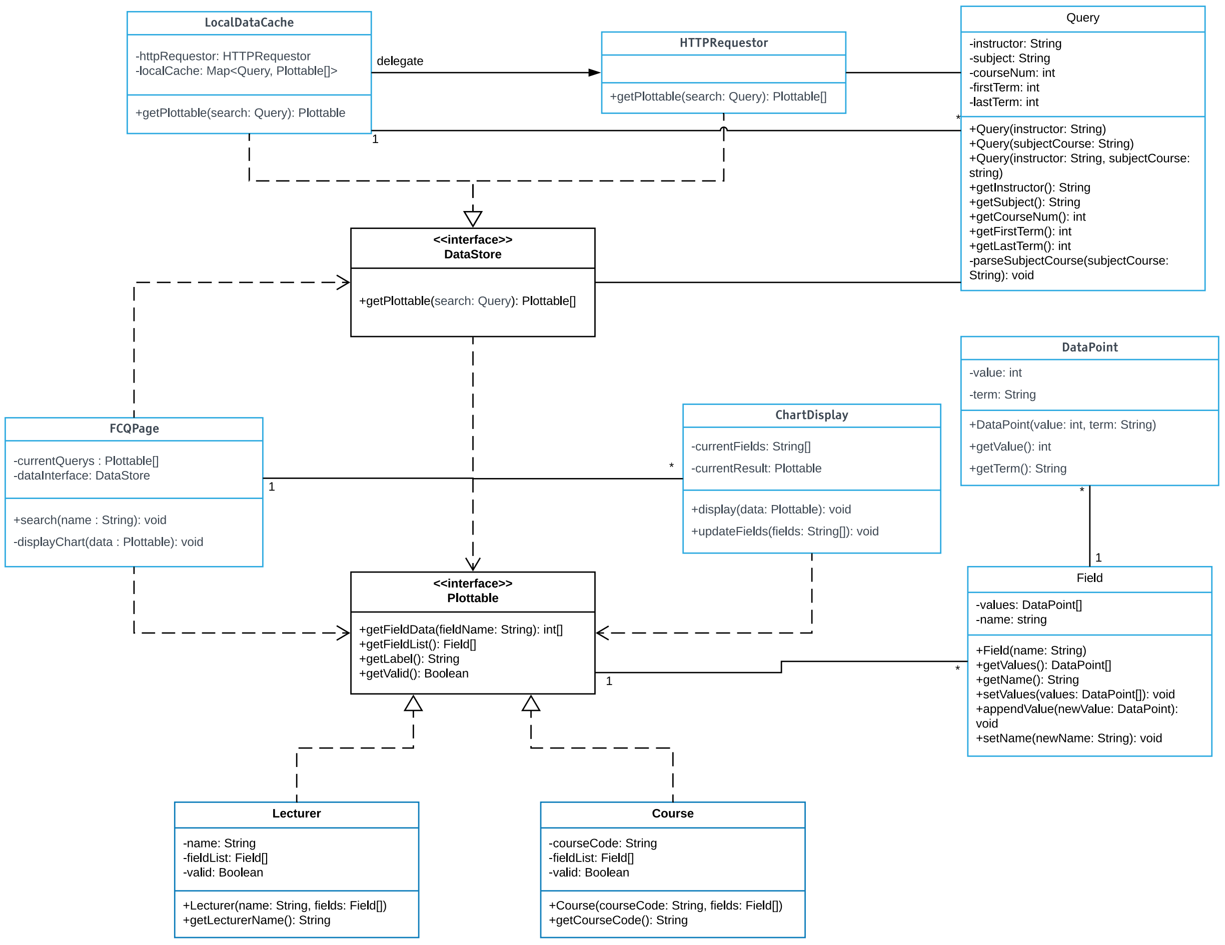
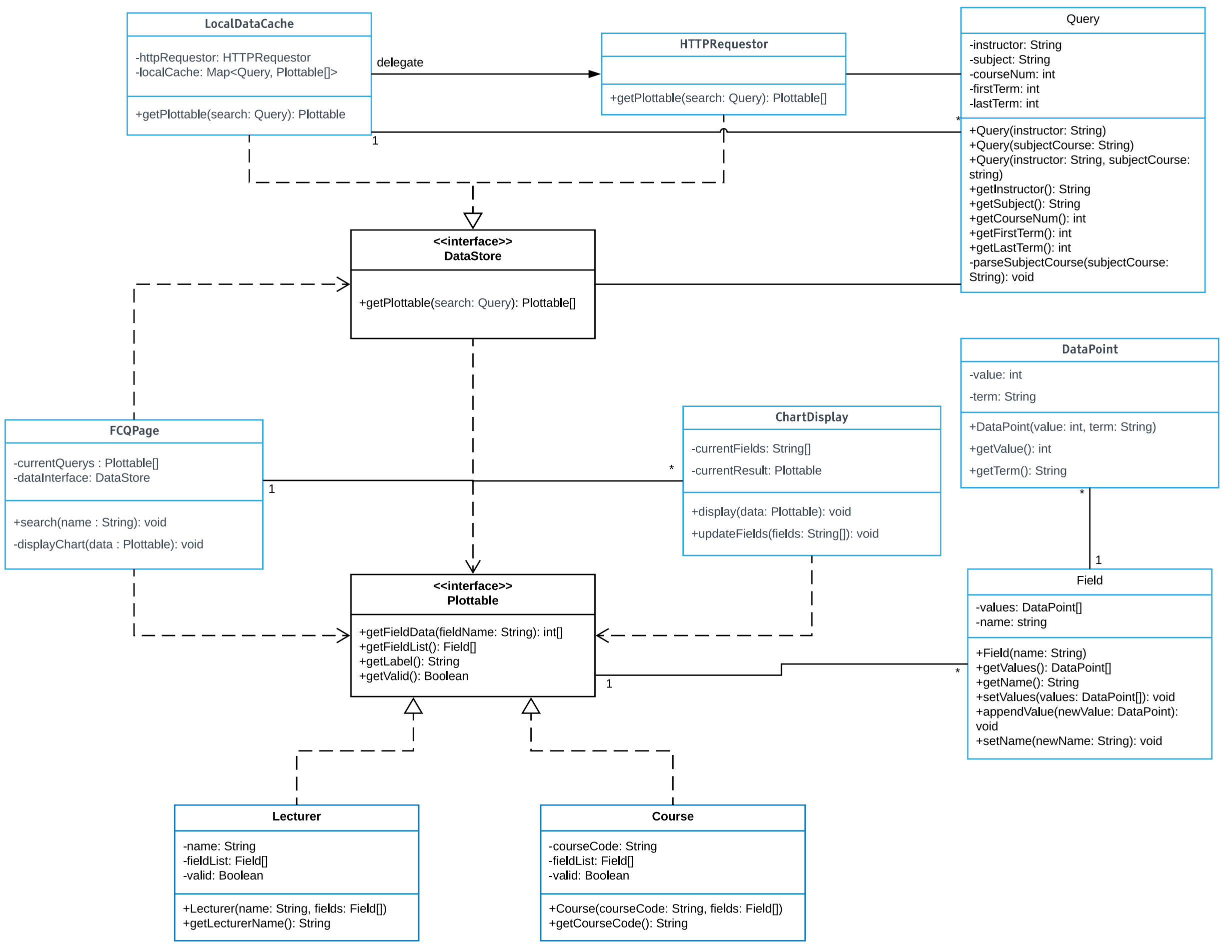
1. **Team:** Samuel Cuthbertson (scuthbert), Connor Hudson (technoboy10)
2. **Title:** fcq.fun
3. **Project Summary:** A CU faculty course questionnaire viewer with improved readability, focused mainly on graphs. Limited to Boulder campus, initially limited to College of Engineering results.
4. **Revised Class Diagram:**

