Pitt CS Online Advisor Project Refactor

Goal: From React + Node.js + mySql

To Front-End (React) + Back-end (Spring Boot, Redis) + DB (mySql)

Be able to deploy with Docker to Pitt Servers

Benefits:

1. Data security, won't expose the database
2. High Available, Load Balance
3. better extendibility, could introduce other microservices (features) in future
4. Easy to use (potential to integrate with peopleSoft, student center)

Current issues:

```
var express = require("express");
var router = express.Router();
var mysql = require("mysql2");
const { isNull } = require("util");

const connection = mysql.createConnection({
  host: "localhost",
  port: "3406",
  user: "root",
  password: "password",
  database: "GradePredictorDB",
});

router.post("/", (req, res, next) => {
  // let id = req.body.id;
  // let curr = parseInt(req.body.curr);
  // let resSem = parseInt(req.body.resSem);
  // let leeway = parseInt(req.body.leeway);
  // let percent = parseFloat(req.body.percent);
  let query = req.body.query;
  connection.query(query, (err, results, fields) => {
    if (err) throw err;
    // let s = "";
    // Object.keys(results).forEach((key) => {
    //   s = s.concat(JSON.stringify(results[key]) + "\n");
    // });
    // res.send(s);
    res.send(results);
  });
});

module.exports = router;
```

```
const connection = mysql.createConnection({
  host: "localhost",
  port: "3406",
  user: "root",
  password: "password",
  database: "GradePredictorDB",
});
```

The server(node.js express) directly uses the root user of mySql Database, and passes sql queries to the DB. It's very dangerous. The correct way is to expose APIs for different tasks (queries) instead of sending SQL from the front end.

```
let sql = `SELECT * FROM Grades WHERE termsFromStudentStart > ${termsFromStudentStart} AND termsFromStudentStart <= ${
  termsFromStudentStart + resultSemester
} AND studentID IN \n (SELECT studentID FROM Students WHERE \n`;
if (completed) {
  sql += `allMandatory = true\n`;
} else {
  sql += `(true)\n`; //This is here so that the following line can start with "AND".
}
let grades = currentStudent.grades;
for (let i = 0; i < grades.length; i++) {
  let grade = grades[i];
  if (grade.termsFromStudentStart > termsFromStudentStart) continue;
  if (mandatory && !MANDATORYCOURSES.includes(grade.catalogNumber))
    continue;
  sql +=
    `AND EXISTS (SELECT 1 FROM Grades WHERE Grades.studentID = Students.studentID AND catalogNumber = ${grade.catalogNumber} ` +
    `AND ABS(termsFromStudentStart - ${grade.termsFromStudentStart}) <= ${semLeeway} ` +
    `AND ABS(parseGrade('${grade.grade}') - parseGrade(grade)) <= ${gradeLeeway})\n`;
  // `AND ABS(parseGrade('${grade.grade}') - parseGrade(grade)) / ((parseGrade('${grade.grade}')+parseGrade(grade))/2)` +
  // `<= ${percentDifference}))\n`;
}
sql += `\n) ORDER BY studentID ASC, catalogNumber ASC, term ASC;`;
// setQuery(sql);
sendQuery(sql);
```

```
    );
  case 1:
    return (
      // <h3>Machine Learning Algorithm 1 results</h3>
      <DetRatioResults
        style={{ margin: 50 }}
        MANDATORYCOURSES={MANDATORYCOURSES}
        currentStudent={currentStudent}
        termsFromStudentStart={termsFromStudentStart}
      />
    );
  case 2:
    // return <h3>Machine Learning Algorithm 2 results</h3>;
    return (
      <BestTraceResults
        style={{ margin: 50 }}
        MANDATORYCOURSES={MANDATORYCOURSES}
        currentStudent={currentStudent}
        termsFromStudentStart={termsFromStudentStart}
      />
    );
```

calculation tasks need to be done in back-end.

There might be other ML algorithms available
Add new RNN to predict when to take the course and what grade may the student get.

Technologies:

Frontend:
- React.js

Backend:
- Spring Boot + mySQL (maven)

Testing:
- Postman, JUnit, Mockito

Deployment:
- Docker
- HAproxy for load balance

Questions & Answers:

1. if you look at the diagrams, the node.js is serving as the "backend" in the new architecture. so, what's new? eg, the first use case after refactor looks pretty similar to the old design.

   In current design, Node js is not a “back-end”. It is only responsible for forwarding a SQL query to the database. In other words, **anyone could send the Node js server a SQL query and Node js will forward the query to the database.**

   So the issue here is, back-end is giving the permission to manipulate DB to front-end. This is a big security issue.

2. you have an attacker scenario, which does not happen in the current architecture because there is no internet connection, all the elements are in a single local computer. did you mean this attack would happen if each piece was on a different server on the network? why couldn't this attack also happen in the new architecture

   We are considering deploying the application. I agree the attack is not likely to happen on a local machine. But when we deploy the application to servers, **the attack is very likely to happen.** The attacker will know the location of the server and send SQL queries to it. The server, however, will always forward these queries to the DB and send back data to the attacker.

