QUEST Team Metrics Dashboard

Proposal 11/01

Collin Draper
Jessica Gomez
Amy Odenthal
Ellie Pearson
Sam Schoberg
Will Schulmeister
Rohith Venkatesh

Table of Contents

1. Introduction

1.1. Vision

Our vision for the QUEST Team Metrics Dashboard is a tool that monitors team health via an aggregation of a variety of metrics measuring both student engagement and sentiment. This dashboard allows QUEST faculty to at a glance determine which teams are performing adequately and which require more assistance, so that more attention can be given where needed. Users will be able to quickly navigate through this dashboard to view a list of teams and their team health.

1.2. Background

The Quality Enhancement Systems and Teams (QUEST) Honors Program is a three-year interdisciplinary honors program for students in the Robert H. Smith School of Business, Clark School of Engineering, and College of CMNS focusing on experiential learning projects surrounding quality management and process improvement. QUEST admits 90 students from each graduating class, with those 90 students splitting into two cohorts. Each cohort goes through a five-course curriculum, with four of those offered classes being team-based. Students work in teams of about five students on semester-long projects and are expected to produce quality results. Keeping track of each team in each class can be an arduous task, and is problematic when considering how many different components there are to keep track of. The program instructors and teaching assistants need a way to monitor the "health" of each team efficiently without having to manually meet with each team to check their progress, or continuously monitoring the course/team's ELMS page.

1.3. Objectives

The objective for this project is to create a tool for monitoring the team health of all QUEST teams across the four courses. Our team aims to offer an at-a-glance assessment (Red - Yellow - Green) for a team's health based on a dynamic set of data sources including: Slack, ELMS, Self-Assessment and more. Team health will be defined on the Red - Yellow - Green scale, with backing from the multiple data points aggregated to a final value to fit onto this scale. The tool will use this definition of team health from our team, with the ability for admin users to configure those thresholds for each color. The team will explore how exactly these metrics can be analyzed and compared to set thresholds to determine if the health of a team is red, yellow, or green. This first requires an understanding of the QUEST Program, and more specifically the learning outcomes of the program in order to define team health according to QUEST. The team aims to then develop a dashboard for QUEST leadership, specifically the director Dr. Joe Bailey, to track teams and identify any warning signs of an unsuccessful team. This product should be dynamic, and allow easy addition of external data sources in the future to ensure the product continues to serve the needs of the QUEST leadership. The data dashboard should also be configurable, so that users can customize the data displayed to their specific use case. The goal is ultimately to provide a system that is a simple and initiative way for Dr. Joe Bailey to determine what teams are making progress and finding success, and what teams may be falling behind and need intervention.

1.4. Audience

The stakeholders for this project include the Quality Guild of the QUEST Program, which is the faculty in charge of the QUEST Program, and UMD Computer Science students Adam Sarsony and James Wang. Adam and James play an important role in this project in providing us with context around the program, users, and possible use cases, but are not considered users for the scope of this document and project. We will have two different types of users for this tool, with Dr. Joe Bailey, a member of the QUEST Quality Guild as the Executive Director of the program having an admin role, and QUEST students as student users.

2. Product

2.1. Product Perspective and Relation to Current Field and Market

2.1.1. CMSC435 Dashboard

2.1.1.1. One product that our QUEST Metrics dashboard is similar to, and in some ways drew influence from, is Purtilo's Project Team Status Dashboard for CMSC435. This dashboard aggregates certain data points that represent team health, repository churn, and meetings/engagement with professors and clients. The dashboard then represents this data by assigning a color to each team that demonstrates their progress on the semester-long project. Our product draws some similarities from this existing tool, namely the utilization of color-coding for ease of use as well as the aggregation of a variety of indicator metrics. However, our tool will differ by performing all of the analysis and retrieving all the metrics automatically, and pushing the aggregation of these metrics to the dashboard in real-time.

2.1.2. QUEST

2.1.2.1. While the product offers new functionality, it is similar in its construction. The program will share a Flask hosting, MySQL database and Python logic with other codebases in the QUEST sphere. This will allow for ease of integration in the future. While there exists some rudimentary database(s) on QUEST students, cohorts, and information, we have been informed to proceed with harvesting data on our own from scratch. QUEST hopes this program's database could fit in as a canonical information base.

2.2. User Characteristics

2.2.1. ADMIN: The first user type will be an admin user role. The QUEST quality guild, specifically Dr. Bailey, will use this user type. This user will have outside knowledge on the health of the teams and will be supplementing their information with hard data and general overviews. This user will not be looking for the program to tell them what to do or when to do it, but simply offer an overview that they may then take action on. This user will have, at minimum, instructor privileges when logged into the UMD CAS.

2.2.2. STUDENT: The second user type will be a student user role. This user will have access to their own student and team data and the ability to fill out the student surveys. QUEST students in the five classes will use this user type.

2.2.3. DEVELOPER: Although not a user, QUEST developers and tech leaders such as Danny Laurence are an important consideration in planning and development. This can be anyone

who has influence on the QUEST codebase. Although they won't be using the product directly, their integrating of the product and continued upkeep after this semester are key points. They will be involved in acceptance testing.

2.3. Constraints

2.3.1. FERPA: The Family Educational Rights and Privacy Act (FERPA) is a federal law regarding the privacy of student education records and the control of disclosure of personally identifiable information from education records. After speaking with Danny Laurence who has experience building tools for QUEST, we have determined that QUEST follows University rules and guidelines in handling student data. Therefore, we will follow the same guidelines in our project regarding data privacy.

2.3.2. Time: The product must pass all acceptance tests (as defined below) by December 5th.

2.3.3. Database: We are limited to QUEST's database capabilities for space and cost.

2.3.4. Slack: We are limited by the QUEST's usage of Slack. If each team communicates within a designated channel, tracking their activity and engagement will be trivial. However, if there is no standardized way QUEST classes are using Slack, tracking slack messages per team will be vastly more difficult.

2.3.5. ELMS: We need to have administrative permissions for accessing the ELMS API in a way that we can scrape necessary information. We will need to configure Dr Bailey's account to access data. To do this, we must make sure that Dr. Bailey's ELMS account has admin privileges over, at minimum, a sub-group including all of the QUEST courses he wishes to utilize our tool with. If this is not the case already, this would require contacting the admin of the sub-group directly above Dr. Bailey to trickle down these necessary admin privileges to his specific user. Furthermore we will have to set up OAuth2 tokens for Dr. Bailey's admin account. These two steps will allow our tool to invoke API calls that can glean data from all of the relevant QUEST classes and from the students within those classes.

3. Data Flow

3.1. Data Collection and Processing

3.1.1. For each data source, Slack, ELMs, and personal surveys, we have distinct plans for collecting, transforming, and eventually storing the data in our master database. All of the metrics mentioned below will be collected and aggregated on the QUEST group and class levels. Our goal is to analyze the data for anomalies that could be indicative of a dip in team engagement and produce visualizations that establish baselines for team activity within the QUEST program.

3.1.2. Slack

3.1.2.1. Value: Slack is a chat based collaboration tool for teams. By accessing message logs for each team and class, we'll be able to track communication patterns for teams and classes. From this data, we could notify concerned parties when there is a dip in communication as well chart communication patterns for teams throughout the semester.

3.1.2.2. Access: We can access Slack workspace data through the Slack Web API with Python. After creating appropriate accesses, we can query all message history for all teams in

a workspace. Communication is a metric we plan on updating daily at midnight by running our data collection scripts with a cron job on our VM.

3.1.2.3. Data: The API response for conversation history is a JSON object that contains the message text, timestamps, channel, author among other less useful fields. See */questmetrics/playground/backend/messages.csv* for reference.

3.1.2.4. Processing: To glean value from the message data, we have to be able to map slack user IDs to individual quest students and their teams in order to see who sent the message and which team they belong to. To accomplish this, we'll lookup their UMD email address using the Slack Web API and then map that to a QUEST team using our database.

3.1.3. ELMS

3.1.3.1. Value: ELMs is the course management software used by QUEST classes. Professors post assignments and readings to it frequently, and engaged students should be actively checking it. We can collect data about student activity on the site using the ELMs API. The combination of different data points mentioned below will be an indicator of how engaged a student or group is with the course material.

3.1.3.2. Access: Working with ELMs API requires access keys from a course administrator. We plan on asking Dr. Bailey to request access keys on our behalf. Documentation on exact steps we will present to him can be found in the *questmetrics/industry_reserach/ElmsAPI/elms_auth_research.pdf* . We plan on gathering ELMs data on a weekly basis with a cron job on our VM that runs on Sunday night.

3.1.3.3. Data: We plan to collect various data points for each student in QUEST. These include login frequency, assignment grades, promptness of assignment submissions, and how often they are viewing files. It's important to note not all of these data points may be used in the final product. After analysis we may determine some ELMs metrics may not be useful for determining team engagement.

3.1.3.4. Processing: Much like the Slack data, we must be able to map a student's individual data to their respective QUEST teams. We'll perform this operation using a team lookup in our database.

3.1.4. Team Health Surveys

3.1.4.1. Value: Individual student users will have access to Team Health surveys that will gauge a team's potential for success and highlight any potential weaknesses. The questions were chosen through research cited in Appendix B. This is the biggest influence on a team's metrics because it will specifically show how the team feels about the progress they have made and how each member feels about their teammates.

