**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. System Design**

* + **1.1 Design Considerations**
  + **1.2 Requirements Traceability Matrix (RTM)**

**2. Design Models**

* + **2.1 Architectural Design**
  + **2.2 Data Design**
    - **2.2.1 Entity-Relationship (ER) Diagram**
    - **2.2.2 Data Dictionary**

**3. State Diagram**

* + **3.1 Key System States in GCTConnect**
  + **3.2 State Transitions & Events**
  + **3.3 Key Considerations for Designing the State Diagram**
  + **3.4 Benefits of the State Diagram**

**4. Design Decisions**

* + **4.1 Architectural Design: Three-Tier Architecture**
  + **4.2 Object-Oriented Design (OOD) Principles**
  + **4.3 Database Design: Normalization & Constraints**
  + **4.4 Choice of Programming Language & Frameworks**
  + **4.5 AI Chatbot Integration**
  + **4.6 Security Measures**
  + **4.7 Algorithmic Choices**
  + **4.8 User Interface (UI) & User Experience (UX) Choices**

**5. Summary**

**6. References**

**GCTConnect**

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

## 

### **Software Design Specification (SDS) - Summary**

The **GCTConnect** software design specification document outlines the architectural and technical design of a **social networking platform** tailored for **Government Graduate College Township, Lahore**. The system is developed using **ASP.NET 8 MVC and Web API** with **C#, HTML, CSS, JavaScript, and jQuery** for front-end and back-end development. The core functionalities include **automated user registration**, **role-based group management**, **AI chatbot integration**, and **secure messaging** between students, faculty, and administration. The system ensures structured communication by dynamically assigning users to groups based on their roles and enabling hierarchical announcement broadcasting. A **relational database** (SQL Server) is employed for managing user accounts, group structures, and chat logs, ensuring data integrity and scalability.

The design follows a **modular and scalable architecture**, allowing **future enhancements** such as **dynamic group creation**, **one-on-one messaging**, and **full profile management**. Security measures, including **user authentication, role-based access control, and encrypted communication**, are implemented to protect sensitive academic data. The AI-powered chatbot, integrated using **NLP models**, provides real-time responses to frequently asked questions, reducing administrative workload. **Version control** is managed through **GitHub**, and the software undergoes continuous testing to ensure reliability. The system is designed to be **user-friendly, efficient, and adaptable**, enabling **seamless communication** within the institution while minimizing manual administrative efforts.

### 

### **System Design**

#### **1. Design Considerations**

The **GCTConnect** system is designed with key considerations to ensure **scalability, efficiency, security, and usability**. The following design aspects were taken into account:

* **Scalability:** The platform is built using **ASP.NET 8 MVC and Web API**, ensuring it can handle a growing number of users without performance degradation. The database is optimized using **SQL Server**, allowing efficient data retrieval and storage.
* **Automation & Role-Based Access:** Automated user registration and dynamic group allocation are implemented to minimize manual effort and errors in user management. Different user roles (students, teachers, hods, top management, administration) are assigned access privileges accordingly.
* **Security & Privacy:** The system includes **secure authentication, encrypted communication**, and **role-based access control** to protect sensitive academic and personal data. Data encryption techniques ensure secure message exchanges.
* **AI Integration:** A chatbot using **Natural Language Processing (NLP)** assists users with general college-related queries such as admissions, event schedules, and announcements.
* **User Experience (UX):** The UI is designed to be **user-friendly, responsive, and accessible** across devices, ensuring easy adoption by students, faculty, and administrators.
* **Maintainability & Version Control:** The project is maintained using **GitHub**, allowing version control and collaboration. The modular code structure ensures easy maintenance and updates.

