IRS v2

Jessica Bishop, Jason Chan,
Sukhmai Kapur, Harrison Li

Motivation

- Data visualization and analysis tool for instructors
- Create web application
  - Understanding full stack web development
  - Learn modern frameworks

Goals

- Dynamically retrieve data from database
  - No retrieving data from JSON files
- Provide meaningful analysis

Technical Stack

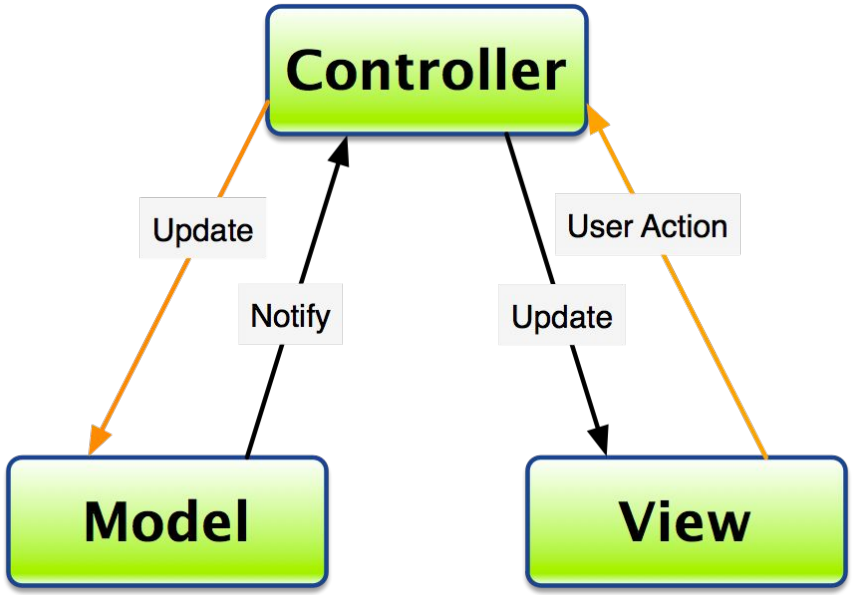
MVC Model

Model - The software that affects the data

Controller - The bridge between the data and the view

View - What the user sees

Each part is modular and can be exchanged

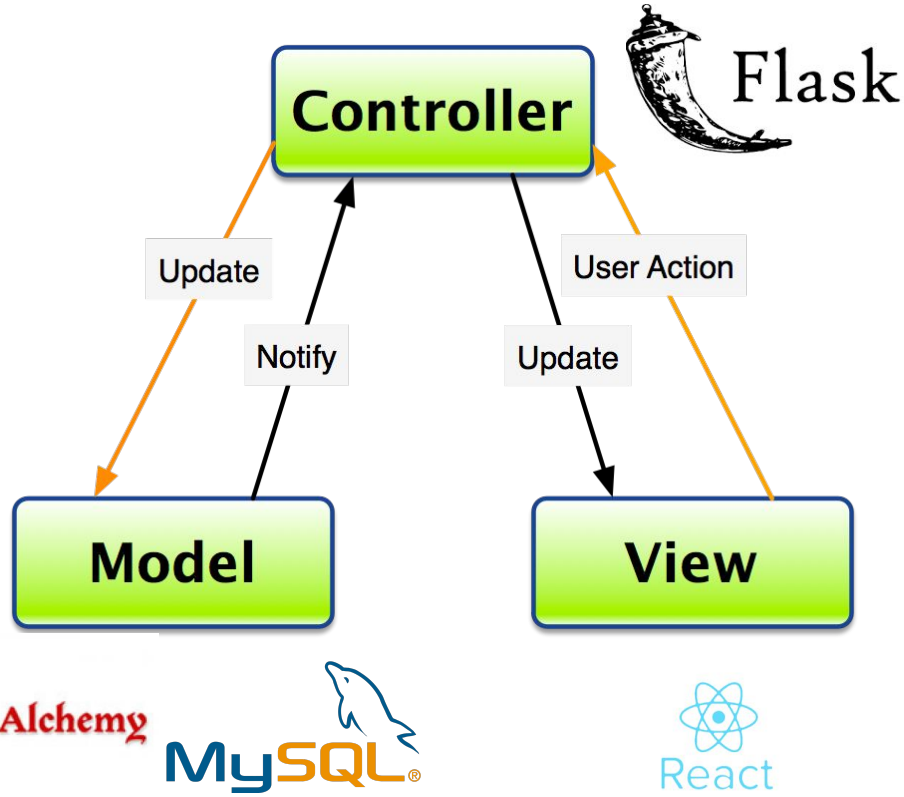


Technical Stack cont.

