## Assignment\_19-08-2024

Author: ThanhTH10

Date: 19/08/2024

1. Explain about interfaces with respect to C++

Abstract class, pure virtual function and virtual destructor

Interfaces in C++: In C++, an interface is a way to define a contract for derived classes, specifying which functions the derived classes must implement. This concept is implemented through abstract classes and pure virtual functions.

**Abstract Class:**

* An **abstract class** is a class that cannot be instantiated directly. It typically contains one or more **pure virtual functions**, which means that derived classes must provide implementations for these functions.
* An abstract class serves as a blueprint for other classes, forcing them to implement specific methods.

class AbstractClass {

public:

    virtual void someMethod() = 0; // Pure virtual function

};

**Pure Virtual Function:**

* A **pure virtual function** is a function declared in a base class that has no definition (implementation) in the base class. It is denoted by = 0 after the function declaration.
* Derived classes **must** provide an implementation of the pure virtual function; otherwise, they too will be considered abstract.

class Interface {

public:

    virtual void doSomething() = 0; // Pure virtual function

};

class DerivedClass : public Interface {

public:

    void doSomething() override {

        // Provide implementation

    }

};

**Virtual Destructor:**

* In C++, when you are using polymorphism (i.e., base class pointers or references to derived class objects), it is crucial to declare the destructor in the base class as **virtual**.
* This ensures that the derived class's destructor is called when a derived object is deleted through a base class pointer, preventing memory leaks or undefined behavior.

class Base {

public:

    virtual ~Base() { // Virtual destructor

        // Clean up resources

    }

};

class Derived : public Base {

public:

    ~Derived() {

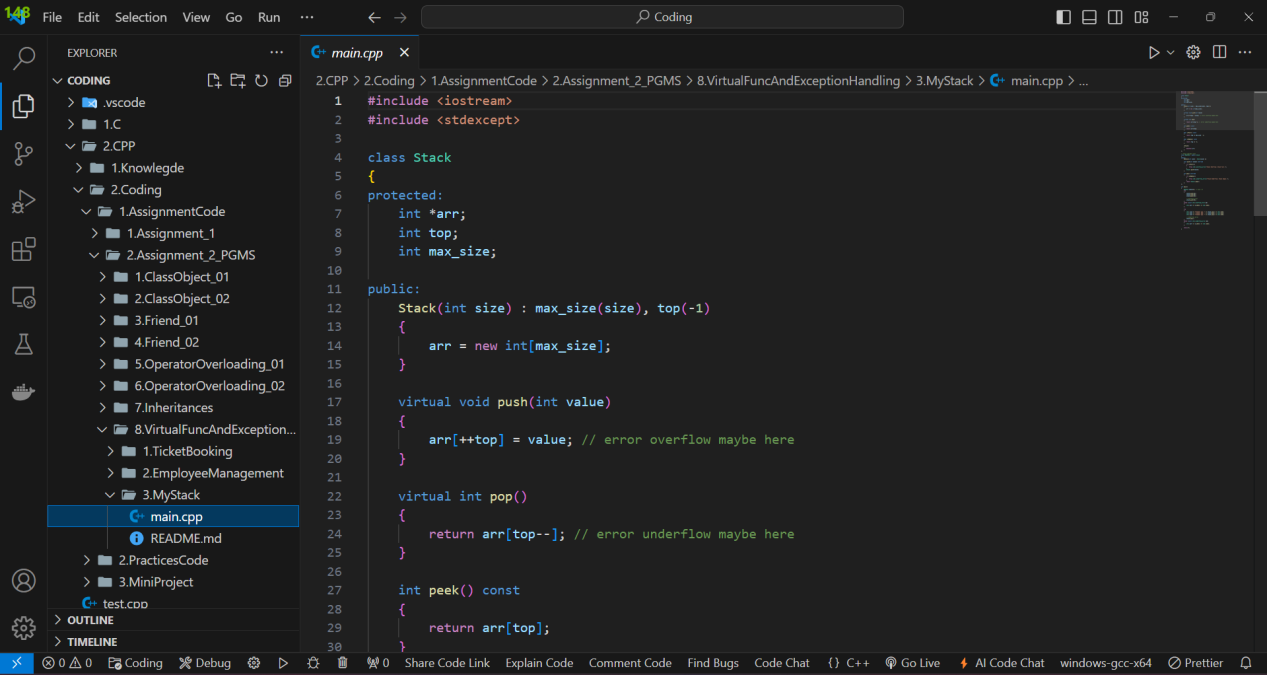
        // Derived class cleanup

    }

};

