ASSIGNMENT 1
COEN 275

Pujitha Kallu
ID : W1653660
pkallu@scu.edu

1.) You are leading a development team at a streaming service company that has recently seen rapid growth. The current system struggles to handle the increasing number of users streaming content simultaneously, causing buffering and downtime. You propose a new Scalable Video Streaming System to address these issues. The CEO supports the idea but emphasizes that the system must be capable of scaling up quickly and maintaining high availability. Answer the following questions:

a)What software development strategy would you choose to implement this new system and why? (Min 150 words, 2 Marks)
Ans:
To implement the new Scalable Video Streaming System, I would choose an Agile development strategy, specifically utilizing the Scrum framework. This approach is well-suited for the given scenario for several reasons:

1. Iterative Development: Agile allows for iterative development with short sprints (typically 2-4 weeks). This is crucial for a rapidly growing streaming service, as it enables us to quickly deliver functional increments of the system. We can start with a Minimum Viable Product (MVP) that addresses the most critical scalability issues and then continuously improve and expand the system's capabilities.

2. Flexibility and Adaptability: The streaming industry is dynamic, with evolving technologies and user expectations. Agile's flexibility allows us to adapt to changing requirements or newly discovered challenges during the development process. This is particularly important when dealing with scalability and high availability, as we may need to adjust our approach based on real-world performance data.

3. Continuous Feedback: Scrum's sprint reviews and retrospectives provide regular opportunities for stakeholder feedback, including the CEO. This ensures that we're consistently aligned with the company's goals of rapid scaling and high availability. We can quickly course-correct if any aspect of the system isn't meeting expectations.

4. Focus on Quality and Testing: Agile emphasizes continuous integration and testing throughout the development process. This is critical for a high-availability system, as it helps us identify and address potential issues early, reducing the risk of downtime or performance problems when scaling up.

5. Cross-functional Teams: Scrum promotes cross-functional teams, which is beneficial for developing a complex system like this. We can bring together experts in video streaming, scalability, cloud infrastructure, and user experience to collaborate closely, ensuring all aspects of the system are well-integrated.

6. Transparency and Progress Visibility: Agile provides clear visibility into the project's progress through tools like burndown charts and sprint backlogs. This transparency will be valuable for keeping the CEO and other stakeholders informed about the system's development and our ability to meet scaling targets.

b) How will you ensure that the system can handle sudden spikes in user activity without affecting performance? (Min 150 words, 2 Marks)
Ans:
To ensure the system can handle sudden spikes in user activity without affecting performance, I would implement a multi-faceted approach focusing on elasticity, load balancing, and content delivery optimization. Here's how we would address this challenge:

1. Auto-scaling Architecture:
We would design the system with auto-scaling capabilities, particularly leveraging cloud services. This would allow the system to automatically provision additional resources (e.g., servers, containers) when user activity increases. We'd set up scaling policies based on metrics like CPU utilization, network traffic, or concurrent user sessions. This elastic infrastructure would expand during peak times and contract during lulls, ensuring efficient resource usage while maintaining performance. In other words by using cloud platforms like AWS, Google Cloud, or Microsoft Azure, we can automatically scale up or down the number of server instances, ensuring sufficient capacity during peak times and cost efficiency during low activity periods.

2. Load Balancing:
Implementing intelligent load balancers would be crucial. These would distribute incoming requests across multiple servers, preventing any single point of failure and ensuring even resource utilization. We'd use algorithms that consider server health, current load, and geographical proximity to users for optimal request routing.

3. Content Delivery Network (CDN):
Utilizing a robust CDN would be essential for handling traffic spikes. By caching content at edge locations closer to users, we can reduce the load on our origin servers and decrease latency. During sudden activity increases, the CDN would absorb much of the additional traffic, maintaining performance even as user numbers surge.

4. Microservices Architecture:
Breaking down the system into microservices would allow us to scale individual components independently. For instance, the video transcoding service could scale separately from the user authentication service, allowing for more granular and efficient resource allocation during activity spikes.

5. Caching Strategies:
Implementing multi-level caching (e.g., in-memory caches, distributed caches) would reduce the load on backend databases and services. This would be particularly effective for frequently accessed data like user profiles or popular video metadata.

6. Asynchronous Processing:
For non-critical operations that don't require immediate processing, we'd use message queues and asynchronous processing. This would help manage sudden increases in workload by deferring non-essential tasks during peak times.

7. Performance Monitoring and Predictive Scaling:
We'd implement comprehensive monitoring to track system performance in real-time. By analyzing historical data and patterns, we could also implement predictive scaling, proactively increasing capacity before anticipated spikes (e.g., during major sporting events or popular show premieres).