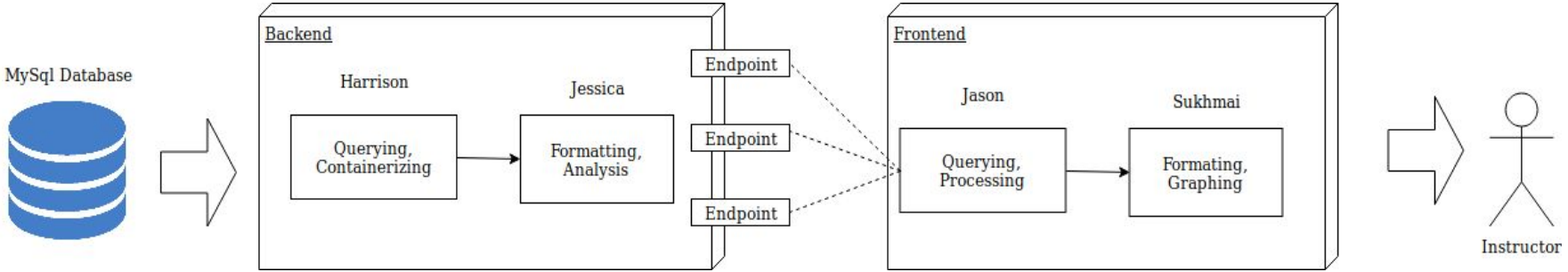
Model - Mysql, SQLAlchemy

Controller - Flask

View - React

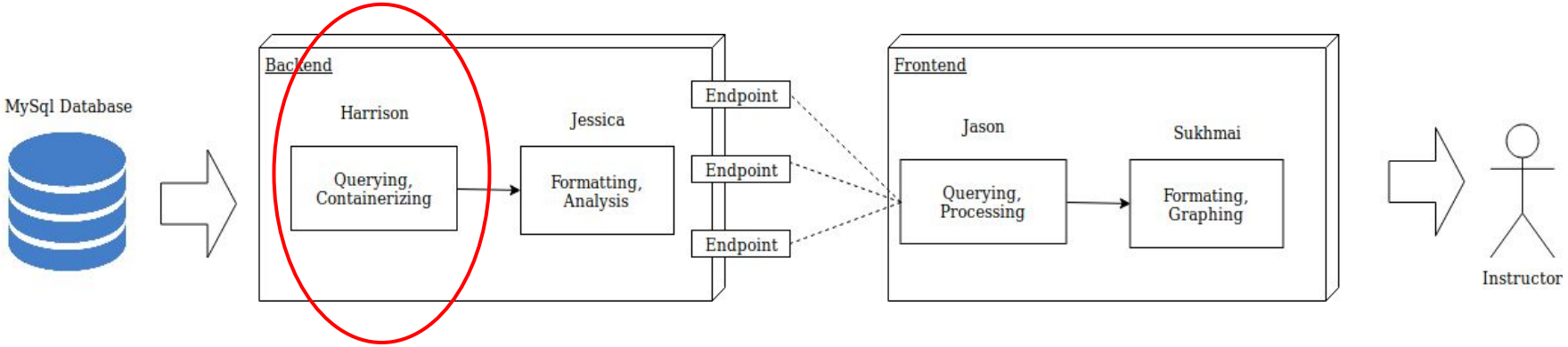


Data Stream Diagram



Harrison

- Data layer
- Data object abstractions



Data Layer

- Abstraction level that centralizes any logics related to obtaining data
- Implemented using SQLAlchemy (ORM - Object Relational Mapping)
  - Picked SQLAlchemy as it is flexible. Very small changes have to be made if the underlying database changes (i.e PostgreSQL vs. MySQL)
  - Allows the developer to think about database and tables in an object oriented way.
  - Presumably better security versus SQL injection attacks. Relying on the SQLAlchemy interface and not using raw SQL adds a level of security
- Focused on making one method that queried all relevant question data by student so that this generic method could retrieve data for many different types of endpoint / analysis requests.
  - **Pro:** Minimized amount of code written
  - **Con:** some analysis requests probably took much longer than if there was a more tailored method.

