**System-Level Design of TA Management Suite**

**1. Architectural Design**

**Three-Tier Architecture**

Presentation Layer (Frontend)

* Built using HTML, CSS, and JavaScript.
* Hosted on Azure App Service.

Business Logic Layer (Backend)

* Developed in Python.
* Hosted on Azure App Service.

Data Access Layer

* Azure SQL Database for structured data.
* Azure Blob Storage for unstructured data (e.g., CVs).

**2. Data Flow & User Interaction**

TA Applicants

* Interact with the frontend to submit applications and CVs, stored in Azure SQL Database and Azure Blob Storage respectively.

Department Staff

* Utilize an admin interface to input course details into Azure SQL Database and view TA applications.

TA Committee Members

* Access data from Azure SQL Database for decision-making through a specialized dashboard.

Instructors

* Submit evaluations via a web interface, stored in Azure SQL Database.

**3. System Components & Interactions**

User Management

* Azure Active Directory: Authentication and role-based access control.
* User Profiles: Stored in Azure SQL Database.

Data Storage

* Azure SQL Database: Handles applications, courses, evaluations, and user profiles.
* Azure Blob Storage: Responsible for storing CVs and other documents.

Backend Logic

* Application Submission & Management: Python backend to handle CRUD operations on TA applications.
* Matching Algorithm: Python-based algorithm to intelligently match TAs to courses. Data will be read from and written to Azure SQL Database.
* Notification System: Azure Notification Hubs integrated with the Python backend to handle notifications.

Performance Evaluation

* Instructors submit evaluations through the frontend, processed by the Python backend, and stored in Azure SQL Database.

Automated Workflows

* Azure Logic Apps or Azure Functions: Automate routine tasks like generating preliminary recommendations for TA assignments, running on a scheduled basis or triggered by specific events.

**4. Security & Compliance**

Data Encryption

* SSL/TLS for data in transit.
* Azure’s built-in encryption for data at rest.

Role-Based Access Control

* Implemented using Azure Active Directory.

Regular Audits

* Periodic checks leveraging Azure monitoring tools.

Backup and Recovery

* Azure SQL Database’s backup features and Azure Blob Storage redundancy.

**5. Scalability & Performance**

Load Balancers

* Azure’s built-in load balancing features for App Service.

Scaling

* Azure App Service allows for easy horizontal scaling.
* Azure SQL Database to be optimized for better performance and scalability.

**6. Monitoring & Maintenance**

Logging and Monitoring

* Utilize Azure Monitoring tools and log storage.

CI/CD Pipeline

* Automated deployment via Azure DevOps, sticking to Python for all backend development.

**User Authentication and Management Subsystem**

**Overview:**

The User Authentication and Management Subsystem is responsible for managing secure user access to the TA Management Suite. It utilizes Azure Active Directory for registration, login, authentication, and role-based access control (RBAC).

**Components:**

Azure Active Directory (Azure AD)

* Manages user identities and roles.
* Interacts with the Python backend and the frontend.

**Data Model:**

User Entity in Azure AD

* Username
* Password (hashed and salted)
* Role (TA Applicant, Department Staff, TA Committee Member, Instructor)

**Functionalities and Interactions:**

**User Registration**

Frontend

* Registration form capturing username, password, and role.

Backend

* Validation of input data.

Azure AD

* Creating a new user identity.
* Assigning the specified role to the user.

**User Login**

Frontend

* Login form capturing username and password.

Backend

* Forwarding credentials to Azure AD for validation.

Azure AD

* Authenticating credentials.
* Returning an authentication token upon successful authentication.

**Role-Based Access Control (RBAC)**

Frontend

* Different views and capabilities based on the user role.

Backend

* Checking the user’s role in Azure AD before allowing specific operations.

Azure AD

* Storing role information for each user.
* Validating roles during each session.

**Password Reset and Recovery**

Frontend

* Provides an option for password reset.

Backend

* Handles password reset requests by generating tokens.

Azure AD

* Validates the token and allows the password reset.

**Logout**

Frontend

* Provides a logout button.

Backend

* Invalidates the session.

Azure AD

* Optionally, revokes the authentication token.

**API Endpoints:**

1. /register - POST: Registers a new user.
2. /login - POST: Authenticates an existing user.
3. /logout - POST: Logs out the authenticated user.
4. /reset-password - POST: Handles password reset requests.
5. /change-password - POST: Allows authenticated users to change their password.

**Security Measures:**

1. Encryption — All data sent over the network is encrypted using SSL/TLS.
2. Rate Limiting — To prevent brute-force attacks.
3. Multi-factor Authentication (MFA) — Optionally implemented via Azure AD.

**Monitoring:**

1. Azure Monitoring Tools — Used for logging, auditing, and real-time monitoring of the authentication operations.

UML Diagrams:

>Class Diagram

A screenshot of a computer

Description automatically generated

this diagram, the classes represent different layers, roles, and components in the system:

* PresentationLayer, BusinessLogicLayer, and DataAccessLayer form the architectural backbone.
* TAApplicants, DepartmentStaff, TACommitteeMembers, and Instructors are roles that interact with the system.
* UserManagement, DataStorage, BackendLogic, and PerformanceEvaluation are specific functional components of the system.

>Use Case

A diagram of a software application

Description automatically generatedDiagram of a diagram of a department

Description automatically generatedA diagram of a company member

Description automatically generatedA diagram of a performance evaluation

Description automatically generated