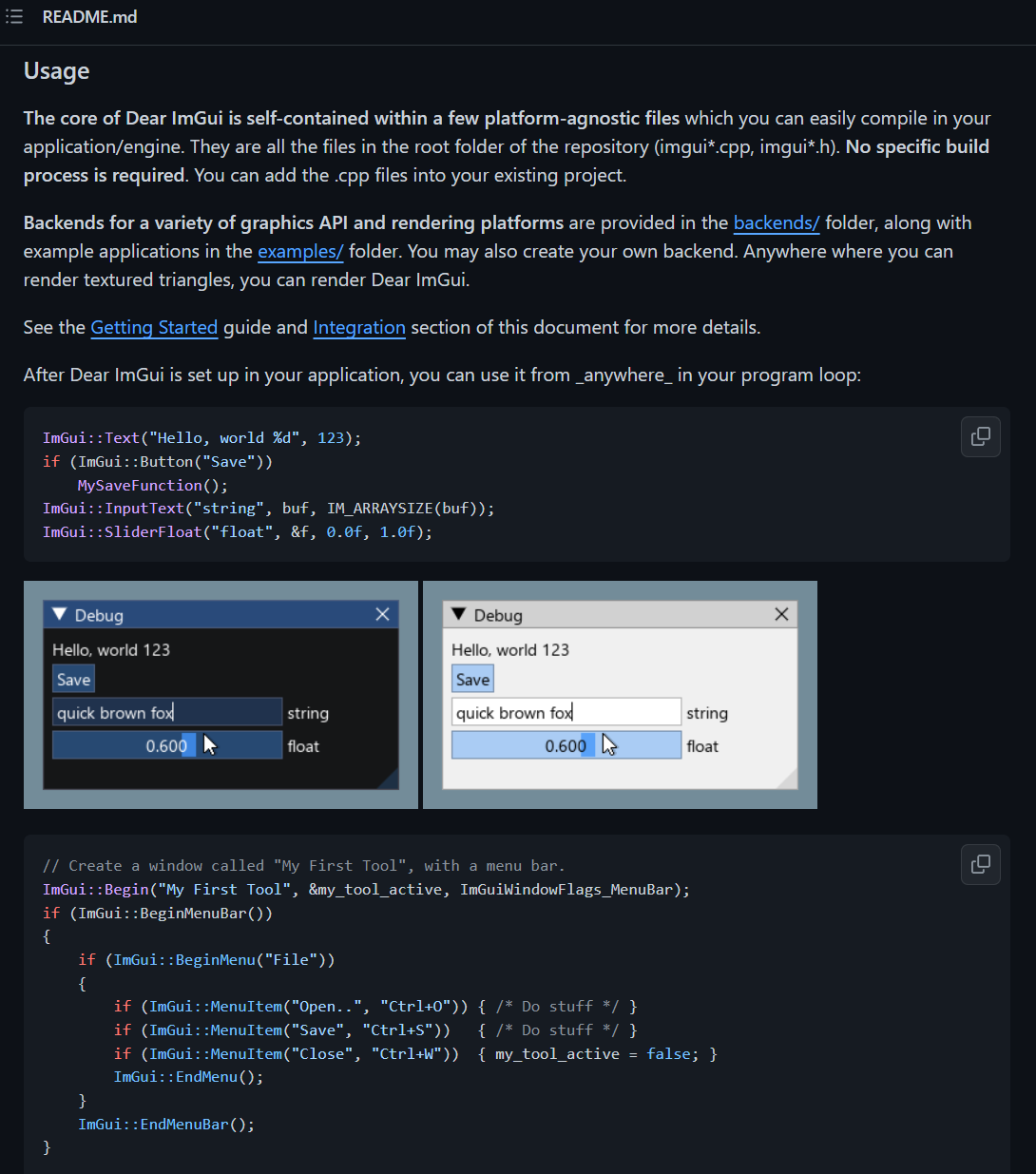
**(Documentation)**

****

The GitHub page for "Dear ImGui" describes it as a bloat-free graphical user interface library for C++ with minimal dependencies. It's designed for rapid development of tools and visualization/debug tools, not end-user applications. Dear ImGui is easy to use, highly portable, and efficient in terms of runtime and memory. It's particularly suited for game engines, real-time 3D applications, and systems with non-standard OS features. The page details usage, integration methods, demos, and provides extensive documentation for both beginners and advanced users​