#### **2. Requirements Traceability Matrix (RTM)**

The **Requirements Traceability Matrix (RTM)** ensures that all functional and non-functional requirements are mapped to their respective design components and implementations.

| **Requirement ID** | **Requirement Description** | **Design Component** | **Implementation Status** |
| --- | --- | --- | --- |
| **FR-01** | User registration by admin with automated email credentials | User Management Module (ASP.NET 8 MVC) | Implemented ✅ |
| **FR-02** | Automatic group creation and assignment based on roles | Group Management Module | Implemented ✅ |
| **FR-03** | Batch creation and deletion for managing student groups | Batch Management Module | Implemented ✅ |
| **FR-04** | Secure login with role-based access control | Authentication & Authorization | Implemented ✅ |
| **FR-05** | AI chatbot for academic queries | AI Chatbot (NLP Model) | In Progress ⏳ |
| **FR-06** | Announcement system for higher management | Announcement Module | In Progress ⏳ |
| **FR-07** | Direct and group messaging features | Chat Module (SignalR) | In Progress ⏳ |
| **FR-08** | User profile management and updates | Profile Management Module | Planned 🚀 |
| **NFR-01** | System must handle 1000+ concurrent users | Scalable Database & API Optimization | Implemented ✅ |
| **NFR-02** | Secure user data and prevent unauthorized access | Encryption & Role-Based Access Control | Implemented ✅ |
| **NFR-03** | Responsive UI for mobile and desktop | UI/UX Design with Bootstrap & jQuery | Implemented ✅ |
| **NFR-04** | Continuous performance monitoring and optimization | Performance Monitoring Module | Pending ⏳ |

### **Design Models**

#### ****3.1 Architectural Design****

#### The **GCTConnect** system follows a **three-tier architecture** to ensure modularity, scalability, and efficient communication between system components.

##### **Three-Tier Architecture Overview:**

* **Presentation Layer (Frontend):**
  1. Built using **HTML, CSS, JavaScript, and jQuery** for an intuitive UI.
  2. Users interact with the system through a web-based dashboard.
  3. Ensures role-based navigation (Students, Teachers, HODs, and Administrators).
* **Business Logic Layer (Backend):**
  1. Implemented using **ASP.NET 8 MVC and Web API** for handling system logic.
  2. Manages **user authentication, group assignments, AI chatbot interactions, and messaging.**
  3. Processes requests from the frontend and interacts with the database.
* **Data Layer (Database):**
  1. Uses **SQL Server** for storing user profiles, messages, announcements, and AI chatbot responses.
  2. Ensures **data integrity, optimized queries, and scalability** for handling high user loads.
  3. Implements **role-based access control and encryption** for security.

