North South University

Software Engineering (CSE - 327)

Software Design Specification (SDS)

Project Title: North South University Advising System

Prepared by:

Alamin Sheikh Naim - 2013556642

Sheik Mehedi Hassan - 2021230642

Section: 7

Date: 14 NOV, 2024

Introduction

1.1 Purpose

The purpose of this document is to describe the software design of the NSU Advising System, which aims to automate and streamline the academic advising process for students, advisors, and administrators at North South University (NSU). This Software Design Specification (SDS) outlines the system architecture, components, and design decisions necessary to implement the system as described in the Software Requirements Specification (SRS).

1.2 Scope

This document defines the system architecture and detailed design of the NSU Advising System. It covers the following areas:

- **System Architecture**
- **Component Design**
- **Data Design**
- **Interface Design**
- **Security and Performance Considerations**
- **Testing Strategy**

1.3 Audience

This document is intended for:

- Software Developers responsible for implementing the system.
- Database Administrators responsible for database setup and maintenance.
- Project Managers overseeing the development and deployment of the system.
- Quality Assurance (QA) Engineers testing the system for functional and non-functional requirements.

System Architecture

2.1 Architecture Overview

The NSU Advising System is designed using a multi-tier architecture to separate concerns and enable scalability. It follows a three-tier architecture model comprising:

1. Presentation Layer (Client-side UI): Provides the interface for students, advisors, and administrators.

2. Application Layer (Business Logic): Contains the core logic for advising, registration, approval, and user management.

3. Data Layer (Database): Stores and retrieves data such as user profiles, courses, and advising sessions.

Architecture Diagram

~~~
```plaintext
+----------------------------------+
|        Presentation Layer       |
| (Student, Advisor, Admin)        |
|                              |
+----------------------------------+        +-------------------+
             |                         |  Web Server (API) |
             |                         +-------------------+
+----------------------------------+                 |
|     Business Logic Layer       |                 |
|  - Registration, Advising       |                  v
|  - Course Selection, Admin      |          +-------------------+
+----------------------------------+         |   Database Layer |
             |                         |(SQL Server, MySQL) |
             |                         +-------------------+
             v
       +------------------------+
       |  External Systems (SSO) |
~~~

```
        +-----------------------+
```

2.2 Design Constraints

Scalability: The system must be capable of handling high traffic during course registration periods. Horizontal scaling is a must for the application layer.

Security: The system must comply with NSU's data privacy policies and security protocols.

Integration: The system must integrate with existing NSU student information systems (e.g., NSU authentication service, course catalogs).

2.3 System Components

1. User Interface (UI)

- Designed with **HTML, CSS, PHP,** and **JavaScript** (React.js or Angular).
- The UI will be responsive, ensuring a smooth user experience across both desktop and mobile devices.

2. Business Logic

- Implemented in Node.js or Java/Spring for backend processing.
- RESTful APIs will facilitate communication between the frontend and backend.

3. Database

- A relational database (MySQL) will be used to store all persistent data.
- Tables include **User**, **Student**, **Course**, **Advising**, and **Registration**.

4. Security Layer:

- Use of SSL/TLS encryption for all communications.
- Single Sign-On (SSO) integration for authentication and authorization.

Design Overview

3.1 User Interface Design

- **3.1.1 Login and Authentication**

  - Login Screen: A secure login page where students, advisors, and administrators authenticate via NSU SSO.
  - Role-based Navigation**: Different users will be presented with different navigation options based on their roles (student, advisor, admin).

- **3.1.2 Dashboard Screens**

- **Student Dashboard:** Displays available courses, academic progress, and pending course approvals.
- **Advisor Dashboard:** Shows a list of students with pending course approvals and their academic progress.
- **Admin Panel:** Admins will manage users, course catalogs, and report generation.

3.2 Component Design

3.2.1 User Registration and Profile Management

Functional Description: Both students and faculty can register via NSU-provided credentials. User roles (student, advisor, admin) are defined during the registration process.

Database Tables:

User Table: Stores credentials, roles, and contact details.

Student Table: Stores academic information such as degree progress, credits, and registered courses.

Advisor Table: Stores faculty information (name, department, etc.).

3.2.2 Course Selection and Advising

Functional Description: Students browse available courses, add them to their advising list, and submit their selections for approval.

Advisor Interaction: Advisors can view the student’s course list, approve/reject courses, and provide feedback.

Database Tables:

- **Course Table:** Stores course information (name, credits, prerequisites).
- **Advising Table:** Tracks advising sessions, including approvals and feedback.

3.2.3 Admin Panel

Functional Description: Admin users can manage the courses, user roles, and system settings.

Database Tables:

- **User Role Table:** Manages different user access levels.
- **Course Catalog Table:** Stores details of the course offerings each semester.

3.3 Database Design

3.3.1 Entity-Relationship (ER) Diagram

