Sanjay Ghodawat University Kolhapur

Established as a State Private University under Govt. of Maharashtra Act No. XL dated 3rd May 2017

Empowering Lives Globally !

PROJECTREPORT

A report submitted in partial fulfillment of the requirements for the

Project

School of Computer Science &

Engineering By

Tejas Dhole
PRNNo: 22SC114281058 Roll No: 45
Sankalp Misal
PRNNo: 22SC114281057 Roll No: 43
Prajwal Nikam
PRNNo: 22SC114281118 Roll No: 44
Nishant Shastri
PRNNo: 22SC114281059 Roll No: 46

Program: BTech Class: SY BTech (Div A)

Under Supervision of

Mrs. Veena Mali

AcademicYear:2022-2023

Sanjay Ghodawat University Kolhapur

Established as a State Private University under Govt. of Maharashtra Act No. XL dated 3rd May 2017

Empowering Lives Globally !

School of Computer Science & Engineering

CERTIFICATE

This is to certify that the "**Project Report**"

On

"C++ Learning and Assessment System"

Submitted by

Tejas Dhole
PRNNo:22SC114281058 Roll No: 45
Sankalp Misal
PRNNo:22SC114281057 Roll No: 43
Prajwal Nikam
PRNNo:22SC114281118 Roll No: 44
Nishant Shastri
PRNNo:22SC114281059 Roll No: 46

Program: BTech CSE Class: SY BTech(DivA)

Is work done by him/her and submitted during the 2023–2024 academic year, in partial fulfillment of the **Project.**

Sanjay Ghodawat University, Kolhapur

Mrs. Veena Mali	**Dr. Deepika Patil**	**Dr. Deepika Patil**	
Project Guide	**PBL Co-ordinator**	**Head, CSE& AIML**	**External**

Sanjay Ghodawat University Kolhapur
Established as a State Private University under Govt. of Maharashtra Act No. XL dated 3rd May 2017
Empowering Lives Globally !

DECLARATION

I the under signed solemnly declare that the report of the project work entitled **"C++ Learning and Assessment System "**which is carried out under the supervision of **Mrs. Veena Mali**I assert that the statements made and conclusions drawn are an outcome of the project work .I further declare that to the best of my knowledge and belief that the project report does not contain any part of any work which has been submitted for the award of any other degree/diploma/certificate in this University or any other University.

Student Name:

Tejas Dhole
PRNNo:22SC114281058 Roll No: 45
Sankalp Misal
PRNNo:22SC114281057 Roll No: 43
Prajwal Nikam
PRNNo:22SC114281118 Roll No: 44
Nishant Shastri
PRNNo:22SC114281059 Roll No: 46

Class:SYBTech(DivA)

ACKNOWLEDGMENT

First, I would like to thank my Head of the department **Dr. Deepika Patil** for constructive criticism through out my project. I would like to thank PBL coordinator **Dr. Deepika Patil** and Department Project Guide **Mrs. Veena Mali** for support and advices to get and complete internship in above said organization. It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals. I am extremely grateful to my department staff members and friends who helped me in successful completion of this project.

ABSTRACT

In the realm of computer science and software development, mastering the fundamentals of a programming language is essential. The "Interactive C++ Language Learning Application" is designed to assist beginners in learning C++, one of the most widely used programming languages. This project leverages an intuitive and interactive console-based application to provide a guided learning experience for students and enthusiasts.

1. Randomized Quizzes: Users can test their knowledge through randomized quizzes that challenge them to apply what they've learned.

2. Timed Quiz Mode: For a more challenging experience, the application offers a timed quiz mode. Users can set the number of questions and a time limit to assess their proficiency under pressure.

3. User-Friendly Interface: The application boasts an easy-to-navigate menu system that enables users to choose between reading lessons, taking quizzes, or exploring available content.