8. Database Optimization:
Employing database sharding and reading replicas would help manage increased database load during activity spikes. This would distribute data across multiple servers and alleviate pressure on the primary database.

2.)You are a software development manager for a company that is working with a client to develop a complex web-based CRM (Customer Relationship Management) system. The client often changes the project scope, adding new features or adjusting the requirements based on market trends. Your

team, which is globally distributed across different time zones, faces coordination challenges and communication bottlenecks. Due to the frequent changes and remote work environment, progress has slowed. What software development methodology would you recommend to manage this project effectively, and why? (Min 200 words, 2 Marks).
Ans:
For a project involving the development of a complex web-based CRM system with a frequently changing scope, remote teams, and communication challenges, I would recommend the **Agile development methodology**, specifically the **Scrum framework**. Agile methodologies are designed to handle dynamic requirements and continuous client feedback, making them well-suited to projects with evolving scopes.

Scrum breaks the development process into smaller, manageable units called **sprints**, typically lasting two to four weeks. Each sprint focuses on delivering a potentially shippable product increment, allowing the team to adapt to changes quickly. This approach is ideal when the client frequently alters requirements, as it provides the flexibility to accommodate new features or adjustments without disrupting the overall project.

The **Scrum framework** also emphasizes frequent communication through **daily stand-up meetings**, sprint planning, and retrospective sessions. This helps in identifying bottlenecks, setting clear goals, and tracking progress, even with a globally distributed team. The use of time-boxed sprints ensures that the team remains focused on achieving specific objectives, minimizing the impact of changing requirements and preventing scope creep.

To overcome the coordination challenges due to different time zones, leveraging **Agile tools** like **Jira, Trello**, or **Slack** can be highly beneficial. These tools facilitate asynchronous communication, task tracking, and collaboration, ensuring that team members stay aligned on the project's goals regardless of their location. Furthermore, the Agile methodology's iterative approach allows for regular feedback from the client at the end of each sprint, reducing misunderstandings and ensuring that the product aligns with the client's evolving expectations.

Agile's emphasis on **adaptability, transparency, and continuous improvement** makes it the most effective methodology for managing projects with unpredictable requirements and remote teams. It enables quicker delivery of features, ensures that the team can pivot based on client needs, and maintains a high level of collaboration across different time zones. This makes the development process more efficient and responsive, leading to a higher-quality CRM system that meets the client's demands.

3.)Write a custom class named Matrix that represents a 2x2 matrix. Implement the following functionalities:
-Constructors (default, parameterized, and copy).
-Overload the + operator to add two matrices and return the result.

-Overload the * operator to multiply two matrices and return the result.
-Write a function that transposes a matrix. Implement the above in a separate header file and demonstrate the functionality in a main.cpp file.(2 Marks)

Sol: #Main.cpp

Matrix > main.cpp > main()

```cpp
#include <iostream>
#include "Matrix.h"

Tabnine | Edit | Test | Explain | Document | Ask
void printMatrix(const Matrix& m) {
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            std::cout << m.get(i, j) << " ";
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}

Tabnine | Edit | Test | Explain | Document | Ask
int main() {
    // Test default constructor
    Matrix m1;
    std::cout << "Default constructor (m1):" << std::endl;
    printMatrix(m1);

    // Test parameterized constructor
    Matrix m2(1, 2, 3, 4);
    std::cout << "Parameterized constructor (m2):" << std::endl;
    printMatrix(m2);

    // Test copy constructor
    Matrix m3 = m2;
    std::cout << "Copy constructor (m3 = m2):" << std::endl;
    printMatrix(m3);

    // Test addition
    Matrix m4 = m2 + m3;
    std::cout << "Addition (m4 = m2 + m3):" << std::endl;
    printMatrix(m4);

    // Test multiplication
    Matrix m5 = m2 * m3;
    std::cout << "Multiplication (m5 = m2 * m3):" << std::endl;
    printMatrix(m5);

    // Test transpose
    Matrix m6 = m2.transpose();
    std::cout << "Transpose of m2:" << std::endl;
    printMatrix(m6);

    return 0;
}
```