Data Object Abstractions

StudentDataContainer

+ Student ID

+ Semester of student ID

- Dictionary whose keys were question IDs, and values were list of dictionaries

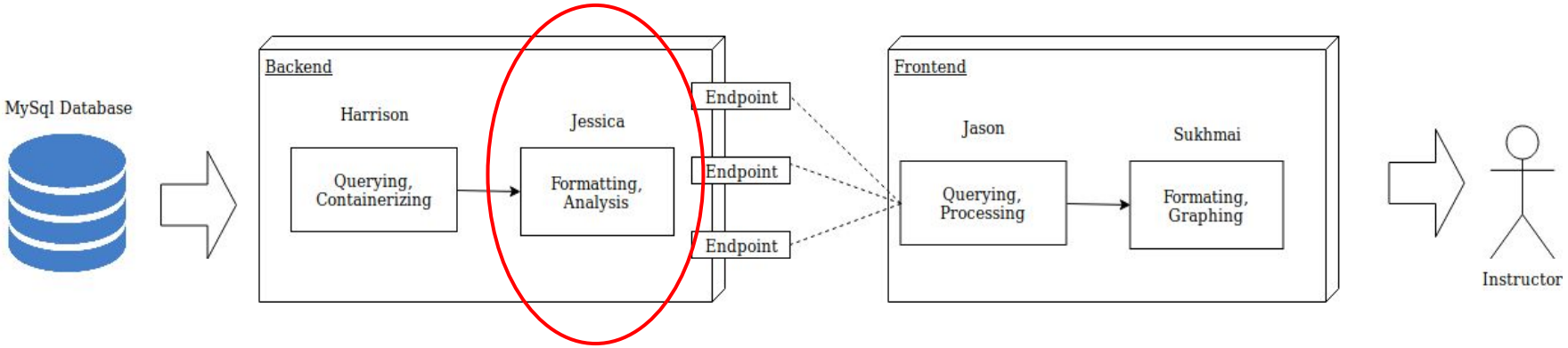
+ Various get methods that held all logic of how to traverse private dictionary and any preliminary summarization that required intimate logic of how data was stored. Put it all in this object for proper abstraction

```
{ Question_id1 :
    {"interfactions": [QuestionInteractionDictionary1,
                       QuestionInteractionDictionary2, ...]
    },
 Question_id2 :
    {"interfactions": [QuestionInteractionDictionary1,
                       QuestionInteractionDictionary2, ...]
    }
  .
  .
  .
}
```

- Instantiated in the Data layer class when querying the database for data
- Allows all clients of data layer not to need to know specifics of how data is organized, just need to know the interface of this object

Jessica

- Data analysis
- Endpoints



Data analysis

- Harrison's backend provided me with a List of Student Data Container objects
- Allowed me to call specific methods on each of the student's data:
  - .get_question_score() returns a dictionary of question ID's and that students respective score
  - .get_all_durations() returns a dictionary of question ID's and the time that student took to answer that question
  - .get_all_question_ids() returns all the questions that the student interacted with
  - ._number_interactions() returns the number of times a student interacted with a question
  - ._times_skipped() returns the number of times a student skipped a question
- I used these methods to run my analysis and organize the data

Pre-test vs Test

- Pre-test: first time the student answered the question (Chapters 1 - 7)
  - Pass in False for test_question parameter in student data container methods
- Test: second time the student answered the question (Chapter 8)
  - Pass in True for test_question parameter in student data container methods
- The Analysis methods return a dictionary with two key values (pre-test and test) and the values as the corresponding data requested

getOverallMean()

{'test': {'mean': <mean score of all test scores>, 'count': number of questions used to calculate mean}, 'pre-test': {'mean': <mean score of all pre-test scores>, 'count': number of questions used to calculate mean} }

- Returns the overall mean of all questions over a given range of semesters

getMeanScorePerQuestion()

{'test': {questionID: averageScore},
 'pre-test': {questionID: averageScore}}

- Returns the average score for each questionID for a range of semesters

getListOfDurationPerQuestionID()

{'test': {questionID: list of durations},
 'pre-test': {questionID: list of durations}}