   In the new design, this will never happen because **it is the back-end that manipulates the DB**. The front-end will hit RESTful API endpoints provided by back-end and the back-end will send back the data. That’s the data flow. **There is no way for the attacker to pass in a SQL query or to hit an endpoint that does not exist.**

   Here is an example. The attacker wants all data from the database. Probably he will hit “{hostname}/getall”? However, our back-end will never provide this API for requests, so the attacker will only receive a HTTP 400 Bad Request response.

3. you show a 2nd use case, which could also happen with the old design, but we do not want that. we want it to be automatically downloaded from the PeopleSoft database.

   First, in the old design every time we need to input ALL data, refer to Nathan. So this cannot happen.

   Sure, we do not want that. We want everything to happen automatically. However, integrating with PeopleSoft will not be the first thing I need to do. After finishing the main components (reaching Milestone 1), I will work on the integration. However, I am not sure if it is possible (I am not familiar with PeopleSoft mechanism). So I

cannot promise I am able to finish it at this time. What I can do is to let the back-end support the integration first.

Then in Milestone 2, I will work on the integration. In my expectation it won't be very complex.

4. what is the advantages of your backend (2 technologies) over node.js?

   This is a good question (comparing Spring Boot with Node.js). I always believe which technology to use is a minor thing. However, current very limited work has been done in the Node.js server (please refer to the first picture in this document).

   The server **only forwards the sql queries**. So **I am going to build up all services from scratch. So I don't think we need to stick with Node.js.**

   There will be a lot of reasons I prefer Spring Boot and Java for this project. A lot of discussions have been addressed on the internet.
   https://www.quora.com/What-are-the-advantages-of-Spring-Boot-over-Node-js-for-a-RESTful-web-service

   Actually, only considering the choice of the language for back-end between Java and Javascript, we need to choose Java. It is easier to maintain since it is a statically-typed language.

   Scripts written in dynamically-typed languages (like Javascript) can compile even if they contain errors that will prevent the script from running properly (if at all). If a script written in a statically-typed language (such as Java) contains errors, it will fail to compile until the errors have been fixed.

   Also, Nathan is proficient in Java. So it will be convenient for both of us to collaborate, if needed.
   Redis is an in-memory database. I will use it to reduce Database I/O. This is a very common solution for this issue. Although as you said, you didn't see there is a bottle-neck for now, we still need to consider the long run. New data comes in every semester. This will be a highlight of this project.

5. last slide is a distributed architecture. is this what you want the end project to be?

   Deployment will be included in my project. So yes, at the end of the semester the application will be deploy to servers in this distributed fashion.

	Old Design	New Design
Security	front-end manipulates DB, very dangerous.	back-end manipulates DB, only gives the user the data we want to show.
Scalability	All work is done in front-end	separated back-end and front-end. easy to scale by

		running more back-end nodes.
Extendability	require more time down the road to add features	able to create another microservice for future features without touching current code
Convenience	need to install on every user's computer to use	users can use it online
Integration	N/A	Could integrate with PeopleSoft, Student Center

1. can you give me examples of security issues? is what you describe in 2 an example?

I think I have included it in the workflow diagrams. If it is not clear we can talk about it in a meeting.

DM 1/31 i guess we will have to talk in a meeting.

Hi Professor, I presented an example in the recording, Hope that one is clear.

2. client can still ask for "all students in 2001" then "all students in 2002" etc. and get all the data. the issue i see is authentication and access control, unless i'm missing something.

No, client cannot because back-end will not provide these kind of API endpoints. HTTP requests are not SQL queries.

DM 1/31 i think we have to define what the FE is doing now, and what the FE will do in the future. maybe we're talking about the same thing. we cannot talk about FE before and FE after, because i assume that whatever is there in the FE now will be put into a server that can be accessible through the web. maybe that's what you mean. in that case, only what is offered by the UI will be accessible, and now, since the user has all the data, it's irrelevant what the UI offers. is that what you mean? it's hard to understand what the issue is.

Firstly, I assume no users will have the original data (xls file), because we want to get rid of manual initialization. Is that right? After deployment, all data is stored in DB. We are discussing potential attacks after deployment, not on a local machine.

When no user has the original data, if the user has the ability to get all data (pass in a SQL query), then it is an issue.

Hope this is clear.

3. (did you revise Milestone 1 after we talked? i thought Nathan said ML will happen offline.) during milestone 1, you said you'll deploy to a local machine, which is what the current status is. the project will need external integration.

ML happened offline. But at the end of the day we need back-end to get data from DB. So it still needs this integration.

I need to develop on my computer first and then deploy to the servers. There is no cloud resources for me to implement CI/CD.

DM 1/31 ok, i think we agree. we need to migrate to a server, and therefore there will be external integration. correct?

Yes, this is correct. External integration we can think of now is the integration with PeopleSoft system.

