**Final Year Design Project System**

**Requirements Specification**

***GCTConnect***

Software Design Specification Document

by

|  |  |  |
| --- | --- | --- |
| **Student Name** | **College Roll No** | **University Roll No** |
| Talha Mahboob | 21-3846 | 060054 |
| Malik Awais | 21-3843 | 060058 |
| Abdul Majid | 21-3832 | 060038 |

Project Advisor:

**Sir Imran**

Faculty of Computing & Information Technology,

University of the Punjab, Lahore, Pakistan.

(2025)

**Table of Contents**

**1. Executive Summary**

**2. Requirements Analysis**

* + **2.1 User Classes and Characteristics**
  + **2.2 Requirement Identifying Techniques**

**3. Functional Requirements**

* + **3.1 User Registration & Authentication**
  + **3.2 Automated Group Creation & Management**
  + **3.3 AI Chatbot for Academic Queries**
  + **3.4 Messaging System & Announcements**
  + **3.5 Batch Management System**

**4. Non-Functional Requirements**

* + **4.1 Reliability**
  + **4.2 Usability**
  + **4.3 Performance**
  + **4.4 Security**

**5. Use Case Analysis**

* + **5.1 User Registration (UC1)**
  + **5.2 Messaging System (UC2)**
  + **5.3 Group Creation and Deletion (UC3)**
  + **5.4 Batch Creation and Deletion (UC4)**

**6. Use Case Diagram**

* + **6.1 User Roles & Responsibilities**
  + **6.2 Admin Responsibilities**
  + **6.3 Principal Responsibilities**
  + **6.4 HOD Responsibilities**
  + **6.5 Teacher Responsibilities**
  + **6.6 Student Responsibilities**

**7. Storyboards for GCTConnect**

* + **7.1 Sending a Message in a Group**
  + **7.2 AI Chatbot Query for Academic Assistance**
  + **7.3 Principal Posting an Announcement**

**8. Summary**

**9. References**

**GCTConnect**

**A complete Social Networking Platform for Government College Township**

### **Executive Summary**

The **Software Requirements Specification (SRS)** for **GCTConnect** outlines the **functional, non-functional, and technical requirements** for a **social networking platform** designed specifically for **Government Graduate College Township, Lahore**. The system aims to **automate communication, streamline user management, and provide AI-driven assistance** for students, faculty, and administration.

The core **functional requirements** include **user registration**, **role-based access control**, **automated group creation**, **real-time messaging**, **AI chatbot integration**, and **announcement broadcasting**. **Admins manage user onboarding**, while students and faculty are automatically assigned to their respective groups based on their department and batch. The AI chatbot assists users with **academic queries**, reducing administrative workload.

The **non-functional requirements** focus on **security, scalability, and performance**. The system ensures **data integrity through SQL Server normalization**, **role-based authentication**, **password hashing**, and **encryption techniques**. The messaging system leverages **ASP.NET SignalR** for real-time communication, ensuring a **responsive and interactive experience**.

The **technical stack** includes **ASP.NET 8 MVC, C#, JavaScript, jQuery, SQL Server**, and **Microsoft SignalR**. The **three-tier architecture** provides **modular design, ease of maintenance, and scalability** for future enhancements such as **dynamic group creation, one-on-one messaging, and profile management**.

This document serves as a **blueprint** for developers, project managers, and stakeholders, ensuring that **GCTConnect** meets institutional networking and communication needs **efficiently and securely** while being adaptable for **future expansions**.

**Requirements Analysis**

##### **1. User Classes and Characteristics**

The **GCTConnect** system is designed for a diverse set of users within **Government Graduate College Township, Lahore**. Each user class has distinct characteristics, roles, and permissions within the system.

|  |  |  |
| --- | --- | --- |
| **User Class** | **Description** | **Characteristics & Responsibilities** |
| **Students** | Undergraduate and graduate students | - Can view announcements, join department-specific and batch-specific groups, send messages, and interact with the AI chatbot. |
| **Teachers** | Faculty members of different departments | - Can communicate with students and colleagues, share academic resources, and send department-wide announcements. |
| **Heads of Departments (HODs)** | Senior faculty managing academic departments | - Can oversee department communication, send important announcements, and manage department-specific chat groups. |
| **Principal** | College's top management authority | - Can send global announcements, monitor communication flows, and manage system-wide updates. |
| **Administrators** | IT and administrative staff managing user registrations and system operations | - Responsible for onboarding new users, managing batch creation and deletion, and ensuring smooth system operation. |