- Returns a list of durations for each question ID for a range of semesters

getMeanDurationPerQuestionID()

{'test': {questionID: mean duration},
 'pre-test': {questionID: mean duration}}

- Returns a mean duration it took for all of the students in a semester for each questionID

percentOfQuestionSkipped()

{'test': {questionID: percent of question skipped},
'pre-test': {questionID: percent of question skipped}}

- Returns the percentage of each questionID that was skipped for a range of semesters

```
{'test': {931: 78.58672376873662, 156: 49.68287526427061, 2181: 98.4251968503937, 3161: 60.879120879120876, 855: 62.365591397849464, 364: 90.5334
8387096774194, 1190: 60.55979643765903, 3164: 77.20739219712526, 990: 69.08315565031982, 467: 39.0625, 445: 97.58064516129032, 866: 70.0214132762
825, 722: 86.73949579831933, 3165: 91.12271540469973, 785: 81.42548596112312, 1191: 86.39798488664988, 816: 89.95901639344262, 898: 73.0848861283
6866953, 122: 96.7935871743487, 3166: 95.75371549893843, 1194: 73.95577395577395, 919: 71.1453744493392, 3226: 75.70149253731343, 863: 66.5551943
634920635, 17: 82.5, 582: 84.31001890359168, 1193: 81.19658119658119, 3577: 54.60251046025105, 390: 76.4080459770115, 112: 52.522935779816514, 35
818181818182, 3179: 33.6, 3169: 75.1592356687898, 424: 80.78895463510848, 3168: 82.32758620689656, 3167: 82.45243128964059, 335: 62.8235294117647
1446, 3174: 76.53508771929825, 423: 29.847494553376908, 1081: 86.92, 530: 58.78661087866109, 57: 78.66108786610879, 410: 77.77777777777777, 3175:
 790: 80.38760511627926, 554: 61.137443364928984, 886: 40.24640657084189, 740: 67.68151053864193, 672: 66.96581196581197, 316: 80.68833652007648,
36, 3585: 39.49579831932773, 2125: 88.15426997245179, 414: 33.76623376623377, 653: 25.34435261707989, 2126: 72.70114942528735, 1026: 86.833333333
6229508197, 3587: 35.44973544973545, 145: 54.54545454545455, 3586: 54.26008968609865, 484: 60.50420168067227, 3308: 18.72146118721461, 1200: 57.8
41046832, 1196: 84.86486486486487, 49: 94.0312195592286, 220: 44.03292181069959, 248: 93.52861035422343, 385: 82.63975155279503, 3332: 75.2895752
84375, 3305: 20.416666666666668, 3302: 31.428571428571427, 1201: 62.35632183908046, 696: 92.44186046511628, 3303: 68.72659176029963, 587: 64.7286
25, 3543: 52.32919254658385, 705: 9.923664122137405, 3317: 34.166666666666664, 5: 63.278688524590166, 516: 87.8698224852071, 680: 61.333333333333
448844885, 3314: 59.451219512195124, 3549: 53.43511450381679, 1203: 92.57142857142857, 3312: 17.60299625468165, 3588: 62.234042553191486, 2185: 5
039215686274, 567: 83.28075709779179, 3551: 31.862745098039216, 3333: 14.117647058823529, 982: 62.30769230769231, 4: 91.56346749226006, 1207: 89.
82: 91.8840579710145, 3589: 32.76836158192091, 2186: 61.67664670658683, 3313: 33.333333333333336, 964: 62.06896551724138, 3548: 55.76923076923077