3.1.4.2. Access: The surveys will be distributed weekly to individuals within the application. They will be hosted on the dashboard itself so that the students can sign in using CAS and complete them.

3.1.4.3. Data: Note the questions from Appendix B for an initial prototype for the types of questions a survey may ask. The questions measure team coordination, different specializations, motivations, credibility and cohesiveness. Each question will be on a

scale rating, with the lowest score indicating that the individual needs attention and the highest score indicating that the individual is content or happy with their team and their progress.

3.1.4.4. Processing: The data will be aggregated by team. The score will be added up for each individual which will become a component of individual and team health scores. Team health survey scores will be the largest indicator on team health since other data sources don't show accurately how each team is interacting.

3.2. Analysis

3.2.1.1. Visualizations: All of the above metrics will be aggregated and displayed for the admin user on the team and class level using Tableau. Slack and ELMs activity and team health survey ratings will be plotted over time. We plan to research other visualization techniques as well and will document them as they are discovered and implemented.

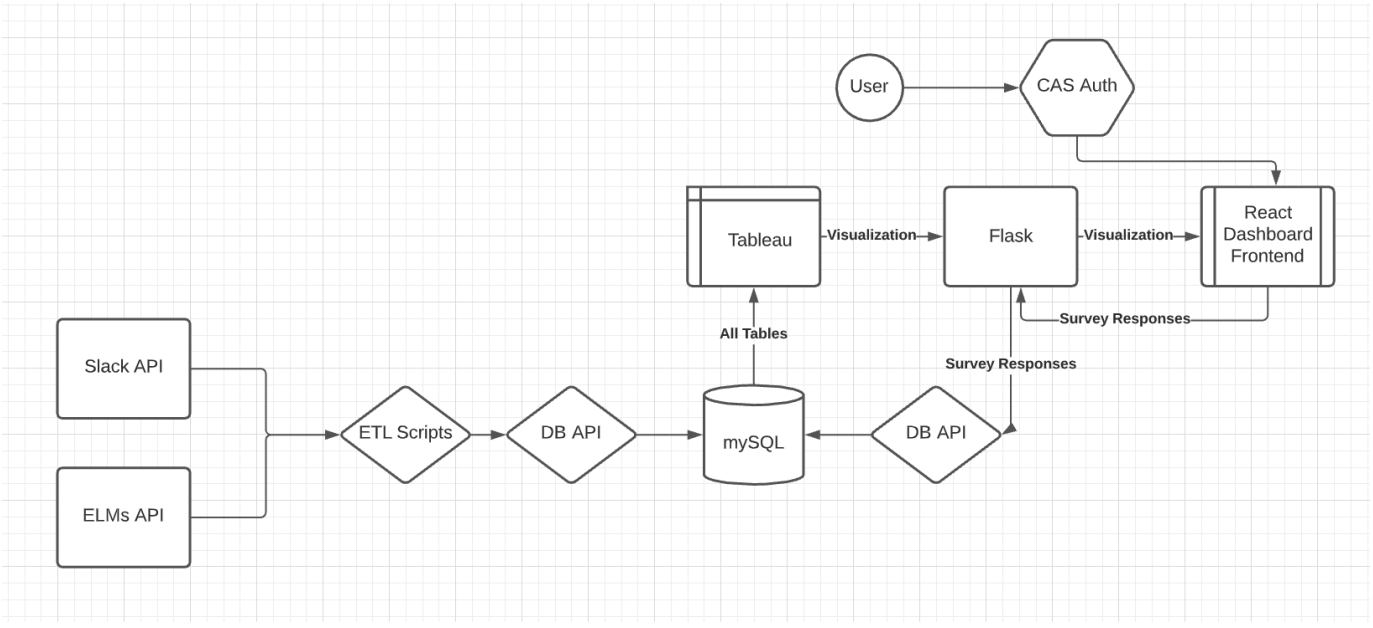
3.2.1.2. Health Rating: All of the metrics above will be weighted and combined to form an overall score that represents team health and engagement on a red, yellow, green (bad, okay, good) scale. We acknowledge this analysis is going to be highly experimental as we discover which data points are more indicative than others at quantifying health and engagement. We will publish and document our research and conclusions as we experiment. Initially, we think this analysis could be a ranking system where we compare teams to one another, but we are also exploring other options.

3.3. Design constraints

3.3.1. The first design constraint is brand and styling since the theme and aesthetic of the overall dashboard should be in accordance with UMD and QUEST styling. The next constraint is that the dashboard is being designed for a computer interface. Next, the product must not only be aesthetic, but must also have seamless navigation between categories and data to enable easy decision-making. Our last constraint is ease of integration. The dashboard needs to be able to accommodate multiple sources of data (plug-and-play), thus must be built with scalability and robustness in mind.

4. Components

4.1. Architecture



Architecture diagram

The graphic above provides a high level overview of the data flow for our system. After reading from the Slack and ELMS APIs, we'll process the data and enter it into a mySQL database. The tableau server reads all the tables from the database and produces the appropriate visualizations. Flask extracts the visualizations from the server and serves them to the react front end. Users login to the react front end using CAS auth and can enter survey responses. Those responses are sent to the database and will be charted in tableau as well.

4.2. Login Page

4.2.1. UMD CAS will be used to handle the login to ensure privileges dictate what functionality a given visitor has access to.

4.3. Dashboard

4.3.1. The frontend will be coded in React with JSX. The main goal is modular display, which will allow users to add in their own data sources. The visualizations from the tableau server will be embedded within the dashboard. The user will be able to see all metrics displayed at the class and group level. The dashboard will not be mobile friendly— it will require Dr. Bailey to use a computer to view the dashboard, and the student users to use their computers in order to complete the survey.

4.4. Tableau Server

4.4.1. The UMD Tableau server will be used to visualize the data and provide analysis. The server is able to link to our mySQL database, pull in current data, and provide drag and drop functionality for plotting metrics against one another. It makes it easy for a user to view high level QUEST class data then drill down to the team level. Finally, the live updating visualizations from the server will be embedded in our front end.

4.5. Internal API

4.5.1. It will be built in accordance with the Standard RESTful API (GET, POST, DELETE, PUT). Basic functions are defined below and are subject to change. This Internal API will be used to separate database functionality from data processing and will include in depth documentation for all endpoints, functionality and return values. By cooperating with Danny Laurence, this Internal API will be usable as a standalone function outside of our product, but it will primarily be used to ensure the data that gets sent for processing is consistent regardless of the changes to the database. QUEST will be responsible for securing deployed data from rogue calls.

4.5.2. GET

4.5.2.1.1. getStudent: retrieve all personal information related to the queried student. Should be a single student. The queried student must exist.

4.5.2.1.2. getGroupSlackInfo: retrieves the JSON object associated with the most recent data from the Slack API for a specific, queried group. The queried group must exist.

4.5.2.1.3. getGroupElmsInfo: retrieves the JSON object associated with the most recent data from the ELMS API for a specific, queried group. The queried group must exist.

4.5.2.1.4. getGroupXInfo: API endpoint to be added in the case of extension. For example, if Github is added as a source, an endpoint called getGroupGithubInfo would be included.

4.5.2.1.5. getGroupHealth: retrieves the health data associated with a specific, queried group. The queried group must exist.

4.5.2.1.6. getStudentReport: retrieves the survey data associated with a specific, queried student. The queried student must exist.

4.5.2.1.7. getStudentHealth: retrieves the health data associated with a specific, queried student. The queried student must exist.

4.5.2.1.8. fetchAllStudents: retrieves all data in the database from the Student table.

4.5.2.1.9. fetchAllGroups: retrieves all data in the database from the Group table.

4.5.2.1.10. getRedGroups: get all groups with Red status

4.5.2.1.11. getGroupsbyStudent

4.5.2.1.11.1. By UID

4.5.2.1.11.2. By Email

4.5.2.1.12. getStudentsByGroup

4.5.2.1.12.1. By group name, return all students' info

4.5.2.1.12.2. In students.py '/students/group/<name>'

4.5.2.2. POST

4.5.2.2.1. newStudent: add a new student to the database

4.5.2.2.2. addStudenttoGroup: adds an existing student to an existing

4.5.2.2.3. newGroup: create an empty group in the Group table.

4.5.2.3. PUT

4.5.2.3.1. updateGroupHealth: replace the value in the Group Health column with the queried value. The queried group must exist.

4.5.2.3.2. updateGroupSlackInfo: replace the value in the Group Slack Info column with the queried value. The queried group must exist.

4.5.2.3.3. updateGroupElmsInfo: replace the value in the Group Elms Info column with the queried value. The queried group must exist.

4.5.2.3.4. updateStudentHealth: replace the value in the Student Health column with the queried value. The queried student must exist.

4.5.2.3.5. updateStudentReport: replace the survey data associated with a specific, queried student. The queried student must exist.

4.5.2.3.6. updateGroupXInfo: API endpoint to be added in the case of extension. For example, if Github is added as a source, an endpoint called updateGroupGithubInfo would be included.

4.5.2.4. DELETE

4.5.2.4.1. clearClass: clear all information associated with a queried class ID, including groups and students.

4.5.2.4.2. removeStudent: remove student from the student table and their associated group.

