PROJECT DELTA

Anirudh Nimmagadda - anirudh.nimmagadda@gmail.com
Bharani Dharan - bharanidharan.nb@gmail.com
Vipul Rawat - vipulrawat007@gmail.com

Table of Contents

Overview

Project Delta is an attempt to provide Pesto's instructors visibility into how effectively the student body is learning the curriculum.

We will achieve this by tracking:

1. How confident students are that they have mastered topics covered in the lectures
2. Students' performance in exercises

… and presenting instructors with visualizations that will allow easy identification of:

1. Struggling students
2. Insufficiently-explained topics
3. Unreasonably difficult exercises (like the telephone regex question we were given on day 2. WTH guys? It was day 2!)

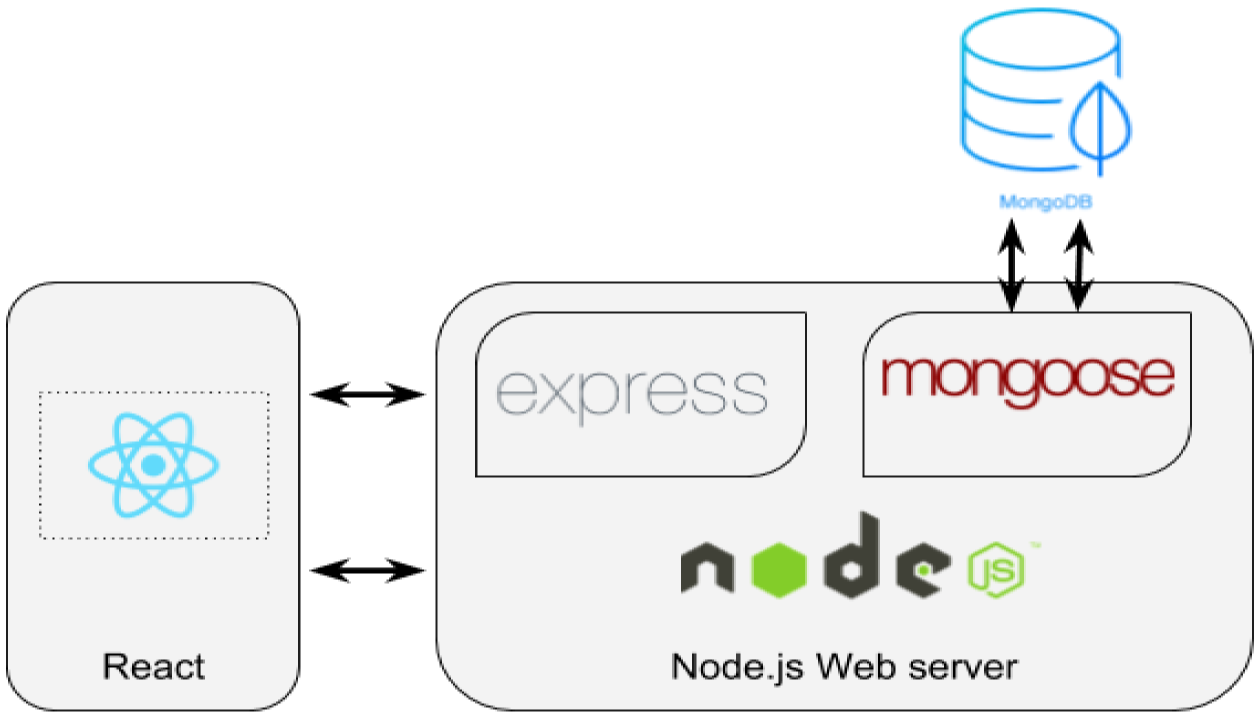
The process already in place at Pesto for this purpose involves instructors creating a new Airtable form for every set of exercises attempted by students. An example form is here.

Students must fill in their name and email address in every form, and the dropdown input used to select solved exercises is inefficient, since only a single option can be selected each time the dropdown is expanded.

Also, there is no way to track how well students understand the set of topics covered in a given lecture under the current process.

System Architecture

The below diagram shows the flow of our application with client, server and database:



For maintaining the above represented system we are following **Monolithic Architecture**. Monolith application means keeping all logically separated projects in a single repository to consolidate all development workflows.