<https://github.com/ocornut/imgui#readme>

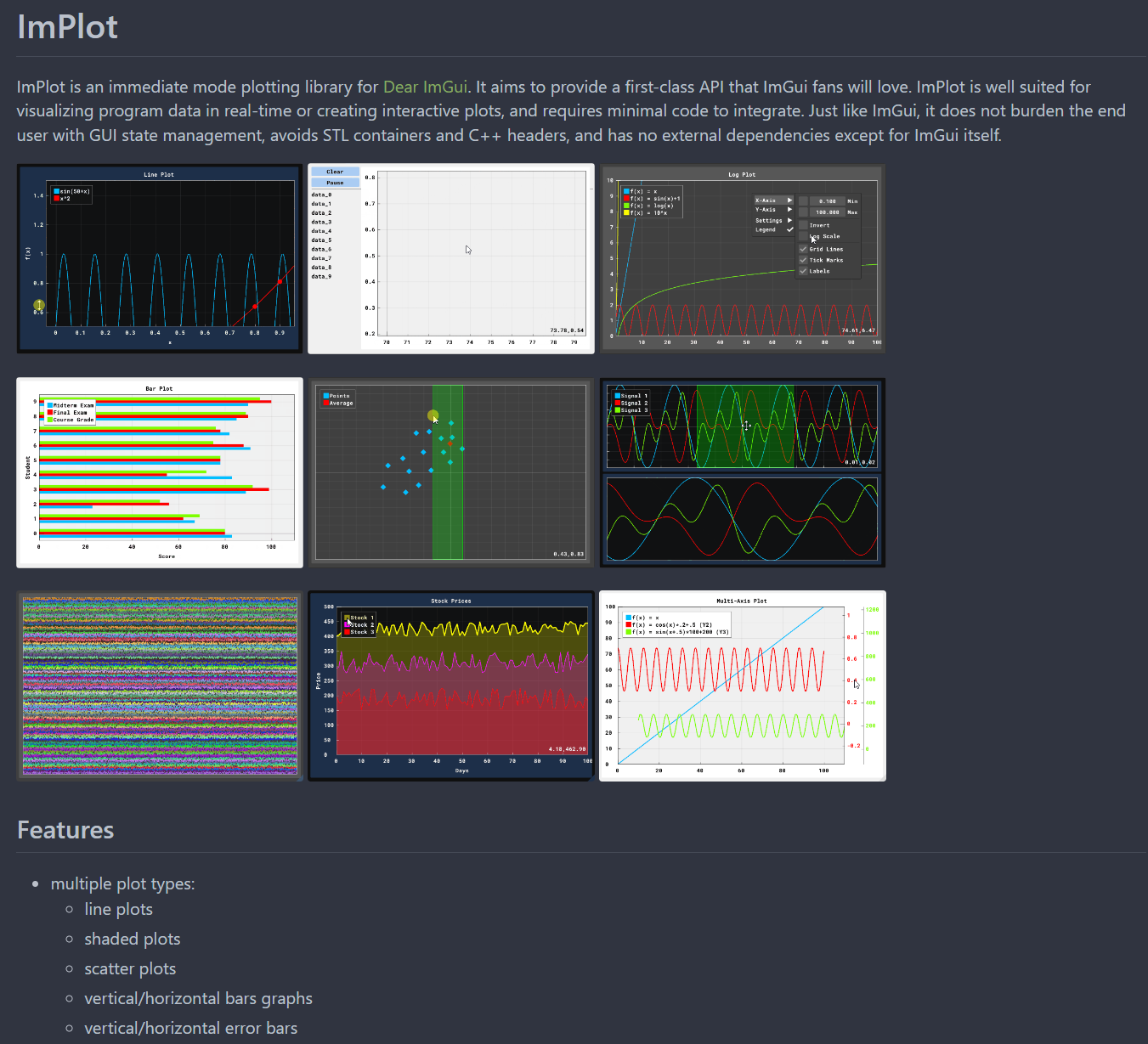
**(Documentation)**



The "ImPlot" repository on FSU's Git server is a fork of an advanced data visualization plugin for the Dear ImGui framework. It provides GPU-accelerated plotting capabilities, ideal for real-time or interactive visualization in C++ applications. The library is designed for ease of use and integration, focusing on function over form. It supports various plot types like line, scatter, bar graphs, and more, with features like configurable axes, subplots, and multiple styling options. It requires ImGui and is suitable for large datasets and real-time applications​

