Contest-submission Design Document

Problem

- We had issues with scoring on spreadsheets: cross referencing, duplicate submissions, and people having the same names was a large bottleneck in the grading process
- Score submission and emailing a large number of students caused much manual work and a large delay

Solution

- We hope to solve these problems by creating a fully functionally Flask app to replace the outdated Google form + Google sheet + gmail stack traditionally used
- Key features
    - Account registration with email confirmation
    - Contest answer submission and updating
    - Automatic grading

Structure

- Frontend will be vanilla html, css, js
- Backend will be Flask + Postgres

Models

- Student
- Contest
- Submission (table with indirect references to student id and contest id)

Database Schema

Table name: student

	id (PK)	token	name	email	password	grade	gender	school	active
req	True	True	True	True	True	True	True	True	True
type	INT	TEXT	TEXT	TEXT	TEXT	INT	TEXT	TEXT	BOOL
value	Primary key	JWT token	User's full name	User's email	Hash of user's password	User's grade	User's gender M, F, O	User's school	If user is active

Constraints

- Unique email,
- Password must be at least 6 characters

Table name: contest

	id (PK)	name	deadline	num_questions	answers	active
req	True	True	True	True	False	True
type	INT	TEXT	TIMESTAMP	INT	INT(array)	BOOL
value	Primary key	Display name of contest (2020 October)	EDT time for deadline of contest 24 hr	Number of questions in contest	Array of answers	If contest submission is open

Constraints

- Answers must have length num_questions
- Name has a specific format

Table name: submission

	id (PK)	student_id	contest_id	answers	raw_score	adj_score
req	True	True	True	False	FALSE	FALSE
type	INT	INT	INT	INT(array)	INT	INT
value	Primary key	Id of student	Id of contest	Array of correct answers	# questions right	Weighted score

Constraints

- Unique constraint on (student_id, contest_id)

Views

- /admin admin panel
- / home page with general information and faqs
- /register page to register
- /sign-in page to sign in
- /profile page to edit profile
- /forgot-password page to recover password
- /contests page to view list of contests current and past
- /contests/<contest_id> page to download a specific contest and submit answers

API Endpoints

- Admin-side
    - /new-contest
    - /edit-contest

        - /download-users
        - /download-contests
        - /score/<contest_id>
        - /download-results/<contest_id>
        - /email-results/<contest_id>
        - /upload-test/<contest_id>
        - /upload-solutions/<contest_id>
- Student-side
    - /new-account
    - /edit-account
    - /submit/<contest_id>

Security

We will use tokens through Flask's JWT library