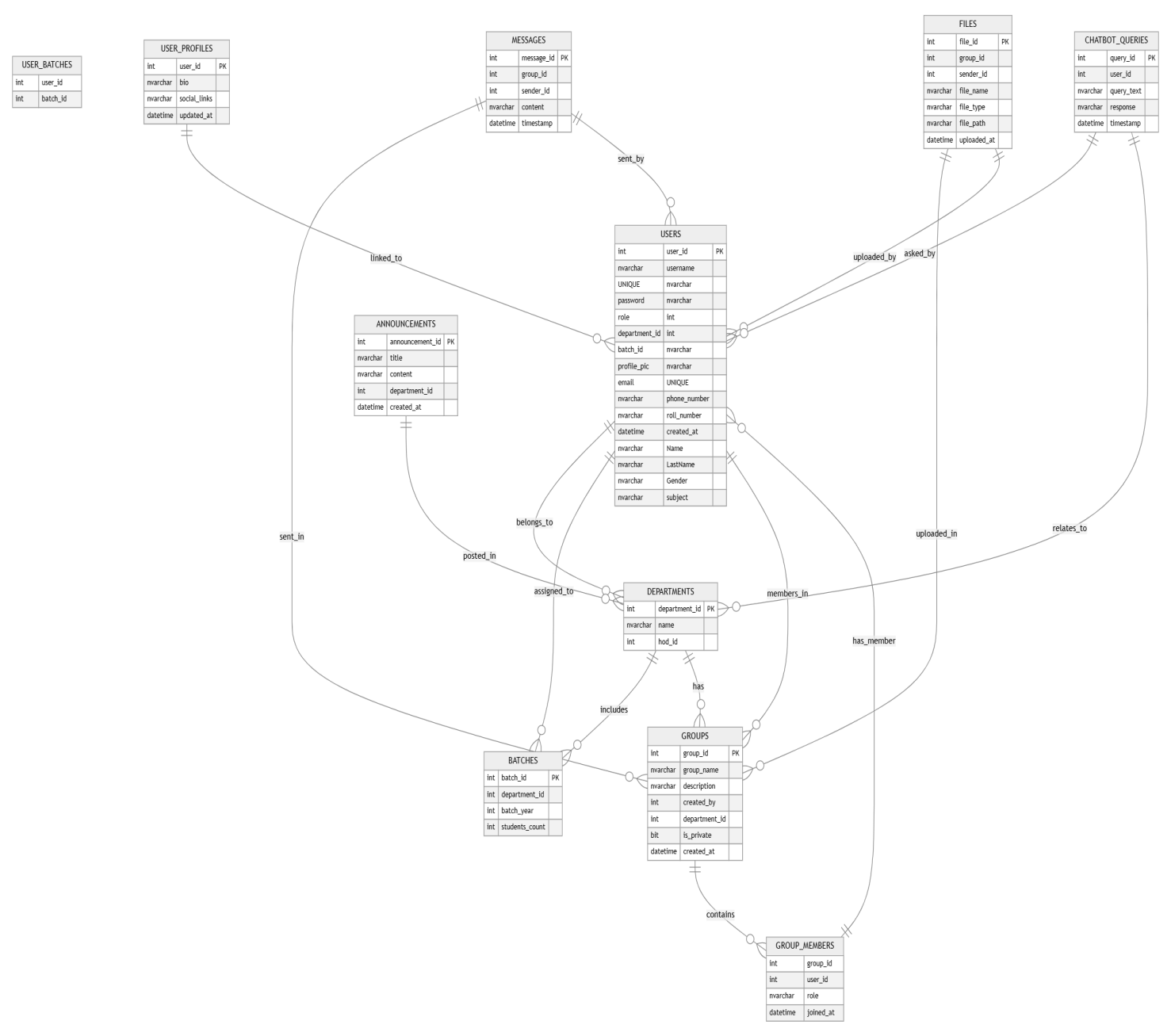
##### **System Components and Interaction:**

* **User Registration:** Admin registers users → System assigns roles automatically → Credentials sent via email.
* **Group Management:** Users are automatically assigned to relevant groups based on role and batch.
* **AI Chatbot:** Provides real-time responses to academic queries using a **predefined knowledge base**.
* **Messaging & Announcements:** Secure, structured communication between students, faculty, and management.