This project aims to address the educational challenges faced by individuals pursuing C++ programming skills. By providing structured lessons and interactive quizzes, it helps users grasp the essential concepts and principles of C++. The Interactive C++ Language Learning Application is designed to be extensible. Following are it's future enhancements:

1. Additional Lessons: Expanding the lesson library to cover more advanced C++ topics.

2. Scoring System: Implementing a scoring system for quizzes to motivate users.

3. Interactive Code Samples: Incorporating interactive coding exercises.

In conclusion, this project provides an accessible and engaging platform for learning C++ programming. By combining educational content with interactive quizzes, it aims to empower individuals to master the fundamentals of C++, setting them on a path to becoming proficient programmers.

TABLEOFCONTENT'S

Introduction

The Console-Based To-Do List Application is a versatile and practical project designed to assist users in managing their tasks and responsibilities efficiently. Built using C++, this application provides a user-friendly command-line interface that empowers users to effortlessly organize their daily, weekly, and long-term tasks in one central location.

It’s key features are :-

- Task Management
- Displaying Task
- Basic errors handling
- Future Enhancement

This To-Do List project is an excellent starting point for those looking to develop their programming skills in C++. It provides a command-line interface for basic task management and can be further customized and improved to meet specific needs and preferences. Whether you're a programming enthusiast or someone seeking a more organized approach to daily tasks, this project can be a valuable tool for managing your to-do list efficiently.

- **ProblemDefinition:**

"Design and develop a digital to-do list application that allows users to efficiently manage their tasks, appointments, and responsibilities. The application should provide a user-friendly interface, offer features for creating, editing, and prioritizing tasks, and support task categorization and due date management. Additionally, the application should be accessible on various platforms (web, mobile, etc.) and ensure data synchronization across devices, all while maintaining the security and privacy of user information."

This problem definition outlines the key requirements and goals for creating a to-do list application. It highlights the need for user-friendliness, task management features, cross-platform support, and data security.

- **Scope:**

1. The scope for a to-do list application can vary depending on its intended audience and specific goals, but here's a general scope that encompasses common features and functionalities.
2. Task Creation: Users can add tasks with titles, descriptions, and due dates.
3. Task Organization: Users can categorize tasks into different lists or categories (e.g., work, personal, shopping).
4. Task Prioritization: Ability to set task priorities (e.g., high, medium, low).
5. Task Completion: Users can mark tasks as completed.
6. Task Editing: Ability to edit task details and due dates.
7. Attachments: Attach files, images, or links to tasks.
8. Subtasks: Break tasks into subtasks for better task management.
9. Notes: Ability to add notes or comments to tasks.

- **Problem Identification:**

1. Task Organization: Many people struggle with organizing and prioritizing their tasks and responsibilities, leading to missed deadlines and increased stress.

2. Task Organization: Many people struggle with organizing and prioritizing their tasks and responsibilities, leading to missed deadlines and increased stress.

3. Time Management: Effective time management can be a challenge, and a to-do list can assist in allocating time to specific tasks.

4. Task Overload: It's easy to become overwhelmed by a long list of tasks. A to-do list can help users break down their workload into manageable steps.

5. Lack of Accountability: Without a system to track progress, individuals may not be accountable for their tasks, impacting their personal and professional lives.

6. Difficulty in Collaboration: In a professional setting, team members may struggle to coordinate tasks and projects. A to-do list app can facilitate collaboration.

Objectives

I. **Task Management:** Create a system that allows users to add, remove, edit, and mark tasks as completed.

II. **User Interaction:** Implement a user-friendly interface for input and navigation, using menus or command-line prompts.

III. **Task Storage**: Utilize appropriate data structures (arrays, vectors, linked lists, etc.) to store and manage tasks effectively.

