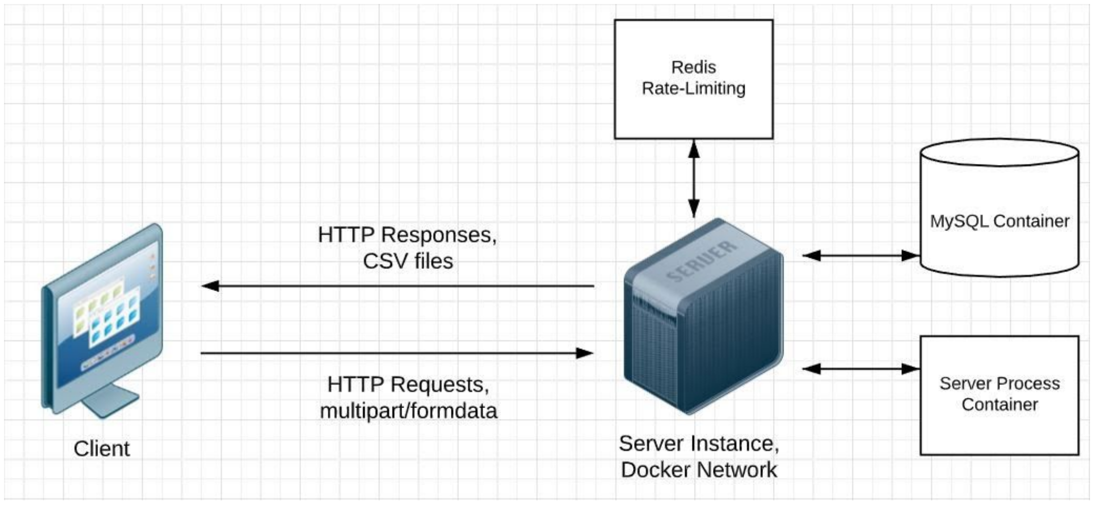
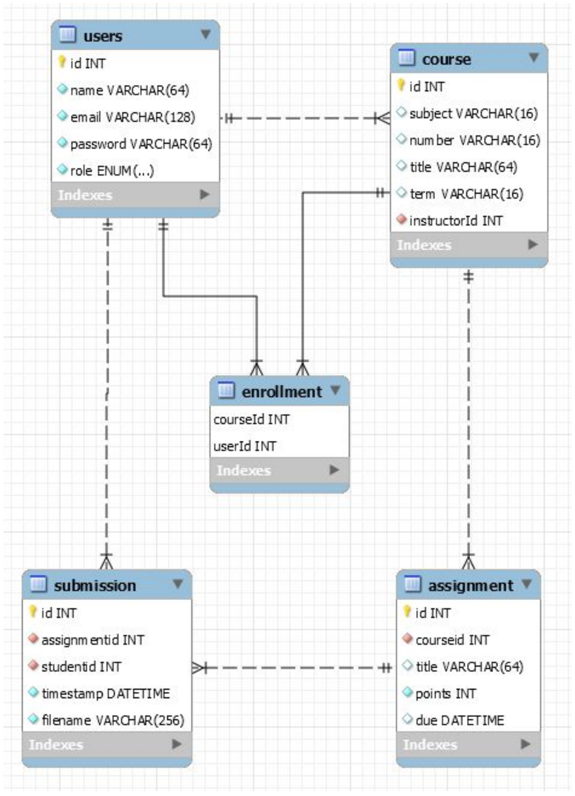
Robert Hudspeth
Conner Rhea
Jason Stalkamp
Zisong Zhang

CS 493 Final Project - Group 12
API Architecture & Design Document

Architecture Diagram



Data Layout



Design Reflection

Given a very short time window to complete this project, we opted for a very straightforward design, as demonstrated in the Architecture Diagram. Conceptually, it's a very simple design where the client and server communicate through HTTP requests and responses, and the server handles the user authentication, rate-limiting, and data storage. With cloud-based applications like the ones we've built over the course of this term, the server does all of the heavy lifting, so there's no computational load or burden placed on the client. This is made even easier since we were given the specific endpoints and specifications in the openapi.yaml file, so thanks for providing that to us!

Because a majority of us are more comfortable using MySQL, we voted to use it as our backend choice, and have provided the Entity-Relationship diagram above in the Data Layout section. Our backend was designed to have no more than 2 degrees of separation between each entity. This means the structure can be easily understood by those new to the system, and data can be assessed relatively quickly when performing retrieval and update queries.

For our data organization, the primary data type is Users, comprised of students, instructors, and admins. If a user is an instructor, then a Course entity inherits that User's id (labeled as instructorId) as a foreign key. A student can be enrolled in multiple courses, and every course roster inherits both the Courses' id and the User's id. A Course has Assignments, which inherit the Courses' courseId. A Submission of an Assignment inherits both the assignmentId of that Assignment, and the userId of the student who submitted it.