4.5.2.4.3. removeGroup: removes group from the Group table and the student team table

4.6. External APIs/ETL Scripts

We will be using two external APIs to fetch data at the start, ELMS and Slack. We'll design a class to encapsulate the API requests for each service. From ELMS, we'll request data regarding the last login time for users and assignment submission data such as: submitted vs yet to submit, time of submission. late submissions, and assignment grades. From Slack, we'll request data about conversation history within a channel.

4.7. Surveys

The survey will be built with HTML and CSS, and will be a React component as a separate page of the dashboard. This survey page will be mobile friendly so that students can complete it on their phones. A student will not be able to access the rest of the dashboard through the survey page itself— the dashboard itself will be protected by UMD CAS authentication. Dr. Bailey will be able to change or add questions tailored for each class through his own login. All survey questions will be required to have responses on a scale from 1 to 5 in order to be considered a part of the metrics.

4.8. Database

4.8.1. We will use a MySQL database to be hosted by QUEST (with the help of Danny Laurence). In the meantime, it will be hosted on the VM given by CMSC435. MySQL suits our needs here because of its flexibility in complex joins that may need to happen across students, groups and any of their individual information.

4.8.2. Tables

4.8.2.1. People (information on all the people), Groups (information on each group associated with each class), Classes (information on the classes), Students (information on people who are current QUEST students), Student Teams (maps students to each of their

teams), Slack Data(information on the Slack channel of a given group), Elms Data (information on the ELMs activity of a given student), Survey Data (information on the Survey data of a given student), XData

4.8.2.1.1. While we will not use People as a standalone table, it is useful for Danny Laurence's ability to fold our database in with the current base. We will be adding students to both Students and People 1:1.

4.8.2.1.2. Student Teams is created to map a student to one or more of their related groups

4.8.2.1.3. The sql code for creating this tables is in Appendix A

4.8.2.1.4. XData refers to any new table to be added in the event of a new data source. If the data is on a student to student basis, it will match Elms Data. If on a group to group basic, it will match the Slack Data

4.8.2.2. Note: the schema above are subject to change in the case of extension of data sources, in which case a column could be added to either table. In this case, all related API calls would be updated, removed, or added to maintain robustness. The schemas' primary keys are also subject to change in the case of improved efficiency or readability. These changes would only be reflected on the bottom layer of the API.

4.8.3. CRUD

4.8.3.1. Data will be deleted and created at the start of each semester to reflect the changing of QUEST members and QUEST teams at these dates. As the code will be hosted by QUEST, we will not oversee this management, but we will provide the necessary API functionality to carry it out from a high level, with CRUD functionality at every level of information stored in the database.

4.9. Hosting (Flask)

4.9.1. The program will be hosted in Flask. We chose Flask for its simplicity and lack of dependencies as well as its ubiquity in other QUEST code. Flask code will be responsible for rendering React templates as well as receiving API calls.

4.10. Functionality Testing

4.10.1. Unit Tests:

4.10.1.1. Build tests either before code inception or without referencing it to ensure complete test coverage and confidence in results. This can then be used as a baseline to test changes against, particularly when new sources of data are added.

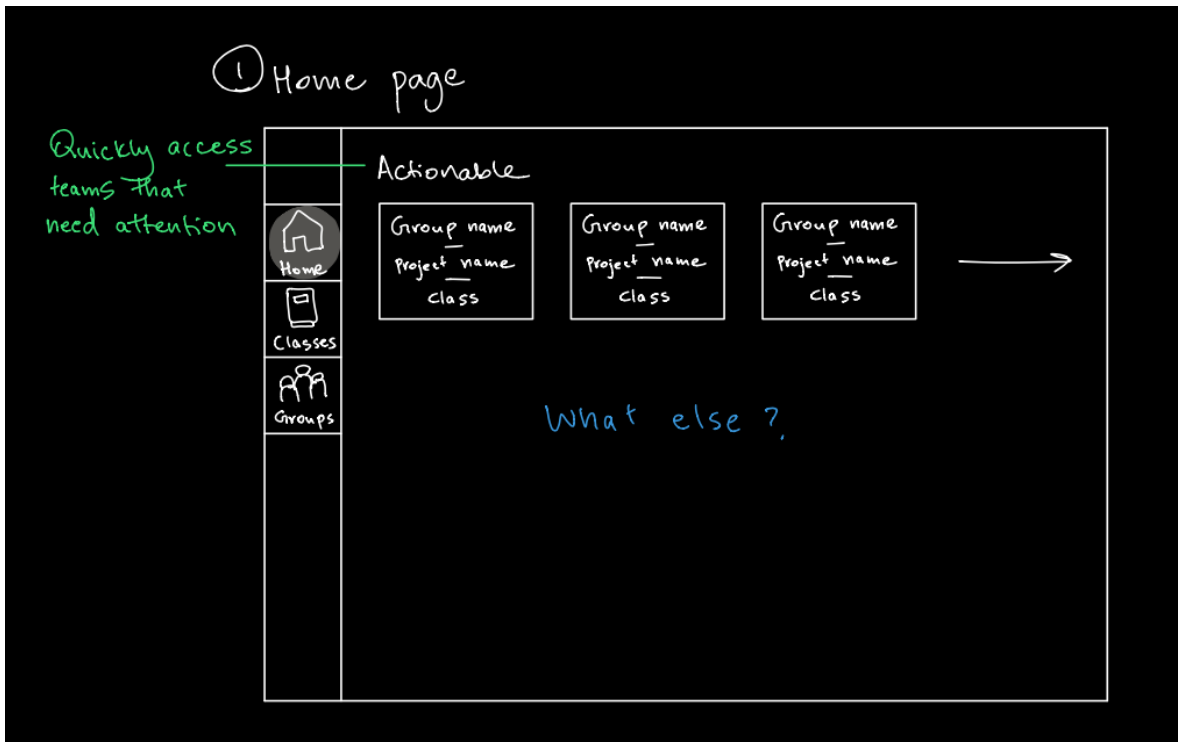
4.10.1.2. PyTest will be the unit testing framework to test functionality of Python code and outputs.

4.10.1.3. Jest will be used as the unit testing framework for React.

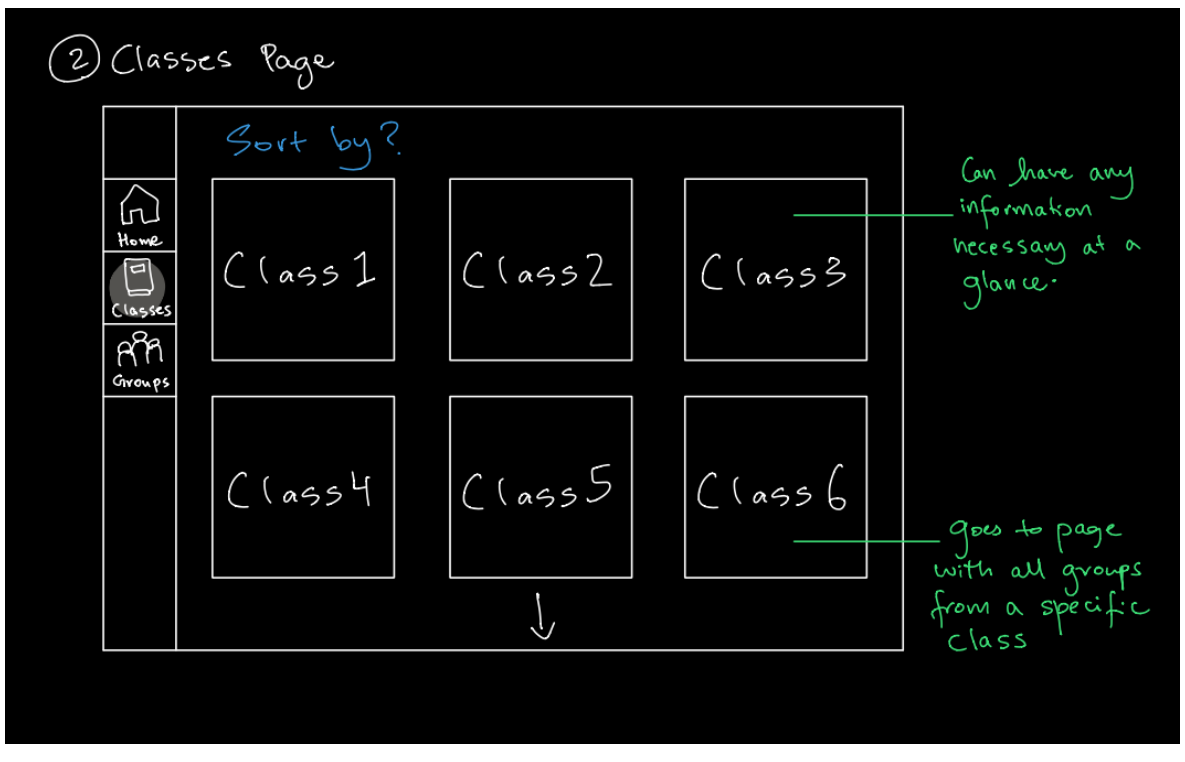
4.10.2. Stress Tests:

4.10.2.1. Locust will be used to stress test the API and Flask hosting. We will make sure our infrastructure can hold up the weight of many calls without breaking data storage. This use case may not be realistic given the worst case number of users being low, but it is important for ensuring robustness.