Each user class has **role-based access control (RBAC)** to ensure **restricted access** based on their level of authority. **For example, students cannot create groups, while teachers, HODs, and administrators have expanded privileges.**

##### **2. Requirement Identifying Techniques**

To accurately define system requirements, a combination of requirement elicitation techniques was used:

|  |  |  |
| --- | --- | --- |
| **Technique** | **Description** | **Application in GCTConnect** |
| **Interviews** | Conducted with faculty, students, and administrative staff to understand their communication challenges. | Identified the need for **automated group creation**, **batch-based user assignment**, and **announcement broadcasting.** |
| **Surveys & Questionnaires** | Online forms were distributed to students and staff to gather opinions on existing communication issues. | Revealed inefficiencies in **WhatsApp-based communication**, leading to the inclusion of **structured chat groups and AI chatbot integration.** |
| **Observation** | Monitored how current communication happens within the college (e.g., via WhatsApp, email, noticeboards). | Highlighted **delays in announcements** and **missed messages**, reinforcing the need for **centralized messaging.** |
| **Document Analysis** | Reviewed institutional policies and existing communication guidelines. | Ensured that GCTConnect aligns with **administrative workflows** and **hierarchical communication structures.** |
| **Use Case Analysis** | Identified possible system interactions from different user perspectives. | Led to defining **role-based access control (RBAC)** and **feature restrictions** per user role. |

#### 

#### **Functional Requirements**

The **GCTConnect** system must support several key **functional requirements** to ensure efficient communication, role-based management, and AI-assisted interactions. These include:

##### **Functional Requirement X (Key Functional Features)**

**User Registration & Authentication:**

Admins can register users and assign roles (Students, Teachers, HODs, Principal).

Users receive login credentials via email.

Secure login with **password hashing** and **role-based access control (RBAC)**.

**Automated Group Creation & Management:**

System **automatically assigns** users to groups based on their role, batch, and department.

Admins can **create, modify, or delete** groups dynamically.

Teachers, HODs, and students can **send messages and share files** within groups.

**AI Chatbot for Academic Queries:**

AI chatbot provides **real-time responses** to admission queries, event schedules, and other academic questions.

Integrated with a **knowledge base** for automated assistance.

**Messaging System & Announcements:**

Students and faculty can send **text messages, documents, and multimedia** in groups.

The Principal and HODs can send **institution-wide or department-wide announcements.**

Messages are stored in **SQL Server** for future reference.

**Batch Management System:**

Admins can **create, update, and delete** student batches.

Students are assigned to batches dynamically upon registration.

#### **Non-Functional Requirements**

In addition to functional capabilities, **GCTConnect** must meet **non-functional requirements** to ensure **reliability, usability, performance, and security.**

##### **Reliability**

The system must provide **99.9% uptime** to ensure uninterrupted communication.

Automatic **error logging and reporting** must be implemented for system monitoring.

**Database backups** must be scheduled to prevent data loss.

##### **Usability**

The user interface must be **intuitive and accessible** for students, teachers, and administrators.

The system should be **responsive** across **desktop, tablet, and mobile** devices.

A **simple and structured UI** with **role-based dashboards** must be provided.

##### **Performance**

The system must handle **1000+ concurrent users** without lag.

Messages and announcements should be **delivered within milliseconds** using **SignalR for real-time communication.**

Database queries must be **optimized** for **fast retrieval and minimal latency.**

##### **Security**

Passwords must be **hashed and encrypted** using secure algorithms.

**Role-based access control (RBAC)** must ensure restricted access to features based on user roles.

**Data encryption** must be applied to sensitive information, such as user credentials and messages.

**SQL Injection Prevention** must be enforced using **parameterized queries.**

### **Use Case Analysis**

Below are the detailed **Use Case Specifications** for four critical functionalities in **GCTConnect**:

* **User Registration (UC1)**
* **Messaging System (UC2)**
* **Group Creation/Deletion (UC3)**
* **Batch Creation/Deletion (UC4)**

Each use case outlines **requirements actors conditions flow of events and exceptions**