https://fsunuc.physics.fsu.edu/git/gwm17/implot

**(Documentation)**



**Comparison Table of Resources**

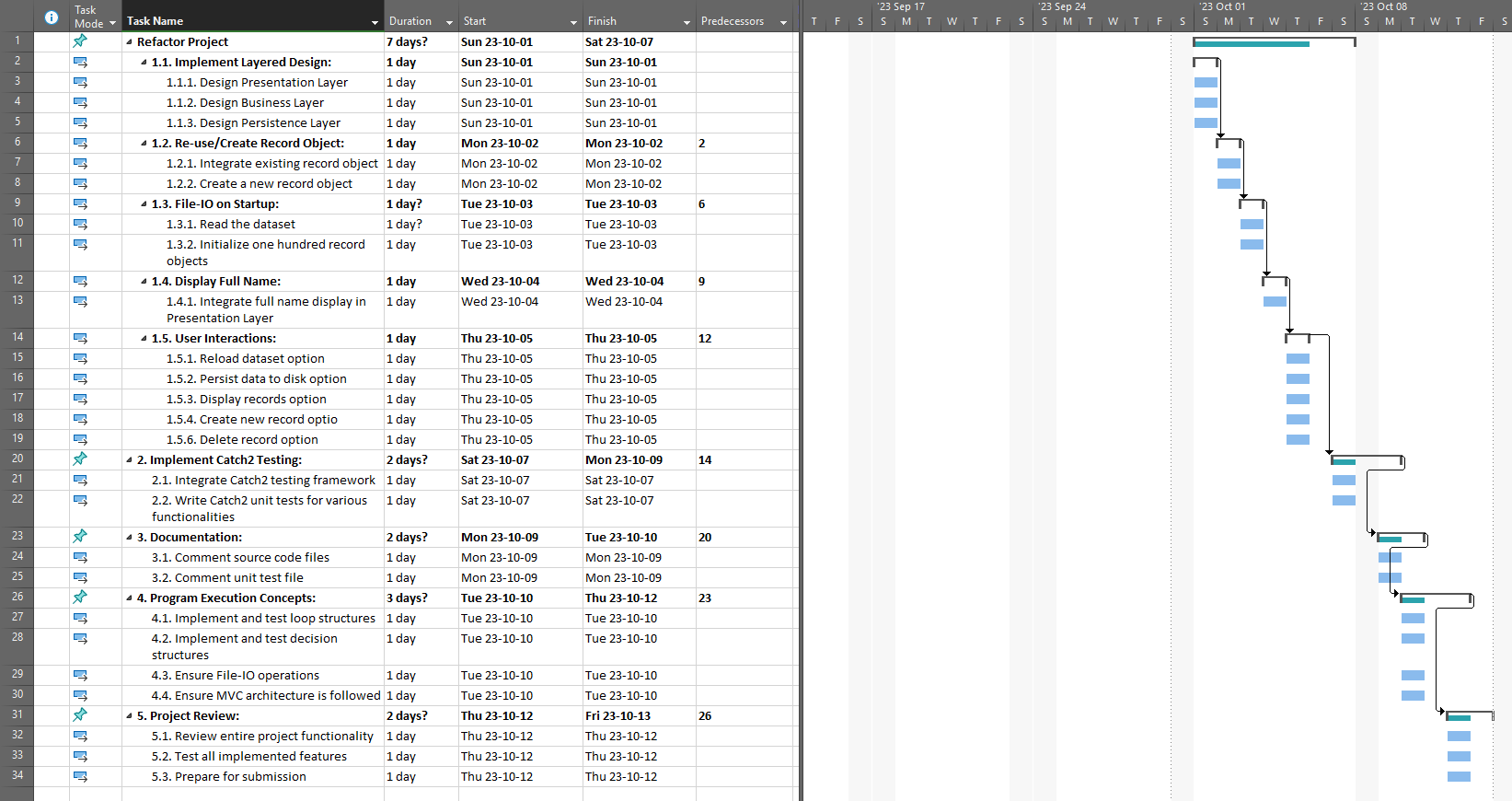
| **Resource** | **Time Consumption** | **Usefulness** |
| --- | --- | --- |
| DevDocs for C++ | High | High |
| HackerRank Dashboard | Low | Medium |
| LearnCpp | Medium | Medium |
| Code with Mosh Courses | Medium | Medium |
| ChatGPT (OpenAI) | Ultra Low | High |
| TutorialsPoint | Low | High |
| GLFW Docs | High | High |
| ImGUI README.md | Medium | High |
| ImPLOT Docs | Medium | High |