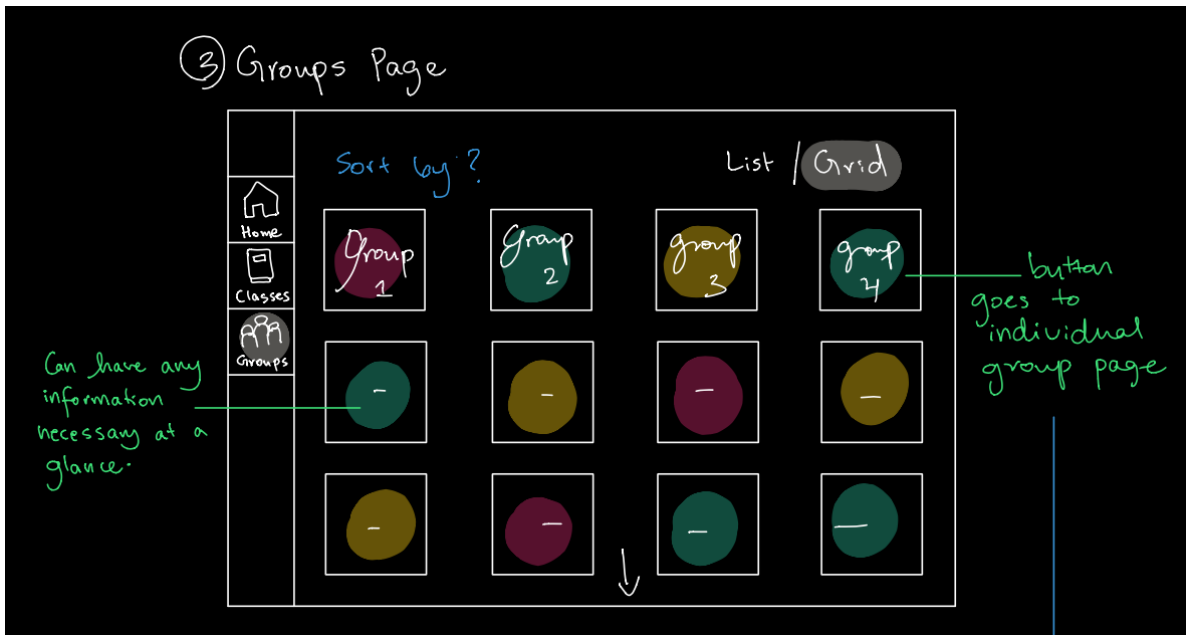
5. Interface



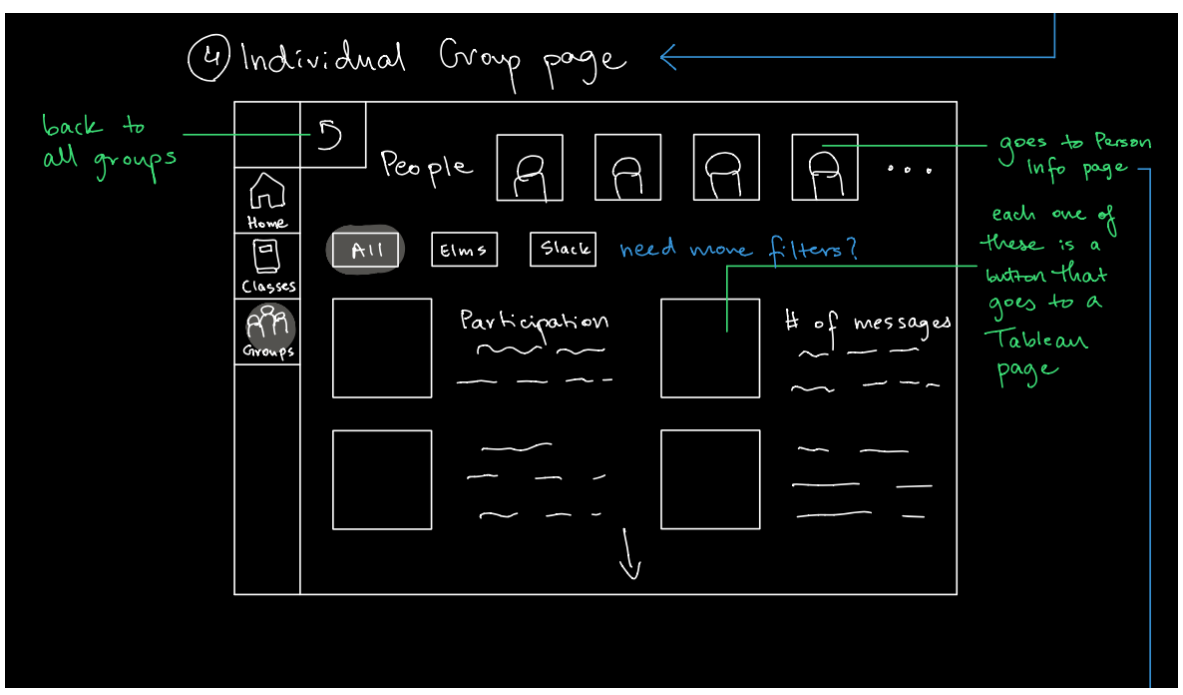
This is the Home page that contains an Actionable Section, which is a list of teams that are colored red, giving the user the opportunity to intervene. Each of the team cards will contain the name of the group and the class that they are a part of, and clicking on them will navigate the user to the specific team's page.



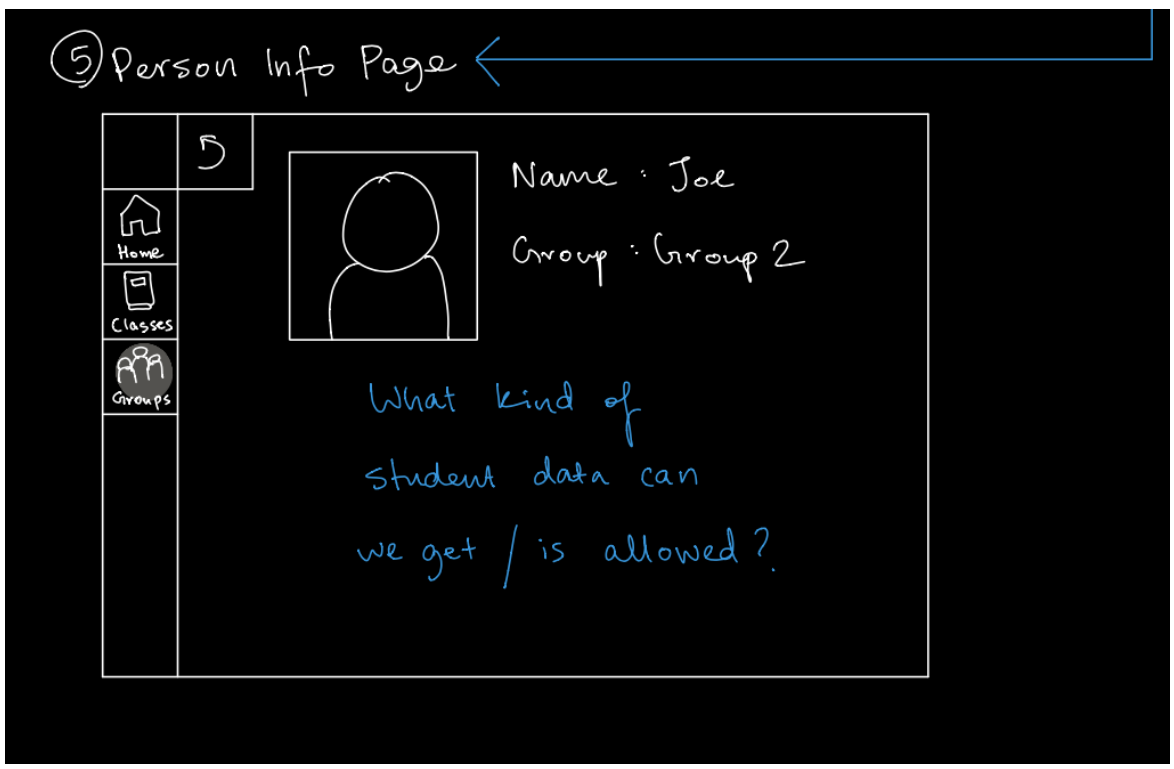
This is the Classes page that contains a list of the classes in the QUEST program. Each of these will contain a color associated with them based on the calculated metrics.



This is the Groups page that contains a list of all the groups in the QUEST program. Each of these will contain a color associated with them based on the calculated metrics.



This is the Individual Group page that contains the people in the group, different Tableau visualizations for the different sets of data, and filters to view the data in different perspectives.



This is the student page that contains information specific to that student: how often they submit their assignment on time, how many messages they send in Slack, etc.

6. Extensibility

Extensibility is a critical feature of our dashboard. This means that in the future developers will easily be able to add additional metrics to our database using the API, visualize those metrics, and finally work those metrics into the overall health measurements for teams and classes. In order to accomplish this, we're going to rely on a combination of modular code and documentation of processes. Additionally, QUEST developers and tech leaders will have access to in-depth documentation on how to extend the program beyond the MVP.

6.1. Modularity on the frontend will help us easily adapt displays for more components. React was chosen specifically for this plan.

