Intro: HokieGuide is an AI-powered chatbot designed to simplify access to academic and financial information for Virginia Tech students. It integrates with existing Virginia Tech systems, such as Hokie Spa and Hokie Scheduler, providing a centralized platform for retrieving course history, graduation requirements, tuition balance, and more using a student's PID. With natural language processing (NLP) capabilities, HokieGuide allows students to interact using simple queries, making it an efficient solution for managing their academic and financial needs.

Fully dressed use cases:

Use Case UC1: Authenticate Student using PID

Primary Actor: Students

Stakeholders and interests:

- Students: Would want a secure and efficient way to access the resources.
- University Administration: Needs to provide secure access and maintain sensitive information.
- IT Department: Responsible for implementing and maintaining the authentication system, ensuring it is well protected and user-friendly

Preconditions:

- The student must have an active university-provided PID and input the correct username and password.
- The student must have a secure, protected internet connection.

Success guarantee:

- If the student enters a valid PID username and password and can successfully validate the Duo Push 2-Factor Authentication, they will be allowed access to the website.

Main success scenario:

- Student navigates to the login page and inputs their PID username and password.
- The system may ask for a 2-Factor Authentication, depending on the last time the student logged in.
- The system will verify the credentials against the database, and if the student's credentials are valid, they will be redirected to the website dashboard.

Extensions:

- **Invalid Credentials:**

    - If the student enters an incorrect username or password, the system displays an error message and prompts the student to try again.
- **Forgot Password:**
    - The student clicks on "Forgot Password" and follows the recovery process to reset their password via email verification.
- **Account Lockout:**
    - After multiple failed login attempts, the account is temporarily locked for security reasons, and the student is informed.

Special requirements:

- The system must comply with data protection regulations (e.g., FERPA, GDPR).
- Passwords must meet complexity requirements (minimum length, special characters, etc.).
- The system should have two-factor authentication (2FA) for enhanced security.

Technology and data variation list:

- **Authentication Methods:**
    - Username/password (primary method)
    - Multi-factor authentication (SMS, email, authentication app)
- **User Interfaces:**
    - Web portal
    - Mobile app for authentication
- **Database Variations:**
    - Centralized database for user credentials
    - Integration with other university systems (e.g., course registration, library access)

Use Case UC2: Check Course Requirements

Primary Actor: Students

Stakeholders and Interests:

- **Students:** Want to easily understand what courses they need to complete their degree requirements.
- **Academic Advisors:** Need tools to assist students in planning their academic paths effectively.
- **University Administration:** Aims to ensure that students are meeting educational standards and requirements.

- **IT Department:** Responsible for maintaining the system, ensuring it is user-friendly and accurate.

Preconditions:

- The student must be logged into their university account.
- The system must have access to the student's academic records (completed courses, grades, etc.).

Sucess Guarantee:

- The student can view a clear and accurate summary of their course requirements and the courses they have already completed.

Main Success Scenario:

- The student logs into their university account.
- The student navigates to the "Course Requirement Checker" feature.
- The system retrieves the student's academic records and program requirements.
- The student views a list of required courses, completed courses, and any outstanding requirements.
- The student can click on each requirement for additional details (prerequisites, recommended semesters, etc.).
- The system provides suggestions for upcoming courses that meet the requirements.
- The student logs out after reviewing their requirements.

Extensions:

- **No Requirements Met:**
    - If the student has not completed any requirements, the system provides guidance on how to get started.
- **Advising Session Request:**
    - The student can request an advising session directly through the checker for personalized assistance.
- **Course Prerequisites Not Met:**
    - The system alerts the student if they attempt to enroll in a course without meeting prerequisites.
- **Curriculum Changes:**
    - If there have been changes to the curriculum, the system informs the student of updates to their requirements.

Special Requirements:

- The system must be updated regularly to reflect any changes in course offerings or degree requirements.
- It should be accessible on multiple devices (desktop, tablet, mobile).
- The system must comply with privacy regulations concerning student records.

Technology and Data Variation List:

- **User Interface:**
  - Web portal and mobile app for accessing the Course Requirement Checker.
- **Data Sources:**
  - Integration with the university's student information system for real-time updates on course availability and student records.
- **Notification System:**
  - Alerts and notifications for students regarding upcoming courses and requirements.

Use Case UC3: View Course Enrollement Status

Primary Actor: Students

Stakeholders and Interests:

- **Students:** Want to easily track their enrollment status for current and future courses.
- **Academic Advisors:** Need to assist students in managing their course loads and resolving enrollment issues.
- **University Administration:** Aims to monitor course capacities and student enrollment trends.
- **IT Department:** Responsible for maintaining the system and ensuring its accuracy and reliability.

Preconditions:

- The student must be logged into their university account.
- The system must have access to the student’s course enrollment records.

Sucess Guarantee:

- The student can view a clear and accurate summary of their enrollment status for current and upcoming courses, including any pending approvals.