#### **3.2 Data Design**

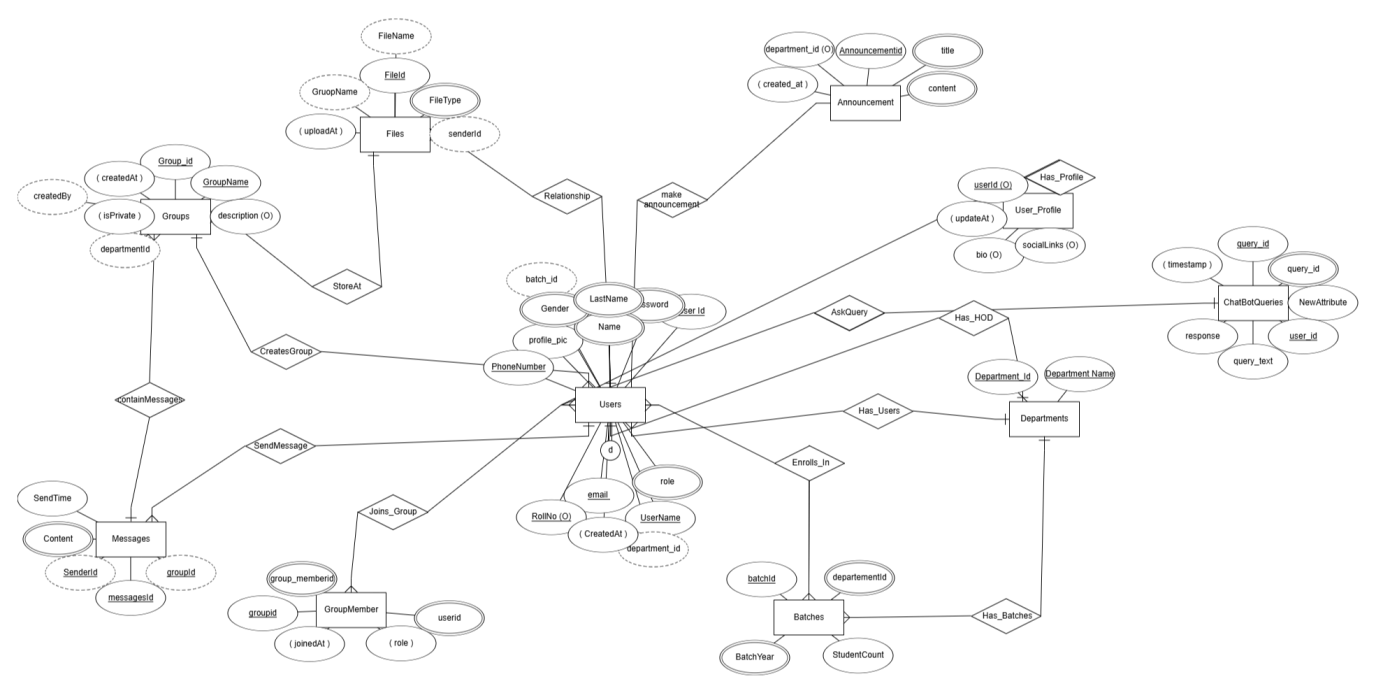
The **database design** follows a **relational model** with well-defined entities and relationships to ensure efficient data management. The key database tables include:

1. **Users Table:** Stores user details, roles, and login credentials.
2. **Groups Table:** Defines academic and administrative groups, linked to users dynamically.
3. **Messages Table:** Stores communication logs between users and groups.
4. **Announcements Table:** Maintains official announcements from higher management.
5. **AI Chatbot Knowledge Base:** Stores predefined responses to queries.
6. **Batch Table:** Handles student batch creation and deletion dynamically.



##### **Entity-Relationship (ER) Diagram:**

* **Users** (UserID, Name, Email, Role, PasswordHash)
* **Groups** (GroupID, Name, Type, CreatedBy, CreatedOn)
* **GroupMember** (UserID, GroupID) (Mapping table for many-to-many relationships)
* **Messages** (MessageID, SenderID, ReceiverID, Content, Timestamp)
* **Announcements** (AnnounceID, Title, Content, CreatedBy, CreatedOn)
* **ChatbotResponses** (QueryID, Question, Response, LastUpdated)