39344262295, 3328: 32.1285140562249, 3310: 25.110132158590307, 658: 76.33440514469453, 278: 73.17073170731707, 40: 96.96969696969697, 437: 63.664
629441624365484, 1084: 0.0, 775: 78.57142857142857, 339: 60.83018867924528, 51: 78.46441947565543, 3319: 77.61732851985559, 2189: 49.6350364963650
103896103, 2193: 80.42704626334519, 3320: 77.5, 2199: 70.22653721682848, 2190: 60.4, 1093: 95.59956171617164, 633: 98.43924214046821, 148: 87.692
.329588014981276, 2207: 95.43973941368078, 2192: 68.04511278195488, 2208: 79.12457912457913, 1208: 90.23569023569024, 27: 78.78787878787878, 247:
 49.11032028469751, 2216: 57.62081784386617, 1091: 89.93055555555556, 2209: 93.77049180327869, 124: 77.15355805243446, 756: 94.56869009584665, 22
063, 3556: 37.786259541984734, 1211: 85.71429285714301, 1098: 64.98516320474778, 120: 77.20364741641338, 164: 57.19844357976654, 1045: 65.0349650
762711864407, 3607: 54.651162790697676, 1212: 49.290780141843975, 644: 71.98227109634543, 1089: 59.872611464968152, 1007: 97.0674486803519, 3594:
606: 59.01639344262295, 3601: 50.33737854251014, 3558: 49.69512195121951, 142: 93.71069182389937, 1099: 69.6319018404908, 361: 95.61128526645768,
95, 1070: 86.68941979522184, 2215: 63.50148367952522, 3599: 37.33031674208145, 1022: 87.98701298701299, 3554: 31.636363636363637, 3605: 56.0, 110
, 1017: 76.5886287625418, 1029: 85.80246913580247, 1217: 84.15841584158416, 3571: 37.05179282868526, 1023: 89.59731543624162, 1086: 82.4074018518
0627177701, 1215: 95.48611111111111, 1076: 93.44262295081967, 1221: 82.20640569395017, 1220: 85.49511924686193, 1058: 57.19178082191781, 3570: 10
89473684, 211: 46.94656488549618, 255: 65.04329004329004, 1043: 91.23376623376623, 276: 55.0, 23: 97.3421926910299, 288: 62.42236614906839, 3199:
9: 0.0, 2240: 0.0, 3180: 45.11494252873563, 885: 33.333333333333336, 3578: 66.57824933687003, 3336: 60.3448275862069, 2211: 82.5503355704698, 215
, 2217: 53.35689045936396, 1072: 76.53061224489795, 3331: 22.926829268292682, 2197: 26.568265682656826, 42: 97.5797587458746, 2161: 94.0594059405
742081447963, 3464: 43.47826086956522, 1060: 86.40939597315436, 1025: 92.95302013422818, 505: 73.13432835820896, 1063: 93.18181818181819, 272: 69
2133: 71.83098591549296, 879: 64.38524590163935, 1216: 84.71760797342192, 1073: 48.41772151898734, 1082: 80.26905829596413, 3318: 34.394904458598
5151516, 1021: 73.07339449541284, 428: 59.90338164251208, 3329: 27.555555555555557, 715: 31.753554502369667, 3561: 45.535714285714285, 3564: 41.1
44: 0.0, 500: 0.0, 501: 0.0, 935: 100.0, 602: 0.0, 41: 0.0, 348: 33.333333333333336, 337: 0.0, 3238: 100.0, 3250: 100.0, 3211: 0.0, 76: 100.0, 67
 3234: 0.0, 555: 50.0, 3327: 0.0, 2142: 50.0, 3563: 0.0, 3560: 83.33333333333333, 378: 50.0, 510: 0.0, 88: 0.0, 3399: 33.333333333333336, 3198: 5
 1004: 100.0, 3268: 50.0, 2131: 50.0, 535: 10.0, 3294: 0.0, 3240: 66.66666666666667, 1152: 100.0, 14: 0.0, 2259: 0.0, 2127: 33.333333333333336, 3
```