1. [https://vscode.dev/](https://vscode.dev/" \t "https://lms.vectorinstitute.in/mod/assign/_blank) write C++ program, compile and execute it

I have experience using VSCode



1. What all phases of software development life cycle

### 1. ****Requirement Gathering and Analysis:****

* **Purpose:** To understand and document what the software needs to accomplish. This phase involves gathering requirements from stakeholders and analyzing them to create a clear and detailed set of specifications.
* **Activities:**
  + Meetings with stakeholders and end-users.
  + Documentation of requirements.
  + Analysis of feasibility and risk.
* **Deliverables:**
  + Requirement Specification Document.
  + Feasibility Study Report.

### 2. ****Planning:****

* **Purpose:** To create a detailed plan for how the project will be executed, including timelines, resources, and budget. This phase sets the groundwork for all subsequent activities.
* **Activities:**
  + Defining project scope and objectives.
  + Estimating time and cost.
  + Developing a project schedule and resource allocation.
* **Deliverables:**
  + Project Plan.
  + Schedule.
  + Resource and Budget Allocation.

### 3. ****Design:****

* **Purpose:** To create a blueprint for the software based on the requirements. This phase involves specifying how the software will be built and how it will function.
* **Activities:**
  + Designing system architecture and components.
  + Creating data models, user interfaces, and detailed design specifications.
* **Deliverables:**
  + Design Document.
  + Data Models and Architecture Diagrams.
  + User Interface Design.

### 4. ****Development (Implementation):****

* **Purpose:** To convert the design into a working software application. This phase involves coding and building the software according to the design specifications.
* **Activities:**
  + Writing code.
  + Integrating components.
  + Performing unit testing to ensure that individual parts work correctly.
* **Deliverables:**
  + Source Code.
  + Executable Software.
  + Unit Test Reports.

### 5. ****Testing:****

* **Purpose:** To ensure that the software is functioning correctly and meets the requirements. This phase involves identifying and fixing defects and verifying that the software performs as expected.
* **Activities:**
  + Conducting various types of testing (e.g., functional, integration, system, acceptance).
  + Reporting and fixing bugs.
  + Validating that requirements are met.
* **Deliverables:**
  + Test Plans and Test Cases.
  + Test Reports.
  + Defect Logs.

### 6. ****Deployment:****

* **Purpose:** To release the software to the end-users. This phase involves installing and configuring the software in the production environment.
* **Activities:**
  + Installing the software on user systems or servers.
  + Configuring the environment.
  + Training users and providing support.
* **Deliverables:**
  + Deployed Software.
  + User Documentation.
  + Training Materials.

### 7. ****Maintenance and Support:****

* **Purpose:** To provide ongoing support and updates after the software is in use. This phase involves fixing any issues that arise and making enhancements as needed.
* **Activities:**
  + Monitoring software performance.
  + Providing technical support.
  + Implementing updates and enhancements.
* **Deliverables:**
  + Maintenance Logs.
  + Update and Patch Releases.
  + Support Documentation.

1. what is virtual memory . Learn about physical and virtual memory mapping

**Virtual memory** is a memory management technique that creates the illusion of a larger memory space by using disk storage to extend the available RAM.

#### Key Concepts:

**Virtual Memory:**

* 1. **Purpose:** Allows systems to use more memory than physically available by swapping data between RAM and disk storage.
  2. **Benefits:**
* Increases system capacity.
* Provides process isolation and security.
* Optimizes RAM usage.

**Physical Memory:**

* **Definition:** Actual RAM installed in the computer.
* **Limitation:** Limited by the size of the RAM modules.

#### Memory Mapping:

1. **Virtual Address Space:** Range of addresses that applications can use, provided by the operating system.
2. **Physical Address Space:** Actual locations in RAM accessed by hardware.