### **3.2.1 Data Dictionary:**

| **Table Name** | **Attribute** | **Data Type** | **Description** |
| --- | --- | --- | --- |
| **Users** | user\_id (PK) | INT (IDENTITY) | Unique identifier for each user |
|  | username | NVARCHAR(255) UNIQUE NOT NULL | Unique username assigned to each user |
|  | password | NVARCHAR(255) NOT NULL | Encrypted password for secure login |
|  | role | NVARCHAR(50) NOT NULL | Defines the user’s role (Admin, Principal, HOD, Teacher, Student) |
|  | department\_id | INT NULL | Department ID associated with the user (NULL for Admin and Principal) |
|  | batch\_id | INT NULL | Batch ID (Applicable only for students) |
|  | profile\_pic | NVARCHAR(255) NULL | Path to profile picture |
|  | email | NVARCHAR(255) UNIQUE NOT NULL | User’s registered email |
|  | phone\_number | NVARCHAR(15) NULL | Contact number of the user |
|  | roll\_number | NVARCHAR(50) NULL | Unique roll number (Required for students) |
|  | created\_at | DATETIME DEFAULT GETDATE() | Timestamp when the user was created |
|  | Name | NVARCHAR(100) NOT NULL | First name of the user |
|  | LastName | NVARCHAR(100) NOT NULL | Last name of the user |
|  | Gender | NVARCHAR(50) CHECK (Male, Female, Other) | Gender selection for the user |
|  | subject | NVARCHAR(255) NULL | Required for Teachers and HODs only |
| **Departments** | department\_id (PK) | INT (IDENTITY) | Unique identifier for departments |
|  | name | NVARCHAR(255) NOT NULL | Name of the department |
|  | hod\_id | INT NULL | ID of the Head of Department (HOD) |
| **Batches** | batch\_id (PK) | INT (IDENTITY) | Unique identifier for student batches |
|  | department\_id | INT NOT NULL | The department associated with the batch |
|  | batch\_year | INT NOT NULL | The academic year of the batch |
|  | students\_count | INT DEFAULT 0 | Number of students in the batch |
| **User\_Batches** | user\_id (PK) | INT | Links a student to their batch |
|  | batch\_id (PK) | INT | Links a batch to a student |
| **Groups** | group\_id (PK) | INT (IDENTITY) | Unique identifier for chat groups |
|  | group\_name | NVARCHAR(255) NOT NULL | Name of the group (e.g., "Computer Science Students 2024") |
|  | description | NVARCHAR(MAX) | Description of the group purpose |
|  | created\_by | INT NOT NULL | User ID of the group creator |
|  | department\_id | INT NULL | If the group is department-specific |
|  | is\_private | BIT DEFAULT 1 | Determines if the group is private or public |
|  | created\_at | DATETIME DEFAULT GETDATE() | Timestamp of group creation |
| **Group\_Members** | group\_id (PK) | INT | ID of the group |
|  | user\_id (PK) | INT | ID of the user assigned to the group |
|  | role | NVARCHAR(50) CHECK ('group\_admin', 'group\_member') | Defines the user’s role in the group |
|  | joined\_at | DATETIME DEFAULT GETDATE() | Timestamp when the user joined the group |
| **Messages** | message\_id (PK) | INT (IDENTITY) | Unique identifier for messages |
|  | group\_id (FK) | INT | Group where the message was sent |
|  | sender\_id (FK) | INT | ID of the user who sent the message |
|  | content | NVARCHAR(MAX) NOT NULL | The actual message text |
|  | timestamp | DATETIME DEFAULT GETDATE() | Timestamp when the message was sent |
| **Files** | file\_id (PK) | INT (IDENTITY) | Unique identifier for uploaded files |
|  | group\_id (FK) | INT | Group where the file was shared |
|  | sender\_id (FK) | INT | ID of the user who uploaded the file |
|  | file\_name | NVARCHAR(255) NOT NULL | Name of the uploaded file |
|  | file\_type | NVARCHAR(50) NULL | Type of file (PDF, PPT, etc.) |
|  | file\_path | NVARCHAR(255) NOT NULL | Path where the file is stored |
|  | uploaded\_at | DATETIME DEFAULT GETDATE() | Timestamp when the file was uploaded |
| **Announcements** | announcement\_id (PK) | INT (IDENTITY) | Unique identifier for announcements |
|  | title | NVARCHAR(255) NOT NULL | Title of the announcement |
|  | content | NVARCHAR(MAX) | Announcement message |
|  | department\_id | INT NULL | Associated department (NULL if general) |
|  | created\_at | DATETIME DEFAULT GETDATE() | Timestamp when the announcement was made |
| **User\_Profiles** | user\_id (PK) | INT | Unique identifier linked to a user |
|  | bio | NVARCHAR(MAX) NULL | Short bio of the user |
|  | social\_links | NVARCHAR(MAX) NULL | Social media or external links |
|  | updated\_at | DATETIME DEFAULT GETDATE() | Timestamp of the last update |
| **Chatbot\_Queries** | query\_id (PK) | INT (IDENTITY) | Unique identifier for chatbot queries |
|  | user\_id (FK) | INT NULL | User ID (if logged in) |
|  | query\_text | NVARCHAR(MAX) | User's question to the chatbot |
|  | response | NVARCHAR(MAX) | The chatbot's response |
|  | timestamp | DATETIME DEFAULT GETDATE() | Timestamp of the chatbot interaction |

* **State Diagram:**

### **State Diagram for GCTConnect**

A **state diagram** is used to **represent the various states an object (or system) goes through during its lifecycle**, based on different events and user interactions. In the case of **GCTConnect**, the **state diagram** will illustrate how the system transitions between different states when users **log in, send messages, interact with the chatbot, create announcements, and manage groups**.