#matrix.h

```h
#ifndef MATRIX_H
#define MATRIX_H

class Matrix {
private:
    double data[2][2];

public:
    // Default constructor
    Tabnine | Edit | Test | Explain | Document | Ask
    Matrix();

    // Parameterized constructor
    Tabnine | Edit | Test | Explain | Document | Ask
    Matrix(double a, double b, double c, double d);

    // Copy constructor
    Tabnine | Edit | Test | Explain | Document | Ask
    Matrix(const Matrix& other);

    // Overloaded + operator
    Matrix operator+(const Matrix& other) const;

    // Overloaded * operator
    Matrix operator*(const Matrix& other) const;

    // Transpose function
    Tabnine | Edit | Test | Explain | Document | Ask
    Matrix transpose() const;

    // Getter for accessing elements
    Tabnine | Edit | Test | Explain | Document | Ask
    double get(int row, int col) const;

    // Setter for modifying elements
    Tabnine | Edit | Test | Explain | Document | Ask
    void set(int row, int col, double value);
};
```

```cpp
// Default constructor
Tabnine | Edit | Test | Explain | Document | Ask
Matrix::Matrix() {
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            data[i][j] = 0.0;
        }
    }
}

// Parameterized constructor
Tabnine | Edit | Test | Explain | Document | Ask
Matrix::Matrix(double a, double b, double c, double d) {
    data[0][0] = a; data[0][1] = b;
    data[1][0] = c; data[1][1] = d;
}

// Copy constructor
Tabnine | Edit | Test | Explain | Document | Ask
Matrix::Matrix(const Matrix& other) {
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            data[i][j] = other.data[i][j];
        }
    }
}

// Overloaded + operator
Tabnine | Edit | Test | Explain | Document | Ask
Matrix Matrix::operator+(const Matrix& other) const {
    Matrix result;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            result.data[i][j] = data[i][j] + other.data[i][j];
        }
    }
    return result;
```

```cpp
// Overloaded * operator
Tabnine | Edit | Test | Explain | Document | Ask
Matrix Matrix::operator*(const Matrix& other) const {
    Matrix result;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            result.data[i][j] = data[i][0] * other.data[0][j] + data[i][1] * other.data[1][j];
        }
    }
    return result;
}

// Transpose function
Tabnine | Edit | Test | Explain | Document | Ask
Matrix Matrix::transpose() const {
    return Matrix(data[0][0], data[1][0], data[0][1], data[1][1]);
}

// Getter
Tabnine | Edit | Test | Explain | Document | Ask
double Matrix::get(int row, int col) const {
    return data[row][col];
}

// Setter
Tabnine | Edit | Test | Explain | Document | Ask
void Matrix::set(int row, int col, double value) {
    data[row][col] = value;
}

#endif // MATRIX_H
```