#### How It Works:

1. **Paging:**

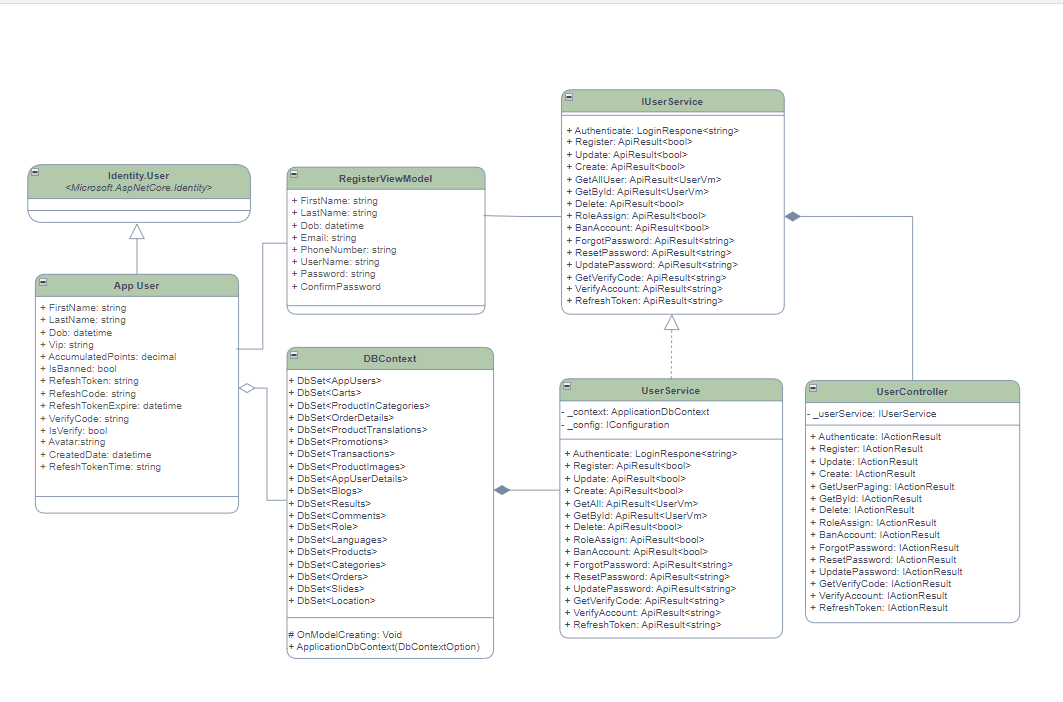
* Divides memory into fixed-size blocks (pages) and maps them to physical memory blocks (page frames).
* Uses a page table to translate virtual addresses to physical addresses.

1. **Segmentation:** Divides memory into variable-sized segments based on logical divisions.
2. **Page Table:** A data structure that keeps track of the mapping between virtual and physical addresses.

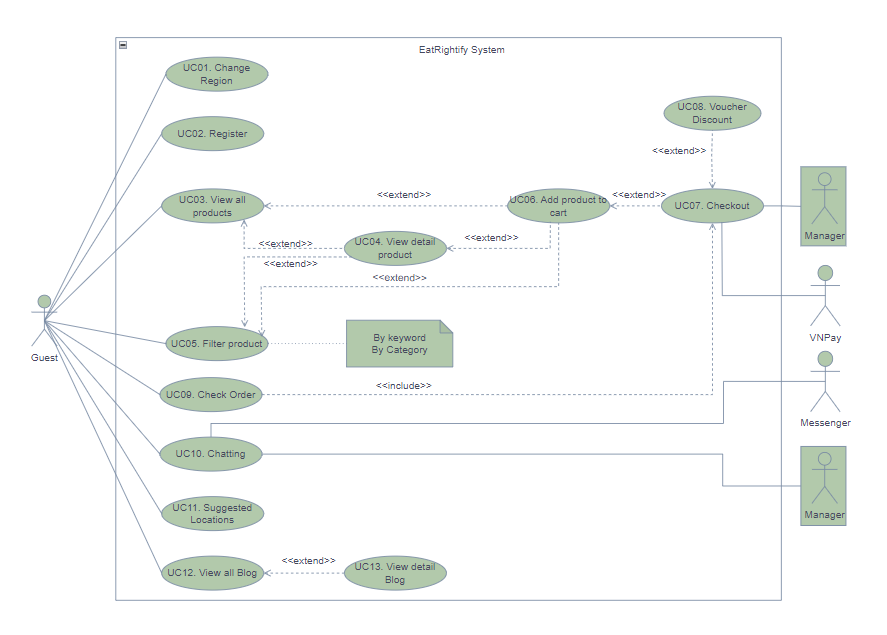
5.Go through all UML diagrams with any available software

