**SDD: Internship Application System**

1. Introduction

The Internship Application System connects students with internship opportunities posted by company HR representatives. It helps students find and apply for internships while enabling HR professionals to manage postings and review applications.

2. System Overview

The system consists of these main modules:

* User Management (Authentication/Profiles).
* Company/HR Management.
* Internship Postings.
* Application Processing.
* Notification System.

3. Module Details

3.1 User Management

* **Main Job**: Handles all user authentication and profile management.
* **Data**: User credentials, roles (Student/HR/Admin), and profile information.
* **Components**: User, Role, Student, Hr, and Admin entities.

3.2 Company/HR Management

* **Main Job**: Manages company information and HR user accounts.
* **Data**: Company details, HR profiles and their company associations.
* **Components**: Company and Hr entities with related services.

3.3 Internship Postings

* **Main Job**: Creates and manages internship opportunities.
* **Data**: Internship details (title, description, requirements, status).
* **Components**: Internship entity with status tracking.

3.4 Application Processing

* **Main Job**: Handles student applications and status updates.
* **Data**: Application records, status changes, timestamps.
* **Components**: Application entity with status lifecycle.

3.5 Notification System

* **Main Job**: Alerts students about application updates.
* **Data**: Notification messages and read status.
* **Components**: Notification entity with read tracking.

4. Component Interactions

* HR users create internships → Students view and apply → System creates application record.
* When HR updates application status → Notification system alerts student.
* Admin can manage all entities through admin interfaces.
* Authentication system validates users before any actions.

5. User Interface Description

Key Screens:

**Student Interface:**

* Dashboard: Shows active internships.
* Internship View: Detailed posting with apply button.
* Notifications: Updates about application changes.

**HR Interface:**

* Internship Management: Create/edit postings.
* Applications Dashboard: View and filter submissions.
* Application Review: Detailed view with status update options.