### **1. Key System States in GCTConnect**

GCTConnect follows several **states**, depending on **user interactions** and **system processes**. Below are the most relevant states for creating the state diagram:

**Idle State**

* The system is waiting for **user interaction** (Login, AI Query, Group Chat, etc.).

**User Authentication State**

* User enters credentials and attempts login.
* If **credentials are valid**, transition to **Dashboard State**.
* If **invalid**, return to **Idle State** with an error message.

**Dashboard State**

* After successful login, the user is redirected to their **role-based dashboard** (Admin, Teacher, HOD, Principal, or Student).
* The user can choose to navigate to different features, such as **chat, announcements, AI chatbot, or group management**.

**Chat State**

* The user selects a **chat group** and enters a message.
* The system **validates the message** and sends it to **group members**.
* If message delivery is **successful**, it transitions to **Message Sent State**.
* If message fails, an error message is displayed, and the system **returns to Chat State**.

**AI Chatbot State**

* The student types a query in the chatbot.
* The chatbot **processes the query** using **NLP** and fetches the appropriate response.
* If the answer is found, the **response is displayed**.
* If the answer is not found, an alternative suggestion is provided, and the system remains in **AI Chatbot State**.

**Announcement State**

* The Principal, Admin, or HOD **creates an announcement**.
* The system **validates the announcement details** and **determines the audience** (College-wide, Department, Batch).
* If all conditions are met, the announcement is **successfully posted** and transitions to **Announcement Sent State**.
* If the announcement fails, an error message is displayed, and the system **returns to Announcement State**.

**Group & Batch Management State**

* The Admin or HOD **creates, modifies, or deletes groups/batches**.
* The system **validates** the changes and updates the database.
* If the operation is successful, it transitions to **Group/Batch Updated State**.
* If unsuccessful, the system remains in **Group Management State** and displays an error.

**Logout State**

* The user **clicks logout** from the dashboard.
* The system **ends the session and clears authentication tokens**.
* The user is **redirected back to Idle State** (Login Page).

### **2. State Transitions & Events**

1. **Idle State → User Authentication State** (Triggered when the user attempts login).
2. **User Authentication State → Dashboard State** (If login is successful).
3. **Dashboard State → Chat State** (When a user selects a group chat).
4. **Chat State → Message Sent State** (When a message is successfully delivered).
5. **Dashboard State → AI Chatbot State** (When a student asks a chatbot query).
6. **AI Chatbot State → AI Response State** (When the chatbot provides an answer).
7. **Dashboard State → Announcement State** (When an admin/principal creates an announcement).
8. **Announcement State → Announcement Sent State** (If the announcement is successfully posted).
9. **Dashboard State → Group Management State** (When an admin modifies groups).
10. **Group Management State → Group/Batch Updated State** (When group modifications are successful).
11. **Any State → Logout State** (When the user clicks logout).

### **3. Key Considerations for Designing the State Diagram**

* **State diagrams should include all major transitions between states**.
* **Decision points (e.g., Valid login? Message sent successfully? Announcement confirmed?) must be clearly defined**.
* **Looping transitions should be included where the user may retry an action (e.g., failed login, invalid message, chatbot reattempts, etc.)**.

### **4. Benefits of the State Diagram**

* **Clearly visualizes how the system behaves in response to user actions**.
* **Helps developers design efficient transitions between states**.
* **Identifies possible errors and failure states for debugging**.
* **Ensures smooth user experience by defining proper state handling mechanisms**.

