**System-level Design**

**Introduction**

The goal of the TA Management Suite is to streamline the management of Teaching Assistants (TAs) at North University. To make this software effective, easy to manage, and straightforward to develop, we’re using a well-established architectural framework called the Model-View-Controller (MVC).

The MVC framework divides the application into three key components:

1. Model: This is the application’s data hub. It holds all the important data and sets the rules for how this data can be changed or manipulated.
2. View: This is the application’s front door. It’s the interface that users see and interact with.
3. Controller: This is the manager that handles the communication between the Model and the View.

Let’s break down how each component functions in our TA Management Suite.

**Models**

**What They Do**

Models are responsible for handling the data and the business logic of the application. They interact directly with the database to retrieve, update, and manipulate data.

**In Our Application**

1. TA Applicant Model: Manages data related to TA applications and CVs.
2. Department Staff Model: Manages course data and preliminary TA-to-course matching.
3. TA Committee Model: Manages decision-making for TA assignments.
4. Instructor Model: Manages TA performance evaluations.

**Views**

**What They Do**

Views take care of the user interface. They display data from the Model in a format that users can understand and interact with.

**In Our Application**

1. TA Applicant View: Shows forms for TA applications and lists available courses.
2. Department Staff View: Offers a dashboard to manage course details.
3. TA Committee View: Provides an interface to review and decide on TA assignments.
4. Instructor View: Gives a platform for submitting TA evaluations.

**Controllers**

**What They Do**

Controllers act as intermediaries between the Model and the View. When you interact with the View (like clicking a button), the Controller is responsible for handling this action.

**In Our Application**

1. TA Applicant Controller: Processes TA application form submissions.
2. Department Staff Controller: Handles course details updates.
3. TA Committee Controller: Manages the decision-making process for TA assignments.
4. Instructor Controller: Processes TA evaluations and updates the Model accordingly.

**Detailed Design: Matching and Recommendation Subsystem**

**Overview:**

The Matching and Recommendation Subsystem is responsible for generating automated, preliminary recommendations for TA-to-course assignments. It leverages a Python-based matching algorithm and Azure services for automation to simplify the decision-making process for the TA Committee.

**Components:**

1. Python-based Matching Algorithm: Custom-built algorithm to match TAs with appropriate courses.
2. Azure Logic Apps or Azure Functions: Automates the matching process and generates preliminary recommendations.

**Algorithm Criteria:**

1. TA’s expertise and qualifications
2. Previous TA experience (if any)
3. Course requirements and qualifications needed
4. Historical performance metrics of TAs (if available)

**Data Models:**

TA-to-Course Matching Data Structure

* TA ID
* Course ID
* Matching Score
* Preliminary Status (Recommended/Not Recommended)

Preliminary Recommendations Data Structure

* Committee Review ID (Primary Key)
* TA ID
* Course ID
* Matching Score
* Timestamps (created\_at, updated\_at)

**Functionalities and Interactions:**

Automatic Matching

Python Algorithm

* Pulls data from Azure SQL Database for TAs and courses.
* Computes matching scores based on the criteria.
* Stores the results in a suitable data structure for further processing.

Azure Logic Apps/Azure Functions

* Triggered at a set frequency or manually by department staff.
* Runs the Python-based matching algorithm.
* Stores preliminary recommendations in Azure SQL Database.

Preliminary Recommendations

Python Algorithm

* Uses the matching scores to generate a list of preliminary TA-to-course recommendations.

Azure Logic Apps/Azure Functions

* Updates the Preliminary Recommendations Data Structure in Azure SQL Database.
* Sends notifications to the TA Committee for review, using Azure Notification Hubs.

**API Endpoints:**

1. /matching/run - POST: Manually triggers the matching algorithm.
2. /recommendations - GET: Retrieves the list of preliminary recommendations.
3. /recommendations/<committee\_review\_id> - GET: Retrieves specific recommendation details.

**Security Measures:**

1. Authentication: The subsystem can only be triggered by authenticated and authorized department staff or automated secure triggers.
2. Authorization: Role-based access control to ensure only authorized personnel can trigger or view recommendations.
3. Data Validation: Data used for matching is validated for integrity.
4. Logging: All matchings and recommendations are logged for auditing purposes.

**Monitoring:**

1. Azure Monitoring Tools: Used for logging, auditing, and real-time monitoring of the subsystem’s activities.

By integrating Python-based algorithms with Azure Logic Apps or Azure Functions, the Matching and Recommendation Subsystem aims to streamline the TA assignment process, making it easier for the TA Committee to finalize decisions.

>State Chart

**States**

New Application: This is the initial state where the TA applicant starts the application process.

FillingForm: In this state, the applicant fills out the application form with personal and academic details.

CVUpload: After completing the form, the applicant uploads their CV.

CourseSelection: The applicant selects the courses they are qualified for and would like to assist with.

ApplicationReview: Before submitting, the applicant reviews the entire application.

Submitted: The application is successfully submitted and is now in the review process.

StatusCheck: This is where the applicant can check the current status of their application. The sub-states here are:

* Pending: The default sub-state, where the application is in the review process but no decision has been made yet.
* Accepted: The application has been approved.
* Rejected: The application has been denied.

**Transitions**

* Initiate Application: Transitions from the start point to the NewApplication state.
* Edit: Allows the applicant to edit the form while in the FillingForm state.
* Submit: Transitions from FillingForm to CVUpload.
* Next: Transitions from CVUpload to CourseSelection and from CourseSelection to ApplicationReview.
* Confirm: Confirms the application, transitioning from ApplicationReview to Submitted.
* Edit Application: Goes back to the FillingForm state for editing.
* View Status: Transitions from Submitted to StatusCheck for checking the application status.
* Application Approved: Transitions from Pending to Accepted within StatusCheck.
* Application Denied: Transitions from Pending to Rejected within StatusCheck.
* Exit: This transition occurs from Submitted and StatusCheck states to the end state, indicating the end of interaction with the system.

A diagram of a software process

Description automatically generated