Main Sucess Scenario:

- The student logs into their university account.
- The student navigates to the "Course Enrollment Status" feature.
- The system retrieves the student's current and upcoming course enrollments.
- The student views the list of enrolled courses, including:
    - Course titles
    - Status (e.g., enrolled, waitlisted, pending approval)
    - Enrollment dates
- The student can click on individual courses for additional details (instructor information, syllabus, etc.).
- The system provides alerts for any enrollment issues (e.g., waitlist status, prerequisites not met).
- The student logs out after reviewing their enrollment status.

Extensions:

- **Enrollment Pending:**
    - If a course enrollment is pending, the system informs the student of the reasons (e.g., instructor approval needed) and any actions required.
- **Waitlisted Courses:**
    - The system displays information on waitlisted courses and the likelihood of enrollment based on current capacity.
- **Enrollment Changes:**
    - If there are changes to the student's enrollment (e.g., course cancellation), the system sends an alert and updates the enrollment status.
- **Drop/Add Period:**
    - During the drop/add period, the system allows students to modify their enrollments easily and provides confirmation of changes.

Special Requirements:

- The system must be updated in real-time to reflect any changes in course enrollment statuses.
- It should provide clear notifications to students about changes in their enrollment.
- The system must comply with privacy regulations concerning student records.

Technology and Data Variation List:

- **User Interface:**

    - Web portal and mobile app for accessing course enrollment status.
- **Data Sources:**
    - Integration with the university's student information system to ensure accurate and up-to-date enrollment data.
- **Notification System:**
    - Alerts and notifications for changes in enrollment status, important deadlines, and potential issues.

Use Case UC4: Analyze GPA Distribution Data

Primary Actor: Students

Stakeholders and Interests:

- **Students:** Seek information on GPA distributions to make informed decisions about course selection and academic performance.
- **Academic Advisors:** Need access to GPA distribution data to guide students in their academic planning.
- **University Administration:** Aims to analyze trends in student performance and improve academic offerings.
- **Faculty:** Interested in understanding grading patterns within their courses and programs.

Preconditions:

- The student must be logged into their university account.
- The system must have access to aggregated GPA data for courses and programs.

Success Guarantee:

- The student can view comprehensive and accurate GPA distribution data, including averages, medians, and grade breakdowns for courses or programs.

Main Sucess Scenario:

- The student logs into their university account.
- The student navigates to the "GPA Distribution Data" feature.
- The system retrieves the aggregated GPA data for selected courses or programs.
- The student selects specific courses or a program to view distribution data.
- The system displays:
    - A graphical representation (e.g., histogram, pie chart) of GPA distribution.
    - Key statistics (mean GPA, median GPA, standard deviation).

    - Percentage of students receiving each grade (A, B, C, etc.).
- The student can filter data by semester, department, or course level.
- The student logs out after reviewing the data.

Extensions:

- **Data Export:**
    - The student can download the GPA distribution data in various formats (CSV, PDF) for further analysis.
- **Comparison Tool:**
    - The system allows students to compare GPA distributions across different courses or programs.
- **Historical Data:**
    - The student can view historical GPA distribution trends over multiple semesters.
- **Feedback Mechanism:**
    - Students can provide feedback on the usefulness of the GPA distribution data and suggest improvements.

Special Requirements:

- The system must ensure data privacy and comply with regulations regarding student information.
- Data must be aggregated to avoid revealing individual student GPAs.
- The user interface should be intuitive and accessible across multiple devices.

Technology and Data Variation List:

- **User Interface:**
    - Web portal and mobile app for accessing GPA distribution data.
- **Data Sources:**
    - Integration with the university's academic records system to access up-to-date GPA data.
- **Visualization Tools:**
    - Utilization of data visualization libraries to create interactive charts and graphs.

Use Case UC5: Retrieve Personalized Class Schedule

Primary Actor:

Stakeholders and interests:

- **Student**: Wants quick and easy access to their class schedule to manage time effectively.
- **Academic Advisor**: Can view the student's schedule to assist with academic planning.
- **University IT Department**: Ensures that the scheduling system is accessible, accurate, and reliable.

Preconditions:

- Student must be logged in and authenticated.
- Student's schedule data must be available and up-to-date in the university's database.

Success guarantee:

- The student successfully views their current semester's class schedule, including course names, times, and locations.

Main success scenario:

- Student navigates to the "Class Schedule" section in the university portal or app.
- System retrieves the current semester's schedule based on the student's ID.
- Schedule information is displayed, including:
    - Course names (e.g., "CS101: Introduction to Computer Science")
    - Course times (e.g., "Mon/Wed/Fri 9:00 AM - 10:15 AM")
    - Course locations (e.g., "Room 203, Engineering Building")
- Student can see the full schedule, navigate to specific days or weeks, and print or download a copy.