### **Use Case #1: User Registration and Authentication**

#### **Table 1: Use Case Description for User Registration and Authentication**

| **UC Identifier** | **UC1** |
| --- | --- |
| **Requirements Traceability** | FR-01: User registration FR-02: Secure login FR-03: Automated email verification |
| **Purpose** | Allow new users (students teachers HODs principal admin) to register and authenticate securely into the system |
| **Priority** | High |
| **Preconditions** | Admin must initiate user registration User must have a valid email address to receive credentials |
| **Postconditions** | User receives login credentials via email successfully logs into the system and is assigned to appropriate role-based groups |
| **Actors** | Admin (triggers registration) User (logs in after receiving credentials) System (verifies and stores user data) |
| **Extends** | None |
| **Main Success Scenario** |  |
| Admin enters user details and assigns role (Student Teacher HOD Principal Admin) |  |
| System generates login credentials and sends them via email |  |
| User receives email and logs into the system |  |
| System verifies credentials and grants access |  |
| User is redirected to a role-based dashboard |  |
| **Alternate Flows** |  |
| If the user is **already registered** system prevents duplicate registration |  |
| If the **email is incorrect or not found** system prompts admin to re-enter |  |
| **Exceptions** |  |
| Email **delivery failure** due to invalid email |  |
| System **database error** while storing user data |  |
| **Includes** | None |

### **Use Case #2: Messaging System (Group and Direct Messages)**

#### **Table 2: Use Case Description for Messaging System**

| **UC Identifier** | **UC2** |
| --- | --- |
| **Requirements Traceability** | FR-04: Real-time chat FR-05: Group messaging FR-06: File sharing |
| **Purpose** | Allow students teachers and administrators to communicate via group and private messages securely |
| **Priority** | High |
| **Preconditions** | User must be logged in and assigned to at least one group |
| **Postconditions** | Message is successfully delivered and stored in the database Recipients receive real-time notifications |
| **Actors** | User (Student Teacher HOD Principal) System (Handles message processing) Database (Stores messages) |
| **Extends** | None |
| **Main Success Scenario** |  |
| User selects a chat group or direct message option |  |
| User enters message text and clicks send |  |
| System **validates message content** (e.g. no restricted words) |  |
| Message is **sent to the database** and forwarded to recipients via **SignalR real-time updates** |  |
| Recipients receive message instantly and can reply |  |
| **Alternate Flows** |  |
| If **user is offline** message is stored and delivered when they log in |  |
| If **message contains restricted content** system blocks it and notifies the sender |  |
| **Exceptions** |  |
| Database connection failure prevents message from being stored |  |
| Network issues cause delayed message delivery |  |
| **Includes** | UC3: File sharing functionality (optional feature) |

### **Use Case #3: Group Creation and Deletion**

#### **Table 3: Use Case Description for Group Management**

| **UC Identifier** | **UC3** |
| --- | --- |
| **Requirements Traceability** | FR-07: Group creation FR-08: Group deletion |
| **Purpose** | Enable admins and authorized faculty members to create or delete discussion groups based on batch department or academic purpose |
| **Priority** | High |
| **Preconditions** | User must have admin teacher or HOD privileges |
| **Postconditions** | Group is created and assigned to relevant users or deleted successfully |
| **Actors** | Admin HOD Teachers System Database |
| **Extends** | None |
| **Main Success Scenario** |  |
| Admin or HOD selects **Create Group** |  |
| System requests **group name description and department (if applicable)** |  |
| System **assigns users** automatically based on role batch or department |  |
| Group is successfully created and available in the system |  |
| **Alternate Flows** |  |
| If **group name already exists** system prompts user to enter a unique name |  |
| If **deleting a group** system asks for confirmation before deletion |  |
| **Exceptions** |  |
| Database connection failure prevents group creation |  |
| System denies group deletion if it is an **official system group** |  |
| **Includes** | None |

### **Use Case #4: Batch Creation and Deletion**

### **Table 4: Use Case Description for Batch Management**

| **UC Identifier** | **UC4** |
| --- | --- |
| **Requirements Traceability** | FR-09: Batch creation FR-10: Batch deletion |
| **Purpose** | Allow administrators to create and manage student batches for academic organization |
| **Priority** | High |
| **Preconditions** | User must have admin privileges |
| **Postconditions** | Batch is successfully created assigned to students or deleted |
| **Actors** | Admin System Database |
| **Extends** | None |
| **Main Success Scenario** |  |
| Admin selects **Create Batch** |  |
| System requests **batch year department and student list** |  |
| Batch is created and students are automatically assigned based on department |  |
| **Alternate Flows** |  |
| If **batch already exists** system prevents duplicate batch creation |  |
| If **deleting a batch** system ensures students are reassigned before deletion |  |
| **Exceptions** |  |
| Database connection failure prevents batch creation |  |
| System prevents batch deletion if students are still linked |  |
| **Includes** | None |