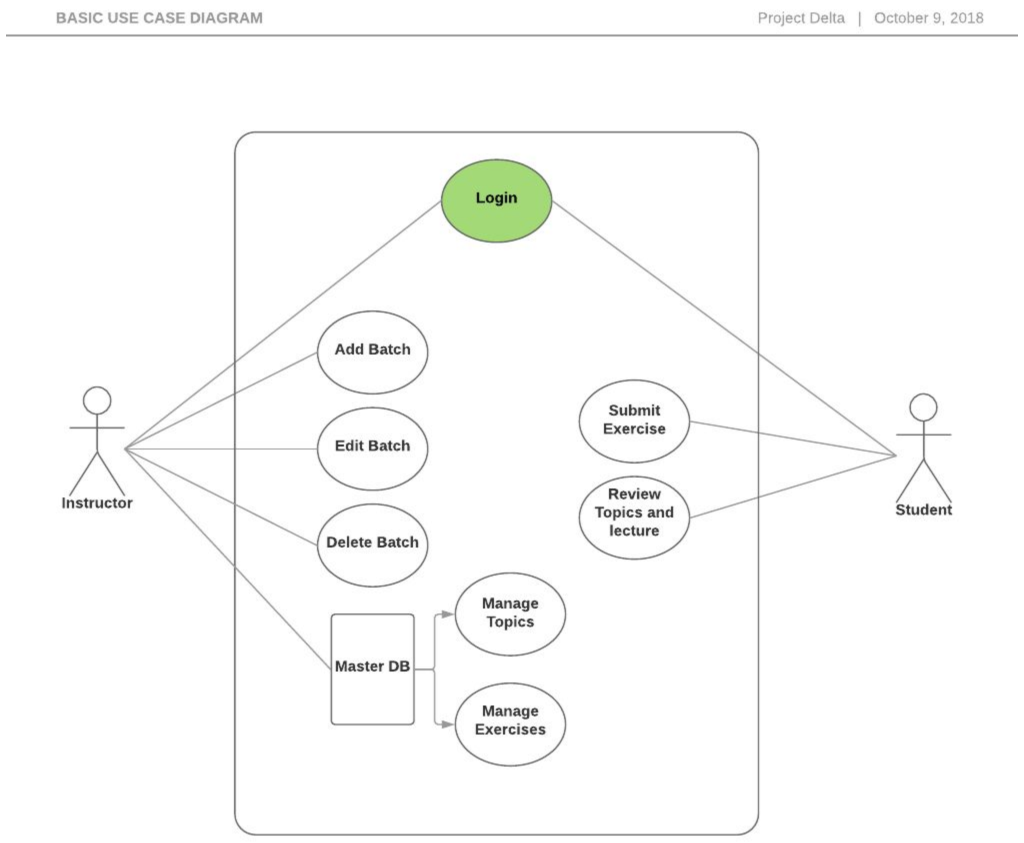
Our project will contain three sub-repositories termed as ‘student’, ‘instructor’ and ‘api’. ‘student’ and ‘instructor’ are the client side repositories which contains the separate front-end applications. ‘api’ is the sub-repo which will interact will the database and provides the endpoints to with data to consume. The whole mono-repo is termed as ‘packages’ in our project.

```
.
├── packages                  # Mono-repository
│   ├── shared                 # Contain shared components
│   ├── api                    # Backend sub-repo
│   ├── students               # Frontend sub-repo for students
│   └── instructors            # Frontend sub-repo for instructors
└── package.json               # Master packages
```

Each sub-repo contains all the individuals *tests* and their own packages. The mono-repo follows the below directory structure:

```
.
.
├── ...
|── api                     # Sub-repo
│   ├── _tests_             # Test files
│   └── src                 # Source file
└── ...
```

Major Features



Birds-eye view of the actions that an instructor and a student can perform.

Different entry points for students and instructors
Student landing page is at https://delta.pesto.com
Instructor landing page is at https://instructors.delta.pesto.com

If the request for this page is made *without* a valid sign-in token, as will be the case when the user navigates to the page directly, they will be shown the sign-in screen from which they can start the magic sign-in process.

If the request for this page contains a valid sign-in token, as will be the case if the user clicks on the link in the email we send them during the magic sign-in process, the user is shown the appropriate dashboard.

User Account Creation
Instructor account creation is handled by direct database insertion. This may be changed in future versions of **Delta**.

Student accounts are auto-created when they go through the magic sign-in process for the first time. On first-time-sign-in, they are shown a profile creation page where they must submit these essential details:

- first & last name
- batch city & number
- profile picture

'Magic sign-in'
When a user tries to sign-in by submitting their email, an alphanumeric string of 20 random characters is created in the back-end, and stored in Redis under a user-namespaced key (ex: **anirudh.nimmagadda@gmail.com:token**) set to expire in 15 minutes.

A link to the dashboard relevant to the user is generated, and the token is added to it as a query parameter. Example links are shown below:

https://delta.pesto.com/?token=qwertasdfgzxcvb12345
https://instructor.delta.pesto.com/?token=QWERTASDFGZXCVB12345

This link is sent to the user as an email. Clicking it after the token has expired will take the user back to the home page, where they can try signing-in again. Otherwise, the server creates a JWT to remember the user, and responds with a set-cookie header containing the JWT and the appropriate dashboard.