Extensions:

- **E1**: If the schedule data cannot be retrieved (e.g., database connection failure), the system displays an error message and provides options for contacting support.
- **E2**: If the student is not enrolled in any classes for the current semester, the system displays a message: "No classes found for the current semester."

Special requirements:

- The schedule display must be responsive and accessible on both web and mobile platforms.

Technology and data variation list:

- Data Source: Schedule data comes from the university's student information system.
- Technologies: Web-based platform (React frontend), backend (Flask/Python), database (MySQL).
- Authentication Method: OAuth2.0 for secure access to student data.

Use Case UC6: Access Professor Office Hours & Contact Info

Primary Actor: Student

Stakeholders and interests:

- **Student**: Wants to know the professor's office hours and contact information for academic help and consultations.
- **Professors**: Want to ensure students have accurate information to connect during office hours.
- **University IT Department**: Ensures that contact information is accurate and up-to-date.

Preconditions:

- Student is logged in and authenticated.
- Student is enrolled in at least one course for the current semester.
- Professor's contact information and office hours are maintained and up-to-date.

Success guarantee:

- The student successfully views the office hours and contact details of all professors associated with their current semester courses.

Main Succes Scenario:

- Student navigates to the "Professor Office Hours" section in the university portal or app.
- System retrieves a list of professors based on the student's class schedule.

- System fetches and displays office hours and contact information (email, phone number) for each professor.
- Student can view additional information such as office location, scheduled office hours, and any available booking links.

Extensions:

- **E1**: If the office hours or contact details are not available, the system displays a message: “Office hours information not available at this time.”

Special requirements:

- Information display must be responsive and user-friendly on both web and mobile platforms.

Technology and Data Variation List:

- Data Source: Professors’ information comes from the university’s human resources or academic database.
- Technologies: Web-based platform, backend system using REST APIs.
- Authentication Method: OAuth2.0 for secure access to student and faculty data.

Use Case UC7: Get AI-Powered Course Recommendations

Primary Actor:

- Student

Stakeholders and Interests:

- **Student**: Seeks personalized recommendations to optimize academic experience based on their interests and performance.
- **Academic Advisor**: Uses recommendations to guide students in course selection.
- **University IT Department**: Ensures the recommendation system is accurate and updated with the latest course offerings.

Preconditions:

- Student is logged in and authenticated.
- Student’s academic history and current enrollment data are available.
- AI model is trained on academic performance, course content, and student interests.

Success Guarantee:

- Student receives a list of elective courses or minors that align with their academic performance, interests, and major.

Main Success Scenario:

1. Student navigates to the "Course Recommendations" section in the university portal.
2. Student inputs their interests or selects areas they want to explore further.
3. AI system analyzes the student's academic history, performance, and inputted interests.
4. System generates a list of recommended courses or minors.
5. Student can view course details, prerequisites, and add courses to their academic plan.

Extensions:

- **E1**: If the AI model cannot provide recommendations (e.g., insufficient data), the system displays a message: "No recommendations available at this time."

Special Requirements:

- AI model must be trained and updated regularly with new course offerings and academic trends.
- Recommendations must consider prerequisites and current course load.

Technology and Data Variation List:

- Data Source: Academic history and course data come from the student information system.
- Technologies: Machine learning model (e.g., TensorFlow, scikit-learn), web-based platform.
- Authentication Method: OAuth2.0 for secure access to student data.

Use Case UC8: View Financial Aid Balance and Status

Primary Actor:

- Student

Stakeholders and Interests:

- **Student**: Wants to know their financial aid balance and usage to plan for tuition payments and expenses.
- **Financial Aid Office**: Manages and updates student financial aid records.
- **University IT Department**: Ensures financial aid data is accurate and updated in real-time.

Preconditions:

- Student is logged in and authenticated.
- Student's financial aid records are available and up-to-date in the financial aid system.

Success Guarantee:

- The student successfully views their current financial aid status, balance, and usage.

Main Success Scenario:

1. Student navigates to the "Financial Aid" section in the university portal or app.
2. System retrieves the student's financial aid status and balance.
3. The following information is displayed:
    a. Total financial aid awarded for the current academic year.
    b. Amount used so far and the remaining balance.
    c. Breakdown by grants, loans, and scholarships.
4. Student can view detailed information for each category and contact the financial aid office if needed.

Extensions:

- **E1**: If the financial aid data cannot be retrieved, the system displays an error message: "Financial aid information not available at this time."

Special Requirements:

- Information must be displayed securely, ensuring that sensitive financial data is protected.

Technology and Data Variation List:

- Data Source: Financial aid information comes from the university's financial aid database.
- Technologies: Web-based platform, backend system using secure API connections.
- Authentication Method: OAuth2.0 with multi-factor authentication for secure access to financial data.