4. so your argument is basically Java vs js. is that right? (i didn't read the quora site, i assume you summarized it. also, i'm sure you can find the opposite on the internet: advantages of noje.js over springBoot. it's the internet! you can find anything there!

This is one perspective. And yes I still don't get the point why we want to stick with Javascript. Spring Boot is the first choice for us so why not to use it? Remember nothing has been done with Node.js so far.

DM 1/31 i looked at the link. not very clear at all why Spring is better than Node. according to them, node offers asynchronous tasks and is faster. note that i don't really care what you use, i want you to reason about it, and not justify by saying "i know java." this is a MS project, right?

I will do it.

5. so your final architecture will include HAProxy, FE, BE, in-memory DB? let's now list all the use cases and see what each part will do.

HAProxy serves as the load balancer.

All features need FE, BE, in-memory DB.

I have addressed this question in the recording, here is the link: https://pitt.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=4b826b6a-ff86-4c6d-b799-acc1002fc6bb

DM 1/31 i have not watched the video.

I don't think the comparison is biased… Since the current design is not even a feasible solution for deployment on servers. If you are concerned about something else, please tell me and I will do the comparison. Thanks.

DM 1/31 as i said one issue is that it is abstract. about biases, the first thing is the security issue discussed above. the second is scalability, and Spring is slower than Node. 3: extensibility is not clear. 4 and 5 are clear.

For scalability, I think "Spring Boot vs Node" does not equal to "New design vs Old design". I will reason about why we want to choose Spring Boot.

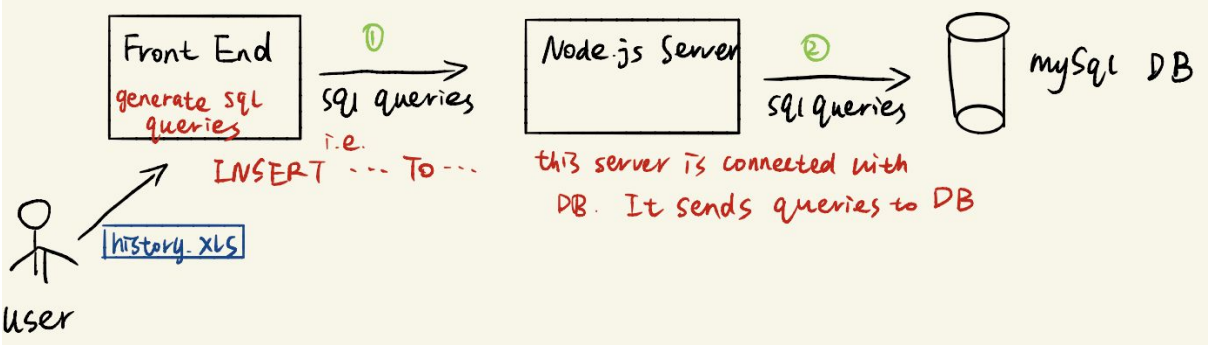
For extensibility, I have talked about it in the recording.

Use Cases Comparison (old design vs new design):

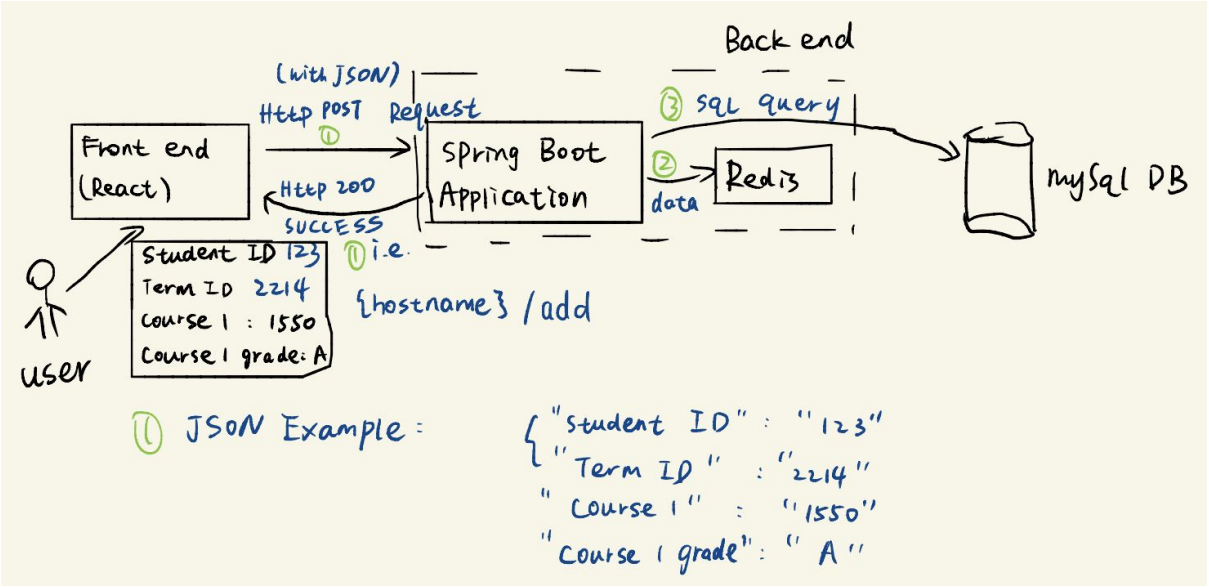
1. **Termly data synchronization (aka Add New Data)**

Description:
We want to add new data to our database after each semester.