Student Methods

- Same functionality: mean question scores, durations, and percents skipped
- Take a student ID as a parameter
- Only returns the data for that specific student

Endpoints

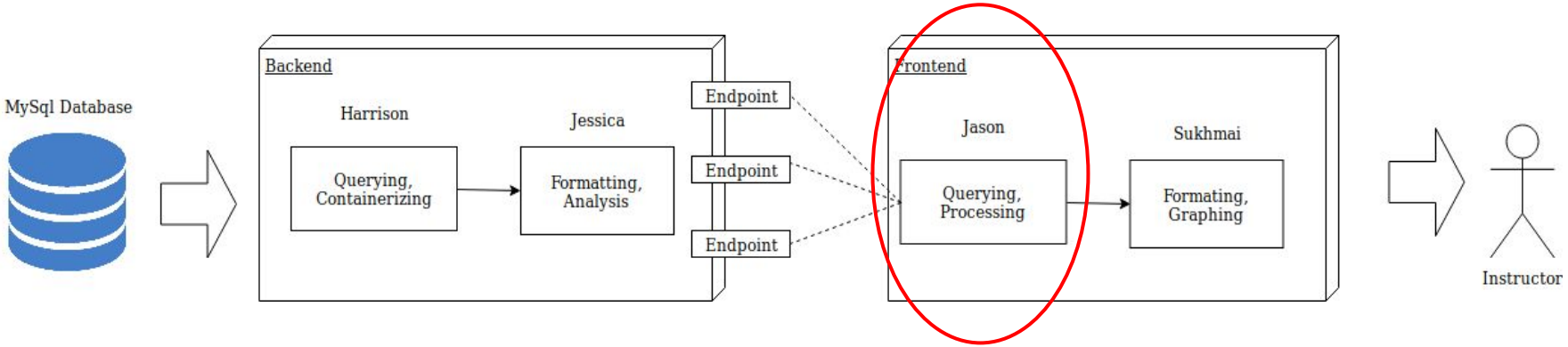
- Allows the front end to make a request to the server to receive the analysed data
  - Each endpoint has a specific URL corresponding to what data will be returned
  - Inside each endpoint method was a call to the analysis methods

'http://127.0.0.1:5000/get_list_of_durations' returns the dictionary of pre-test and test data containing the list of durations per question ID

- **This is the backbone of making our data fetching dynamic**

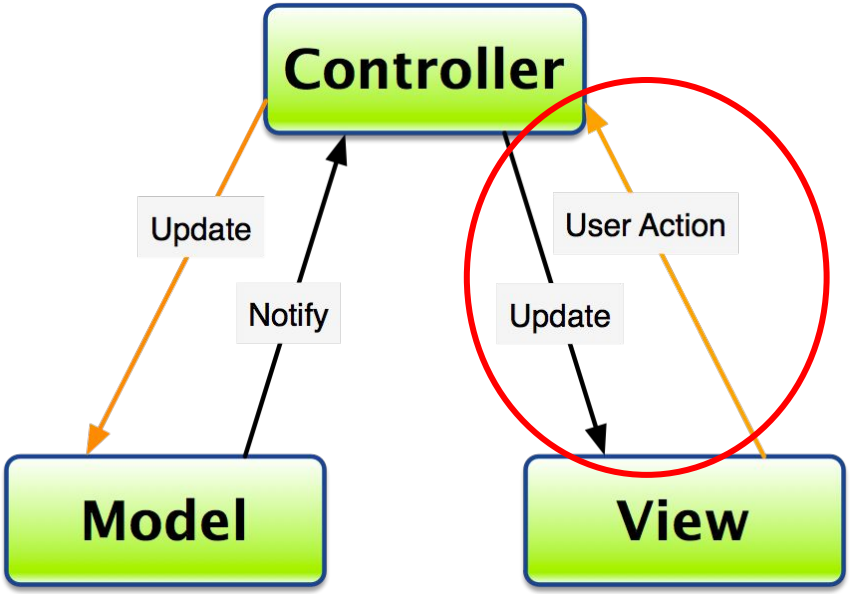
Jason

- Backend Communication
- Data Processing



Querying

- Communicated with backend using HTTP Requests
  - Used nodejs framework axios
    - Simple promise-based framework