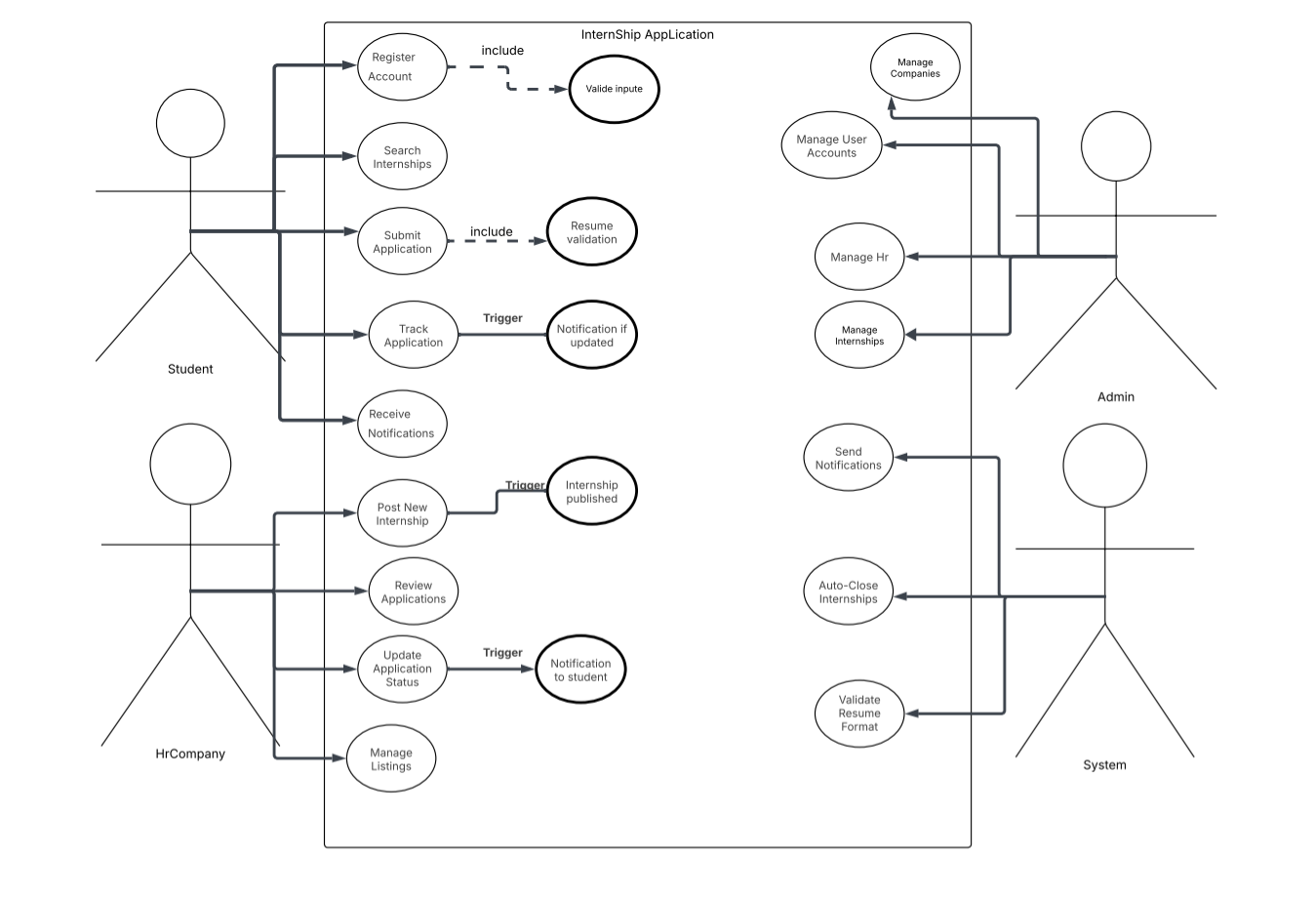
**Admin Interface:**

* User Management: CRUD operations for all users.
* System Monitoring: Analytics and reporting.

**Common Screens:**

* Login/Registration: Standard auth flow with role-based redirect.
* Profile Management: Edit personal and account information.

Use Case Diagram



ERD

A screenshot of a computer

AI-generated content may be incorrect.

Activity Diagram

A diagram of a software system

AI-generated content may be incorrect.

A diagram of a flowchart

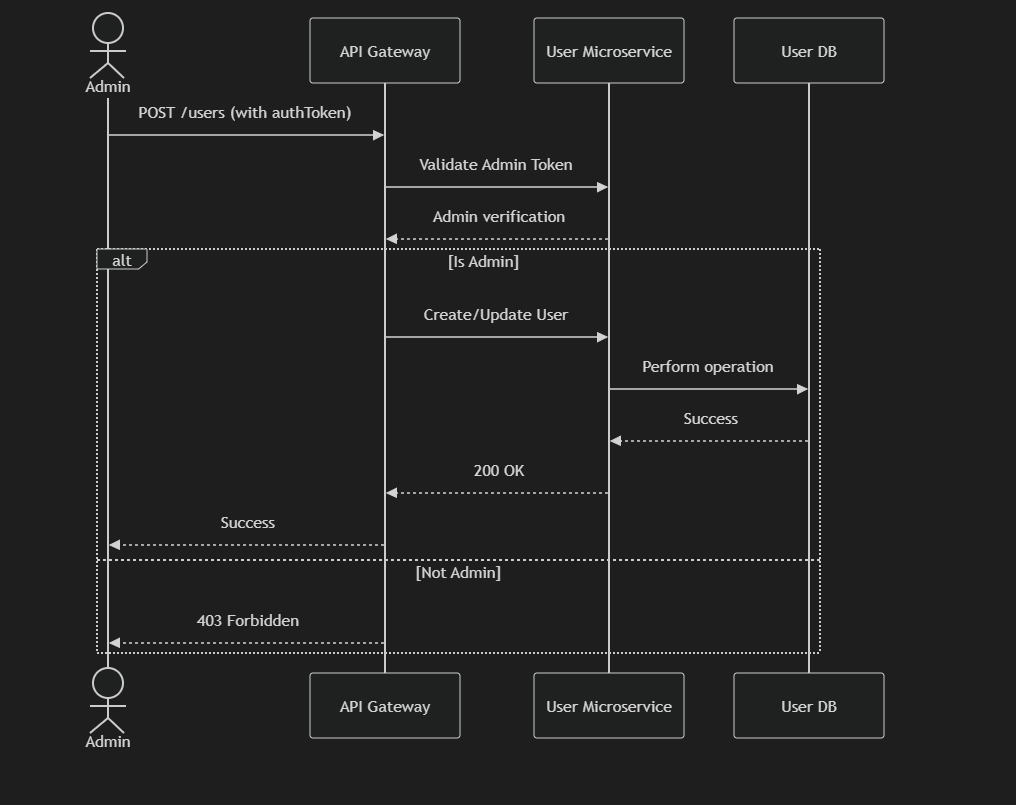
AI-generated content may be incorrect.

A diagram of a flowchart

AI-generated content may be incorrect.

Sequence Diagram

* 1. Admin Manage Users



* 1. HR Creates an internship

A screenshot of a computer screen

AI-generated content may be incorrect.

* 1. HR Reviews Applications (Status Update)

A screenshot of a computer program

AI-generated content may be incorrect.

* 1. Student applies for an internship

A screenshot of a computer

AI-generated content may be incorrect.