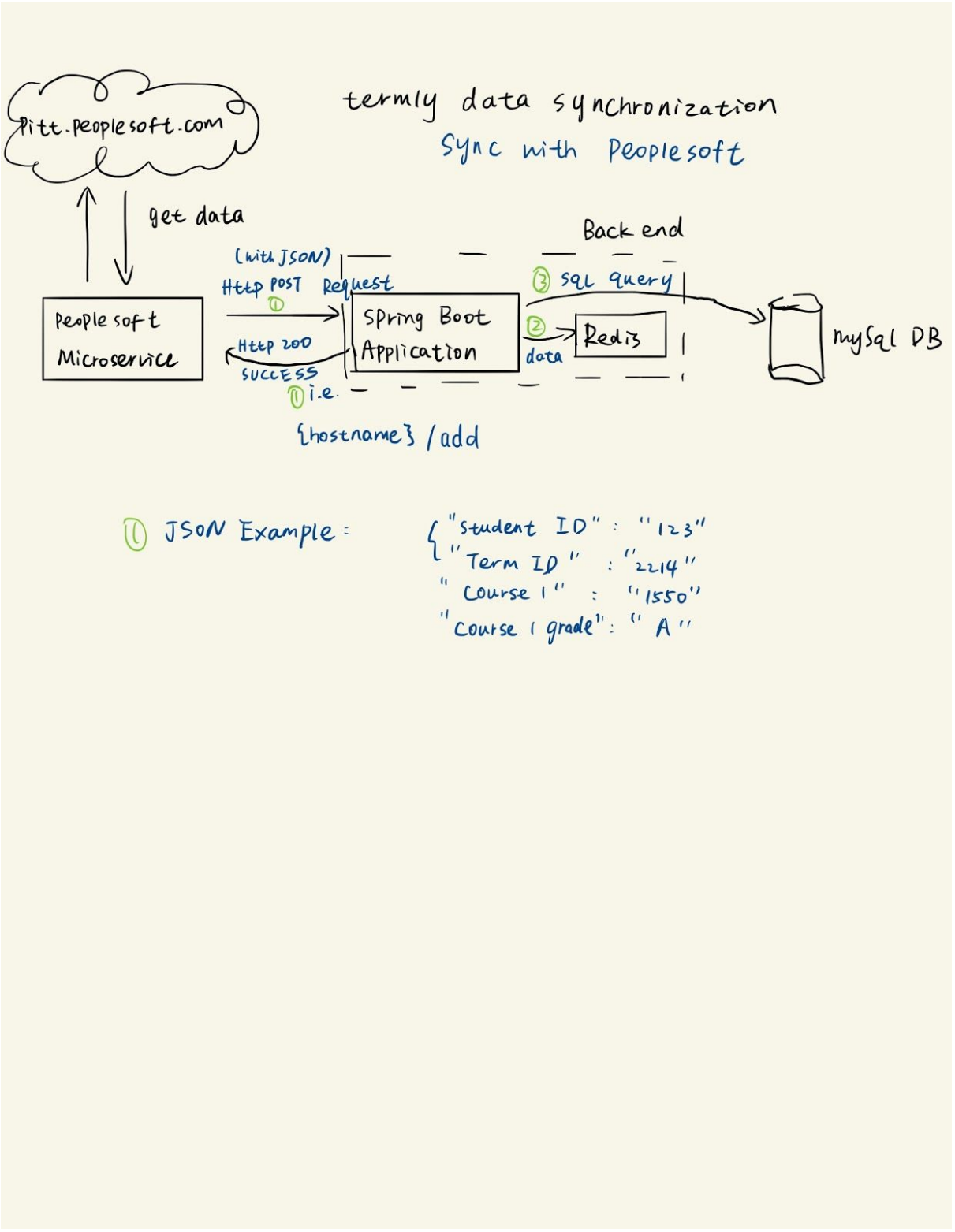
Current Flow:



Flow After Refactor:



If we want to integrate with PeopleSoft:

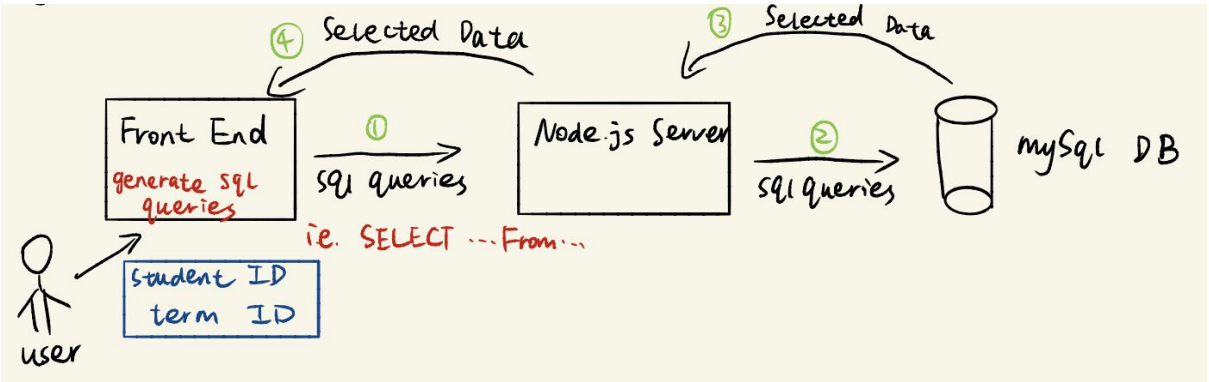


2. **Student Grade Prediction (advisor version)**

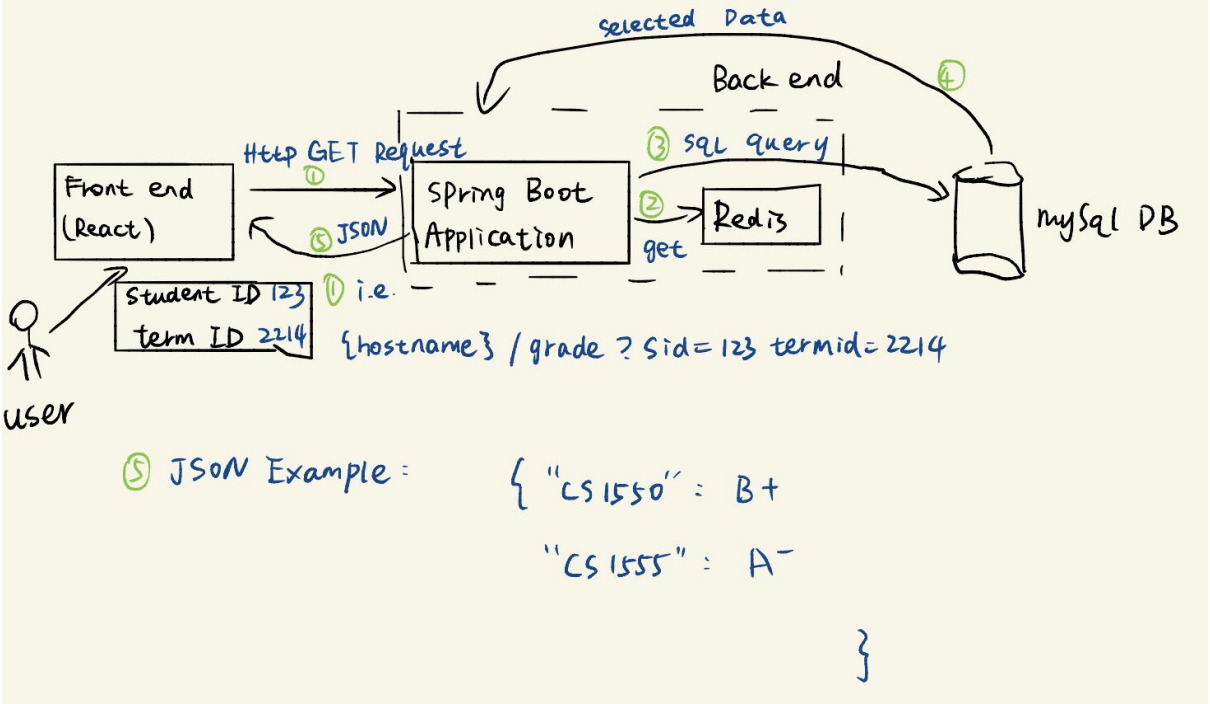
Description:

Given the student ID and current term ID, plus search settings (number of semesters to look into the future, number of semesters leeway, grade leeway by percent), returns back the grade prediction on all available CS courses.

Current Flow:



Flow After Refactor:

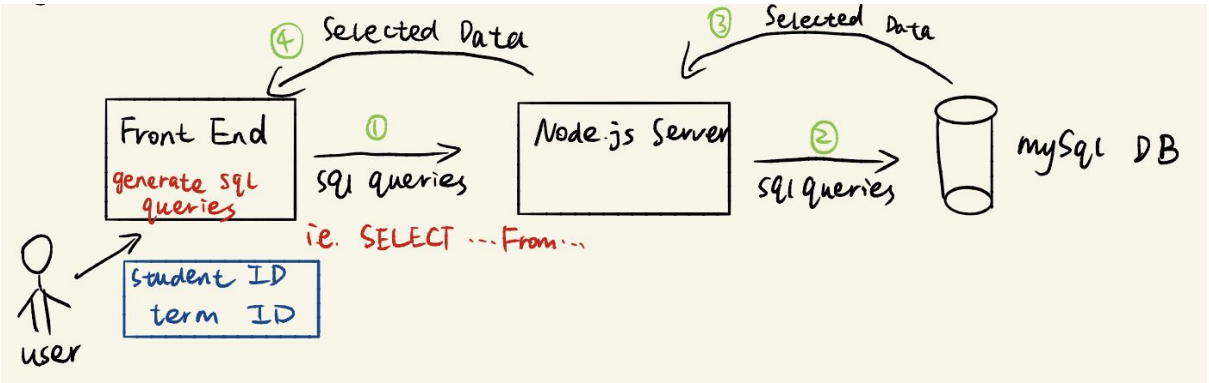


3. **Student Grade Prediction with ML (advisor version)**

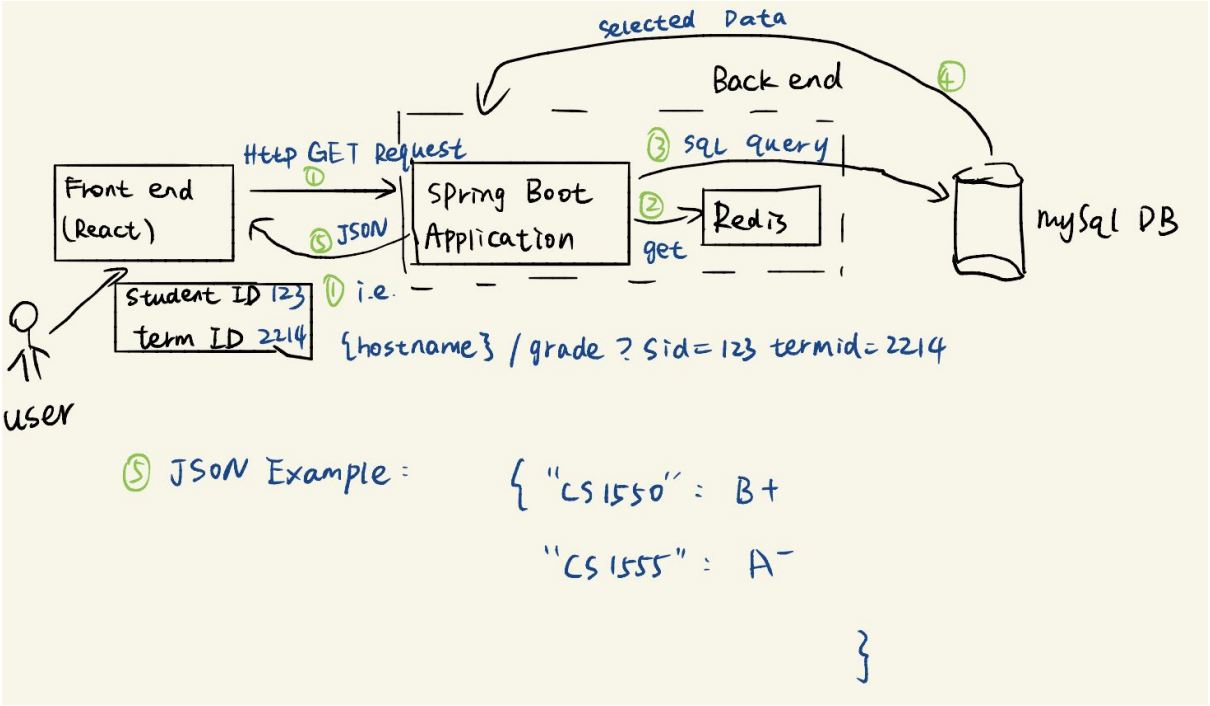
Description:

Given the student ID and current term ID, plus search settings (number of semesters to look into the future, number of semesters leeway, grade leeway by percent), **match the student with existing ML clusters,** returns back the grade prediction on all available CS courses.

Current Flow:



Flow After Refactor:



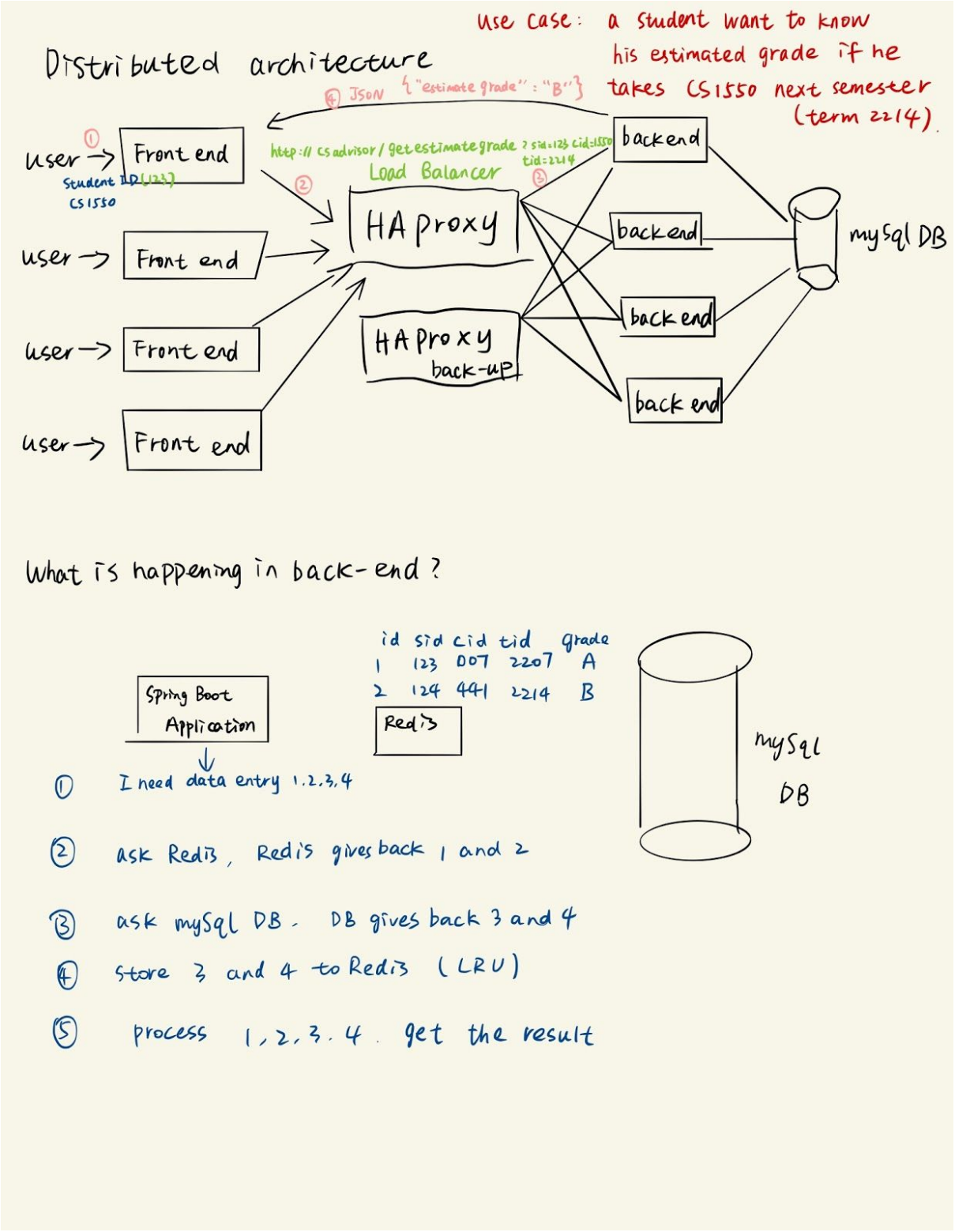
Note: the data flow is the same, the difference is how does back-end consume the data.

4. **Student Grade Prediction (student version)** (new feature proposed)

Description:
Given the student ID, current term ID and course ID. Returns the grade prediction of this student taking the course next semester. Would be very helpful for students before the registration period.

Current Flow: N/A

Flow After Refactor:



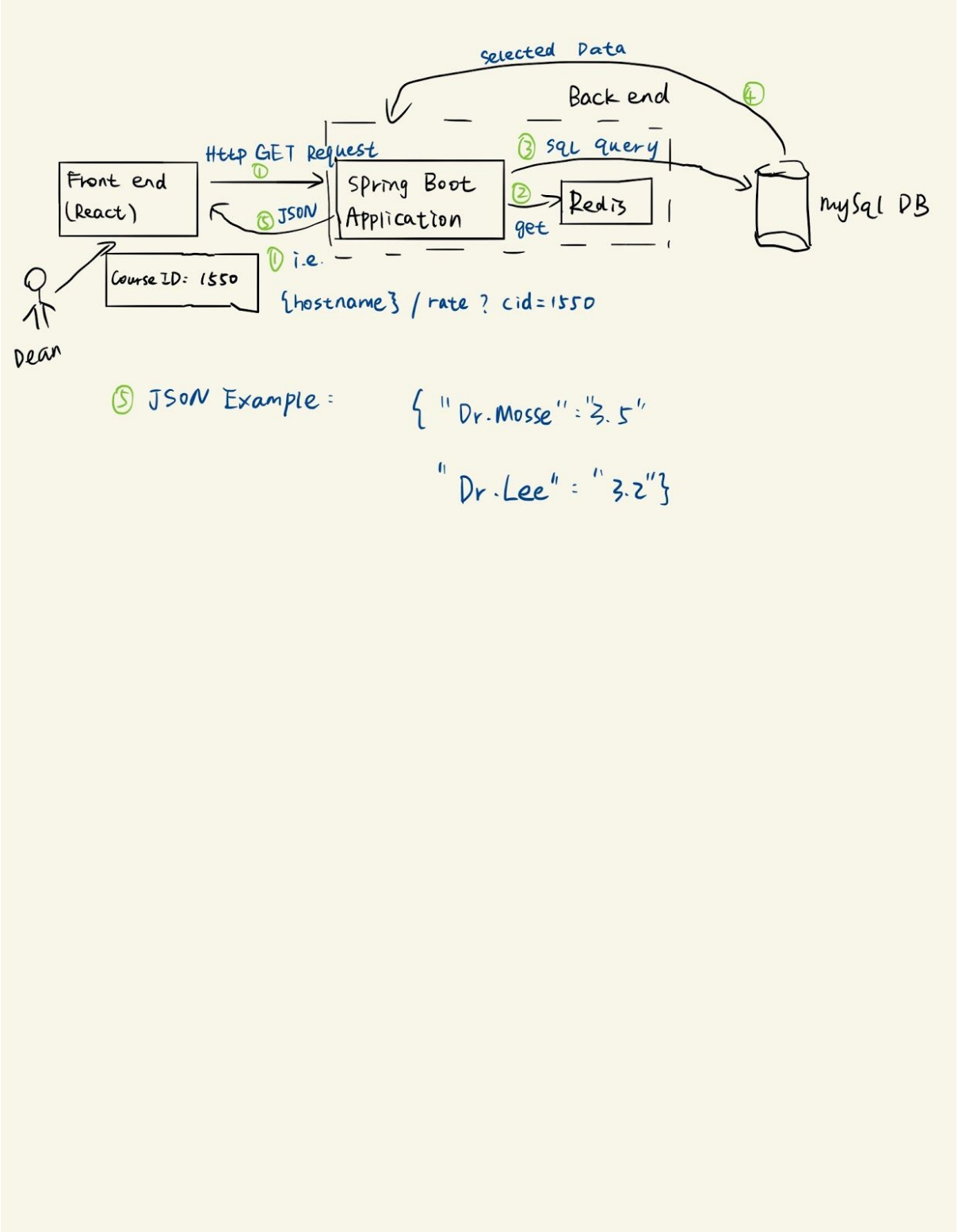
5. **Professor Teaching Performance Analyzer** (new feature proposed)