6.2. Backend Containerization:

6.2.1. We plan to host our database and run our data collection and processing scripts on the provided VM. We understand eventually the quest dev team may port the application to an AWS server for production. To mitigate some of the pains it takes to run the codebase on a new server, we're going to investigate ways to preserve the production environment. We identify docker and VM snapshots as potential promising solutions. With either method, we'll be able to provide the Quest devs with a way to recreate our environment in future builds. Before commiting to either option, we will research and publish information to the svn. Fortunately, the final decision on this issue can wait until we're nearly complete with the system.

6.3. Separation of functionality on the backend:

6.3.1. Additions to database schema will propagate up to the API service layer logic and the call on the Flask server.

6.3.2. Modifications to database schema will only propagate up to the API service layer logic.

6.3.3. Changes in data delivery to the front end and Tableau will also only be evident on the API service layer.

6.4. Documentation of our internal API as well as the process of adding schema to accommodate new data will allow for continued ease of extensibility.

6.5. Documentation of our Tableau workflow:

6.5.1. Provided there is new data in the database, we'll detail how to go about creating visualizations for both the class and group levels.

7. Acceptance Criteria

To respect people's time and variable schedules, we're reserving the week of November 7-14, 2020 for meeting(s) for the below tests. If all of the tests pass, the system will be considered a success. Any less is a failure. We consider all of these to be critical for even a minimally viable product. Students may participate in different tests so long as they satisfy the criteria for each one.

7.1. Security: Can Dr. Bailey log in successfully? Do logins have the appropriate privileges?

7.1.1. Plan

7.1.1.1. We will recruit three QUEST students in different classes and groups as well as Dr. Bailey to attempt to authenticate to the QUESTmetrics system. Of the students, one will be expected to use mobile and two will be expected to use a modern desktop browser.

7.1.1.2. Without giving explicit login instructions, we will direct the users to the QUESTmetrics login page and instruct them to authenticate. After the authentication process, all above assertions must be true for the test to pass.

7.1.1.3. Once on the home page, we will ask them to stay with screens shared. The team will decide for themselves if the information being displayed is appropriate and the users will also be asked if they feel the information is correct for their permissions. Feedback will be recorded.

7.1.2. Criteria

7.1.2.1. All subjects must all confirm authentication is identical to other campus systems. If the process is longer in any way, the test fails.

7.1.2.2. All subjects must be able to refresh the page and remain authenticated. If they are forced to log in again, the test fails.

7.1.2.3. All subjects must land on their appropriate homepages after auth. If Dr. Bailey does not see information at the top level or if a student sees administrative information, the test fails.

7.2. Clarity: From the homepage, can Dr. Bailey clearly see an overview of major metrics for all the QUEST classes? Are teams we consider "struggling" clearly labeled as such? Is it clear to the user what the graphs are displaying?

7.2.1. Plan

7.2.1.1. Only Dr. Bailey is required for this test as students will not have the privileges to see metrics across other teams and groups.

7.2.1.2. Populate the QUEST database with one struggling ("RED" health) team.

7.2.1.3. From the home page, we will ask Dr. Bailey to identify a struggling team. We will then ask him to find more information on their status and give us an explanation for why they are performing the way they are. We expect him to click directly on the team on the home page.

7.2.1.4. From the Group page, we will ask Dr. Bailey if he feels the data is clear and ask him to change to a specific subset of it without instructions on how to do it. Feedback will be recorded.

7.2.2. Criteria

7.2.2.1. Dr. Bailey will be able to see team members, current health diagnosis and health history (since the beginning of data collection) for any team he wishes within 3 clicks. If there are any more, the test fails.

7.2.2.1.1. Note: The number of clicks here (3) represents the minimum number of clicks expected to go from the home page to the team-specific information page.

7.2.2.2. Dr. Bailey is able to identify a struggling team within 15 seconds of being on the home screen. He then clicks directly on their profile and can identify why they are struggling.

7.2.2.3. Dr. Bailey considers the data as clear and is able to switch to a different representation of data.

7.3. Flexibility: Can Dr. Bailey see all metrics at the class level and at the group level?

7.3.1. Plan

7.3.1.1. From the home page, we'll ask Dr. Bailey to navigate to the class view page. From there we'll ask him to name all the classes and describe the information displayed for each class.

7.3.1.2. From the class page, we'll ask Dr. Bailey to navigate to the specific groups within a class of his choosing.

7.3.1.3. From the groups page, we'll ask Dr. Bailey to list all the groups within a class and describe the information displayed for each group.

7.3.2. Criteria

7.3.2.1. Dr. Bailey can navigate to the class page within one click of the home page.

7.3.2.2. Dr. Bailey will be able to confirm the names of all of the classes in QUEST and interpret the class level statistics from the classes page.

7.3.2.3. Dr. Bailey, upon prompting and within one click, will be able to navigate to the groups within a class page. He will confirm those groups are accurate for the class and accurately interpret the information displayed.

7.4. Flexibility: Can Dr. Bailey modify the questions in the survey?

7.4.1. Plan

7.4.1.1. From the home page, Dr. Bailey will be able to navigate to the survey page. From there, he will be able to add and edit the survey question.

7.4.1.2. Dr. Bailey will be able to choose which class he will be assigning this survey to.

7.4.2. Criteria

7.4.2.1. Dr. Bailey can navigate to the survey page with one click of the home page.

7.4.2.2. Dr. Bailey can add and delete survey questions to the survey form.

7.4.2.3. Dr. Bailey can edit the survey questions inside the form.

7.4.2.4. Dr. Bailey can choose a class to assign the survey to from a drop down menu.

7.5. Student Access: Can the students fill out the survey provided to them?

7.5.1. Plan

7.5.1.1. We will recruit 2 students from QUEST. One will use mobile and one will use a modern browser on desktop.

7.5.1.2. The students will log in and will be able to access their survey in 1 click. We will then ask if the survey's location was clear. Feedback will be recorded.

7.5.1.3. The students will each fill out their surveys.

7.5.1.4. Independently, the QUESTmetrics team will check if the survey submission is reflected in the database.

7.5.2. Criteria

7.5.2.1. If the students are unable to access their survey in 1 click or they deem the process unclear, the test fails.

7.5.2.2. If the changes are not reflected in the database, the test fails.

7.6. Extensibility: Can a developer add a new metric with minimal effort (See Extensibility section)?

7.6.1. Plan

7.6.1.1. This task will be developmental and will take place asynchronously. We will ask Dr. Bailey for a data source which could improve the robustness of the assessment. If he is impartial to a decision, we will suggest GitHub. We are aware of Github usage for some teams in QUEST. We were discouraged from including Github in the MVP, but inclusion of this datapoint at a later date could be a great demonstration of extensibility.

7.6.1.2. We will then assume development and documentation to be finished within 2 days of the request.

7.6.1.3. We will plant a team that is considered struggling because of its newly implemented data.

7.6.1.4. Dr. Bailey will navigate to the team and then its new data. We will ask him if the display is clear and complete from his perspective.

7.6.2. Criteria

7.6.2.1. If the implementation, testing, and documentation process takes place over longer than two days, the test will fail.

7.6.2.2. If Dr. Bailey does not consider the new data clear and complete, the test will fail.

7.7. Integration: Can a developer build our codebase on a different server (AWS or other VM)?

7.7.1. Plan

7.7.1.1. This task will take place asynchronously and rely on deployed proof of completion as well as a follow up conversation with the subjects.

7.7.1.2. We'll recruit either Adam or James to serve as QUEST student developers, and will spin up a new, clean server either as a VM on either Purtilo's cluster or on AWS (depending on customer preference). We'll provide them with the applications code repository and direct them to the readme. We'll tell them to build the application on this new server using the documentation.

7.7.2. Criteria

7.7.2.1. The developer can successfully build and start the front and back end servers using the provided docker containers (or VM snapshot) within a day.