Use Case UC9: Check Tuition Balance

- **Primary Actor**: Student
- **Stakeholders and Interests**:
    - **Student**: Wants to view their current tuition balance, including payment history and upcoming payment deadlines.
    - **University Administration**: Ensures students have clear visibility of financial obligations.
- **Preconditions**:
    - The student must be logged into the system.
    - The student must have an existing financial account within the university system.
- **Postconditions**:
    - The student is informed of the current tuition balance.
    - The student can take further actions such as viewing past payments or upcoming due dates.
- **Main Success Scenario**:
    i. The student navigates to the "Finance" section of the web application.
    ii. The system retrieves the student's financial information from the university's database.
    iii. The tuition balance and detailed breakdown are displayed.
    iv. The student can view the balance, past payments, and due dates.
- **Extensions**:
    - 3a. **Data retrieval failure**: If the system cannot retrieve financial data, an error message is displayed, and the student is prompted to retry.
- **Special Requirements**:
    - Response time for retrieving the balance should be under 3 seconds.
    - Data must be presented in an easy-to-read format.
- **Frequency of Use**: Likely multiple times per semester.

Use Case UC10: Match Scholarships to Student Profile

- **Primary Actor**: Student
- **Stakeholders and Interests**:
    - **Student**: Wants to find relevant scholarships they are eligible for.
    - **Financial Aid Office**: Wants to assist students in securing scholarships.
- **Preconditions**:
    - The student must be logged into the system.
    - The student's profile information must be up-to-date (e.g., GPA, program of study).
- **Postconditions**:
    - A list of scholarships matching the student's profile is provided.
- **Main Success Scenario**:
    1. The student navigates to the "Scholarship Matching" tool.
    2. The system prompts the student to confirm profile details.
    3. The student confirms or updates their profile information.
    4. The system matches available scholarships based on the student's profile.
    5. A list of matching scholarships is displayed, including application deadlines and eligibility criteria.
- **Extensions**:
    - 4a. **No matching scholarships found**: Display a message indicating no matches and suggest options to broaden the criteria.
    - 3a. **Incomplete profile information**: Prompt the student to complete their profile.
- **Special Requirements**:
    - The system should ensure data security as it involves sensitive personal information.
- **Frequency of Use**: Typically a few times per semester, particularly during application periods.

Use Case UC11: Receive Drop/Withdraw Deadline Notifications

- **Primary Actor**: Student
- **Stakeholders and Interests**:
    - **Student**: Wants to be informed of important deadlines to drop or withdraw from classes without penalties.
    - **University Administration**: Wants to ensure students are aware of deadlines to avoid administrative issues.
- **Preconditions**:

    - The student must be logged into the system.
    - The system must have access to the student's current course schedule.
- **Postconditions**:
    - The student is notified of upcoming drop or withdrawal deadlines.
- **Main Success Scenario**:
    6. The system checks the student's current schedule and identifies upcoming deadlines.
    7. A notification is sent to the student through their preferred communication channel (e.g., email, SMS, in-app notification).
    8. The student views the notification and takes appropriate action if desired.
- **Extensions**:
    - 2a. **Invalid communication channel**: If the preferred communication channel is not available, the system defaults to email notification.
- **Special Requirements**:
    - Notifications should be sent at least one week before the deadline.
- **Frequency of Use**: Occurs a few times per semester.

Use Case UC12: Sync Calendar with Academic Events

- **Primary Actor**: Student
- **Stakeholders and Interests**:
    - **Student**: Wants to have a unified view of academic events and deadlines.
    - **University Administration**: Wants students to stay on track with their schedules.
- **Preconditions**:
    - The student must be logged into the system.
    - The student has a compatible calendar application (e.g., Google Calendar, Outlook).
- **Postconditions**:
    - The student's academic calendar is updated with relevant university events.
- **Main Success Scenario**:
    9. The student navigates to the "Calendar Sync" settings.
    10. The system prompts the student to connect to their preferred calendar application.
    11. The student grants permission for access.
    12. The system synchronizes relevant academic events (e.g., deadlines, class schedules) with the student's personal calendar.
- **Extensions**:

    - 3a. **Permission denied**: If the student does not grant permission, display a message explaining that sync is not possible.
    - 4a. **Calendar sync failure**: If synchronization fails, provide an error message and troubleshooting steps.
- **Special Requirements**:
    - Calendar events should be updated automatically whenever changes occur in the university schedule.
- **Frequency of Use**: Typically used at the start of each semester, with periodic updates as changes occur.

Use Case UC13: Estimate Course Difficulty with AI
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want accurate difficulty predictions to make informed course selection decisions.
- **Advisors**: Need a reliable tool to guide students during registration.
- **Course Instructors**: Interested in how their courses are being evaluated in terms of difficulty.

Preconditions:

- The student has logged into the system.
- The system has access to the historical performance data of students in various courses.

Success Guarantee:
The system provides an accurate estimation of course difficulty based on previous student performance and various course characteristics.
Main Success Scenario:

1. The student selects the courses they are interested in.
2. The system processes the historical data and other factors.
3. The system displays a predicted difficulty score or description for each course.
4. The student can view course difficulty estimates and make informed registration decisions.

