**QuizNose Documentation**

1. Summary/About the project
2. What problem does this solve
   1. Decoupling features and functions
3. What it was built with
4. High Level Diagram/Explanation
5. Features:
   1. Login/Verification
   2. Bulk question/quiz upload
   3. Test taking feature using stored questions
6. Current Objectives
   1. Deployment on Containers
   2. Email notification with test scores
   3. Messaging between teachers and students
   4. Multisession Testing

**About this Project**

Open source educational platform that allows for student teacher interaction with the goal to subject aptitude assessment for the modern classroom. Created with a microservice architecture, QuizNose can be scaled to meet consumer design. This will allow for higher volumes of traffic without worry of performance slow down as a monolithic application.

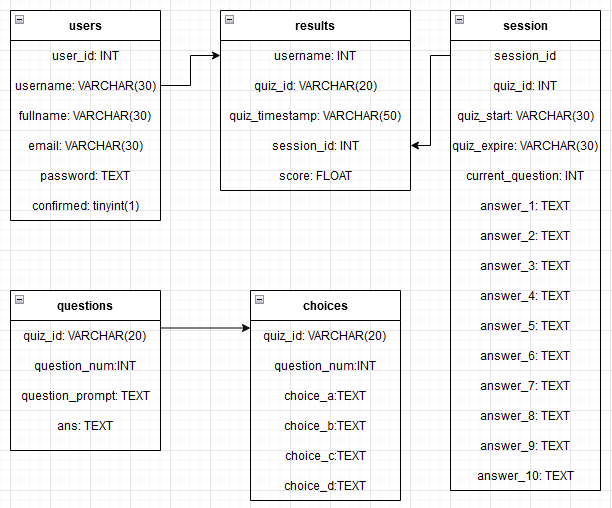
What Problems does this solve?

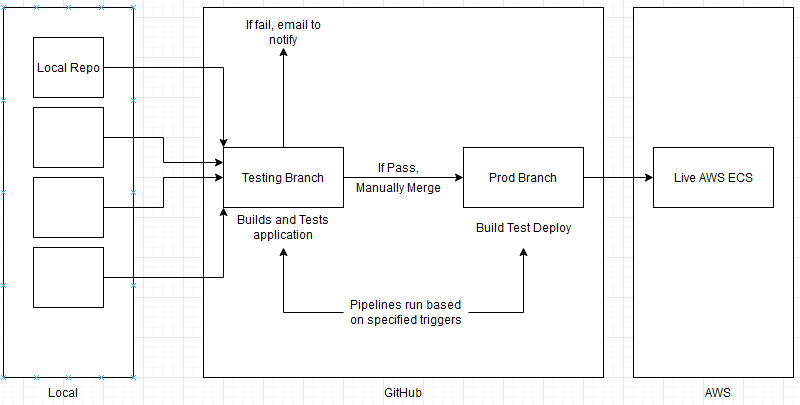
wThe application provides teachers insight on the performance of their students and has a granular view of the student’s understanding of a subject. By decoupling, we are able to ensure that the application is able to run smoothly and integrate any additional changes without drastically affecting other parts downstream.

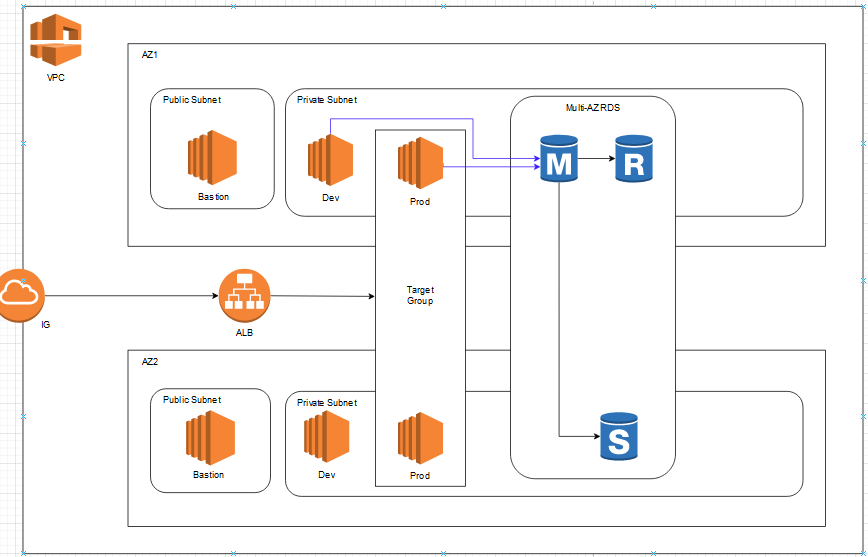
What this was built with:

* Flask
* Python
* HTML
* JavaScript
* CSS
* Terraform
* Docker

**High Level**







Features

1. Login/Verification

Using Flask modules and pip modules.

Front end uses pip module WTForms to connect to the frontend to POST data to the flask backend. The data is stored as a temp var and compares with data on the backend. This is used for login verification, checking for existing username. If it exists, it pulls the matching row and its data. It will grab the password hash stored in the DB as a temp var.

For password verification passlib that uses sha256 verification to match password. If it matches, it will redirect to the appropriate dashboard, either student or teacher. If it fails, it will flash a popup that says the password is incorrect or the account is not verified.

1. Bulk question/Quiz upload

Uses Flask modules and packages: os, csv modules.

It will allow users to upload a csv file on an html page. It will check for file type, if successful, it will save it locally on the server, temporarily. Once saved, the csv reader function will run and open the file and iterate through the csv file, skipping the header. When iterating, it will insert the data into the appropriate tables.

1. Test Taking

Uses Flask modules and packages. This is connected to a MySQL database that houses all of the data to be served.

There are files for CRUD functions to create the tables in the database, Insert into tables, delete tables. There is also a file to pull the data from the db to be served on the front end.

We are using the results table to track a student’s progress through a quiz. The current\_question will start at 1 when the student starts. The quiz will start to serve based on the current\_question. The database will be updated when the student submits an answer for a question. This will trigger a POST api call to the backend. This will search for the response that the student gave and update the database with the answer they had chosen and increment the current\_question. This will then have the backend reload the page with the next question or the value that current\_question is set to.

There is additional functionality where a student can traverse the quiz questions thru the next question and previous question buttons. If a student does not submit an answer, a POST api call will be made and trigger logic to continue the increment without writing to the db. The back button will allow students to return to a previous question. This will create a GET api call that triggers logic to reserve the page with the previous question contents. An answer selected this way will overwrite what the student had selected before.

After the current\_question > 10, the quiz logic will redirect the user to the results page. The page will display the score the student received. There is a comparison between the correct answer and the student’s answer and then a score is calculated. This score will be updated in the database as well as shown on the results page.