Snappet Challenge

Evan Sahit

For the technical interview, I need to code something up to discuss in a in-person meeting at Snappet HQ.

The challenge

- For the Snappet Challenge, I need to create a clear overview for a teacher of all the work his/her students did that day. The data is provided as a CSV and JSON file, where I can choose which to use.
- I can only use the data on the date **2015-03-24 11:30:00 UTC**.
- The data includes the following fields:
  - **SubmittedAnswerId**
    - Integer.
    - ID of the submitted answer.
  - **SubmitDateTime**
    - Date time in UTC.
  - **Correct**
    - Integer.
    - A boolean represented by 1 or 0.
  - **Progress**
    - Integer
    - A negative or positive integer which represents the progress of the student AFTER they completed a submission.
  - **UserId**
    - Integer.
    - ID of the user making the submission.
    - Only 20 students.

- **ExerciseId**
  - Integer
  - ID of the exercise for the submission is made.
- **Difficulty**
  - Double
  - Decimal number with 2 digits after the decimal point representing the difficulty of the exercise.
- **Subject**
  - String.
  - Represents subject which the exercise falls into.
  - Could be: Spelling, Begrijpend lezen, Rekenen.
- **Domain**
  - String.
  - Represents which domain within the subject this exercise pertains to.
  - Could be: Getallen, Meten, Taalverzorging, Verbanden, Verhoudingen.
- **LearningObjective**
  - String.
  - Represents the learning goal of this exercise.

The dashboard

I need to think from the perspective of a teacher and implement something that would be truly useful for them. How can this data best be visualized in order to give teachers a clear picture of what their students have worked on that day?

- In order to cut down the amount of data that has to be searched through, a second filtered dataset can be created which only contains data for the day **2015-03-24** up until **11:30:00 UTC**.
  - This cuts the dataset down from 30k+ to 3k+.

    - This does hurt the extensibility of this API as all the endpoints are specifically designed for this specific subset of the total dataset, so maybe not a good idea.

- Dashboard showing the following pieces of information for the start of the target day up until the cutoff time:
    1. Bar chart showing the different subjects on the x-axis and the number students that have worked on those subjects on the y-axis.
        a. However, if all students follow the exact same schedule of subjects on this day, then this chart won't contribute much as all the bars would be the same height.
    2. A bar chart visualizing the class's daily progress for that day for each subject.
        - This could be a diverging bar chart where a colored in bar right of center means improvements and a colored bar left of the center means a degradation in class progress for that subject.
        - Can be per subject how many progress points the class increased or decreased by summing up their progress for each subject.
        - This can be also as simple as "<subject>: <class_progress_sum>". Where positive values for <class_progress_sum> can be (different shades of) green and negative numbers can be (different shades of) red and values around 0 are (different shades of) yellow.
    3. A bar chart visualizing the class's daily progress for each learning goal.
        - Similar to chart **(2)**, but this focusses on learning goals instead of subjects.
    4. Pie chart visualizing the number of exercises completed per subject.
        a. Clicking on a segment could lead to a page where per subject another pie chart is shown visualizing the number of exercises completed per domain handled within the subject.
    5. A bar chart visualizing per subject the amount of correct and incorrect answers.
        a. Per subject 2 bars can be grouped together; one for correct and the other for incorrect answers.
    6. A bar chart visualizing the overall progress per student.

- Since there are only 20 students, a bar chart showing the student's progress for that day can be shown. This can be a diverging bar chart as well.
- Clicking on a student('s bar chart) could lead to a detail page where more vizualizations are shown for that student such as their progress for each subject.

Tech stack

To speed up development, I am going to use tech I am already familiar with, being mainly JavaScript/TypeScript and Python. I'm going to use a the frontend framework React and the backend framework FastAPI. This is essentially a data visualization task, so a purely dashboard approach such as something like Streamlit would work, but we're trying to showcase some full-stack skills here.

I am planning on letting the frontend purely be responsible for displaying the data and the backend be responsible for everything to do with data. So all data manipulation or filtering is going to be done by the backend. The frontend just sends requests describing what it needs.

Frontend

- Responsible for displaying the data and handling user interaction.
- No data processing, so no going into the dataset and filtering out what is needed for a given visualization by the frontend itself.
- Frameworks/ libraries
  - `React` (technically just a library)
  - `React Router` to handle frontend routing.
  - `Recharts` to handle all the actual data visualization.

Backend

- Will be using `Python` as I am quite familiar with it and a great tool for working with data, especially considering the data-related tooling in the `Python` ecosystem.
- Responsible for providing the frontend with the data that it needs.

- The CSV file will be used as this project's "database".
- Will respond with the filtered data as described by the frontend.
- Take in the CSV and do the necessary filtering to return the needed data and return as `JSON` to the frontend.
- Frameworks/ libraries
  - `FastAPI` as a backend framework which will respond to frontend requests.
  - `Pandas` to work with CSV files.