IV. **Task Sorting:** Allow users to sort tasks based on due date, priority, or completion status.

System Requirements Specification

- **SOFTWAREREQUIREMENTS**
    - Dev C++ / VS code

- **HARDWAREREQUIREMENTS**
    - 4GBRAM
    - 512GBHDD
    - i3processor

Methodology

- **Algorithm:**

1. Adding a task Algorithm:-

Algorithm AddTask(taskList, newTask):

Input: taskList (array of tasks), newTask (string)

Output: Updated taskList

1. Create a new task object with newTask description.
2. Append the new task object to the taskList.
3. Return the updated taskList.

2. Deleting a task Algorithm:-

Algorithm DeleteTask(taskList, taskIndex):

Input: taskList (array of tasks), taskIndex (integer)

Output: Updated taskList

1. Check if taskIndex is valid (within the range of taskList).
2. If valid, remove the task at taskIndex from taskList.
3. Return the updated taskList.

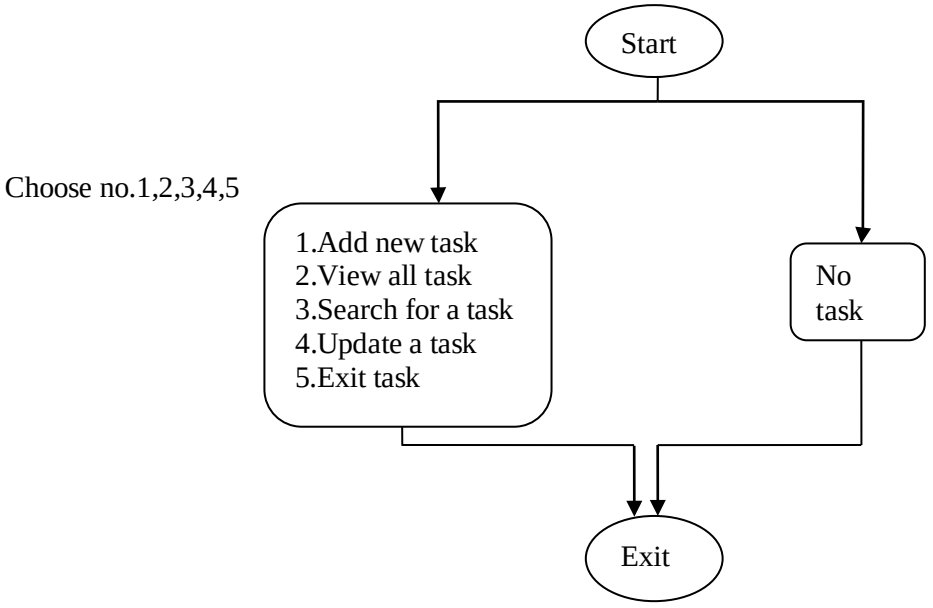
3. Displaying a task Algorithm:-

Algorithm DisplayTasks(taskList):

Input: taskList (array of tasks)

1. Loop through each task in taskList.
2. Print task details (description, due date, etc.) to the screen.
3. Repeat for all tasks in the list.

- **FlowDiagram(Flow Chart):**



Implementation

1. Choose Technology Stack:

Select the programming languages, frameworks, and libraries for web and mobile development. For example, you might use JavaScript for web development and a framework like React or Angular, and Java or Swift for mobile app development.

2. Database Design:

Design the database schema to store user accounts, tasks, categories, and other relevant data. Use a relational or NoSQL database based on your project's requirements.

3. Task Management:

Create the core functionality for task management, including adding, editing, deleting, and completing tasks. Implement data validation to ensure task details are correct.

4. Task Organization:

Allow users to organize tasks into lists or categories. Implement features like creating, editing, and deleting lists.

5. User Interface (UI):

Design a user-friendly and intuitive UI for both web and mobile platforms. Ensure a responsive design for various screen sizes and orientations.

“Introduction of Object-Oriented Programming using C++”

Certainly, here's an essay-style presentation of the history of C++ and an introduction to Object-Oriented Programming (OOP) using C++.