Extensions:

- If no data is available for a course, the system informs the user with a message explaining that an estimation cannot be made.

Special Requirements:

- Real-time processing of large datasets without significant delays.

Technology and Data Variation List:

- Access to data sources such as grade distributions, student feedback, and instructor difficulty ratings.

Use Case UC14: Receive Class Registration Reminders
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want timely reminders to ensure they don't miss registration deadlines.
- **University Administration**: Aims to ensure that all students register for classes on time to avoid system overloads.

Preconditions:

- The student has an active enrollment status and is within the registration period.

Success Guarantee:
The student receives timely reminders about registration deadlines based on their individual academic schedule.
Main Success Scenario:

1. The system identifies the student's registration window.
2. The system sends reminders through email or app notifications.
3. The student receives the reminder and completes their registration on time.

Extensions:

- If the student misses the registration deadline, the system notifies them of the next available registration period.

Special Requirements:

- The system must account for varying registration times for different groups (e.g., by academic year or major).

Technology and Data Variation List:

- Integration with the university's registration system for accurate schedule data.

Use Case UC15: Track Upcoming Assingments
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Need an organized view of their upcoming assignments to manage their time effectively.
- **Instructors**: Benefit from having students who are well-prepared and submit assignments on time.

Preconditions:

- The student has enrolled in courses and the course syllabi are available to the system.

Success Guarantee:
The student has a clear view of their upcoming assignments and deadlines.
Main Success Scenario:

1. The system retrieves the assignment details from the course management platform.
2. The student logs in and views a list of upcoming assignments.
3. The system highlights the nearest deadlines and provides notifications.

Extensions:

- The system can display assignment descriptions, grading rubrics, and links to resources if available.

Special Requirements:

- The system must be able to sync with multiple course platforms (e.g., Canvas, Blackboard).

Technology and Data Variation List:

- Integration with course management systems and calendar apps.

Use Case UC16: Retrieve Exam Schedule
Primary Actor: Students

Stakeholders and Interests:

- **Students**: Need accurate exam schedules to avoid conflicts and ensure adequate preparation time.
- **Instructors**: Want their exam times correctly displayed to students.
- **University Administration**: Needs a smooth process for students to access exam schedules.

Preconditions:

- The university has published the exam schedule.
- The student is enrolled in courses that have scheduled exams.

Success Guarantee:
The student can retrieve their personalized exam schedule accurately and on time.
Main Success Scenario:

1. The student logs into the system.
2. The system retrieves the student's exam schedule based on their course enrollment.
3. The student views their personalized exam schedule.

Extensions:

- The system sends reminders about upcoming exams as the exam date approaches.

Special Requirements:

- Real-time updates in case of exam schedule changes.

Technology and Data Variation List:

- Integration with the university's scheduling system to ensure exam time accuracy.

Use Case UC17: Read Professer Reviews and Ratings
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want reliable reviews to help choose professors based on teaching style, course difficulty, and feedback.
- **Professors**: May use the feedback for personal development and course improvement.

- **University Administration**: Interested in overall course and faculty performance to maintain academic standards.

Preconditions:

- The system has access to student reviews and ratings from various sources.

Success Guarantee:
The system provides an aggregated summary of professor reviews and ratings in an easy-to-read format for students.
Main Success Scenario:

1. The student searches for a professor by name or course.
2. The system gathers reviews from multiple platforms (e.g., university review boards, third-party sites).
3. The system displays an aggregated score, along with pros/cons, and written reviews.
4. The student uses this information to make an informed decision when choosing a professor.

Extensions:

- If there are no reviews available for a professor, the system notifies the student that reviews are unavailable.

Special Requirements:

- The system should be able to process and aggregate data from multiple sources and maintain the anonymity of reviewers where required.

Technology and Data Variation List:

- Integration with multiple review platforms or databases to provide comprehensive data.

Use Case UC18: Trac Real-Time Waitlist Position
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want to track their real-time position on course waitlists to make informed registration decisions.
- **Advisors**: Use waitlist data to help students adjust their schedules as needed.

- **University Administration**: Needs to manage enrollment limits effectively.

Preconditions:

- The student is on a course waitlist and has logged into the system.

Success Guarantee:
The system displays the student's real-time waitlist position and updates it dynamically as positions change.
Main Success Scenario:

1. The student logs into the registration system.
2. The system retrieves the student's current waitlist position for a specific course.
3. The system displays the real-time position on the waitlist, along with estimated movement.
4. The student can decide whether to wait or register for another course.

Extensions:

- If the student is moved off the waitlist into the course, the system sends a notification.
- If the course reaches capacity with no further movement, the system alerts the student.

Special Requirements:

- The system must update in real-time as students are added or removed from the waitlist.

Technology and Data Variation List:

- Integration with the university's registration and course management system to ensure accurate and up-to-date information.