Description:

Provide this feature to the department administrators. Analyze each professor's teaching performance on different CS courses. Given the course name, return the average student grade taught by each professor.

Current Flow: N/A

Flow After Refactor:



6. I have come up with a lot of ideas… We could integrate SCI Courses site with the app, so that only shows available courses to the user (i.e. not every semester has a CS2550 section) We could also set a higher weight for courses that have to be taken to graduate, so that providing more “accurate” recommandation.

Milestone 1 (end of March):
Back-end:
finished feature 1, 2, 3, 4 (committed) and 5 (bonus)

finished back-end support (API endpoints) for peoplesoft integration.

Front-end:
Refactor the website to make it look like other pitt websites. Front-end will be still using React. However, I will re-structured all layers and create new pages, to improve user experience. Since the current way to present data is clear and intuitive, I will keep the current data lay-out method.

Deployment: during this development phase I will run the application on my local machine

Milestone 2 (end of the semester):

Integrate Jiaye’s ML algorithm with back-end:
Jiaye wants to try some RNN approaches. He will train the model and provide me with a pre-trained model. I will keep the pre-trained model in the back-end and use it to process requests.

peopleSoft integration microservice (I cannot commit to this, because I am not sure if it is possible and how complex the system is). But I will try my best, and I think I am likely to be able to finish it.

deploy to pitt server, open for usage, testing.

I am not able to integrate the application with Pitt SSO. Instead I will create an accounting service in the application to support users registration and login. This will be a good start and if in future we want to integrate with Pitt SSO, we can start from here.

The reason is, there is already too much work. In my opinion, integration with peopleSoft is more important, so I will take that as a priority.

Node.js, Spring Boot Comparison: Pros and Cons

Pros:

Node.js	Spring Boot
Javascript based, a rapidly growing community	Java based, a mature and thriving community
Great for I/O tasks	Support for multi-threading
lightweight	**Many easy-to-use dependencies: Spring Data, Spring Cloud...**
	Java is statically-typed
	Maintainability and long term support

multi-threading will increase I/O capability. One use case is writing tons of data to the database (which is likely to happen in termly data synchronization).

Cons:

Node.js	Spring Boot
Does not support multi-threading	high memory utilization
lack of strict type checking (runtime problems)	may include unused dependencies, makes deployment binary file size grow big. Not likely to be an issue in this application.
Not suitable for heavy computing tasks, performance bottlenecks	

Spring Data, Spring Cloud and Spring Web exactly meet this project's need. So there won't be a lot of unused dependencies.

My way to train a model to predict grade of a student in x semester:

1. **train_x** should at least include *semester #*, *course #*, **train_y** is the *grade*.
2. can use LSTM or use dense layers only with a lookback window, don't sure which one will give a better result.

3. if more information about the student can be given, the prediction should be more accurate, but that may cause discrimination problem.

Toolkit for ML:

Tensorflow

output:

a training script

a tensorflow model file (which can be load on the web using tensorflow.js and do prediction in the browser without leaking any information to the user or getting any information to the user)

a web service that simply using the model to do prediction and return the result