C++ is a programming language with a rich history that extends back to the early 1980s. Bjarne Stroustrup, its creator, envisioned a language that would blend the efficiency and power of the C programming language with the structure and organization offered by Object-Oriented Programming (OOP). This ambitious goal led to the birth of C++. Stroustrup's initial attempt, known as "C with Classes," introduced the concept of classes and laid the foundation for C++'s object-oriented features. Over time, C++ evolved and expanded its capabilities, eventually becoming one of the most influential programming languages in the world. The first official C++ language standard was published in 1998, and since then, C++ has undergone several standard revisions, each bringing new features and enhancements to the language.

Object-Oriented Programming (OOP) is a programming paradigm that promotes the organization of code into objects, which are instances of classes. A class defines a blueprint for creating objects, encapsulating both data (attributes) and functions (methods) that operate on the data. In C++, these classes serve as the foundation for object-oriented design. For example, consider a class "Car." This class could have attributes such as "color" and "speed," along with methods like "accelerate" and "brake."

One of the pivotal concepts in OOP is encapsulation. In C++, encapsulation involves bundling data and the functions that manipulate it into a single unit, the class. This practice ensures data hiding,

where the internal details of an object are concealed from the outside. Only the necessary functionalities are exposed, enhancing code security and modularity.

Inheritance is another cornerstone of OOP supported by C++. Inheritance allows a new class (the derived class) to inherit properties and behaviors from an existing class (the base class). This facilitates code reuse and extension. For instance, you can create a base class "Vehicle" with common attributes and methods, and then derive specific vehicle types like "Car" and "Motorcycle" from it. This way, you can share code and characteristics among related classes, promoting a more efficient and maintainable codebase.

In conclusion, C++, with its rich history and extensive capabilities, is a prominent language for exploring Object-Oriented Programming. By understanding the historical context and the core OOP principles like classes, encapsulation, and inheritance, you can embark on a journey to create robust and modular software solutions in C++. This mastery of OOP will equip you with the essential tools to become a proficient C++ programmer and unlock the language's full potential in your coding endeavors.

Advantages of learning C++:

The advantages of learning C++:

1. Versatility: C++ is a versatile programming language that can be applied in various domains, including systems programming, game development, and even web development. Its flexibility allows you to work on a wide range of projects.

2. High Performance: C++ is known for its high performance. It provides low-level memory control and efficient resource management, making it suitable for resource-intensive applications, such as game engines and real-time systems.

3. Object-Oriented Programming (OOP): C++ supports OOP principles, enabling you to design and manage complex systems by creating reusable and organized code through classes and objects.

4. Standard Library: C++ comes with a comprehensive Standard Library that provides a wealth of pre-built functions and data structures. This library can significantly speed up development and reduce the need for writing low-level code.

5. Cross-Platform Compatibility: C++ code can be compiled and executed on various platforms, making it a practical choice for developing software that needs to run on multiple operating systems.

6. Community and Resources: There is a large and active C++ community with abundant online resources, forums, and tutorials. This makes it easier to find support and solutions to programming challenges.

7. Legacy Code Compatibility: Many existing applications and libraries are written in C++. Learning C++ allows you to work with or modify these codebases, a valuable skill in the software development industry.

8. System Programming: C++ is commonly used in system programming, including operating systems, device drivers, and other low-level tasks. Understanding C++ is essential for tasks involving interactions with hardware.

9. Game Development: C++ is a popular language in the gaming industry due to its ability to provide the level of control and performance required for game development.

10. Embedded Systems: C++ is widely used in embedded systems, which power devices ranging from smartphones to Internet of Things (IoT) devices.

11. Career Opportunities: Proficiency in C++ opens up a wide array of career opportunities in various domains, including software development, research, and financial engineering.

12. Educational Value: Learning C++ helps you understand core programming concepts deeply, such as memory management, data structures, and algorithms.

13. Problem-Solving Skills: C++ encourages strong problem-solving and algorithmic thinking. The challenges of managing memory and optimizing code contribute to enhancing your problem-solving skills.