Use Case UC19: Suggest Study Groups Based on Courses
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want to connect with peers for collaborative study based on shared courses, schedules, or interests.
- **University Administration**: Seeks to promote student engagement and collaboration.

Preconditions:

- The student has logged into the system and enrolled in courses.

Success Guarantee:
The system successfully suggests study groups tailored to the student's courses, availability, and preferences.
Main Success Scenario:

1. The student logs in and selects a course for which they want a study group.
2. The system analyzes the student's schedule, learning preferences, and available peers.
3. The system suggests a list of potential study groups or individual partners.
4. The student joins a study group or connects with study partners.

Extensions:

- If no matching study group is found, the system offers to create a new group and invites peers with similar interests.

Special Requirements:

- AI algorithms to match students based on course load, availability, and learning style.

Technology and Data Variation List:

- Integration with the student course management system and peer matching algorithms.

Use Case UC20: Schedule Appointment with Advisor
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want a quick and easy way to schedule appointments with academic advisors.
- **Advisors**: Need a clear view of their availability and schedule to efficiently manage student appointments.
- **University Administration**: Interested in ensuring that all students receive adequate advising support.

Preconditions:

- The student is eligible for advising and the advisor's availability is up-to-date in the system.

Success Guarantee:
The student successfully schedules an appointment with their academic advisor at a suitable time.
Main Success Scenario:

1. The student selects an advisor from the system.
2. The system shows the advisor's available appointment times.
3. The student selects a suitable time and confirms the appointment.
4. The system sends a confirmation and reminder to both the student and advisor.

Extensions:

- If no suitable time is available, the system suggests alternative advisors or offers a waitlist for cancellations.

Special Requirements:

- Integration with the advisor's calendar system for real-time availability.

Technology and Data Variation List:

- Integration with scheduling software and email notifications.

Use Case UC21: Get Directions to Classroom Locations
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Need accurate directions to find their classrooms quickly, especially at the beginning of a term or on a large campus.
- **University Administration**: Wants to ensure students can navigate the campus easily without disruption to classes.

Preconditions:

- The student has logged into the system, and the classroom location data is available.

Success Guarantee:
The system provides accurate and clear directions to the student's selected classroom.
Main Success Scenario:

1. The student selects the classroom they need directions to.

2. The system retrieves the classroom's location and current campus map.
3. The system provides step-by-step directions (e.g., on foot, accessible routes).
4. The student follows the directions to the classroom.

Extensions:

- The system offers alternate routes in case of construction or other obstacles.

Special Requirements:

- Real-time updates for changes in building access, closed areas, or detours.

Technology and Data Variation List:

- Integration with campus maps and GPS-based systems.

Use Case UC22: Track Scholarship Application Status
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want to track the status of their scholarship applications to stay informed and organized.
- **University Administration**: Needs a reliable way to manage and communicate scholarship application statuses.
- **Scholarship Committees**: Require a system to easily review applications and update statuses.

Preconditions:

- The student has applied for scholarships and is logged into the system.

Success Guarantee:
The student can track the status of their scholarship applications in real time.
Main Success Scenario:

1. The student logs in and views their scholarship dashboard.
2. The system displays a list of all active scholarship applications and their current status (e.g., submitted, under review, decision pending).
3. The student receives updates on any changes in application status.

Extensions:

- The system provides detailed feedback or instructions if the student is required to submit additional information.

Special Requirements:

- Must handle large volumes of applications without performance degradation.

Technology and Data Variation List:

- Integration with the scholarship application platform and real-time status updates.

Use Case UC23: Predict Final Grades Using AI
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want to estimate their final grades to adjust study habits and manage time effectively.
- **Advisors**: May use grade predictions to provide academic guidance.
- **Instructors**: Can see how well students are expected to perform in their courses.

Preconditions:

- The student is enrolled in courses and performance data (e.g., assignments, exams) is available.

Success Guarantee:
The system provides an accurate prediction of the student's final grades based on their current performance.
Main Success Scenario:

1. The student selects a course for which they want a grade prediction.
2. The system analyzes the student's performance on completed assignments and exams.
3. The system predicts the final grade based on current performance and course grading criteria.
4. The student views the predicted grade and adjusts study plans accordingly.

Extensions:

- If there is insufficient data, the system provides an estimated range instead of a precise prediction.

Special Requirements:

- The prediction algorithm must be accurate and adaptable to different grading systems.

Technology and Data Variation List:

- Integration with course management platforms and access to grading rubrics.

Use Case UC24: Recieve AI-Powered Career Guidance
Primary Actor: Students
Stakeholders and Interests:

- **Students**: Want personalized career guidance based on their skills, interests, and academic performance.
- **Career Advisors**: Use the system to help guide students toward relevant career paths.
- **University Administration**: Interested in tracking the effectiveness of career services for students.

Preconditions:

- The student has provided data about their academic history, interests, and career goals.

