CS599_project_proposal

1. A list of all of the members of the team.

- Feipeng Yue (Graduate Level) yuef@oregonstate.edu
- Chan-Wei Yeh (Graduate Level) yehcha@oregonstate.edu
- Jirayu Saengwannakool (Graduate Level) saengwaj@oregonstate.edu
- Di Zhang (Graduate Level) zhangd5@oregonstate.edu

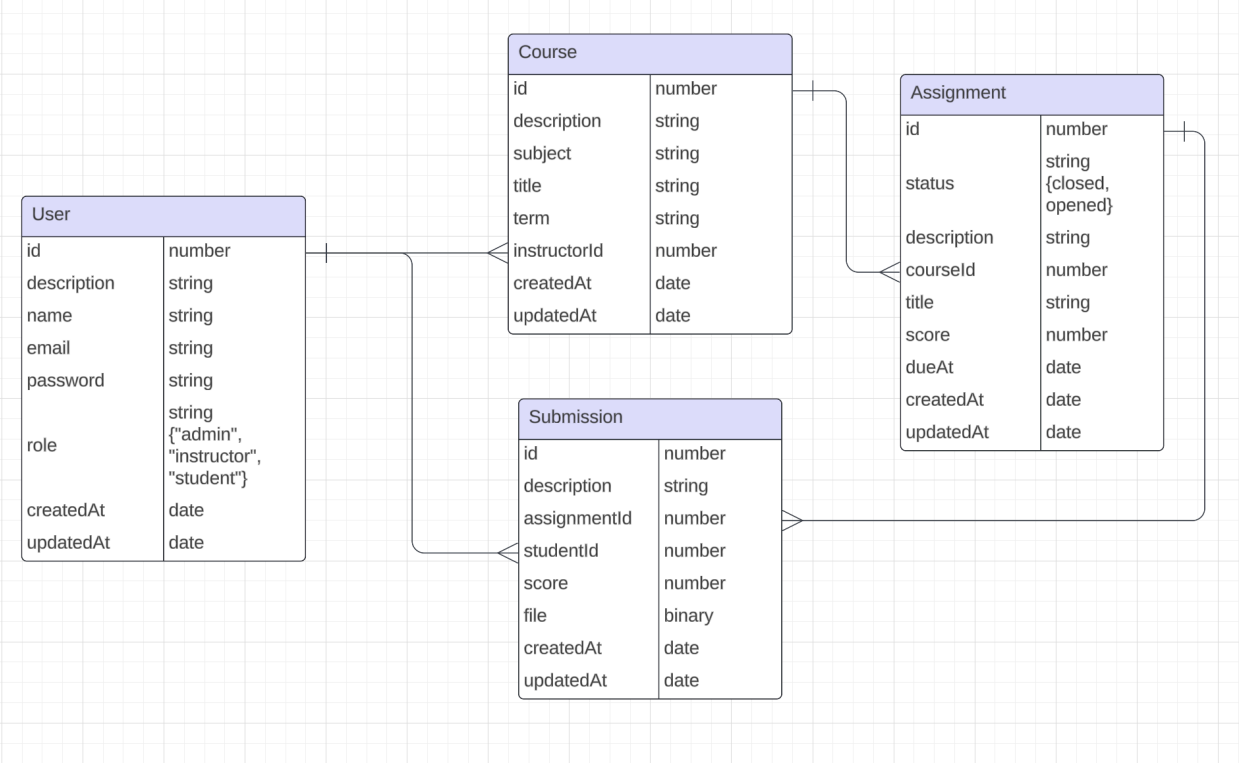
2. An outline of the data entities API will store, including descriptions of the attributes