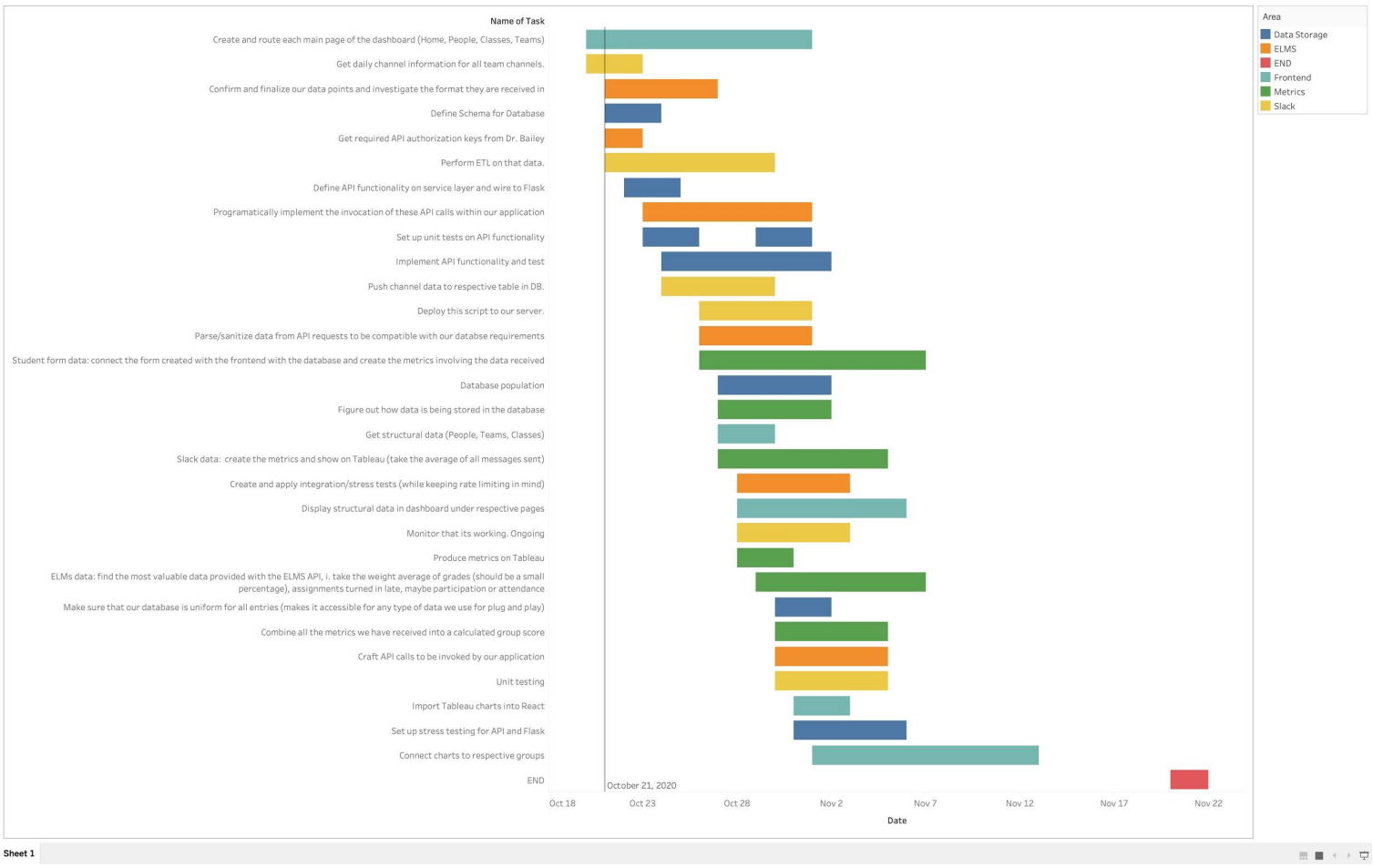
7.7.2.2. The developer can successfully create and link the backend to the database within a day.

7.7.2.3. Developer will confirm when asked that the documentation is clear and reproducible.

7.8. Overall Experience: Is the experience, as a whole, positive?

7.8.1. After the above tests are passed, ask the participant (whether it be Dr. Bailey or student) to rate the experience on a scale from 1-10. If, at the end, the average is not above 8, the test will be considered failed. Obviously, feedback will be expected in any case and will be used to improve the experience.

8. Timeline



9. Cost

9.1. In order to calculate the cost, multiple researched methods were used. One of the easiest, most common ways of estimating cost is the Top Down Estimation. This method requires holistic data from previous projects, so each of us on the team used our experiences on our own personal projects instead. We combined this with the Three Point Estimation method, and each of us expressed our best-case estimate, most-likely estimate, and worst-case estimate. Aggregating all of this data, we came up with three different estimates: best case being 119 hours, most probable case being 146 hours, and the worst case being 175 hours. Each person was surveyed individually and answered with no bias, thus the aforementioned numbers are our team's beliefs.

9.2. The second method that we used in order to estimate the cost was Bottom Up Estimation. More specifically, we created a Work Breakdown Structure where we discussed each of our tasks and laid them out in a Gantt chart to help us visualize the timeline and estimate the cost with better accuracy. Using the above Gantt chart and assuming that the person works an average of one productive hour per day, we estimate that this project will take around 192 hours. This number

accounts for possible issues that may come up during development as well, and thus may be a slight overestimation.

9.3. Two other cost estimation methods that we looked into but did not use are: Parametric estimation (as we did not have previous mathematical data to extrapolate), and the PERT chart (as a lot of our parts happen concurrently and the Gantt chart was better suited to represent our tasks).

9.4. Functional Point Analysis

9.4.1. Counts of each measurement parameter

9.4.1.1. External Inputs (5): Login, survey, Slack data, ELMS data, threshold configuration

9.4.1.2. External Outputs (1): Charts that Tableau produces

9.4.1.3. External Inquiries (2): Auto-sending reminder emails, CAS login

9.4.1.4. External Interfaces (2): Tableau, React site

9.4.1.5. Internal Files (1): Our database

9.4.2. FPA Calculations

Measurement Parameter	Count	Weighing Averages			Low UFP	Medium UFP	High UFP
		Low	Medium	High			
Number of external inputs (EI)	5	3	4	6	15	20	30
Number of external outputs (EO)	1	4	5	7	4	5	7
Number of external inquiries (EQ)	2	3	4	6	6	8	12
Number of internal files (ILF)	1	7	10	15	7	10	15
Number of external interfaces (EIF)	2	5	7	10	10	14	20

Total	42	57	84
FP	44.94	60.99	89.88
Functional Units/week	11.235	15.2475	22.47
Productivity = FP/PM	6.42	8.712857143	12.84

9.4.3. FP Results Explanation

9.4.3.1. Using this style of cost estimation, we have received a low, medium, and high estimate for the effort required for our project in the unit of function points (FP). The function point is a standard unit of measure used to estimate the complexity of a software project. Our estimation shows our project to fall in the range of 44.94 FP and 89.88 FP. The subsequent line demonstrates this total breakdown on a per-week scale to get a better idea on how our sprints should be broken up. Finally, the productivity factor has a unit of FP/PM, where PM represents effort in the unit of person-months. Our estimation shows that our product will require an effort that falls in the range of 6.42 - 12.84 FP/PM.

10. Press Release

QUEST Team Health Dashboard

This dashboard will provide the QUEST Quality Guild, specifically Dr. Joe Bailey, with a tool that monitors team health across the four different courses in the QUEST curriculum, providing key insights into teams that are doing well, and teams that may need assistance.

SUMMARY: As the Executive Director of the QUEST Program, Dr. Joe Bailey is lacking an efficient and clear way to monitor team health for QUEST teams across all four courses. This dashboard aims to fill this void by capturing metrics to determine the overall health of a team by placing that health score on a Red-Yellow-Green scale to show a fast and easy assessment while also providing a depth of data to investigate further. With its ease of use and intuitive nature, the tool will enable Dr. Bailey to quickly notice when teams are suffering, and intervene before there is an irreversible impact on the progression of students and their projects.

PROBLEM: With about 45 teams across five different QUEST courses, Dr. Joe Bailey finds it difficult to track the health of every team to ensure all students are achieving learning outcomes and staying on track. Dr. Bailey is interested in learning which teams are making progress, and which teams may need further assistance if they are not performing well. A multitude of metrics and different data points can be used to determine what truly defines a healthy team, and Dr. Bailey needs a way to determine how these data points can all fit together and produce a health score for a team.

SOLUTION: The QUEST Team Health Dashboard will serve as an at-a-glance tool for Dr. Bailey to check in on the health of every team across the different QUEST courses. Using a variety of data sources including Slack, ELMS, and surveys hosted on the tool itself, an aggregate score of team health will be determined and then compared to set thresholds on a Red-Yellow-Green scale to produce a final color evaluation for each team. The dashboard will be modular in design, allowing users to configure what teams or classes they want to view at a time, and also extensible in allowing additional data sources to be incorporated with minimal overhead. This dashboard will not just be for Dr. Bailey, though. Students will be expected to be active in filling out self-assessments and their effort will be rewarded through similar understanding of their own progress in the program.

QUOTE FROM SPOKESPERSON: “With a culmination of data from a variety of sources, this dashboard gives a holistic overview of team health, addressing both objective metrics such as assignment submission times, and subjective data points regarding team sentiment.”

HOW TO GET STARTED: The intuitive nature of this tool will allow easy integration into Dr. Bailey’s existing workflow of assessing QUEST teams. Using UMD CAS login, he can sign into the tool and view the team health of every team based on our chosen data sources. The dashboard will allow for dynamic filtering of both what data to display and how to display it. If you are a student, simply log in with your own CAS login and fill out the surveys as they come.

CUSTOMER QUOTE: “The Red-Yellow-Green scale displayed on the dashboard is extremely intuitive and gives me an easy way to quickly check up on teams and identify those red teams who need help.”

CLOSING AND CALL TO ACTION: We believe the developer-friendly nature of this dashboard will speak for itself in use and further extension by Dr. Bailey and his students. For further reading on how the dashboard works on a finer level, look through the documentation and see exactly what it can do.

11. Advertising Plan

11.1. As we are fitting in with QUEST infrastructure, our advertisement will be focused around convincing Dr. Bailey and other Quality Guild members of the program’s robustness and efficiency. A major part

of this advertisement will be a "user manual"-style document, intended to demonstrate the functionality and user experience of our dashboard. The manual will expand on documentation by focusing on those who are looking to onboard this product for their own course and may not be as familiar with the specs and benefits. It is important to demonstrate how extensible and modular our product is in order to properly advertise to clients outside of Dr. Bailey's specific use-case. As such, it will be significantly less concerned with code-related details and more on the use. This advertisement document can be developed concurrently with the product itself, save for specific UI-related information (to be accompanied with screenshots).