* 1. Student Views Internships (Paginated Search)

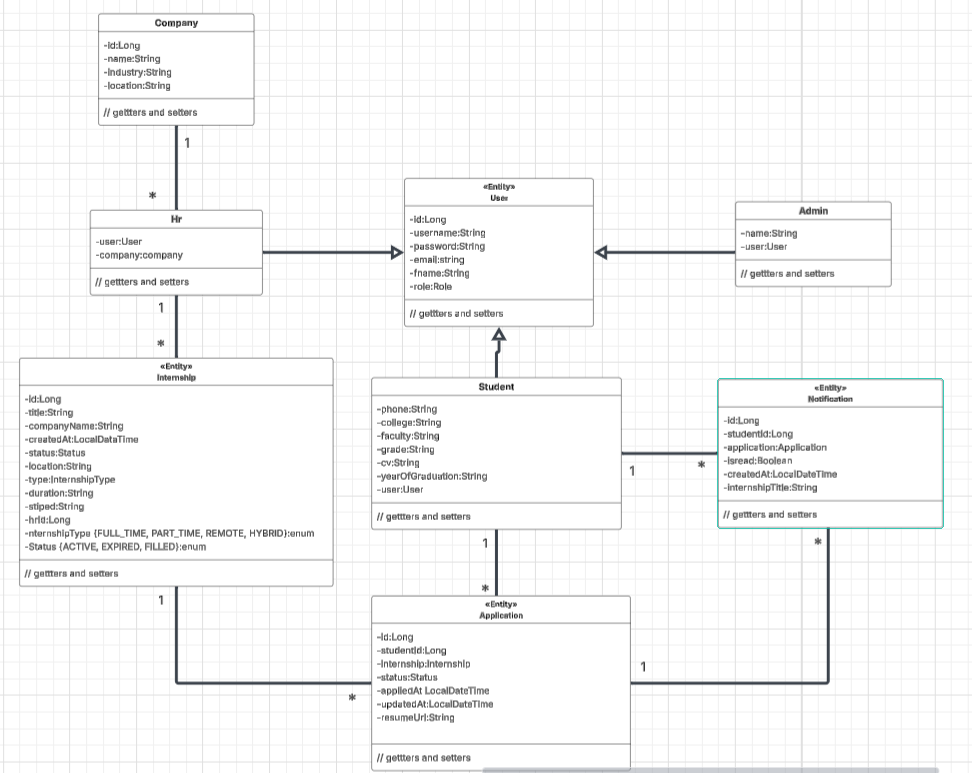
A black screen with white text

AI-generated content may be incorrect.

Class Diagram

A diagram of a company

AI-generated content may be incorrect.



OCL

USER

context User

inv usernameUnique: User.allInstances()->isUnique(username)

inv emailUnique: User.allInstances()->isUnique(email)

inv: self.oclIsKindOf(Student) or self.oclIsKindOf(HR) or self.oclIsKindOf(Admin)

STUDENT

Context Student

inv: self.applications->forAll(a | a.student = self)

inv: self.notifications->forAll(n | n.recipient = self)

inv YearOfGraduationNotEmpty: not self.yearOfGraduation.oclIsUndefined() and self.yearOfGraduation <> ''

context Student::updateStudent()

pre: self.user.role.name = 'STUDENT'

context Student::applyToInternship(internship: Internship): Application

pre: not self.applications->exists(a | a.internship = internship)

pre: internship.status = Status::ACTIVE

post: result.oclIsNew()

post: self.user.applications->exists(a | a.internship = internship and a.studentId = self.user.id

post: result.status = 'Pending'

HR

context HR

inv: self.internships->forAll(i | i.postedBy = self)

inv: self.company <> null

context HR::postInternship(title: String, description: String, deadline: Date): Internship

pre: self.company <> null

post: result.oclIsNew()

post: internship.status = Status::ACTIVE

post: result.postedBy = self

post: result.company = self.company

context HR::updateApplicationStatus(application: Application, newStatus: String)

pre: application.internship.postedBy = self

pre: newStatus in Set{'Accepted', 'Rejected'}

post: application.status = newStatus

post: application.student.notifications->exists(n | n.message = 'Your application status has changed to ' + newStatus and n.isRead = false)

ADMIN

context Admin

inv: self.companies->forAll(c | c.addedBy = self)

inv: self.hrs->forAll(h | h.addedBy = self)

context Admin::addCompany(name: String, description: String): Company

post: result.addedBy = self

post: result.oclIsNew()

context Admin::addHR(user: User, company: Company): HR

pre: company <> null

post: result.company = company

post: result.addedBy = self

post: result.oclIsNew()

COMPANY

context Company

inv nameUnique: Company.allInstances()->isUnique(name)

inv: self.hrs->forAll(h | h.company = self)

inv: self.internships->forAll(i | i.company = self)

INTERNSHIP

context Internship

inv: self.postedBy <> null

inv: self.company <> null

inv: self.postDate <= Date.now()

inv: self.deadline >= Date.now() or self.deadline = null

inv: self.applications->forAll(a | a.internship = self)

context Internship::createInternship(hrId: Long): Internship

pre: HR.allInstances()->exists(h | h.id = hrId)

post: result.oclIsNew()

post: Internship.allInstances()->includes(result)

APPLICATION

context Application

inv studentExists: Student.allInstances()->exists(id = studentId)

inv: self.student <> null

inv: self.internship <> null

inv: self.status in Set{'Pending', 'Accepted', 'Rejected'}

inv: self.applicationDate <= Date.now()

NOTIFICATION

context Notification

inv: self.sentDate <= Date.now()

inv: self.isRead implies self.readDate >= self.sentDate

context Notification::markAsRead()

post: isRead = true