Name	Type	Description
User		
id	number	Increment number of user to identify id
description	string	An object representing information about a Tarpaulin application user.
name	string	example: “Zhang Di” Full name of the User.
email	string	example: saengwaj@oregonstate.edu Email address for the User. This is required to be unique among all Users.
password	string	example: hunter2123123 The User's plain-text password. This is required when creating a new User and when logging in.
role	string	default: student Permission role of the User. Can be either “admin”, “instructor”, or “student”. Enum: [admin, instructor, student]
createdAt	date	ISO Date of creating time

updatedAt	date	ISO Date of updating time
Course		
description	string	An object representing information about a specific course.
subject	string	example: CS Short subject code.
id	number	example: 599 Course number.
title	string	example: Cloud Application Development Course title.
term	string	example: sp23 Academic term in which Course is offered.
instructorId	number	ID for Course instructor. Exact type/format will depend on your implementation but will likely be an integer. This ID must correspond to a User with the 'instructor' role.
createdAt	date	ISO Date of creating time
updatedAt	date	ISO Date of updating time
Assignment		
id	number	To identify assignment id
status	string	Status of assignment which can be either opened or closed
description	string	An object representing information about a single assignment.
courseId	number	ID of the Course associated with the Assignment. Exact type/format will depend on your implementation but will likely be an integer
title	string	example: Assignment 2 Assignment description.

score	number	example: 100 Possible points for the Assignment.
dueAt	string	example: “2024-05-14T17:00:00-07:00” Date and time Assignment is due. Should be in ISO 8601 format.
createdAt	date	ISO Date of creating time
updatedAt	date	ISO Date of updating time
Submission		
id	number	To identify id of submission
description	string	An object representing information about a single student submission for an Assignment.
assignmentId	number	ID of the Assignment to which the Submission corresponds. Exact type/format will depend on your implementation but will likely be an integer
studentId	number	ID of the Student who created the submission. Exact type/format will depend on your implementation but will likely be an integer
score	number	The grade, in points, assigned to the student for this submission, if one is assigned. Should not be accepted during submission creation, only via update.
file	binary	When the Submission is being created, this will be the binary data contained in the submitted file. When Submission information is being returned from the API, this will contain the URL from which the file can be downloaded.
createdAt	date	ISO Date of creating time example: 2022-06-14T17:00:00-07:00 Date and time Submission was made. Should be in ISO 8601 format.

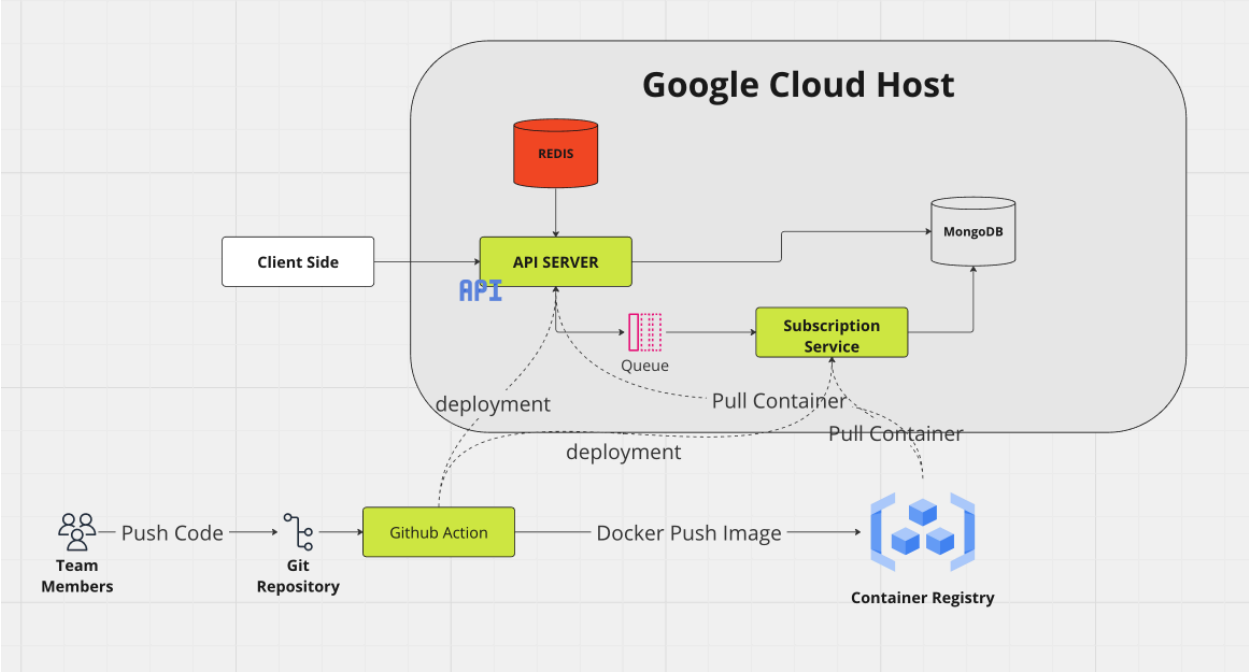
updatedAt	date	ISO Date of updating time