## .

## ****Use Case Diagram:****

The **Use Case Diagram** for **GCTConnect** illustrates the **interactions between different user roles** and their respective **tasks within the system**. This diagram provides a **clear visualization of role-based access and functionality**, ensuring that each **user type has defined privileges and actions**.

## ****User Roles & Their Responsibilities****

### ****1. Admin****

**The Admin is responsible for managing users, groups, and system settings.**

* **User Registration (UC1):** The **Admin** registers users (students, teachers, HODs, and the principal) in the system.
* **Login & Authentication (UC2):** The Admin logs in to manage platform functionalities.
* **Create & Delete Groups (UC3):** The Admin can **automatically or manually** create and delete groups based on departments, batches, and faculty.
* **Create & Delete Batches (UC4):** The Admin has the ability to create or remove student batches.
* **Manage User Roles (UC8):** The Admin assigns user roles (e.g., student, teacher, HOD, principal) and **grants permissions** accordingly.

**Importance:** The Admin acts as the **primary controller** of user access and system configurations.

### ****2. Principal****

**The Principal has administrative privileges but focuses on high-level announcements and monitoring.**

* **Login & Authentication (UC2):** The Principal logs in to access the platform.
* **View Announcements (UC6):** The Principal can view official college-wide announcements.
* **Post Announcements (UC9):** The Principal can send **official announcements** to students, teachers, and HODs.

**Importance:** The Principal ensures that important updates and college-wide communications are **delivered effectively**.

### ****3. Head of Department (HOD)****

**HODs play a managerial role within their respective departments.**

* **Login & Authentication (UC2):** HODs log in to access their department’s communication and announcements.
* **Create & Delete Groups (UC3):** HODs can create departmental discussion groups for faculty and students.
* **Send Messages (UC5):** HODs can send important messages within their assigned department.
* **View Announcements (UC6):** HODs receive college-wide and department-specific announcements.
* **Post Announcements (UC9):** HODs can post announcements for their department.

🔹 **Importance:** The HOD ensures that **departmental communication remains structured and effective**.

### ****4. Teacher****

**Teachers communicate with students, share academic discussions, and view institutional updates.**

* **Login & Authentication (UC2):** Teachers log in to access student discussions and course-related groups.
* **Send Messages (UC5):** Teachers communicate with students in subject-based or batch-specific groups.
* **View Announcements (UC6):** Teachers can read important institutional and department-level announcements.