11.2. The advertisement itself can be found http://128.8.127.113. The download button at the end downloads a zip file containing the pilot code. Instructions for using it are in the documentation button.

12. Prototyping

12.1. Below are prototypes of some of the components detailed above. We believe these prototyping exercises hit all major components of our system and display potential value in our ideas. All of the code is available in */questmetrics/playground/*.

12.2. Slack Module:

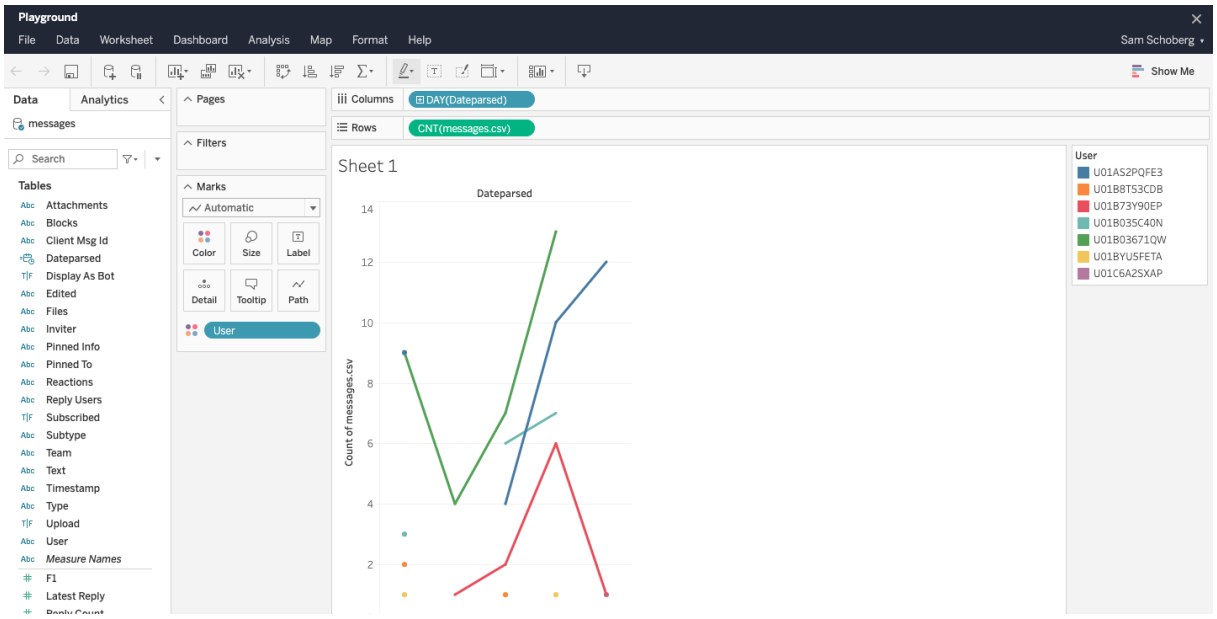
12.2.1. We created a python Slack module to encapsulate the logic to get message and channel history for each team. We learned it will be helpful to modularize each data-grabbing component. It lives in /questmetrics/playground/backend/Slack.ipynb. Using it we were successfully able to retrieve history for channels in our Questmetrics workspace.

12.2.2.

	client_msg_	type	text	user	ts	team	blocks	reactions	bot_id	bot_profile	thread_ts	reply_count	reply_users	latest_reply	reply_users	subscribed	subtype	root	files	upload
0	9658b767-dc	message	meeting not	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich	[{'name': 'exclamation', 'users': ['U01B035C40N'], 'count': 1}, {'name': 'pray', 'users': ['U01B035C40N'], 'count': 1}]												
1	cbbfc145-0a	message	im about to	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich	[{'name': 'heart', 'users': ['U01B035C40N', 'U01BYU5FETA'], 'count': 2}]												
2	f26a3717-b9	message	:beers:	U01B03671C	1.603E+09	T01BD0NFH	[{'type': 'rich	[{'name': 'cry', 'users': ['U01AS2PQFE3'], 'count': 1}]												
3		message	shot	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'image', 'block_i	B01CF8S5FM	{'id': 'B01CF8S5FMK', 'deleted': False, 'name': 'giphy', 'updated': 1602626725, 'app_id': 'A0F827J2C', 'icons': {'image_36': 'https://a.slack-edg											
4		message	shots	U01B035C4C	1.603E+09	T01BD0NFH	[{'type': 'image', 'block_i	B01CF8S5FM	{'id': 'B01CF8S5FMK', 'deleted': False, 'name': 'giphy', 'updated': 1602626725, 'app_id': 'A0F827J2C', 'icons': {'image_36': 'https://a.slack-edg											
5	5e53ae3a-c7	message	Hi team, thi	U01B73Y90E	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'G+o=4', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': "Hi team, this is Sam's friend Jack. Just wanna apologize for that bo													
6	eaf7a2d5-b5	message	Good meetir	U01B73Y90E	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'vHVB', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'Good meeting. Happy saturday.'}]}]}]													
7		message	beer	U01B73Y90E	1.603E+09	T01BD0NFH	[{'type': 'ima	[{'name': '+1	B01CF8S5FM	{'id': 'B01CF8S5FMK', 'deleted': False, 'name': 'giphy', 'updated': 1602626725, 'app_id': 'A0F827J2C', 'icons': {'image_36': 'https://a.slack-edg											
8	bc00e972-1b	message	nevermind i	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich	[{'name': 'raised_hands', 'users': ['U01B03671QW'], 'count': 1}]												
9	5eb5a059-c6	message	we,Äôre thi	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'Tca', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'we,Äôre thinking of doing a second meeting a little later in the week													
10	ed51ff04-87	message	<https://tre	U01B73Y90E	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'o+Z', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'link', 'url': 'https://trello.com/invite/b/h8ZVqhqo/bf03594fa85b7487db17d7b41d4													
11	23df3dcb-bb	message	I'll be on in a	U01B03671C	1.603E+09	T01BD0NFH	[{'type': 'rich	[{'name': '+1', 'users': ['U01B73Y90EP'], 'count': 1}]												
12	0d5c28b7-ff2	message	get out a lit	U01B73Y90E	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'CYv', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'get out a little earlier '}, {'type': 'emoji', 'name': 'slightly_smiling_fac													
13	3ba876c4-be	message	I'm down	U01B035C4C	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': '/CPXI', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': "I'm down"}]}]}]													
14	4797e8f1-58	message	can we just	U01B73Y90E	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'jOqQ', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'can we just hop on now if people who are coming are around?'}]}]}]													
15	da717e7e-c6	message	<https://um	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'LQk', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'link', 'url': 'https://umd.zoom.us/my/rohithv'}]}]}]													
16	f0164d59-11	message	alright	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'VP4', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'alright'}]}]}]													
17	bbed8183-at	message	I think so	U01B035C4C	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'LLy', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'I think so'}]}]}]													
18	74bebd52-8	message	its just with	U01AS2PQF	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'XtL', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'its just with our group right?'}]}]}]													
19	faa85cd6-45	message	we got a lin	U01B035C4C	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': 'EA6SW', 'elements': [{'type': 'rich_text_section', 'elements': [{'type': 'text', 'text': 'we got a link for today?'}]}]}]													
20	22cda6c0-4e	message	Feel better!	U01B03671C	1.603E+09	T01BD0NFH	[{'type': 'rich_text', 'block_id': '9DJ', 'elements': [{'t	1.603E+09	1	1	1.603E+09	['U01B6SDR4	FALSE							

12.3. Tableau:

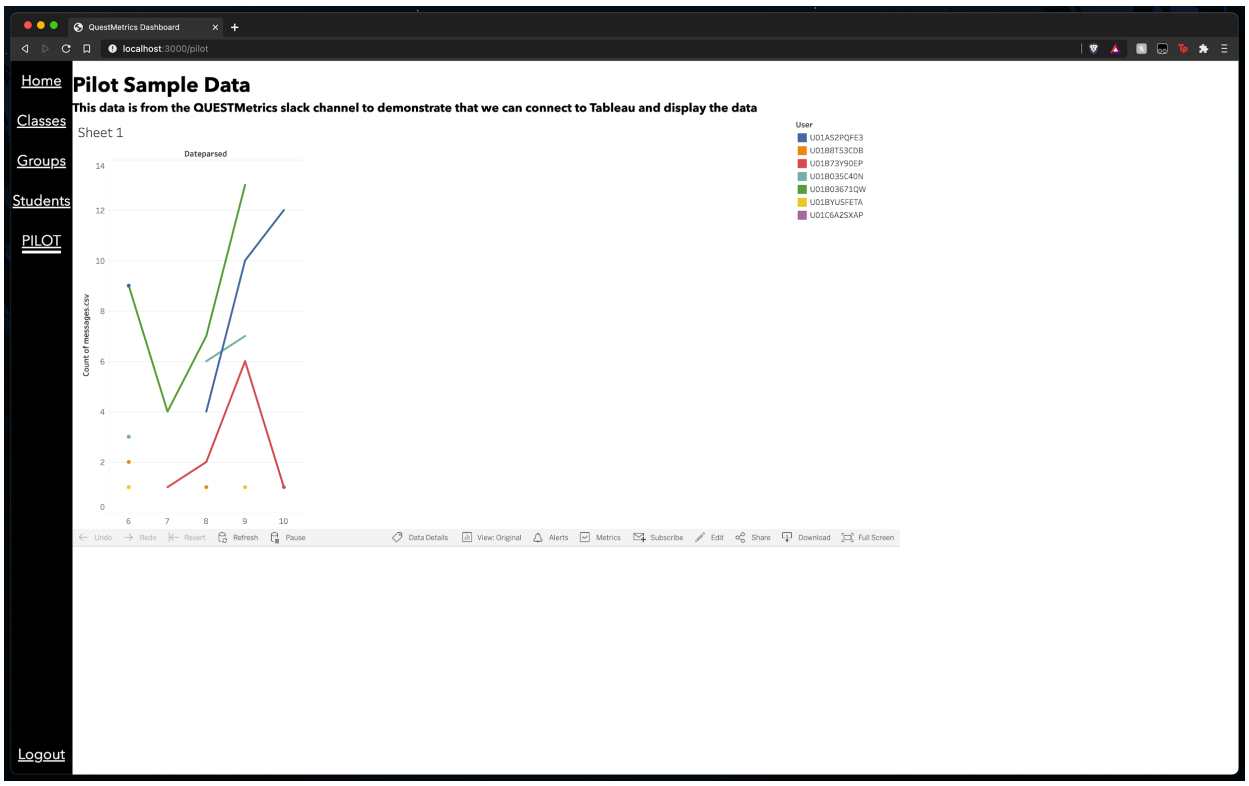
12.3.1. We were successfully able to push the data from the Slack module to a mySQL database on the VM. We were able to connect Tableau online to the database and display table data in a meaningful way (message frequency by user in a channel over time).