   Added since Part 3: Query class, DataPoint class, modifications to other classes to use Query and DataPoint. NB: We will need to revise this diagram once more to allow for the use of asynchronous function calls, as is required of us by some TypeScript libraries.


LocalDataCache
-httpRequestor: HTTPRequestor
-localCache: Map<Query, Plottable[]>
+getPlottable(search: Query): Plottable
delegate
HTTPRequestor
+getPlottable(search: Query): Plottable[]
Query
-instructor: String
-subject: String
-courseNum: int
-firstTerm: int
-lastTerm: int
+Query(instructor: String)
+Query(subjectCourse: String)
+Query(instructor: String, subjectCourse: string)
+getInstructor(): String
+getSubject(): String
+getCourseNum(): int
+getFirstTerm(): int
+getLastTerm(): int
-parseSubjectCourse(subjectCourse: String): void
<<interface>>
DataStore
+getPlottable(search: Query): Plottable[]
DataPoint
-value: int
-term: String
+DataPoint(value: int, term: String)
+getValue(): int
+getTerm(): String
FCQPage
-currentQuerys : Plottable[]
-dataInterface: DataStore
+search(name : String): void
-displayChart(data : Plottable): void
ChartDisplay
-currentFields: String[]
-currentResult: Plottable
+display(data: Plottable): void
+updateFields(fields: String[]): void
Field
-values: DataPoint[]
-name: string
+Field(name: String)
+getValues(): DataPoint[]
+getName(): String
+setValues(values: DataPoint[]): void
+appendValue(newValue: DataPoint): void
+setName(newName: String): void
<<interface>>
Plottable
+getFieldData(fieldName: String): int[]
+getFieldList(): Field[]
+getLabel(): String
+getValid(): Boolean
Lecturer
-name: String
-fieldList: Field[]
-valid: Boolean
+Lecturer(name: String, fields: Field[])
+getLecturerName(): String
Course
-courseCode: String
-fieldList: Field[]
-valid: Boolean
+Course(courseCode: String, fields: Field[])
+getCourseCode(): String
1
*
1
*
*
1
1
*

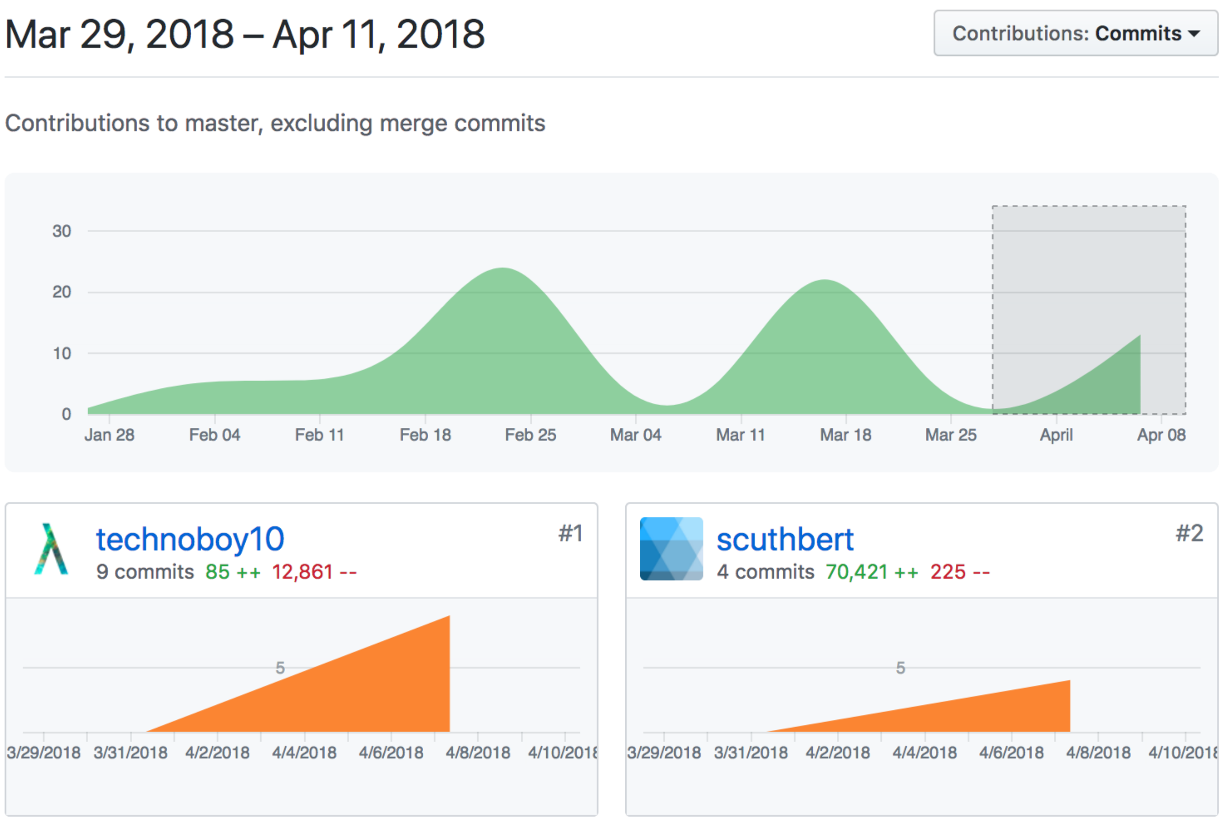
5. **Class Diagram of Implemented Classes:**