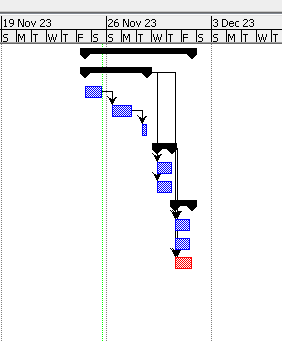
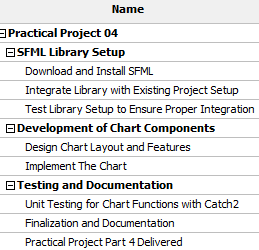
**WBS, Project Management Software, Reflection on Time Estimation**

The WBS allows me to decompose the entire project into manageable steps or tasks, providing a clear hierarchy of what needs to be done. On the other hand, the Gantt chart visualizes the project timeline, showing when each task should start, its duration, and when it's expected to be completed. These tools together ensure that I remain on track and that every aspect of the project is tackled systematically.

With these tools and design decisions at my disposal, I'm more confident about the project's direction and milestones.

****

****



In the course, my initial time estimations were often underestimated, but as I progressed, using Work Breakdown Structures and Gantt charts helped me gauge time more accurately. This experience taught me valuable skills in project time management, leading to a noticeable improvement in efficiently allocating and managing time in the second half of the course.

**References**

1. Sqlite Home Page, https://www.sqlite.org/index.html
2. JetBrains, “Clion: A cross-platform IDE for C and C++ by JetBrains,” JetBrains, https://www.jetbrains.com/clion/
3. Catchorg, “Catchorg/Catch2: A modern, C++-native, Test Framework for unit-tests,” GitHub, https://github.com/catchorg/Catch2
4. “Download,” Download (SFML), https://www.sfml-dev.org/download.php
5. “Tutorials for SFML 2.6,” 2.6 Tutorials (SFML / Learn), https://www.sfml-dev.org/tutorials/2.6/
6. “How to get SFML to work with Clion,” YouTube, <https://www.youtube.com/watch?v=PInJvpQSRHw&t=2s>
7. Ocornut. (n.d.). DEAR IMGUI: Bloat-free graphical user interface for C++ with minimal dependencies. GitHub. https://github.com/ocornut/imgui
8. Epezent. (n.d.). Epezent/Implot: Immediate mode plotting. GitHub. https://github.com/epezent/implot
9. Documentation. GLFW. (n.d.). https://www.glfw.org/documentation.html
10. Doxygen. (n.d.). https://www.doxygen.nl/

**Discussion Board Post Archive**