### **state diagram**

### **Design Decisions**

### The **GCTConnect** system was designed by making several key architectural, structural, and algorithmic decisions to ensure **efficiency, scalability, and maintainability**. Below are the major design choices and their justifications:

### **1. Architectural Design: Three-Tier Architecture**

#### **Choice:**

We implemented a **three-tier architecture**, which includes the **Presentation Layer (Frontend), Business Logic Layer (Backend), and Data Layer (Database).**

#### **Why?**

* **Separation of Concerns:** Each layer is independent, making the system modular and easier to manage.
* **Scalability:** The backend and database layers can be upgraded without affecting the frontend.
* **Security:** Sensitive operations are handled in the backend, preventing direct access to the database.

### **2. Object-Oriented Design (OOD) Principles**

#### **Choice:**

We adopted the **Object-Oriented Design (OOD) principles** in system development, following the **Model-View-Controller (MVC) pattern**.

#### **Why?**

* **Encapsulation:** Separates data (models) from business logic (controllers) and presentation (views).
* **Reusability:** Code is modular and reusable, reducing duplication.
* **Scalability:** New features can be added without modifying existing components.

### **3. Database Design: Normalization & Constraints**

#### **Choice:**

We used **3rd Normal Form (3NF) Normalization** in database design.

#### **Why?**

* **Avoid Data Redundancy:** Breaking data into multiple tables prevents duplication.
* **Efficient Query Performance:** Optimized data retrieval reduces query execution time.
* **Maintain Data Integrity:** Used **primary keys, foreign keys, unique constraints, and check constraints** to ensure data validity.
* **Examples of Constraints Used:**

**Role-based constraints:** Ensures that only specific user types (Students, Teachers, HODs) have batch and department associations.

**Unique Index on Student Roll Number:** Guarantees uniqueness for student records only.

### **4. Choice of Programming Language & Frameworks**

#### **Choice:**

* **Backend:** ASP.NET 8 MVC and Web API with C#
* **Frontend:** HTML, CSS, JavaScript, jQuery
* **Database:** SQL Server

#### **Why?**

#### **ASP.NET 8 MVC:**

#### Provides a structured framework for large-scale applications.

Supports **dependency injection, security features, and RESTful API development**.

* **C#:**

Object-oriented, robust, and optimized for enterprise applications.

* **SQL Server:**

Supports complex relationships, constraints, and scalable transactions.

### **5. AI Chatbot Integration**

#### **Choice:**

Used **Natural Language Processing (NLP)** for chatbot development.

#### **Why?**

Provides **instant responses to student queries** about admissions, schedules, and events.

Reduces **manual workload for administrative staff**.

### **6. Security Measures**

#### **Choice:**

**Role-Based Access Control (RBAC):** Enforces different permissions for Students, Teachers, HODs, Admins, and Principal.

**Data Encryption:** Passwords are **hashed** using secure algorithms.

**SQL Injection Prevention:** Uses **parameterized queries** to prevent malicious database access.

#### **Why?**

Ensures **data privacy and security** for sensitive academic information.

### **7. Algorithmic Choices**

#### **Choice:**

**Batch Creation & Deletion Algorithm:** Uses SQL **triggers and stored procedures** for handling student batch assignments.

**Messaging Optimization:** Implemented **SignalR for real-time messaging** to ensure **efficient and fast chat communication.**

#### **Why?**

**Triggers and stored procedures** ensure that batch updates are executed automatically without human intervention.

**SignalR provides real-time messaging** with lower latency compared to traditional polling mechanisms.

### **8. User Interface (UI) & User Experience (UX) Choices**

#### **Choice:**

**Bootstrap & jQuery** for a **responsive and interactive UI**.

**Role-based dashboard UI** to display only relevant information to users.

#### **Why?**

Ensures **a smooth and accessible experience across devices** (PCs, tablets, and mobile phones).