Signed-in experience for students
The goal is to collect student student feedback on:

1. How well they feel they learned a particular topic through its associated lecture
2. How they did on the exercises

Signed-in students see a dashboard containing a choose-able list of DAYS (1 through 20). Each DAY's view has a set of lecture topics that were covered on that DAY, and the accompanying set of exercises. A low-fidelity mockup of this dashboard can be viewed here.

The student can rate their level of understanding of a topic through the lecture on a sliding scale from 1 (did not understand at all) - 10 (understood fully).

Most exercises have accompanying unit tests. When an exercise has tests, it is possible to check whether a student completed it or not by parsing the output of tests run by Travis CI on the student’s latest github push. We call these exercises ‘automatic exercises’, and the rest ‘manual exercises’

Our program does the above mentioned parsing and marks automatic exercises as solved/unsolved without the student’s intervention. Manual exercises must be marked solved by the student (only if he/she has actually solved them!).

Once the student has rated the DAY’s topics, and marked the manual exercises he has solved, he can submit the topics and exercises form. This will save his ‘performance’ for that DAY into the MongoDB database backing **Delta**.

Signed-in experience for instructors
Instructors should be able to do the following things:

1. View visualizations of how their students are performing at learning topics and solving exercises
2. Create/read/edit/deactivate lecture topics
3. Create/read/edit/deactivate exercises
4. Create/read/edit/deactivate batches

Visualizing student performance
The primary component of the instructor dashboard is a pair of charts - one graphing students’ topical understanding, and the other graphing their performance on the exercises.

Both charts contain dropdown inputs that allow viewing the data in different ways. A few useful visualizations are listed below:

1. For particular <batch + DAY + topic>, graph all students’ understanding of topic as a descending-height bar graph
    - This viz allows the instructor to identify students who are struggling with a particular topic
2. For particular <batch + DAY + student>, graph the selected student’s rating across all of that DAY’s topics as a desc-height bar graph
    - This viz allows the instructor to identify the topics a particular student is weakest at
3. For particular <batch + DAY>, graph exercises (X-axis) vs % of students in batch who solved them (Y-axis)
    - This allows the identification of very difficult exercises that no/few students are able to solve

Create/View/Edit/Deactivate lecture topics
We maintain a ‘master’-list of lecture topics as a single collection in our MongoDB database. The instructor will have access to pages that allow him/her to view and edit this list.

Each entry in the list is a single lecture topic identified by name (ex: ‘git rebase’, ‘mongodb insert’), and having a category (ex: ‘git’, ‘mongodb’). The instructor must also specify the DAY on which he/she plans to lecture on this topic, though this setting can be overridden for any specific batch later.

Create/View/Edit/Deactivate exercises
We maintain a similar ‘master’-list of exercises. Each exercise has an identifying name. The instructor must also specify the lecture topic the exercise is associated with.

Create/View/Edit/Deactivate batches
The instructor can create a new batch by specifying three things: the city, a batch number, and the start date of the batch.

When an instructor creates a batch, we use the information already in the ‘master’ lists of topics and exercises to create batch-topic and batch-exercise associations. These batch-specific lists are what signed-in students belonging to that batch will be asked to mark.

Creating these batch-specific copies of topics and exercises has multiple benefits. Specifically:

1. Instructor can deactivate a topic from the master list if he/she feels it is no longer useful for future batches. However, batches that are already in-progress, where the topic has been covered, will still have access to it
2. Instructor can cover an unannounced topic for an in-progress batch by adding it to the appropriate batch-specific list, without worrying about having to deactivate it prior to new batch creation

Version 2 Features

The following is a list of features that have been discussed as good-to-haves, but are not currently slated for implementation during the month allotted to the project:

1. **[Batch DAY-date update]** A single batch typically has 20 DAYs of lectures held Monday through Friday over a four-week period. So, for a typical batch, DAY-date associations are easy to work out based on its start date. However, a public holiday or similar event may cause a lecture day to be cancelled, affecting the DAY-date associations for that DAY and all following DAYs. The instructor should be able to update the DAY-date associations of a batch when this happens.
2. **[Multiple instructors]** Currently, the plan is to implement the single instructor and single center version but in future there will be multiple instructor and multiple centers handling their own data of topics and exercises.Like, one instructor in delhi has his list of 5 topics but another instructor in pune suppose added 1 more topic along with those 5 topics then his list will be different than of previous one. And the same may occur with exercises also. So, in future version the schema should be design as to handle multiple centers and instructors.