**System-level Design**

**Introduction**

The TA Management Suite aims to provide a seamless experience for managing Teaching Assistants at North University. The architecture for this suite is designed to be modular and scalable, with each functional module being an independent entity.

**Modules**

**TA Application Service**

Responsibilities

* Manage the submission of TA applications.
* Store CVs and other relevant documents.

Features

* Form submission interface for TA applicants.
* Secure storage for CVs using Azure Blob Storage.

**Course Management Service**

Responsibilities

* Enable department staff to input courses that require TAs.

Features

* Dashboard for adding and editing course details.
* Data storage in Azure SQL Database.

**TA Matching Service**

Responsibilities

* Match TA applicants with courses based on skills and qualifications.

Features

* Algorithms to make preliminary recommendations.
* Automated routine tasks via Azure Logic Apps or Azure Functions.

**Decision-Making Service**

Responsibilities

* Support TA committee members in finalizing TA assignments.

Features

* Decision-making interface for committee members.
* Record-keeping of final TA assignments.

**TA Evaluation Service**

Responsibilities

* Collect and manage TA evaluations from instructors.

Features

* Evaluation submission forms for instructors.
* Store evaluations in Azure SQL Database.

**Notification Service**

Responsibilities

* Notify TA applicants and other stakeholders about application status and decisions.

Features

* Utilize Azure Notification Hubs for status updates and alerts.

**Data Storage and Management**

* Azure SQL Database will be used to store structured data such as application statuses, course details, and TA assessments.
* Azure Blob Storage will store unstructured data like CVs.

**User Authentication and Access Control**

* Azure Active Directory will handle user authentication.
* Role-based access control will define roles like TA applicant, department staff, TA committee members, and instructors.

**Automation and Communication**

* Routine tasks such as generating preliminary TA assignment recommendations can be automated using Azure Logic Apps or Azure Functions.
* Azure Notification Hubs will be used to manage notifications, thereby informing applicants and other stakeholders of relevant updates and decisions.

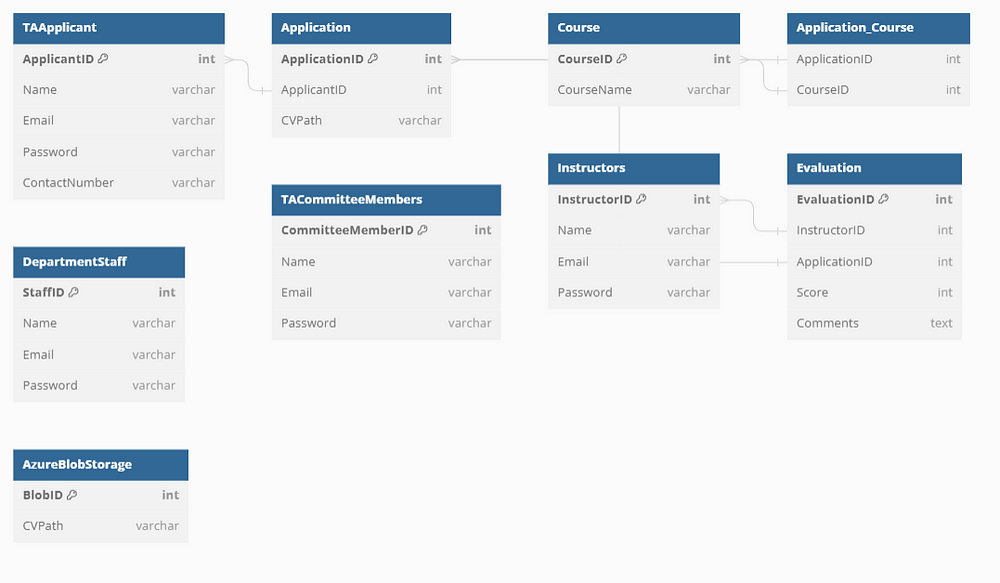
**Security and Compliance**

* All sensitive data will be protected through Azure services that adhere to industry standards for security and data protection.

**Conclusion**

This design aims to be flexible, robust, and scalable, facilitating easy development and deployment of each functional module while ensuring high availability and performance. The use of Azure services further ensures the system’s reliability and security.

>ER Diagram:



**Detailed Design: Course Management Subsystem**

**Overview:**

The Course Management Subsystem is tailored for department staff, enabling them to efficiently manage courses that require Teaching Assistants (TAs). This subsystem allows for the addition, modification, and deletion of courses, along with the ability to specify the qualifications and requirements for each course.

**Components:**

1. Admin Interface for Department Staff: A web interface where courses can be managed.
2. Azure SQL Database: Holds all the structured course data, including qualifications and requirements.

**Data Model:**

Course Entity in Azure SQL Database

* Course ID (Primary Key)
* Course Name
* Course Code
* Semester/Year
* Qualifications Required (e.g., specific skills, coursework, etc.)
* Number of TAs Required
* Status (Open/Closed)
* Timestamps (created\_at, updated\_at)

**Functionalities and Interactions:**

Adding New Courses

Admin Interface

* Provides a form to input all course details including name, code, semester, qualifications required, and the number of TAs required.

Python Backend

* Validates the form data.
* Inserts the new course record into the Azure SQL Database.

Updating Existing Courses

Admin Interface

* Provides an “Edit” option next to each course in the list of courses.
* Allows for modifying all fields.

Python Backend

* Validates the updated data.
* Updates the specific course record in the Azure SQL Database.

Deleting Courses

Admin Interface

* Provides a “Delete” option next to each course.

Python Backend

* Deletes the specific course record from the Azure SQL Database.

Viewing All Courses

Admin Interface

* Displays a list of all courses along with key details.

Python Backend

* Queries the Azure SQL Database to pull all course records for display.

Specifying Qualifications and Requirements

Admin Interface

* Allows for input of qualifications and requirements in a text area or predefined fields during the course creation or updating process.

Python Backend

* Saves this information in the Azure SQL Database under the respective course entity.

**API Endpoints:**

1. /courses/create - POST: Allows department staff to add new courses.
2. /courses - GET: Retrieves all courses.
3. /courses/<course\_id> - GET: Retrieves specific course details.
4. /courses/<course\_id>/update - PUT: Allows updating of course details.
5. /courses/<course\_id>/delete - DELETE: Allows deletion of a specific course.

**Security Measures:**

1. Authentication: Only authenticated department staff can access the admin interface.
2. Authorization: Role-based access control to ensure only authorized staff can make changes.
3. Data Validation: All form data is validated to ensure integrity.
4. Audit Logs: All changes are logged for auditing purposes.

**Monitoring:**

1. Azure Monitoring Tools: Utilized for logging, auditing, and real-time monitoring of operations within the subsystem.