In summary, learning C++ offers a broad skill set and a vast range of applications. It equips you with the ability to work on diverse projects, from low-level system programming to high-performance game development, making it a valuable language in the world of programming.

Source Code:

```
#include <iostream>
#include <vector>
#include <fstream>

using namespace std;

vector<string> todoList;

void displayTasks() {
cout << "To-Do List:\n";
for (int i = 0; i < todoList.size(); ++i) {
cout << i + 1 << ". " << todoList[i] << endl;
}
}

int main() {
int choice;
string task;

// Load tasks from a file if available

while (true) {
cout << "\nMenu:\n";
cout << "1. Add Task\n";
cout << "2. Display Tasks\n";
cout << "3. Mark Task as Completed\n";
cout << "4. Save and Exit\n";
cout << "Enter your choice: ";
cin >> choice;
cin.ignore(); // Clear the newline character from the input buffer

switch (choice) {
case 1:
cout << "Enter a new task: ";
getline(cin, task);
todoList.push_back(task);
break;
case 2:
displayTasks();
break;
case 3:
int index;
cout << "Enter the task number to mark as completed: ";
cin >> index;
if (index >= 1 && index <= todoList.size()) {
// Mark task as completed or remove it from the list
// todoList.erase(todoList.begin() + index - 1);
} else {
cout << "Invalid task number." << endl;
```

```
}
break;
case 4:
// Save the to-do list to a file and exit
// ofstream outFile("todolist.txt");
// for (const string& task : todoList) {
//     outFile << task << endl;
// }
// outFile.close();
return 0;
default:
cout << "Invalid choice. Try again." << endl;
}
}

return 0;
}
```

Result

```
Menu:
1. Add Task
2. Display Tasks
3. Mark Task as Completed
4. Save and Exit
Enter your choice: 1
Enter a new task: warmup :5:30am
Menu:
1. Add Task
2. Display Tasks
3. Mark Task as Completed
4. Save and Exit
Enter your choice: 2
To-Do List:
1. warmup :5:30am

Menu:
1. Add Task
2. Display Tasks
3. Mark Task as Completed
4. Save and Exit
Enter your choice:
```

Conclusion & Future Scope

- **Future Scope:**

1.You can convert your to-dos into not-to-dos by applying one of the following principles: (1) elimination, (2) automation, and (3) delegation, in that order.

2.If your task needs to get done but can be automated in some way, take your time and brainstorm possible solutions.

3.If elimination and automation aren't viable options, consider delegating your task to someone, even when you know that you're the most capable person to do that task.

4.If you're in favor of bullet journals, write down a simple list of activities and tasks that you refuse to do anymore. Your list isn't a mere collection of tasks. If you want to kick the habit of checking emails first thing in the morning or saying yes to every project, add them to your list.

5.Q1 and Q2 stuff will land on your plate. Note, however, that Q2 tasks are the ones that create momentum: these are the important things that will never get full attention unless you make them a priority. Q3 tasks can be automated or delegated; Q4 tasks are perfect candidates for elimination.

- **Conclusion:**

1. Whether you need a to-do list or not is of personal preference. If you, however, keep a to-do list, it's imperative that you get the inversion: a not-to-do list.

2. It helps to weed out the unwanted stuff and focus on the things that matter most. Creating a not-to-do list should take no more than half an hour. Once you get it, however, you'll recognize that your to-do list shrinks to its half over time.

3. Schedule a one-on-one meeting with yourself and compile a simple list of tasks and activities that you want to get rid of. Cut the nonsense.

- **Reference**

Books:

1. Object Oriented Programming in Turbo C++ - Robert Lafore (Galgotia).

2. Object Oriented Programming with C++ - Sourav Sahay (Oxford) Second Edition.

The following are the reference websites

1.https://www.javatpoint.com/cpp-oops-concepts