Success Guarantee:
The system suggests suitable career paths, job opportunities, or internships based on the student's profile.
Main Success Scenario:

1. The student inputs their academic history, skills, and career preferences.
2. The system analyzes the input data using AI algorithms.
3. The system suggests career paths, potential job openings, or internships.
4. The student reviews the suggestions and explores further opportunities.

Extensions:

- If the student is unsure about their preferences, the system offers a career questionnaire to narrow down options.

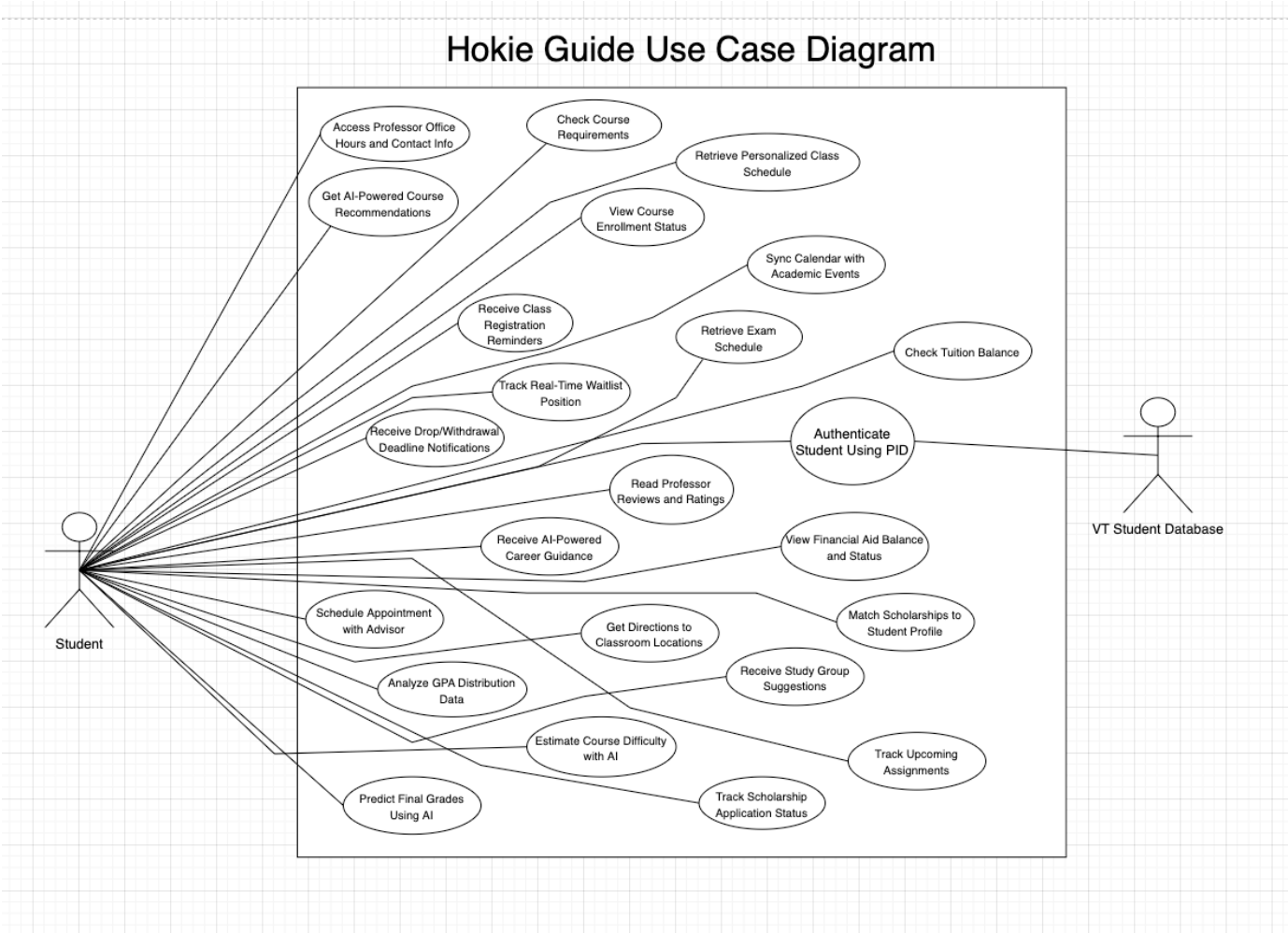
Special Requirements:

- AI algorithms must process diverse student data and provide relevant, up-to-date suggestions.