12.3.2. 

12.4. Front End:

12.4.1. We were successfully able to embed the Tableau visualizations in a local react-app.

12.4.2. From the root directory you can run npm start to build the project locally. From there you will have to create a Tableau online account to view the data.

12.4.3. 

13. Glossary

13.1. API

Application Programming Interface. Defines interactions between our database and our frontend. Also allows for the data to be used in future projects.

13.2. Canvas

Canvas is a tool built by Instructure that allows instructors to manage their students by publishing assignments, information, and grades, and being able to retrieve assignments for grading.

13.3. CAS

CAS is an enterprise-level single sign-on service that is used by UMD for logging in to a variety of services and applications via a directory ID.

13.4. Cohort

A group of QUEST students who were admitted at the same time and are therefore on the same curriculum schedule.

13.5. ELMS

ELMS stands for Enterprise Learning Management System, which is an enterprise-level flavor of Canvas that is used by UMD to publish courses and communicate with students

13.6. Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

13.7. JSX

JSX is an XML/HTML-like syntax used by React that extends ECMAScript so that XML/HTML-like text can co-exist with JavaScript/React code.

13.8. Locust

An open source load testing tool written in Python. Define user behavior and flood the system. Will be used to stress test the API and Flask infrastructure for acceptance testing.

13.9. MySQL

MySQL is an open-source relational database management system. Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language.

13.10. QUEST

QUEST refers to the Quality Enhancement Systems and Teams Honors Program, which is a three-year interdisciplinary honors program for students in the Robert H. Smith School of Business, Clark School of Engineering, and College of CMNS.

13.11. React

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies.

13.12. REST

Representational state transfer is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the internet.

13.13. Slack

Slack is a channel based messaging collaboration tool for teams. A slack workspace is a group of channels and members. Within a workspace, members belong to different channels

where they can send messages and files to other members within that channel. Channels are often delineated by purpose or team.

13.14. Tableau

Tableau is an interactive data visualization software. Given a dataset, a user can produce dashboards to display different qualities of data. Users are also able to "drill down" to different levels of granularity within the data.

14. References

14.1. ELMS/Canvas API

14.1.1. Documentation: https://canvas.instructure.com/doc/api/

14.2. UMD CAS Login

14.2.1. UMD FAQ: https://umd.service-now.com/kb_view.do?sysparm_article=KB0013650

14.2.2. CAS Documentation: https://apereo.github.io/cas/5.2.x/index.html

14.3. Locust

14.3.1. Documentation: https://locust.io/

14.4. Slack API

14.4.1. Documentation: https://api.slack.com/apis

14.4.2. Postman Setup: https://api.slack.com/tutorials/slack-apps-and-postman

14.5. Flask

14.5.1. Documentation: https://flask.palletsprojects.com/en/1.1.x/

14.6. MySQL

14.6.1. Documentation: https://dev.mysql.com/doc/

14.7. Django

14.7.1. Documentation: https://docs.djangoproject.com/en/3.1/

15. Appendix A

```
CREATE TABLE `people` (
    `personId` int(11) NOT NULL AUTO_INCREMENT,
    `firstName` varchar(255) NOT NULL,
    `lastName` varchar(255) NOT NULL,
    `email` varchar(255) NOT NULL,
    UNIQUE (`email`),
    PRIMARY KEY (`personId`)
);
CREATE TABLE `groups` (
    `groupId` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(255),
    `watch` boolean NOT NULL, (refers to Dr. Bailey's status of monitoring the group)
    `classId` int(11) NOT NULL,
    CONSTRAINT `fk_groups_classId_classes_classId` FOREIGN KEY (`classId`) REFERENCES `classes`
(`classId`),
    PRIMARY KEY (`groupId`)
);
```

```
CREATE TABLE `classes` (
    `classId` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(255),
    UNIQUE (`name`),
    PRIMARY KEY (`classId`)
);

CREATE TABLE `students` (
    `studentId` int(11) NOT NULL AUTO_INCREMENT,
    `uid` int(11),
    `personId` int(11) NOT NULL,
    PRIMARY KEY (`studentId`),
    UNIQUE (`uid`),
    CONSTRAINT `fk_students_personId_people_personId` FOREIGN KEY (`personId`) REFERENCES
`people` (`personId`)
);
CREATE TABLE `student_teams` (
    `studentId` int(11) NOT NULL,
    `groupId` int(11) NOT NULL,
    PRIMARY KEY (`studentId`, `groupId`),
    CONSTRAINT `fk_student_teams_studentId_students_studentId` FOREIGN KEY (`studentId`)
REFERENCES `students` (`studentId`),
    CONSTRAINT `fk_student_teams_groupId_groups_groupId` FOREIGN KEY (`groupId`) REFERENCES
`groups` (`groupId`)
);
CREATE TABLE `slackData` (
    `groupId` int(11) NOT NULL,
    `data` blob, (blob is a mysql type for binary values. Will be used to store the csv file. THIS COULD
CHANGE TO A SERIES OF COLUMNS DEPENDING ON IMPLEMENTATION FEASABILITY)
    PRIMARY KEY (`groupId`),
    CONSTRAINT `fk_slackData_groupId_groups_groupId` FOREIGN KEY (`groupId`) REFERENCES `groups`
(`groupId`)
);
CREATE TABLE `elmsData` (
    `studentId` int(11) NOT NULL,
    `classId` int(11) NOT NULL,
    `data` blob,
    PRIMARY KEY (`studentId`, `classId`),
    CONSTRAINT `fk_elmsData_classId_classes_classId` FOREIGN KEY (`classId`) REFERENCES `classes`
(`classId`),
    CONSTRAINT `fk_elmsData_studentId_students_studentId` FOREIGN KEY (`studentId`) REFERENCES
`students` (`studentId`)
```

```
);
CREATE TABLE `surveyData` (
   `studentId` int(11) NOT NULL,
   `groupId` int(11) NOT NULL,
   `data` blob,
   PRIMARY KEY (`studentId`, `groupId`),
   CONSTRAINT `fk_surveyData_groupId_groups_groupId` FOREIGN KEY (`groupId`) REFERENCES
`groups` (`groupId`),
   CONSTRAINT `fk_surveyData_studentId_students_studentId` FOREIGN KEY (`studentId`) REFERENCES
`students` (`studentId`)
);
```

16. Appendix B

The following research and health questions were recommended to us by the customers. They attempt to create a holistic picture of team health based on responses from individual team members.

Transactive Memory System Scale Items (Lewis, 2003)

https://drum.lib.umd.edu/bitstream/handle/1903/19880/King_umd_0117E_18182.pdf?sequence=1&isAllowed=y

Specialization

1. Each team member has specialized knowledge of some aspect of our project.
2. I have knowledge about an aspect of the project that no other team member has.
3. Different team members are responsible for expertise in different areas.
4. The specialized knowledge of several different team members was needed to complete the project deliverables.
5. I know which team members have expertise in specific areas.

Credibility

1. I was comfortable accepting procedural suggestions from other team members.
2. I trusted that other members' knowledge about the project was credible.
3. I was confident relying on the information that other team members brought to the discussion.
4. When other members gave information, I wanted to double-check it for myself. (reversed)
5. I did not have much faith in other members' "expertise." (reversed)

Coordination

1. Our team worked together in a well-coordinated fashion.
2. Our team had very few misunderstandings about what to do.
3. Our team needed to backtrack and start over a lot. (reversed)
4. We accomplished the task smoothly and efficiently.
5. There was much confusion about how we would accomplish the task. (reversed)

Items evaluated to measure discriminant validity:

Motivation

1. I was highly motivated to perform this task well.
2. My team was highly motivated to perform this task well

Cohesiveness

1. This group worked as a cohesive team in order to complete the task.
2. There was a feeling of 'team spirit' within this workgroup.