UML Diagram

3. A list of the different services we plan to use to power your API (e.g. MySQL, MongoDB, RabbitMQ, Redis, etc.).
Tech Stacks

- **API Services** will be written by **TypeScript** which can be compiled to Regular Node.js
- Choosing **MongoDB** as a Database to store all collections that relate to Tarpaulin api
- **Redis** as a memory caching to store temporary data which are often query and using rate limit feature.
- **RabbitMQ** to make api not be throttling during registration and could store a queue of processing such as csv file output.
- **Docker** Container, all services (apis, database, redis, rabbitmq) will be containerized as a docker container.
- **Git**, using git to collaborate with team members and store codes.
- **Google Cloud Service**



3rd Party Libraries

- **CSV** library, to export data from database and process data into csv file
- **Mongoose**, the database connector
- **Express**, providing http server to client side.
- **Amqp**, is a message queue connector.

4. Cloud Host Service (Grad Section)

- The cloud host service: Google Cloud (https://cloud.google.com/)
- The services that will be implemented in the project
    - **Google Kubernetes Engine Quickstar**t: This could deploy a containerized application on Google Cloud. (https://cloud.google.com/kubernetes-engine?hl=en)
    - **Google Compute Engine**: This will be used for deploying API servers directly on a virtual machine. (https://cloud.google.com/products/compute?hl=en)
    - **MongoDB** on Google Cloud.
    - **Redis** on Google Cloud: API caching.
    - **Google Cloud Build**: Setting up pipelines for continuous integration.

5. **A description of the division of labor for the project between the members of the team (see below for more info about this).**

- Feipeng Yue (Graduate Level)
  Users API: Implement all endpoints related to user management, ensuring they follow the OpenAPI specification.
  Authorization: Ensure all endpoints require proper authorization.

- Chan-Wei Yeh (Graduate Level)
  Assignments API: Implement all endpoints related to assignments, ensuring they follow the OpenAPI specification.
  Pagination: Implement pagination for the appropriate endpoints.

- Jirayu Saengwannakool (Graduate Level)
  Submissions API: Implement all endpoints related to submissions, ensuring they follow the OpenAPI specification.
  Rate Limiting: Implement rate limiting for the API.

- Di Zhang (Graduate Level)
  Courses API: Implement all endpoints related to courses, ensuring they follow the OpenAPI specification.
  Infrastructure and Deployment Process: Handle Docker containerization, ensure services are run in Docker containers, and manage the deployment process.

Each person should also contribute to the following:
  Tests/Requests: Write and prepare a set of tests/requests to demonstrate the API's functionality during the grading demo.
  High-Quality Design and Implementation: Ensure that the API has a high-quality design and implementation overall.

Other source

- **Project Requirement:**
  https://docs.google.com/document/d/1A-i0wsm3nTcKpkzOjcajy1OpDV1ZXcUPYJeo1vJ9tjQ/edit#heading=h.1bg6tvj8exsh

- **Project proposal details:**
  https://docs.google.com/document/d/16U2uLCTHKwX7Cik-vzGQzh7TndFNNvej8DUD0Tq9fHc/edit