~~~~
```plaintext

+-------------------+       +-------------------+       +-------------------+
|       User        | 1 ---- N |     Student       | 1 ---- N |     Advising      |
|-------------------|       |-------------------|       |-------------------|
| user_id (PK)      |       | student_id (PK)   |       | advising_id (PK)  |
| username          |       | user_id (FK)      |       | student_id (FK)   |
| password_hash     |       | degree_program    |       | advisor_id (FK)   |
| role              |       | completed_credits |       | selected_courses  |
| email             |       | current_courses   |       | approval_status   |
+-------------------+       +-------------------+       +-------------------+
        |                          |
        | 1                        | N
        |                          |
~~~~

```
        v                    v

+------------------+      +------------------+       +------------------+

|     Course       |      |  Registration    |       |    Admin         |

|------------------|      |------------------|       |------------------|

| course_id (PK)   |      | registration_id (PK)|    | admin_id (PK)    |

| course_name      |      | student_id (FK)  |       | user_id (FK)     |

| prerequisites    |      | course_id (FK)   |       | permissions      |

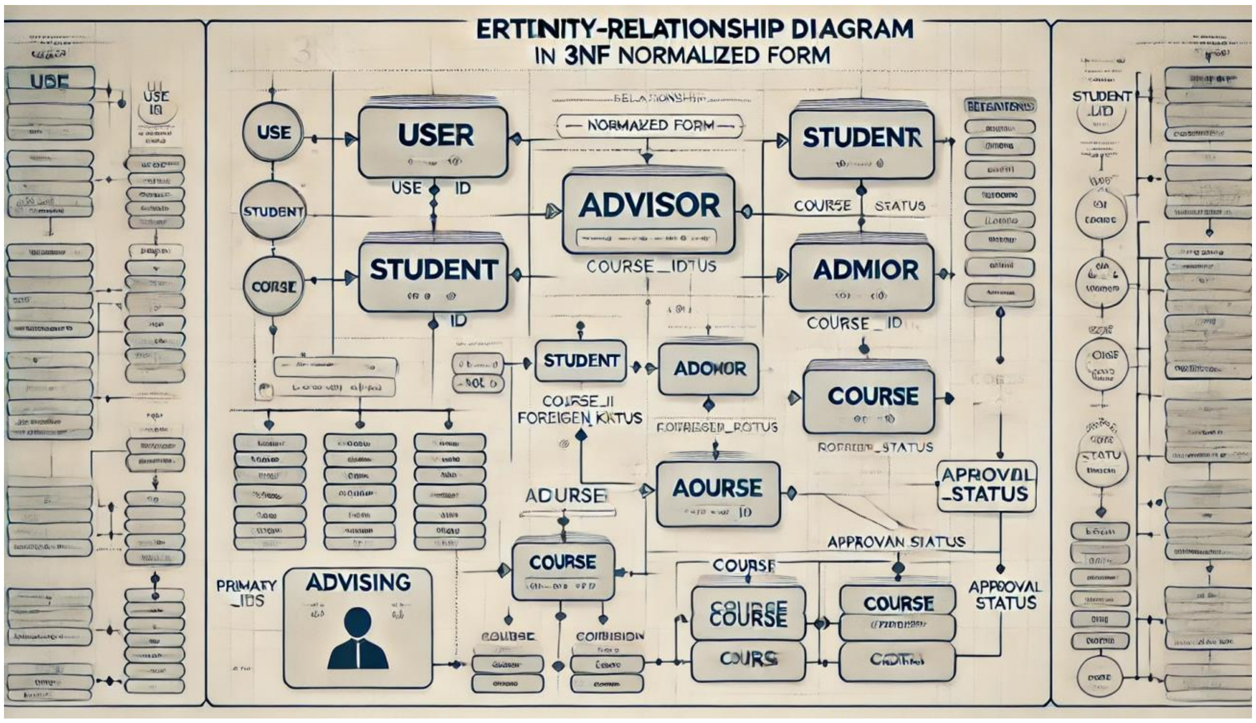
+------------------+      | registration_status|     +------------------+

                    +------------------+
```

3.3.2 Normalization

The database is normalized to **Third Normal Form** (3NF) to eliminate redundancy and ensure integrity.