   This is deceivingly the same class diagram as above, since we have all the function signatures in code. However, some significant functionality is yet lacking and will be discussed below.


LocalDataCache
-httpRequestor: HTTPRequestor
-localCache: Map<Query, Plottable[]>
+getPlottable(search: Query): Plottable
delegate
HTTPRequestor
+getPlottable(search: Query): Plottable[]
Query
-instructor: String
-subject: String
-courseNum: int
-firstTerm: int
-lastTerm: int
+Query(instructor: String)
+Query(subjectCourse: String)
+Query(instructor: String, subjectCourse: string)
+getInstructor(): String
+getSubject(): String
+getCourseNum(): int
+getFirstTerm(): int
+getLastTerm(): int
-parseSubjectCourse(subjectCourse: String): void
1
*
<<interface>>
DataStore
+getPlottable(search: Query): Plottable[]
DataPoint
-value: int
-term: String
+DataPoint(value: int, term: String)
+getValue(): int
+getTerm(): String
FCQPage
-currentQuerys : Plottable[]
-dataInterface: DataStore
+search(name : String): void
-displayChart(data : Plottable): void
ChartDisplay
-currentFields: String[]
-currentResult: Plottable
+display(data: Plottable): void
+updateFields(fields: String[]): void
1
*
*
1
Field
-values: DataPoint[]
-name: string
+Field(name: String)
+getValues(): DataPoint[]
+getName(): String
+setValues(values: DataPoint[]): void
+appendValue(newValue: DataPoint): void
+setName(newName: String): void
<<interface>>
Plottable
+getFieldData(fieldName: String): int[]
+getFieldList(): Field[]
+getLabel(): String
+getValid(): Boolean
1
*
Lecturer
-name: String
-fieldList: Field[]
-valid: Boolean
+Lecturer(name: String, fields: Field[])
+getLecturerName(): String
Course
-courseCode: String
-fieldList: Field[]
-valid: Boolean
+Course(courseCode: String, fields: Field[])
+getCourseCode(): String

6. **Summary:** Since last iteration, we have created a backend server to handle fetching of FCQ data and started connecting this to the HTTPRequestor class (and by extension, the LocalDataCache class). We ran into serious issues relating to how TypeScript handles HTTP requests and asynchronous results, so the implementation is not complete yet. We also implemented a significant portion of the ChartDisplay class, adding a DataPoint class in the process, to provide the needed data to the `ChartDisplay.display()` function without hacky workarounds. We also changed the relationship from FCQPage to ChartDisplay from a 1-1 to a 1-Many, as part of supporting multiple results from a singe search (i.e. a search for "Liz" should return multiple professors).
7. **Breakdown of Work:** Connor: Implemented ChartDisplay component, configured data visualization libraries, helped revise the class diagram. Sam: Wrote a AWS Lambda to make the CUFCQ database accessible through an HTTP GET call. Imported excel sheet into json, began moving datastore to returning a list of Plottables instead of a singe Plottable.