TODO

Backend

- [x] ~~Create endpoints for each type of chart.~~
  - ~~All endpoints must will receive the follwing query parameters which will be used to filter the dataset for a given date time:~~
    - ~~`date`~~
      - ~~date in year/month/day format~~
    - ~~`start-time`~~
      - ~~start time in hr:min:millisec format~~
    - ~~`end-time`~~
      - ~~end time in hr:min:millisec format~~
  - [x] ~~`/classes`~~
    - ~~This endpoint will handle all data requirements for class-level visualizations such as charts **(2)**, **(3)**~~
    - [x] ~~`/classes/progress-per-subject`~~
      - ~~For chart **(2)**.~~
      - ~~Per subject, sum the progress points for all exercises for the whole class.~~
      - ~~Return a JSON object with the following key-value pairs:~~
        - ~~`Spelling: <sum_progress_points>`~~ -

    - ~~`Begrijpend Lezen: <sum_progress_points>`~~
    - ~~`Rekenen: <sum_progress_points>`~~ ~~-~~

- [x] ~~`/classes/progress-per-learning-objective`~~
    - ~~For chart **(3)**.~~
        - ~~Per learning objective, sum the progress points for all exercises for the whole class.~~
    - ~~Return a JSON object with the following key-value pairs:~~
        - ~~`<learning_objective>: <sum_progress_points>`~~ ~~-~~
        - ~~`_`~~ ~~-~~
- [x] ~~`/classes/progress-per-student`~~
    - ~~For chart **(6)**.~~
    - ~~Per student, sum the progress points for all exercises completed.~~
    - ~~Return a JSON object with the following key-value pairs:~~
        - ~~`<student_id>: <sum_progress_points>`~~ ~~-~~
        - ~~`_`~~

- [x] ~~`/subjects`~~
    - This endpoint will handle all data requirements for subject-level visualizations such as charts **(1)**, **(4)** and **(5)**.
- [x] ~~`/subjects/students-per-subject`~~ ~~-~~
    - ~~For chart **(1)**.~~
    - ~~Per subject, sum the number of students that have worked on each subject.~~
    - ~~Return a JSON object with the following key-value pairs:~~
        - ~~`Spelling: <number_of_students>`~~ ~~-~~
        - ~~`Begrijpend Lezen: <number_of_students>`~~
        - ~~`Rekenen: <number_of_students>`~~ ~~-~~
- [x] ~~`/subjects/exercises-per-subject`~~
    - ~~For chart **(4)**.~~

- ~~Per subject, sum the number of excerises completed for each subject.~~
- ~~Return a JSON object with the following key-value pairs:~~
  - ~~Spelling: <number_of_exercises>~~
  - ~~Begrijpend Lezen: <number_of_exercises>~~
  - ~~Rekenen: <number_of_exercises>~~

- [x] ~~/subjects/excercises-per-domain~~
  - ~~Per domain of a subject, sum the number of excerises completed for each domain.~~
  - ~~Return a JSON object with the following key-value pairs:~~
    - ~~Domain: <number_of_exercises>~~
- [x] ~~/subjects/peformance-per-subject~~
  - ~~For chart **(5)**.~~
  - ~~Per subject, sum the number of correct and incorrect answers for each subject.~~
  - ~~Return a JSON object with the following key-value pairs:~~
    - ~~Spelling:~~
      - ~~correct: <number_of_correct>~~
      - ~~incorrect: <number_of_incorrect>~~
    - ~~Begrijpend Lezen:~~
      - ~~correct: <number_of_correct>~~
      - ~~incorrect: <number_of_incorrect>~~
    - ~~Rekenen:~~
      - ~~correct: <number_of_correct>~~
      - ~~incorrect: <number_of_incorrect>~~

- [x] ~~/students~~
  - ~~This endpoint will handle all data requirements for student-level visualizations~~
- [x] ~~/students/progress-per-subject~~

- ~~Per student, sum the progress points per subject.~~
- ~~Return a JSON object with the following key-value pairs:~~
    - ~~<student_id>:~~
        - ~~Spelling: <progress_points>~~ -
        - ~~Begrijpend Lezen: <progress_points>~~
        - ~~Rekenen: <progress_points>~~ -

Frontend

- [x] ~~Create Header component.~~
- [x] ~~Create Dashboard page.~~
- [x] ~~Create Footer component.~~
- [x] ~~Create generic Chart component for each chart.~~
- [x] ~~Create data fetching functions for each chart to use in hooks.~~
- [x] ~~Create data fetching hooks each chart for data, loading and errors.~~
- [x] ~~Make site responsive.~~
- [x] ~~Finish styling.~~