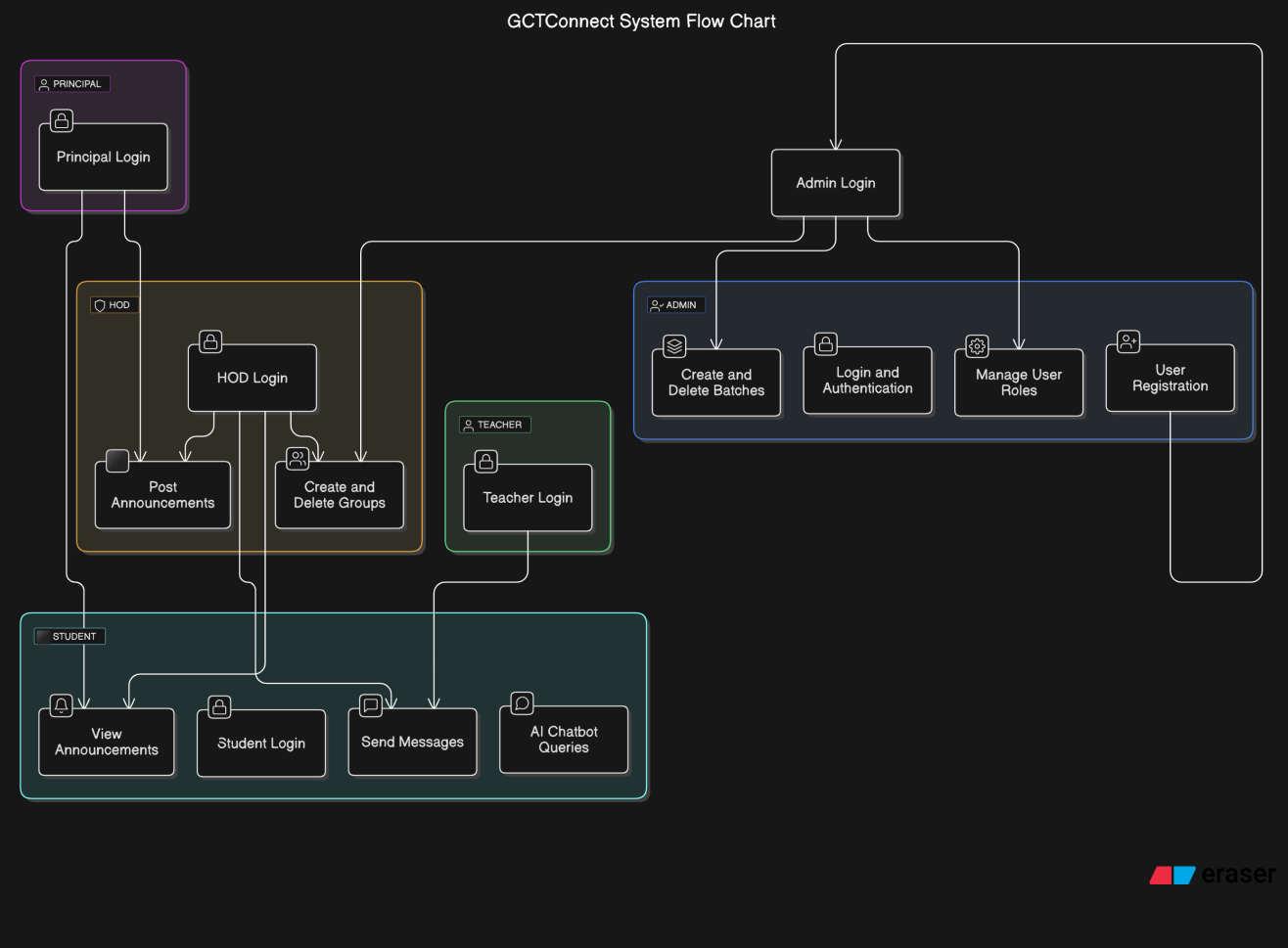
**Importance:** Teachers facilitate **student engagement** by maintaining **active communication channels**.

### ****5. Student****

**Students are the primary users, interacting with peers, faculty, and the AI chatbot.**

* **Login & Authentication (UC2):** Students log in to access their communication groups.
* **Send Messages (UC5):** Students can participate in discussions within their assigned groups.
* **View Announcements (UC6):** Students receive updates from teachers, HODs, and the principal.
* **AI Chatbot Queries (UC7):** Students can interact with the AI chatbot to get **academic and college-related information (e.g., admissions, event schedules, FAQs).**

**Importance:** Students **actively engage with the system** for communication, academic discussions, and quick AI-powered assistance.



### **Storyboards for GCTConnect**

Storyboards provide a **visual representation** of **user interactions** with the system, showing **how users perform different tasks and how the system responds**. Below are three **storyboards** representing key **user actions in GCTConnect**, excluding **Login and User Registration**.

## **Storyboard 1: Sending a Message in a Group**

### **User Goal:** A student or teacher wants to send a message in a group chat.

### **Steps & System Response:**

**User Action:** User opens GCTConnect and navigates to the chat section.  
**System Response:** Displays available chat groups.

**User Action:** User selects a relevant group (e.g., “CS Batch 2024” or “Faculty Chat”).  
**System Response:** Loads the group chat history.

**User Action:** User types a message and presses "Send".  
**System Response:** Verifies permissions (ensures the user belongs to the group).

**User Action:** If verified, the message is delivered to all group members.  
**System Response:** Notifies group members of the new message.

**User Action:** Other users respond in real-time.  
**System Response:** Updates the chat history dynamically.

## **Storyboard 2: AI Chatbot Query for Academic Assistance**

### **User Goal:** A student wants to ask the AI chatbot about an academic-related query.

### **Steps & System Response:**

**User Action:** Student opens GCTConnect and selects the AI chatbot option.  
**System Response:** Displays the chatbot interface with a prompt to ask a question.

