COP 3710 Intro Data Eng.
Professor Paul Allen

Christina Aragon
Austin Hood
Reeve Blake

Internship Group Project

Throughout the system, the database application system for an online student internship database consists of different types of properties that our group had to identify. As we moved forward with creating a database, identifying the scope of the project was essential. The scope can help distinguish where data should be placed and is organized for an easy reference during the early development of the program. Initially reading through the functional dependencies helped us distinguish certain entities, relationships, and other important details needed for the product. Assessing these dependencies eliminated certain assumptions we may have had before looking through the functional dependencies. Although some assumptions were settled by referencing the dependencies, some were not specified. Some of the unassessed assumptions included details such as ‘What institution students are attending.’, ‘What type of degree these internships are curated towards.’, ‘If there is an age restriction for these internships.’, and ‘Each student can only apply for each internship position one time’. This can be displayed in many ways such as creating tables like Entity Relationship Diagram or an Application Architecture Diagram.

Functional Dependencies are an important factor to consider during the creation of the diagrams. The team browsed the functional dependencies for any potential entities that are needed to create the proposed database. We were able to identify four different types of entities from the dependencies that include ‘Projects’, ‘Students’, ‘Internships’, and ‘Companies’. Each property contained in the entities and all tables that may be needed to create the working database were identified. After these tables were located, we then began creating the DDL statements to get an idea of the general structure of the database. Once we knew that the DDL statements worked and the structure was correct, we then made the architecture diagram to visualize all of the connections and foreign keys. An ERD was then made by analyzing the functional dependencies to see the cardinalities of the relations that were specified for this project. Adjustments were made to the DDL statements after this diagram was made because there were some mistakes made. DDL views were then made to display the information that users would want to see, this was to avoid writing the same SQL queries repeatedly.

The product is based on helping students acquire an internship and log it into the database system if not found. In order to create this database, we have identified different aspects of the system so then we can integrate them into code seamlessly. The group uses Node.Js/Express.Js and SQLite to create the application. In doing so we create different views and features for users to use. These features include a basic Log-In view followed by a navigation menu with buttons to search internship's, create a new internship listing, or to directly place an ID number to search a specific listing. We also implemented another feature that shows if the user may view if they successfully got the internship or not. Furthermore, users are able to view data to gauge an understanding of specific views. As a whole, the database application system offers an online student internship database consisting of different types of properties that our group had to identify. If this were to be implemented in a production system it would go fairly well because both Node.JS and SQLite scale into large production software well. They are both used commonly in the industry to create these types of applications.

There should be some clarification on how to use our database because we tried to go an alternate route to explore other options. In order to get information you query a REST API with get requests to API endpoints and provide query values as part of the URL.

For example, the search positions query

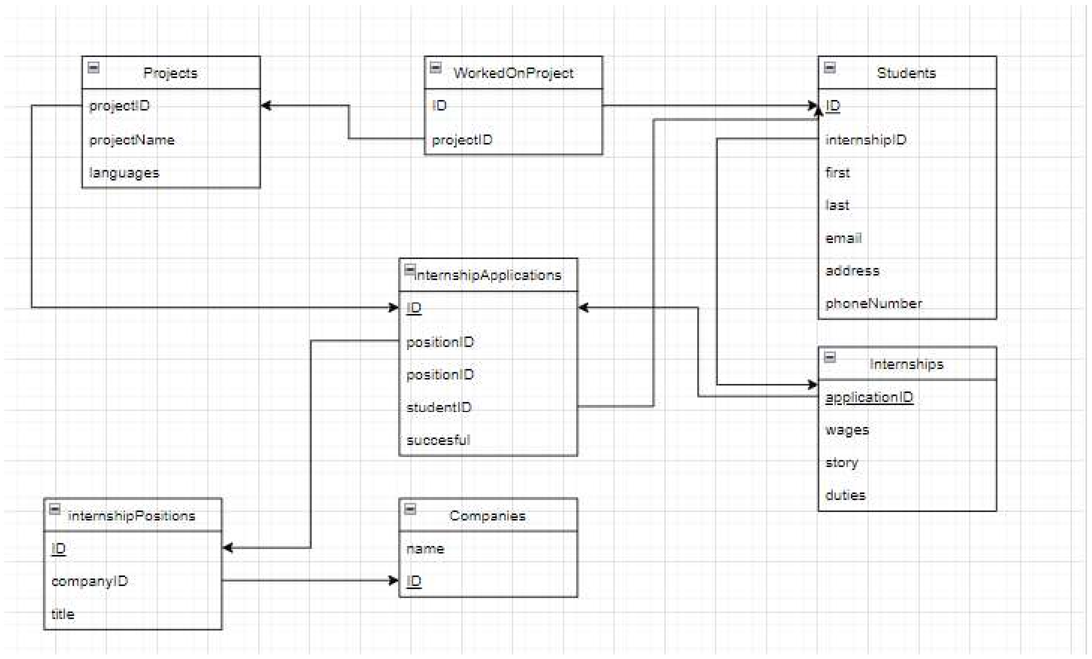
`GET (http://localhost:8080/api/searchPositionsByCompany?company=Arthrex)`

Returns a json response after collecting the internship positions from the database

Or, deleting a student with a specific ID

`POST (http://localhost:8080/api/deleteStudent?id=1)`

This could be added into an HTML document, but we suck at HTML, so we didn't attempt that ugliness.



Projects
projectID
projectName
languages
WorkedOnProject
ID
projectID
Students
ID
internshipID
first
last
email
address
phoneNumber
internshipApplications
ID
positionID
positionID
studentID
succesful
Internships
applicationID
wages
story
duties
internshipPositions
ID
companyID
title
Companies
name
ID