8. **GitHub Graph**:



Mar 29, 2018 – Apr 11, 2018

Contributions: **Commits**

Contributions to master, excluding merge commits

technoboy10 #1
9 commits 85 ++ 12,861 --

scuthbert #2
4 commits 70,421 ++ 225 --

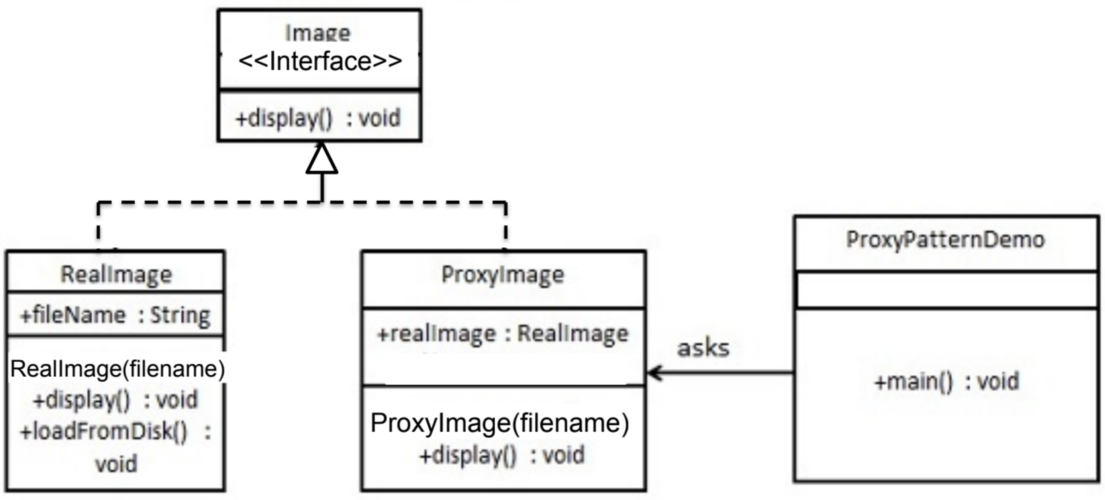
9. **Estimate Remaining Effort:**

   In term of actually finishing everything planned, we're likely now about two thirds of the way there. (Lecturer displays, but Courses do not). However, the deeper we get the farther our scope expands, so estimating is tricky. Our primary remaining task is properly implementing the DataStore functionality, specifically using asynchronous functions.

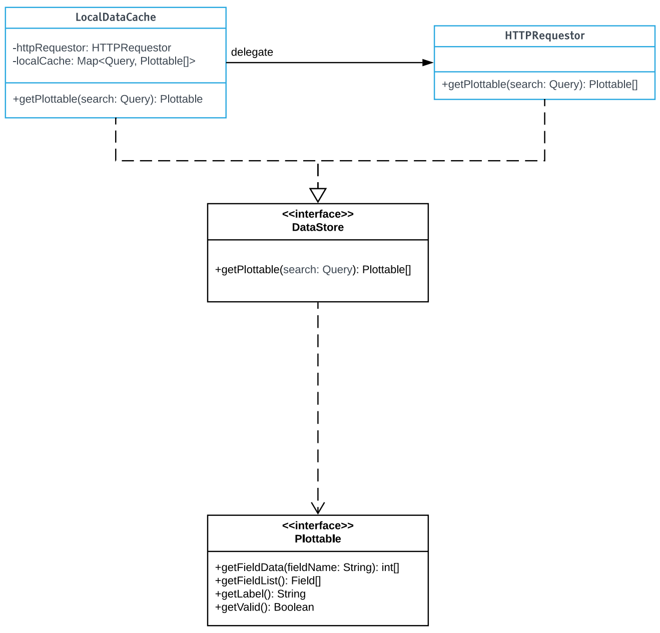
10. **Design Patterns**

    In our system, we prominently use the Proxy design pattern for our data fetching code in order to alleviate excessive database calls. We also use Strategy as an integral part of DisplayChart and DataStore, in order to support searching/displaying for both lecturers and courses.