**User Action:** Student types a query, such as “What is the admission deadline?”  
**System Response:** Processes the input using NLP (Natural Language Processing).

**User Action:** System fetches relevant information from the database.  
**System Response:** Displays an accurate response (e.g., “The admission deadline is March 30, 2025.”).

**User Action:** Student can ask follow-up questions (e.g., “Where can I submit documents?”).  
**System Response:** Continues responding based on database knowledge.

**User Action:** Student receives helpful information and leaves the chatbot.  
**System Response:** Saves previous queries for future reference.

## **Storyboard 3: Principal Posting an Announcement**

### **User Goal:** The principal wants to post an announcement for all students and faculty.

### **Steps & System Response:**

**User Action:** The principal logs in and selects the **Announcements** module.  
**System Response:** Displays a form to create a new announcement.

**User Action:** Principal enters an announcement title (e.g., “Exam Schedule Released”) and the content.  
**System Response:** Provides options to select the audience (College-wide, Department-specific, or Batch-specific).

**User Action:** Principal selects “College-wide” and clicks **Post Announcement**.  
**System Response:** Saves the announcement in the database and sends notifications to all users.

**User Action:** Students and faculty receive notifications.  
**System Response:** Users can open and view the announcement.

**User Action:** Announcement is available in the system for future reference.  
**System Response:** Ensures access to past announcements in the announcement history section.

These **storyboards** illustrate the **step-by-step interactions** of users with **GCTConnect**, showing **how messages, AI queries, and announcements are handled**. These visuals **help developers and designers understand user flow** and improve the system experience.

### **Summary**

GCTConnect is a **social networking and communication platform** designed to **streamline communication** among students, teachers, HODs, and higher management at **Government Graduate College Township, Lahore**. This project addresses the **inefficiencies of traditional communication methods**, such as WhatsApp groups, by introducing **automated group management, real-time messaging, AI-powered chatbot assistance, and a structured announcement system**.

The system is developed using **ASP.NET 8 MVC, C#, Microsoft SQL Server, JavaScript, jQuery, and SignalR for real-time communication**. It includes three main development phases:

1. **Core system development**, including **user registration, automated group assignments, messaging, and an admin dashboard**.
2. **Integration of an AI chatbot** to assist with **academic inquiries, admission queries, and event-related information**.
3. **Advanced features**, such as **direct messaging, dynamic group creation, user profile management, and deployment**.

Key functionalities are **visualized through sequence diagrams, use case diagrams, and flowcharts**, ensuring a structured **development process**. Security concerns such as **data breaches, system outages, and insider threats** are addressed through **role-based access control, encryption, and system monitoring**.

By providing a **centralized communication hub** with **automated role-based messaging and AI-driven assistance**, GCTConnect enhances **collaboration, academic discussions, and efficient information dissemination** within the college.

### **References**

**ASP.NET Core 8 Documentation** – Microsoft Docs:  
<https://docs.microsoft.com/en-us/aspnet/core>

**SignalR for Real-Time Communication** – Microsoft Learn:  
<https://learn.microsoft.com/en-us/aspnet/core/signalr>

**Entity Framework Core for Database Management** – Microsoft Docs:  
<https://docs.microsoft.com/en-us/ef/core>

**AI Chatbot Integration & NLP Techniques** – Towards Data Science:  
<https://towardsdatascience.com/natural-language-processing>

**Database Normalization and Optimization** – SQL Server Best Practices:  
<https://docs.microsoft.com/en-us/sql/relational-databases>

**Critical Path Method for Project Scheduling** – Project Management Institute:  
<https://www.pmi.org/learning/library/critical-path-method>

**Software Design Principles and Patterns** – Clean Architecture by Robert C. Martin:  
<https://www.oreilly.com/library/view/clean-architecture>

**Use Case Diagrams & Sequence Diagrams** – UML 2.0 Documentation:  
[https://www.uml.org](https://www.uml.org/)

**Cybersecurity Best Practices for Web Applications** – OWASP:  
<https://owasp.org/www-project-top-ten/>

**Project Development with GitHub & Version Control** – GitHub Docs:  
<https://docs.github.com/en>

These references provide a **solid foundation** for understanding the **technologies, methodologies, and best practices** used in developing GCTConnect.