Output:

```
~ — m
Last login: Mon Oct 14 14:47:49 on ttys003
/Users/pujitha/Downloads/Subjects/OOAD/OOd_lab/Matrix/matrix_program ; exit;
/Users/pujitha/.zshrc:.:9: no such file or directory: /opt/homebrew/opt/asdf/libexec/asdf.sh
pujitha@Pujithas-MBP-2 ~ % /Users/pujitha/Downloads/Subjects/OOAD/OOd_lab/Matrix/matrix_program ; exit;
Default constructor (m1):
0 0
0 0

Parameterized constructor (m2):
1 2
3 4

Copy constructor (m3 = m2):
1 2
3 4

Addition (m4 = m2 + m3):
2 4
6 8

Multiplication (m5 = m2 * m3):
7 10
15 22

Transpose of m2:
1 3
2 4


Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

4.)Design a class CustomVector that implements a dynamic array with the following:

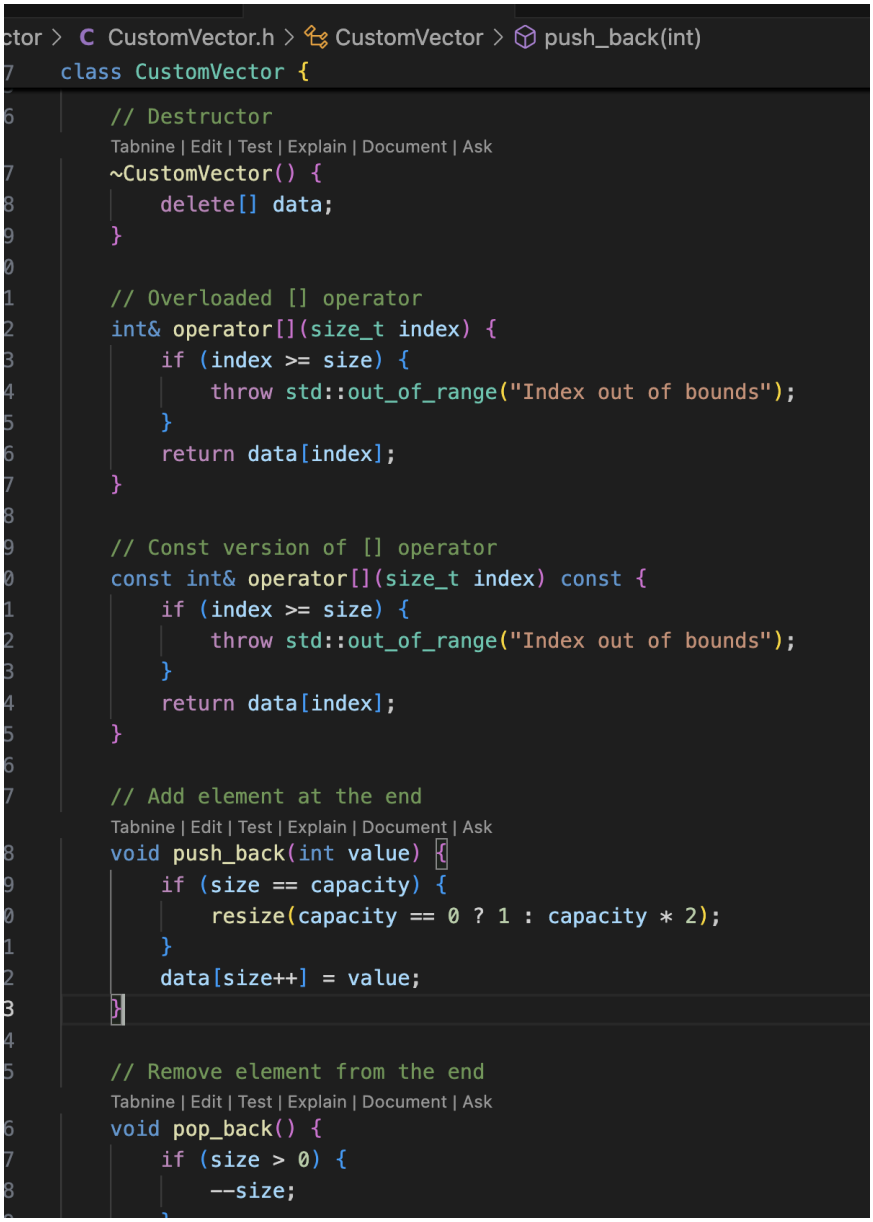
Constructors (default, parameterized, copy).
Overload the [] operator to access elements of the array.
Implement functions to add elements at the end and remove elements from the end.
Implement a destructor to properly manage memory. Test your class with a program that performs addition and removal of elements.(2 Marks)

Ans: CustomVector.h

CustomVector.cpp | CustomVector.h

Vector > CustomVector.h > CustomVector > push_back(int)

```h
#ifndef CUSTOM_VECTOR_H
#define CUSTOM_VECTOR_H

#include <stdexcept>
#include <algorithm>

class CustomVector {
private:
    int* data;
    size_t size;
    size_t capacity;

    Tabnine | Edit | Test | Explain | Document | Ask
    void resize(size_t newCapacity) {
        int* newData = new int[newCapacity];
        std::copy(data, data + size, newData);
        delete[] data;
        data = newData;
        capacity = newCapacity;
    }

public:
    // Default constructor
    Tabnine | Edit | Test | Explain | Document | Ask
    CustomVector() : data(nullptr), size(0), capacity(0) {}

    // Parameterized constructor
    Tabnine | Edit | Test | Explain | Document | Ask
    CustomVector(size_t initialCapacity) : size(0), capacity(initialCapacity) {
        data = new int[capacity];
    }

    // Copy constructor
    Tabnine | Edit | Test | Explain | Document | Ask
    CustomVector(const CustomVector& other) : size(other.size), capacity(other.capacity) {
        data = new int[capacity];
        std::copy(other.data, other.data + size, data);
    }
```


ctor > C CustomVector.h > CustomVector > push_back(int)
class CustomVector {
// Destructor
Tabnine | Edit | Test | Explain | Document | Ask
~CustomVector() {
delete[] data;
}
// Overloaded [] operator
int& operator[](size_t index) {
if (index >= size) {
throw std::out_of_range("Index out of bounds");
}
return data[index];
}
// Const version of [] operator
const int& operator[](size_t index) const {
if (index >= size) {
throw std::out_of_range("Index out of bounds");
}
return data[index];
}
// Add element at the end
Tabnine | Edit | Test | Explain | Document | Ask
void push_back(int value) {
if (size == capacity) {
resize(capacity == 0 ? 1 : capacity * 2);
}
data[size++] = value;
}
// Remove element from the end
Tabnine | Edit | Test | Explain | Document | Ask
void pop_back() {
if (size > 0) {
--size;