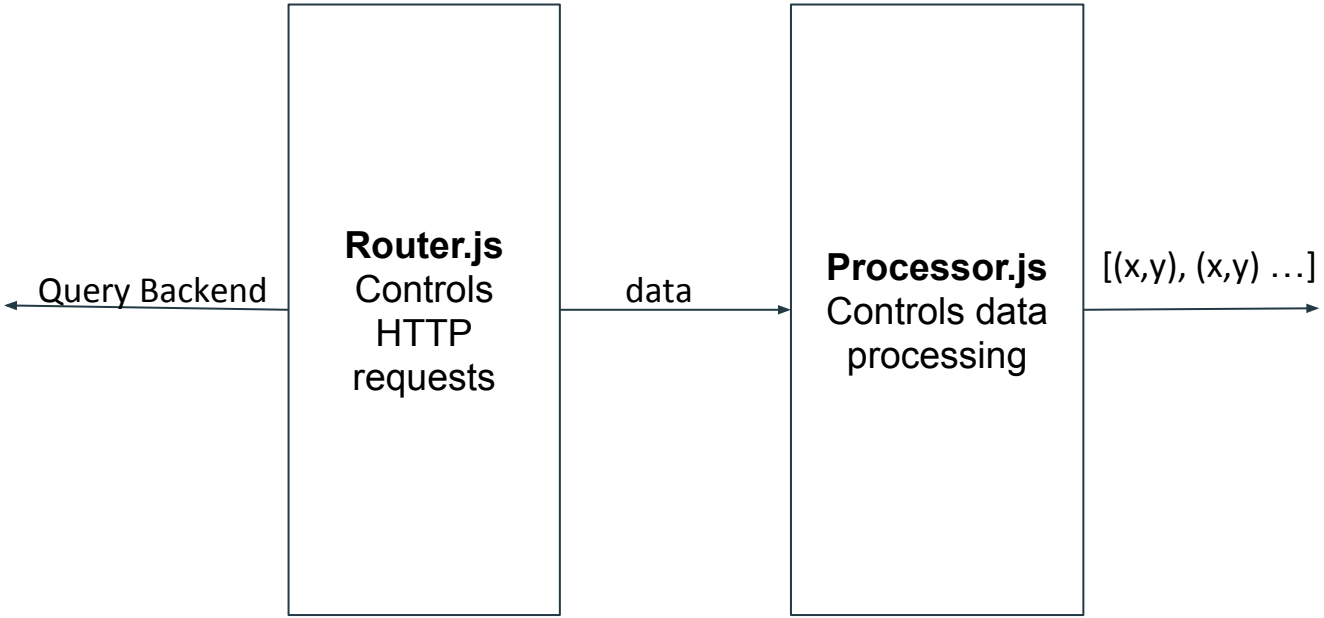


Processing

- Took data from backend and changed it into a list of (x,y) values so that it would be easy to graph

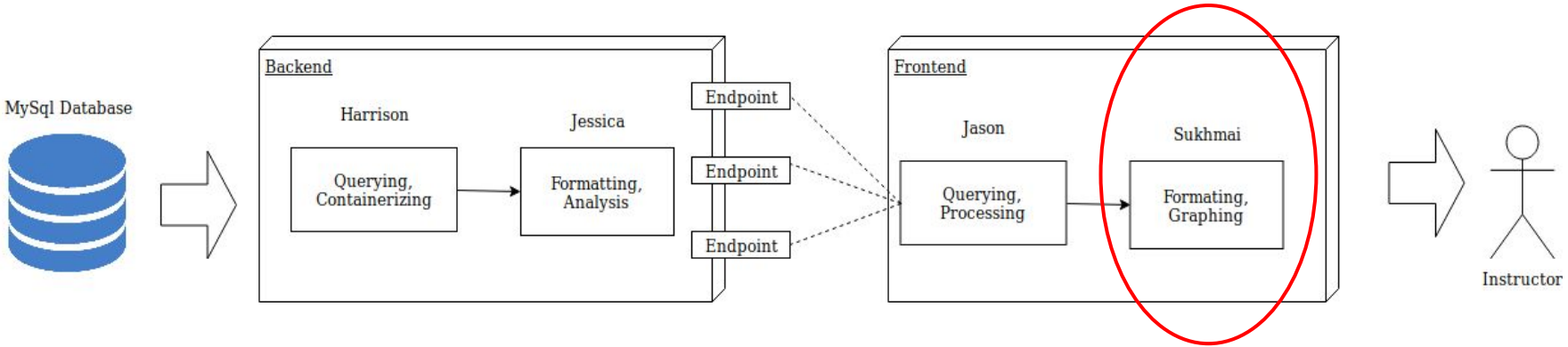
```
export function avgDiffPerCat(params) {
    return getAvgDiffPerCat(params).then(function(data) {
        let x = [];
        let y = [];
        Object.keys(data).forEach(function(topic) {
            x.push(topic);
            y.push(data[topic]);
        });
        return {x: x, y: y};
    });
}
```