3.3.3 ERD



3.3.4 Entities and Attributes

1. **User**
   - **user_id** (PK)
   - username
   - password_hash
   - role (student, advisor, admin)
   - email
2. **Student**
   - **student_id** (PK)
   - user_id (FK)
   - degree_program
   - completed_credits
   - current_courses
3. **Advisor**
   - **advisor_id** (PK)
   - user_id (FK)
   - department
4. **Admin**
   - **admin_id** (PK)
   - user_id (FK)
   - permissions
5. **Course**
   - **course_id** (PK)
   - course_name
   - prerequisites
6. **Advising**
   - **advising_id** (PK)
   - student_id (FK)
   - advisor_id (FK)
   - selected_courses
   - approval_status
7. **Registration**
   - **registration_id** (PK)
   - student_id (FK)
   - course_id (FK)
   - registration_status

Relationships

- A **User** can have one role: student, advisor, or admin.
- A **Student** is linked to a **User** (1:1).
- An **Advisor** is linked to a **User** (1:1).
- An **Admin** is linked to a **User** (1:1).
- A **Student** can interact with multiple **Courses** through **Advising** and **Registration** (1:N).
- An **Advisor** oversees multiple **Students** (1:N).
- A **Student** can have multiple advising sessions (**Advising**) (1:N).
- **Courses** can have multiple **Registrations** (1:N).

Security Design

4.1 Authentication and Authorization

Authorization: Role-based access control (RBAC) ensures that:

- **Students** can only view and select courses.
- **Advisors** can approve/reject courses for their advisees.
- **Administrators** have full control over user and course management.

4.2 Data Encryption and Protection

- Data Encryption: All sensitive data (e.g., passwords) is stored using bcrypt hashing.
- Communication between the client and server is encrypted using SSL/TLS.

Data Privacy

- Adhere to NSU's data privacy policies, ensuring compliance with regulatory standards like GDPR and HIPAA (if applicable).

UI Components

5.1 Login Page

- **Fields**:
  - Username
  - Password
- **Buttons**:
  - Login
- **Additional Elements**:
  - NSU logo and system name
  - "Forgot Password?" link
  - Role-based redirection (Student, Advisor, Admin)

5.2 Student Dashboard

Sections:

1. **Academic Progress**:
    - Progress bar for completed credits vs required credits
    - List of current courses
2. **Course Selection**:
    - Search bar for available courses
    - Course list with details (name, prerequisites, credits)
    - Checkbox to select courses
    - Submit button for advising
3. **Advising Status**:
    - List of submitted courses
    - Approval/rejection status and comments from the advisor

---

5.3 Advisor Dashboard

Sections:

1. **Student Advising**:
    - List of students with pending advising requests
    - Search bar to find students
2. **Course Approval**:
    - Student details (name, ID, degree program)
    - Selected courses with course details
    - Approve/Reject buttons with optional feedback field
3. **Reports**:
    - Generate reports for student performance or advising history

---

5.4 Admin Panel

Sections:

1. **User Management**:
    - Add, edit, and delete users (student, advisor, admin)
    - Role assignment dropdown
2. **Course Management**:
    - Add, edit, delete courses
    - Upload course catalog in bulk
3. **System Monitoring**:
    - Logs for activity (advising, registrations, approvals)

Performance and Scalability

6.1 Database Performance

Indexing: Indexes will be created on frequently queried fields such as `student_id`, `course_id`, and `registration_status` to improve query performance.

Caching: Use Redis for caching frequently accessed data like course listings and student profiles to reduce database load.

6.2 Load Balancing and Scalability

- **Horizontal Scaling**: The application will support horizontal scaling to handle high traffic during peak times, such as registration periods.
- **Balancing:** A load balancer will distribute incoming user requests evenly across multiple servers to ensure optimal performance.

Testing and Quality Assurance

7.1 Unit Testing

Each functional module (e.g., user registration, course selection) will undergo unit testing. Tools like **JUnit** or **Mocha** (for JavaScript) will be used for this purpose.

7.2 Integration Testing

The system's components will be tested in combination to ensure proper data flow and interaction between the user interface, business logic, and database.

7.3 Performance Testing

The system will undergo load testing using tools like JMeter to simulate concurrent users during peak registration times and ensure the system can handle the load.

Conclusion

This Software Design Specification (SDS) provides the foundation for the implementation of the **NSU Advising System**, ensuring the system's architecture, components, and security are designed to meet both the functional and non-functional requirements outlined in the SRS. By following this design, the system will be efficient, scalable, and secure, delivering an optimal user experience for students, advisors, and administrators.