```
    // Get current size
    Tabnine | Edit | Test | Explain | Document | Ask
    size_t getSize() const {
        return size;
    }

    // Get current capacity
    Tabnine | Edit | Test | Explain | Document | Ask
    size_t getCapacity() const {
        return capacity;
    }
};

#endif // CUSTOM_VECTOR_H
```

#customvestor.cpp

```
tor > CustomVector.cpp > main()
#include <iostream>
#include "CustomVector.h"

Tabnine | Edit | Test | Explain | Document | Ask
void printVector(const CustomVector& vec) {
    std::cout << "Vector contents: ";
    for (size_t i = 0; i < vec.getSize(); ++i) {
        std::cout << vec[i] << " ";
    }
    std::cout << std::endl;
    std::cout << "Size: " << vec.getSize() << ", Capacity: " << vec.getCapacity() << std::endl;
}

Tabnine | Edit | Test | Explain | Document | Ask
int main() {
    // Test default constructor
    CustomVector v1;
    std::cout << "Default constructor:" << std::endl;
    printVector(v1);

    // Test parameterized constructor
    CustomVector v2(5);
    std::cout << "\nParameterized constructor (capacity 5):" << std::endl;
    printVector(v2);

    // Test push_back
    std::cout << "\nAdding elements to v2:" << std::endl;
    for (int i = 0; i < 7; ++i) {
        v2.push_back(i * 10);
        printVector(v2);
    }

    // Test copy constructor
    CustomVector v3 = v2;
    std::cout << "\nCopy constructor (v3 = v2):" << std::endl;
    printVector(v3);
```

```
    // Test pop_back
    std::cout << "\nRemoving elements from v2:" << std::endl;
    for (int i = 0; i < 3; ++i) {
        v2.pop_back();
        printVector(v2);
    }

    // Test [] operator
    std::cout << "\nAccessing elements of v2 using []:" << std::endl;
    for (size_t i = 0; i < v2.getSize(); ++i) {
        std::cout << "v2[" << i << "] = " << v2[i] << std::endl;
    }

    // Test exception handling
    try {
        std::cout << "\nTrying to access out-of-bounds element:" << std::endl;
        std::cout << v2[10] << std::endl;
    } catch (const std::out_of_range& e) {
        std::cout << "Caught exception: " << e.what() << std::endl;
    }

    return 0;
}
```

Output:

```
Last login: Mon Oct 14 16:38:14 on ttys003
/Users/pujitha/.zshrc:.:9: no such file or directory: /opt/homebrew/opt/asdf/libexec/asdf.sh
/Users/pujitha/Downloads/Subjects/OOAD/OOd_lab/Vector/vector_program ; exit;
pujitha@Pujithas-MBP-2 ~ % /Users/pujitha/Downloads/Subjects/OOAD/OOd_lab/Vector/vector_program ; exit;
Default constructor:
Vector contents:
Size: 0, Capacity: 0

Parameterized constructor (capacity 5):
Vector contents:
Size: 0, Capacity: 5

Adding elements to v2:
Vector contents: 0
Size: 1, Capacity: 5
Vector contents: 0 10
Size: 2, Capacity: 5
Vector contents: 0 10 20
Size: 3, Capacity: 5
Vector contents: 0 10 20 30
Size: 4, Capacity: 5
Vector contents: 0 10 20 30 40
Size: 5, Capacity: 5
Vector contents: 0 10 20 30 40 50
Size: 6, Capacity: 10
Vector contents: 0 10 20 30 40 50 60
Size: 7, Capacity: 10

Copy constructor (v3 = v2):
Vector contents: 0 10 20 30 40 50 60
Size: 7, Capacity: 10

Removing elements from v2:
Vector contents: 0 10 20 30 40 50
Size: 6, Capacity: 10
Vector contents: 0 10 20 30 40
Size: 5, Capacity: 10
Vector contents: 0 10 20 30
Size: 4, Capacity: 10

Accessing elements of v2 using []:
v2[0] = 0
v2[1] = 10
v2[2] = 20
v2[3] = 30

Trying to access out-of-bounds element:
Caught exception: Index out of bounds

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

Reference:

1. a) Software Development: https://www.intelivita.com/blog/software-development-strategy/
2. https://www.tricentis.com/learn/an-in-depth-look-at-spike-testing-with-examples
3. https://www.scrum.org/resources/what-scrum-module
4. https://www.w3schools.com/cpp/cpp_constructors.asp