Design

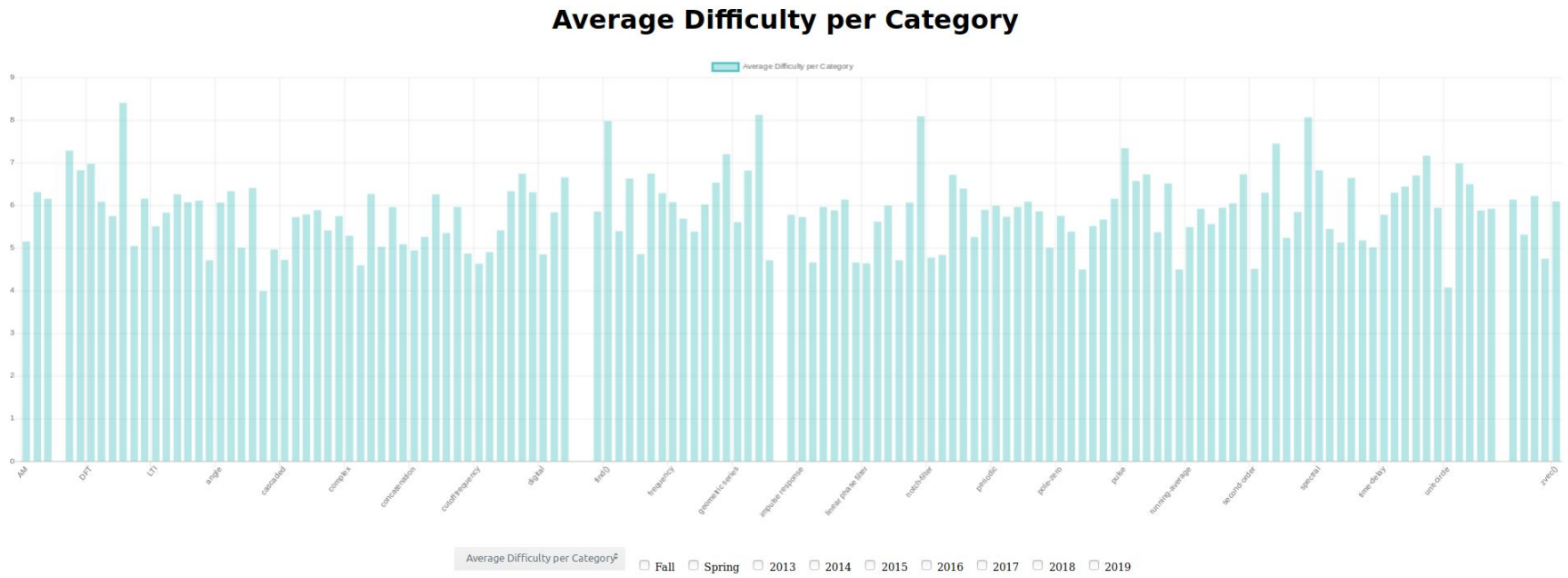

Query Backend
Router.js
Controls
HTTP
requests
data
Processor.js
Controls data
processing
[(x,y), (x,y) …]

Sukhmai

- React
  - Chartjs



Main View


Average Difficulty per Category
Average Difficulty per Category
0
1
2
3
4
5
6
7
8
9
AM
DFT
LTI
angle
cascaded
complex
concatenation
cutoff frequency
digital
find()
frequency
geometric series
impulse response
linear phase filter
notch filter
periodic
pole-zero
pulse
running average
second order
spectral
time delay
unit circle
zvec()
Average Difficulty per Category
Fall
Spring
2013
2014
2015
2016
2017
2018
2019

Using GitHub

- We had a dev branch, a master branch, and each of our own branches
- We would work on our own branches and make pull requests to dev
- Would merge to master during milestones
- Often had to merge from each other's branches

IRS-v2 in the future

- How can IRS-v2 be related to the other projects today?
- For labs and GUIs
  - The labs and GUIs provide the data that we pull and analyse to present to the teachers and TAs
- For Machine Learning
  - We can combine the machine learning with the analysis to provide even more useful information to the teachers and the TAs

Conclusion

- To be improved in future semesters…
  - Improve time it takes to query the database
    - Clean up database
  - Improve efficiency
    - Hit database once and store data in static storage instead of having to hit endpoints every time we want to receive data
  - More Analysis displayed
    - Create filters for specific students
  - Connect to online server instead of local server
  - Improve software development process. Try to follow continuous integration best practices more closely - i.e have smaller feature branches that are more frequently merged back into dev branch and then having other feature branches rebase off of that every day in order to not stray too far from the current state of dev

Demo