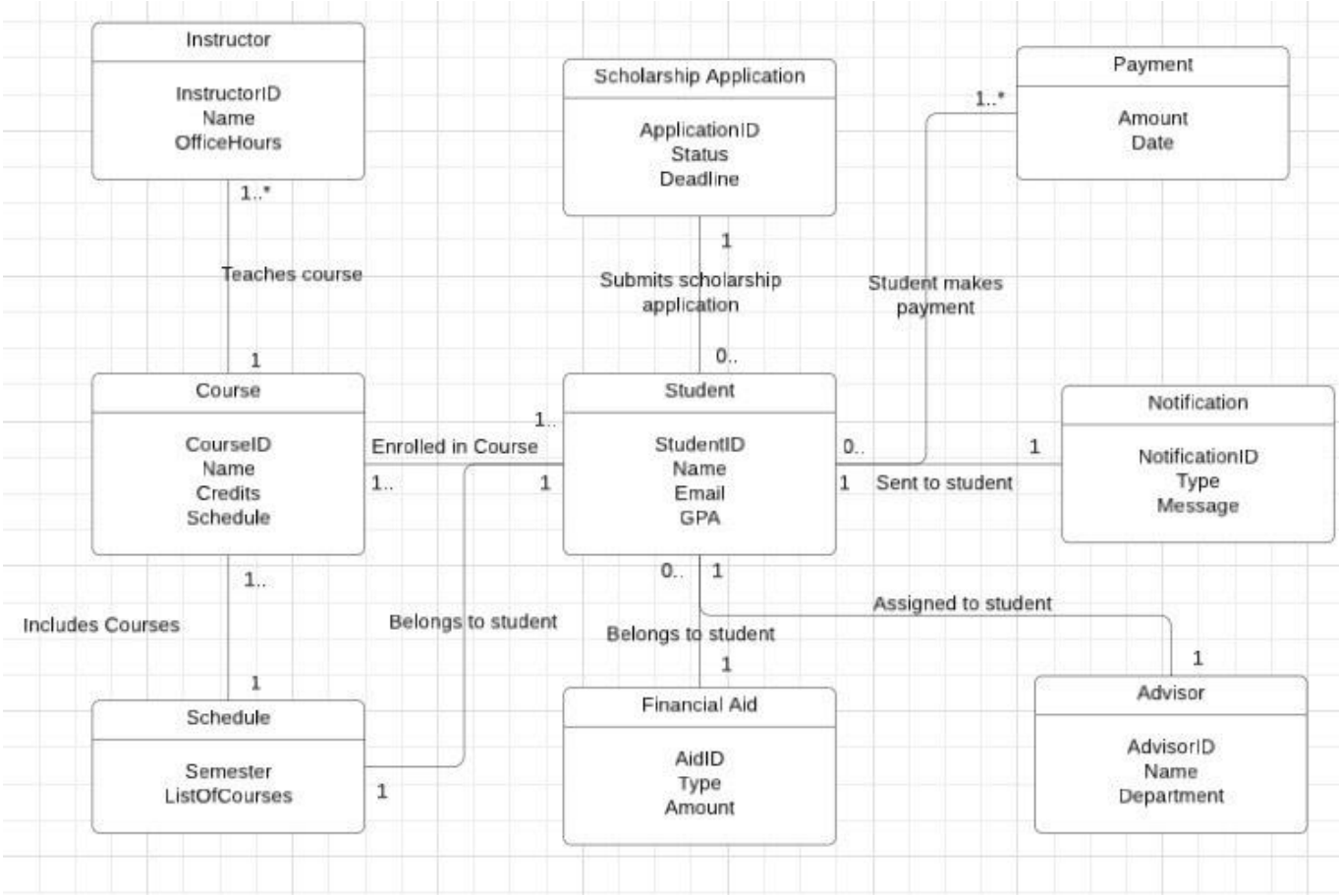
Technology and Data Variation List:

- Integration with job databases, career services platforms, and AI-based profiling algorithms.

Use Case Diagrams:



Conceptual Class Diagrams:



Supplementary Specifications:

1. Data Privacy and Compliance

HokieGuide will adhere to FERPA (Family Educational Rights and Privacy Act) compliance, ensuring that all student information is handled according to the regulations governing educational data. The system will not store or share any personally identifiable information (PII) without explicit user consent.

2. System Availability and Reliability

HokieGuide will maintain a 99.9% uptime, ensuring continuous availability for students during critical periods such as course registration, midterms, and finals. Downtime for maintenance or updates will be scheduled during off-peak hours to minimize disruption.

3. Performance

All queries made to HokieGuide should have a response time of less than 2 seconds for academic and financial information requests. This performance standard will be met even under high traffic, such as when hundreds of students access the system simultaneously during registration periods.

4. Accessibility and Usability

HokieGuide will comply with WCAG 2.1 standards to ensure accessibility for all users, including those with disabilities. The interface will support screen readers and include keyboard navigation to enhance usability.

5. Scalability and Flexibility

HokieGuide's infrastructure will be designed to scale dynamically based on user demand. During high-traffic periods, the system will automatically provision additional resources to maintain performance and availability, ensuring a seamless experience for students.

6. Logging and Monitoring

The system will have robust logging and monitoring mechanisms in place to track system performance, identify potential issues, and log significant user interactions while ensuring that no sensitive student information is stored in the logs. Monitoring dashboards will be accessible to developers and administrators for proactive management and troubleshooting.