### **Summary**

The **System Design** chapter provided a structured overview of the key design ideas, refinements, and critical decisions made during the development of **GCTConnect**. The chapter highlighted the **three-tier architecture**, which ensures **modularity, scalability, and maintainability** by separating the **presentation layer (frontend), business logic layer (backend), and data layer (database).**

The **object-oriented design (OOD) principles** and **MVC pattern** were adopted to enhance **code reusability, scalability, and separation of concerns.** The **database design followed 3rd Normal Form (3NF)** to minimize redundancy and ensure **data integrity through constraints and relationships** such as **role-based access control, batch and department mapping, and student roll number uniqueness.**

Security was a top priority, leading to the implementation of **role-based access control (RBAC), password hashing, encrypted communication, and SQL injection prevention.** Additionally, **AI chatbot integration using Natural Language Processing (NLP)** was designed to assist students and faculty with **automated responses for queries related to admissions, schedules, and announcements.**

Several algorithmic choices were made to optimize system performance, such as **batch creation & deletion automation using SQL triggers and stored procedures, and SignalR for real-time messaging.** These choices improved efficiency and automated various administrative processes.

The design choices in **GCTConnect** align with the project’s objectives of **improving academic communication, automating user and group management, integrating AI assistance, and enhancing overall system security.** The refined system architecture and database design **provide a scalable and efficient solution for institutional networking.**

This chapter plays a crucial role in ensuring that the system is **technically sound, efficient, and capable of meeting user expectations while supporting future enhancements.** 🚀

### **References**

#### **1. Architectural Design & Object-Oriented Design**

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. Wiley.

Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley.

#### **2. Database Design & Normalization**

Codd, E. F. (1970). "A relational model of data for large shared data banks." *Communications of the ACM, 13*(6), 377-387.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts* (7th ed.). McGraw-Hill.

Elmasri, R., & Navathe, S. B. (2017). *Fundamentals of Database Systems* (7th ed.). Pearson.

#### **3. Security & Authentication**

Viega, J., & McGraw, G. (2001). *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley.

OWASP Foundation. (2021). *OWASP Top Ten 2021: The Ten Most Critical Web Application Security Risks*. Retrieved from <https://owasp.org/www-project-top-ten/>

Bishop, M. (2005). *Introduction to Computer Security*. Addison-Wesley.

#### **4. AI Chatbot & Natural Language Processing (NLP)**

Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

Jurafsky, D., & Martin, J. H. (2021). *Speech and Language Processing* (3rd ed.). Pearson.

IBM Watson. (2020). "AI-Powered Chatbots in Education." Retrieved from https://www.ibm.com/cloud/watson-assistant

#### **5. Messaging System & SignalR Implementation**

* Microsoft Docs. (2023). *ASP.NET Core SignalR: Build Real-time Web Apps*. Retrieved from<https://docs.microsoft.com/en-us/aspnet/core/signalr/>
* Troelsen, A., & Japikse, P. (2021). *Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming*. Apress.

#### **6. Software Engineering & System Design Principles**

* Sommerville, I. (2015). *Software Engineering* (10th ed.). Pearson.

Pressman, R. S. (2019). *Software Engineering: A Practitioner’s Approach* (9th ed.). McGraw-Hill.

IEEE. (1998). *IEEE Standard 1016-1998: IEEE Recommended Practice for Software Design Descriptions*. Retrieved from<https://ieeexplore.ieee.org/>

#### **7. Web Development Technologies (ASP.NET, C#, SQL Server)**

Freeman, A. (2021). *Pro ASP.NET Core 6 MVC*. Apress.

Esposito, D. (2020). *Modern Web Development with ASP.NET Core 3: An Architect’s Guide*. Microsoft Press.

Microsoft Docs. (2023). *ASP.NET Core Documentation*. Retrieved from<https://docs.microsoft.com/en-us/aspnet/core/>

#### **8. Project Documentation & UML Modeling (Sequence Diagram)**

Rumbaugh, J., Booch, G., & Jacobson, I. (2004). *The Unified Modeling Language Reference Manual* (2nd ed.). Addison-Wesley.

Larman, C. (2011). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Pearson.