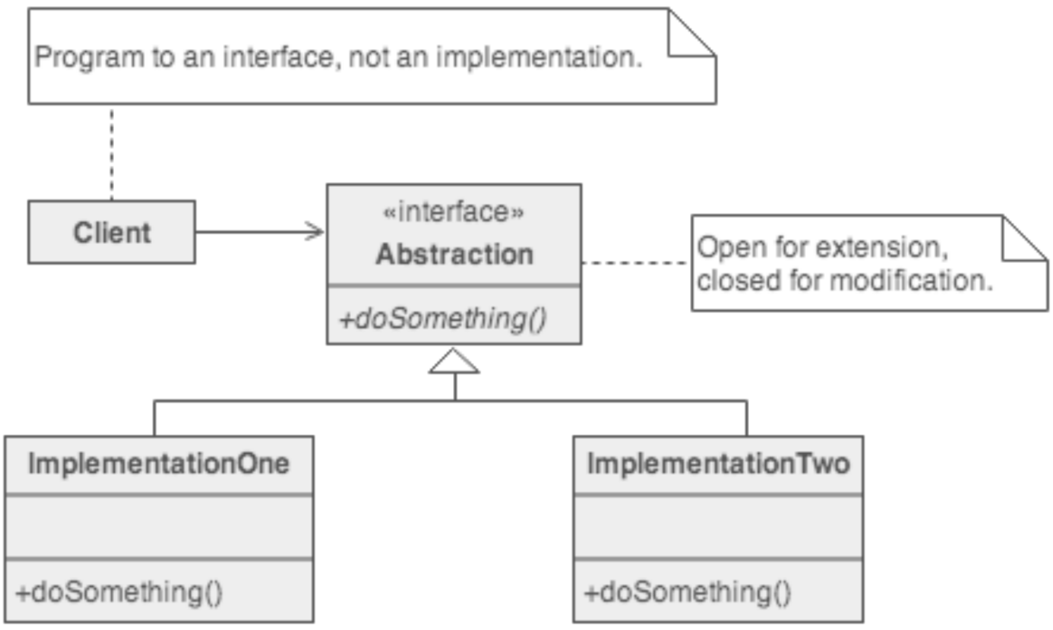
    - Proxy
      - From slides:



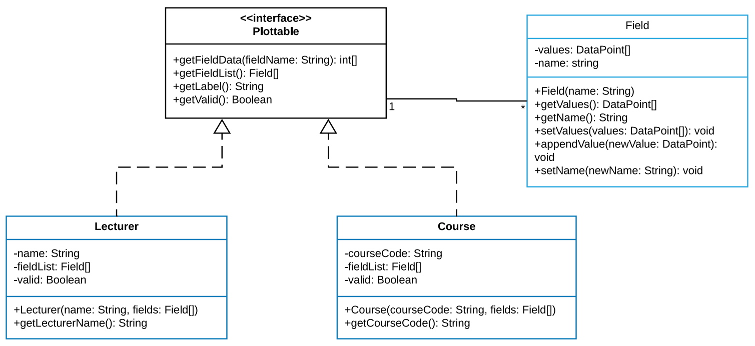
- From our class diagram:


LocalDataCache
-httpRequestor: HTTPRequestor
-localCache: Map<Query, Plottable[]>
+getPlottable(search: Query): Plottable
delegate
HTTPRequestor
+getPlottable(search: Query): Plottable[]
<<interface>>
DataStore
+getPlottable(search: Query): Plottable[]
<<interface>>
Plottable
+getFieldData(fieldName: String): int[]
+getFieldList(): Field[]
+getLabel(): String
+getValid(): Boolean

- Strategy
  - From slides:



- From our class diagram:



11. **Next Iteration:**

    Sam: Will continue with DataStore and FCQPage, adding support for searching by class/additional parameters included in the new Query object.

    Connor: Will ensure that connecting the DataStore and ChartDisplay works properly, and then style the interface so it is more usable for a customer.