I using <https://app.diagrams.net/>

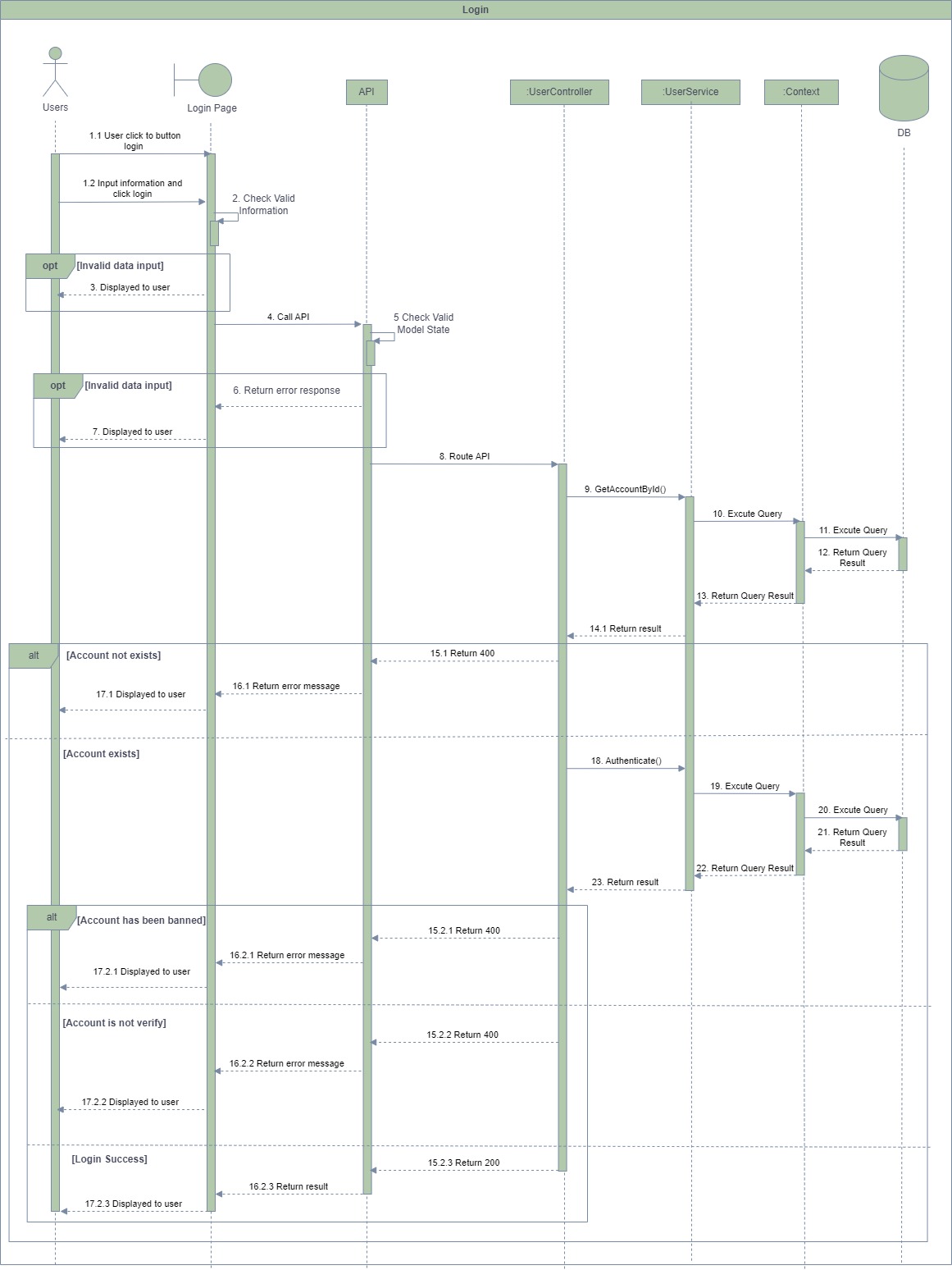
- Class diagram:



- Usecase diagram



- Sequence diagram:



- Package diagram:

