Canvas LTI Student Climate Dashboard

DESIGN DOCUMENT

Team: sddec21-19
Client: Henry Duwe
Adviser: Nick Fila
Team Members
Joshua Slagle
Kira Pierce
Andrew Dort
Emma Paskey
Zach Borchard
Team Email: sddec21-19@iastate.edu
Team Website: http://sddec21-19.sd.ece.iastate.edu/
Revised: 04/23/2021

Executive Summary

Development Standards & Practices Used

Below is a list of all the standards that we will use. These are a lot of today's industry standards for creating software. We will be following these standards very closely as a team. This will give us the ability to write software that is easy to make constant changes quickly across all platforms and push out the product quickly to the consumers.

- TDD
- Agile
- Unit test / Acceptance test
- CI/CD
- Docker
- Mutation Testing
- SOLID Principles
- Kurbenetes

Summary of Requirements

1.1.1 Functional Requirements

- Professor should be able to view class Journey Map
- Professor should be able to view individual students' Journey Maps
- The system should be able to pull data directly from the Canvas API
- The system should be able to accept and use data directly provided to it from a user
- Students should be able to view their own Journey Map
- The system should be able to predict how a group of students with given personas will react to an existing class outline
- Professors should be able to create their own data/feedback metrics
- Professors should be able to create a new Journey map
- Professor should be able to create groups of data together
- Professors should be able to view journey maps with only a specific set of variables taken into account
- The system should create personas on a per-course basis
- The system should be able to convert data into Journey Maps
- The professor should be able to interact with the Journey Maps
- The professor should be able to inspect data at a more granular level through some action

1.1.2 Non-Functional Requirements

- Data integration should be modular for future extensions
- Student's data should not be accessible by other students
- All students, TAs, and Professors should be able to access the application with no crashes
- The system should be modular
- The system should be easily extensible
- Data Storage should not violate FERPA
- The system should use open source datasets in the absence of actual data for training (i.e. sentiment analysis)
- The system should be able to use mock data
- The system should be accessible at all times
- A journey map should be cold constructed within 60 seconds.
- A journey map should be warm constructed within 5 seconds.
- The system should update real time

1.1.3 User Interface Requirements

- The user interface needs to utilize color
- The user interface color should be customizable
- Design Layout to be accessible (Ex:color blindness)
- The UI needs to be interactive (multiple journey maps at once, toggle items on/off)
- The UI should be customizable ^
- The UI needs to be resizable and accommodate multiple screen sizes
- The UI should be able to show particular student subsets
- The UI should be able to zoom in/out.
- The UI should show an appropriate level of information with respect to the zoom level.
    - The zoomed in view should display data at a more granular level
    - The zoomed out view displaying things should be more abstract

Applicable Courses from Iowa State University Curriculum

Along with the basics (Com S 227, Com S 228, and other introductory courses) the following courses were highly applicable to our project:

- COM S 309 - Software Development Practices
- COM S 363 - Introduction to Database Management Systems
- COM S 362 - Object-Oriented Analysis and Design
- SE 319 - Construction of User Interfaces
- SE 329 - Software Project Management
- SE 339 - Software Architecture and Design

New Skills/Knowledge acquired that was not taught in courses

- Docker (Containerization)
- Kubernetes (Container Management)
- Istio (Service Mesh)
- Helm (Infrastructure Setup)

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Acknowledgement

A very special thanks to our advisor Nick Fila and client Henry Duwe for giving us technical assistance in regards to Canvas data curating and project requirements, in addition to being extremely helpful and encouraging in the entire design process.

1.2 Terms and Definitions

Persona - An overarching behavioral type driven by a defining set of characteristics. Used to generalize a group of similar students and model their responses to crafted scenarios.

Journey Map - A visual representation of a student's experience throughout the duration of the class. The x-axis represents time, discretized as a series of events. The y-axis represents the resonance of the class with the student.

Resonance - The level of impact the class has on a student and their academic or professional career. Note that this can be positive or negative.

1.3 Problem and Project Statement

The problem we are addressing is Canvas's lack of extensive data analysis tools for measuring and visualizing the impact courses have on students. Specifically, Canvas lacks the support to automatically collect student input from climate surveys and show trends in resonance throughout a semester. Our client wants one place to easily track the success of his students to help him improve his current and future classes.

Our solution to this is to implement a web-based journey map that will analyze feedback given by students, alongside their grades and the time it takes them to turn in assignments, to display how well they are progressing and emphasize trends that make the class better or worse for different types for students. This will also include the ability to do automatic resonance prediction and rudimentary data analysis for the client's use.

1.4 Operational Environment

This product will operate in an online environment and need to be built securely to reduce risk of data breaches. Specifically, the computer software will be hosted on a VM provisioned to us by ETG running Ubuntu 20.04. Inside of the Linux VM, we will be running the container orchestrator Kubernetes to spin up and scale our application's pods (containers). It is assumed that the user has Docker installed on their computer to be able to run the local web page / database cache. This assumption is conservative and will not be needed if we can push all containers and data processing pipelines to the VM, from which the webpage would be accessed by means of a web browser such as Chrome. This migration is our goal, but is pending an ongoing investigation into the very specific requirements FERPA has for student data BITS and virtual machines on non-locally owned hardware.

1.5 Requirements

We have three sections of requirements here: the traditional Functional Requirements, the traditional non-functional requirements, and then a section dedicated specifically to the user interface requirements since that portion is a large area of focus for our application.

1.5.1 Functional Requirements

- Professor should be able to view class Journey Map
- Professor should be able to view individual students' Journey Maps
- The system should be able to pull data directly from the Canvas API
- The system should be able to accept and use data directly provided to it from a user
- Students should be able to view their own Journey Map
- The system should be able to predict how a group of students with given personas will react to an existing class outline
- Professors should be able to create their own data/feedback metrics
- Professors should be able to create a new Journey map
- Professor should be able to create groups of data together
- Professors should be able to view journey maps with only a specific set of variables taken into account
- The system should create personas on a per-course basis
- The system should be able to convert data into Journey Maps
- The professor should be able to interact with the Journey Maps
- The professor should be able to inspect data at a more granular level through some action

1.5.2 Non-Functional Requirements

- Data integration should be modular for future extensions
- Student's data should not be accessible by other students
- All students, TAs, and Professors should be able to access the application with no crashes
- The system should be modular
- The system should be easily extensible
- Data Storage should not violate FERPA
- The system should use open source datasets in the absence of actual data for training (i.e. sentiment analysis)
- The system should be able to use mock data
- The system should be accessible at all times
- A journey map should be cold constructed within 60 seconds.
- A journey map should be warm constructed within 5 seconds.
- The system should update real time

1.5.3 User Interface Requirements

- The user interface needs to utilize color
- The user interface color should be customizable
- Design Layout to be accessible (Ex:color blindness)
- The UI needs to be interactive (multiple journey maps at once, toggle items on/off)
- The UI should be customizable ^
- The UI needs to be resizable and accommodate multiple screen sizes
- The UI should be able to show particular student subsets
- The UI should be able to zoom in/out.
- The UI should show an appropriate level of information with respect to the zoom level.

- The zoomed in view should display data at a more granular level
- The zoomed out view displaying things should be more abstract

1.6 Intended Users and Uses

The most important end user is the professors that will use this to reflect on their class delivery, both as a whole and as separate components. For example, a professor could tell by the journey map that, although the personas of the students may have varying degrees of success in the class, no one is struggling or holds bad sentiment towards it and its delivery. Or on the other hand, say one persona appears as having low - or negative - resonance towards the class. The instructor can now click on the specific events that drove this persona's resonance down to get a breakdown of what went wrong. This can be thought of as similar to the debugging of a classroom experience.

1.7 Assumptions and Limitations

Assumptions:

- We will have a dummy Canvas class to test our extension on.
- The end product will be used at Iowa State University.
- Users will not be extremely technically proficient.

Limitations:

- We likely will not be able to use real student data for testing.
- End product will need to be completed within the time frame of the Senior Design course.
- We do not know what information professors outside our client would like to collect to use within their journey maps.

1.8 Deliverables

At the end of next semester, we will have produced for the client:

- A Functional UI that displays an interactive journey map that details a set of personas' resonance scores and reactions to class events (quizzes, assignments, exams, etc).
- A module that can group students into a set of personas based on their Canvas data and instructor notes.
- A data processing module that can predict, to the client's satisfaction, how certain personas react to class events.
- A customizable set of data analytic pipelines that the instructor can choose to use or modify to predict persona resonance.

Each of the modules will be held in container(s) and can be easily handed over - along with the network description on how they interact - to the client at the end of the year.

2 Project Plan

2.1 Task Decomposition

In order to solve the problem at hand, it helps to decompose it into multiple tasks and subtasks and to understand interdependence among tasks. Our tasks, laid out below, are grouped into four main categories based on the underlying functionality of the layer. The infrastructure layer focuses on the application's running location, initial configuration, and inter-application communication. The backend layer focuses on the structure and storage of all the application data. Heavily interconnected to the backend, but necessarily distinct, is the data analysis intermediate layer. This layer focuses on the abstracted retrieval of data from the backend (to promote a modular application) and a collection of similarly modular modules that can act on this data in pipeline format.

The frontend layer focuses on displaying the data from the data analysis intermediate layer and interacting with the user in a pleasant fashion. Finally, the Canvas Integration layer is the layer that actually interacts with Canvas to pull down the current class and student information. This is separate from the backend setup because this data is distinct and separate from our application data. Thus, it is modified outside of our application and we cannot track changes made without pulling in the data each time.

The Canvas LTI Student Climate Dashboard (CLSCD) will be considered complete when the following tasks and subtasks are considered complete:

- Infrastructure Setup
  - Kubernetes Set Up on Machine
  - Service Mesh set up for pod-to-pod communication
- Backend Setup
  - Databases Designed
  - Databases are setup (User Information, Saved Filters, Course Data, Student Data)
- Data Analysis Intermediate Layer Complete
  - Data Retrieval Modules Implemented and Tested
  - Data Processing Modules Implemented and Tested
- Frontend Implementation
  - Pages Designed
  - Pages are Implemented
    - § Base Layout
    - § Displaying Data Analysis Layers
    - § Interactive Features Enabled
- Canvas Integration
  - Modules for pulling different types of data down from Canvas created
  - Method for Caching – and deleting – data retrieved from Canvas designed and implemented
  - Modules for retrieving data from cache implemented

2.2 High Level Design

2.2.1 System Description - High Level

At a high level, our system will behave as shown in Figure 1., the high level conceptual sketch. We will have two databases, one online and one local. The online database (labeled here as *Non Student Application Data)* will be for application information such as settings, login information, and instructor notes. In the local *Canvas Cache*, we will have data pulled from Canvas for easier retrieval later. Each database will have a module(s) for accessing the tables/schemas, with a second, intermediate *Data Interface Layer* acting as an adapter between the rest of the application and the databases. The *Data Processing Layer* will consist of all modules related to the persona creation, resonance prediction, sentiment analysis, and supporting modules. All of these feed into the Journey Map creator used by the user interface and can provide information directly to the UI.

2.2.2 Conceptual Sketch

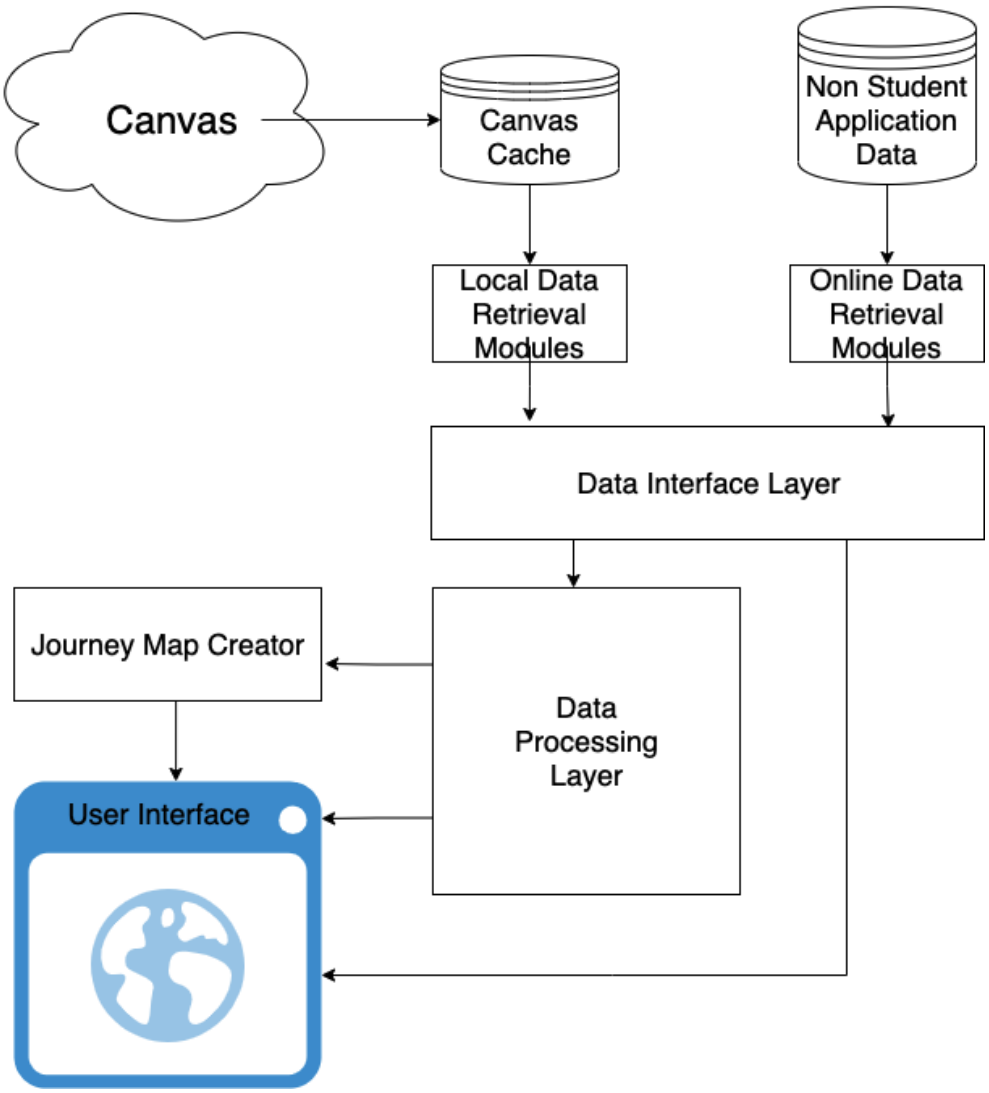

Figure 1. - Conceptual Sketch

2.3 Risks And Risk Management/Mitigation

We've explored a ton of risks, their mitigations, which overarching task they affect on our project plan, and an estimated probability of the risk happening in the table below. One specific risk, not listed in the general list, is that we rely on the dataset curated for us by our advisor and client to model and train our resonance prediction model. If their efforts fall short of what we need – which is doubtful, because they've been great so far – we will use the StudentLife Dataset provided by Dartmouth College to get the bare minimum skeleton requirements of: grades, classes, and Piazza interactions we need (Wang, Rui, et al. "StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones."). From there, our team would put ourselves in each persona's shoes and craft textual responses to complete our dataset.

Task	Risk	Mitigation	Estimate Probability
Infrastructure Complete	Cannot acquire VM from ETG to host Application.	Can self host (No cost), use a cloud provider (AWS, Azure, etc), or grab another machine from IT.	0.05
Infrastructure Complete	Infrastructure Machine becomes inaccessible/goes down for outside reasons.	Create Cluster/Machine using the Infrastructure-As-Code approach to make the entire cluster portable.	0.10
Infrastructure Complete	Infrastructure Machine does not have outside Internet access to setup cluster with.	Can setup a cluster without internet access (pain) or work with IT to gain the correct domain whitelist.	0.01
Backend Setup	Data gets deleted out of database.	Keep backup copies in Git.	0.20
Backend Setup - Design	Exact data we will be hosting is not known until late in development.	Break up the data into tables/schemas that use primary keys to connect data we know will be in there to any data added later.	0.75
Data Analysis Intermediate Layer	Exact data we will be hosting is not known until late in development.	Create stubbed data retrieval methods that can be easily replaced once data is known.	0.75

Frontend Implementation	UI requirements change late in the game.	Making use of the Agile methodology this should not be as big of an issue. Modular design of the frontend components in tandem with the data analysis layer abstraction should allow us to flip displays easily.	0.50
Canvas Integration	Learning curve of Canvas API is greater than expected.	Create a spike to investigate the Canvas API as a dedicated task.	0.50
Canvas Integration	Canvas Rate limiter makes it mathematically impossible to meet time requirements.	Investigate the possibility of parallelizing the data collection from different IP addresses or cleansing HTTP headers.	0.15
Canvas Integration	Data received from Canvas API is not cleansed.	Create an additional module to cleans data from Canvas – extra, unplanned work.	0.40
Canvas Integration	Some desired data cannot be retrieved using the Canvas API.	Create an additional data-scraping module – extra, unplanned work.	0.20
General Risks	Source Code is deleted from the repository.	Backup git repository to local machine or another git repository nightly.	0.05
General Risks	Git repository becomes mangled because of bad push.	Backup git repository to local machine or another git repository nightly.	0.50
General Risks	A team member becomes gravely ill or is no longer able to complete work.	Enforce team documentation procedure to ensure handoff of work is smooth and unburdening.	0.03

Table 1. Risks & Mitigations

2.4 Project Proposed Milestones, Metrics, and Evaluation Criteria

Our project has 11 key milestones, some of which can be worked towards in parallel by different team members. These milestones, along with their completion criteria, are listed below in no specific order of completion other than following the task list order in section 2.1.

Milestone	**Measurement**
Kubernetes Cluster is set up (Infrastructure)	Pods and Services are able to spin up on VM.
Service Mesh is Set Up (Infrastructure)	Pods and Services are able to communicate through the chosen service mesh.
Databases are Designed (Backend)	Each possible data point expected can be logically mapped to a database table/schema.
Databases are Implemented (Backend)	All desired queries can be successfully run against the logical database.
Data Retrieval Modules Implemented (Data Analysis)	Entire collection of data retrieval modules return, without error or being stubbed, the correct information as determined by unit tests from a test database 100% of the time.
Data Processing Modules Implemented (Data Analysis)	Entire collection of data processing modules return, without error or being stubbed, the correct information as determined by unit tests from a test database 100% of the time.
All Pages are Designed	The rough go-ahead has been received in writing from client to implement page sketches.
All Pages are Implemented	Pages appear correctly and interactively as confirmed in writing by client for each page.
Canvas Data can be Pulled	All data listed in 'Required Data' Document can be pulled (or scraped) from the Canvas API or page as confirmed by 100% test coverage.
Canvas Data can be Cached and Retrieved from Cache	By visual inspection. Data exists in cache and can be retrieved as confirmed by passing 100% of tests on the test Canvas page.

Entire Application is up and running	Two-part criteria: 1) All tests comprehensively pass throughout the application. 2) User has tested the application thoroughly and has expressed their thoughts on completion in writing.

Table 2. Milestones & Measures

2.5 Project Timeline/Schedule

This Gantt chart assumes a working day has 4 hours instead of 8 to account for the fact that we are in school with other classes, but also work on the weekends.

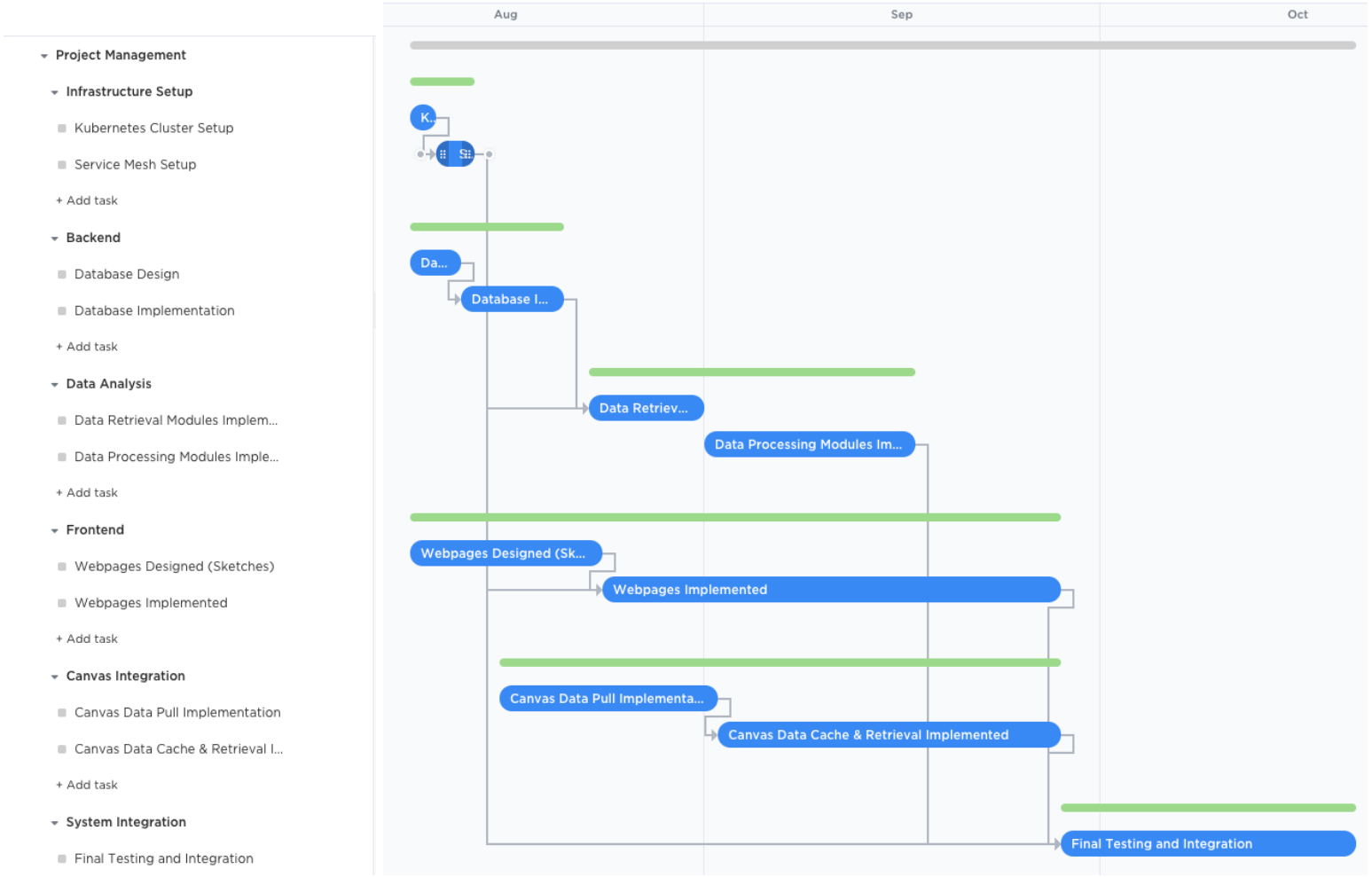

Figure 2. Gantt Chart

2.6 Project Tracking Procedures

Our team, using the Agile approach, will use Trello to keep track of our tasks and sprints for the entirety of the project. Our actual work history and versioning will be recorded and kept track of in the GitLab provided by ISU. Work that is not in the form of source code, infrastructure setup, or container images (e.g. requirements document, design documents, meeting notes) will be kept in our team's Google drive. Communication related tasks (meetings, decisions, and general conversation) will be done through Slack for text and Zoom for video.

2.7 Personnel Effort Requirements

2.7.1 People-Hour Estimation

To estimate the number of hours to develop the project, we will use mainly team experience in similar projects as encouraged by M. Nasir in sections 2.1.1 and 2.13 of the paper "A Survey of Software Estimation Techniques and Project Planning Practices"(M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices"). To apply these methods, we first broke down the application into the tasks in Table 3, then applied our combined experience to estimate people-hours from the bottom up. This is similar to the 'story-pointing' in Agile teams but with hours.

Task	**Description**	**Textual Reference**	**People-Hours**
Kubernetes Cluster is set up (Infrastructure)	Set up a basic Kubernetes cluster on the Virtual Machine provided by ETG.	Our members have done this many times. From experience, we know that this task can be completed in three hours or less including debugging random issues. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	3
Service Mesh is Set Up (Infrastructure)	Enable a service mesh to act as the basic communication tool between pods.	Our members have done this many times. From experience, we know that this task can be completed in four hours or less including debugging random issues. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	4
Databases are Designed (Backend)	All database tables/schemas are designed on paper and ready for implementation.	From project experience, our team knows that it takes around 15 hours to design (well) a database with ~ 5 tables.	15

		(M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	
Databases are Implemented (Backend)	All database tables/schemas are implemented in the chosen database engine.	Team Experience applied here. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	24
Data Retrieval Modules Implemented (Data Analysis)	All queries for data needed from the backend are created in modular methods and tested to be correct.	Team Experience applied here. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	30
Data Processing Modules Implemented (Data Analysis)	All data processing functions needed for data analysis are created in modular methods and tested to be correct.	Team Experience applied here. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	60
All Pages are Designed	All web pages desired sketched out in moderate detail and given the go ahead by the client.	Team Experience applied here. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	50
All Pages are Implemented	All pages appear correctly and interactively as confirmed in writing by the client for each page.	Team Experience applied here. From experience it takes around 30 hours to build a complex webpage and 10 to build a simple page. Using 2 complex and 3 simple pages we get our estimate. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	120
Canvas Data can be Pulled	Functions are written to pull data from the Canvas API and web-scrape if needed.	Team Experience applied here. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	60
Canvas Data can be Cached and Retrieved from Cache	Functions and endpoints are created so that cached Canvas data can be accessed from web requests.	Team Experience applied here. (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices")	80

Table 3. Person-Hour Estimate

2.7.2 Initial Inadequate Method

An attempt was made to estimate the effort and time using the COCOMO method; however, the results obtained were wildly different from what the team knows to be true. Specifically, we tried to use the following approach:

- Start with the functional point analysis technique to generate the UFP
- Adjust the UFP for complexity
- Compute the lines of code from programming average
- Use the COCOMO method to estimate time

This method gave wildly large numbers that deviate quite significantly from what the team knew to be reality. For example, our experience tells us that it does not take 4.3 thousand lines of code to write a basic backend that performs 25 different queries, five of which are complex. We also know most certainly that it does not take six months with two developers working on it full time; we have accomplished similar backend work within a few days in our own ComS 363 class. We believe this discrepancy is due to the fact that COCOMO was initially created after studying dozens of projects in the industry, where the environment is much different than it is here.

Our sentiments appear to be backed up by leaders in the industry, as in this article: https://lwn.net/Articles/659241/. Thus, we elected not to proceed down this route and rather continue to use past projects as a baseline, as recommended by (M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices"), as detailed in section 2.7.1. Below is a copy of our Value Adjustment Table initially attempted for reference.

Value Adjustment Table when attempting the COCOMO:
0 - No Influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 - Essential

Value Adjustment Factor	0	1	2	3	4	5
The system requires reliable backup and recovery.					X	
Specialized data communications are required.	X					
There are distributed processing functions.	X					
Performance is critical.			X			
The system runs in an existing, heavily utilized operational environment.	X					

The system requires on-line data entry.				X		
The on-line data entry requires transactions over multiple screens/operations.	X					
ILFs are updated on-line.		X				
The inputs, outputs, files or inquiries are complex.			X			
The internal processing is complex.			X			
The code is designed to be reusable.						X
Conversions /installation are included in the design.			X			
The system is designed to facilitate change and ease of use.					X	
TOTAL	25					

Table 4. Attempted COCOMO VAF Table

Thus, our Product Complexity Adjustment Factor (CAF) would have been $0.65 + (0.01 * 25) = 0.90$.

2.8 Other Resource Requirements

For operation, a virtual machine running Ubuntu 20.04 has been requested from ETG for the remainder of the year to run our Kubernetes Cluster and host our application. For modeling and training purposes, we have requested a set of 50-60 'Mock Students' from our client/advisor that they have kindly offered to supply.

2.9 Financial Requirements

This section is not relevant but kept for completion. No financial requirements.

3 Design

3.1 Previous Work And Literature

There are similar products that exist in the market, largely specializing in business and marketing needs. UXPressia is a significant company in this market; their product allows users create multiple, phased journeys and descriptive personas, assign personas to one or more journeys, collaborate with other team members in real-time, create impact maps, and generate high-resolution graphics of their work (UXPressia, 2020).

The key difference between the proposed project and similar tools in the marketplace is that this project is specific to an academic setting and is meant to utilize student feedback. Many of the

collaborative features utilized in software on the market are not relevant to this project and may pose security risks regarding student data. Additionally, this project intends to help professors organize and identify patterns in student feedback. This is done with the goal of improving the student experience in a course, especially in terms of course resonance.

In the academic world, there is another analysis tool created by Michigan called MyLA - or My Learning Analytics (*My Learning Analytics Support*, University of Michigan) - that is currently being advertised by Unizin, an organization that deals with the digitization of the university experience. Our tool provides two key features that MyLA does not:

- Our tool provides predictive analytics instead of just statistical analysis of Canvas Data
- Our visualization is much more rich and inviting than MyLA's. Where they have numerical values and bar charts, we will have an interactive journey map that displays trends much more easily. The three figures below show their three visualizations.

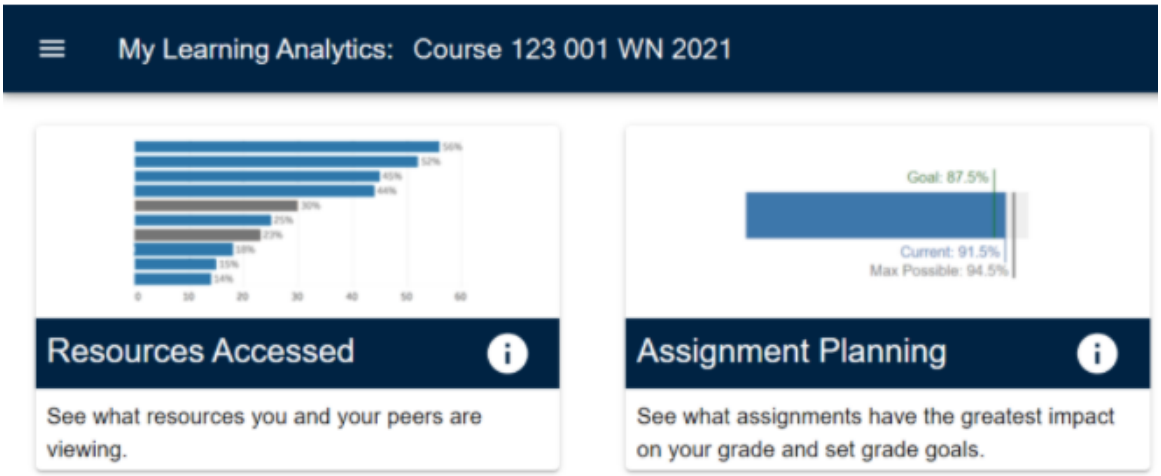

Figure 3.1 - MyLA Resources Accessed

Figure 3.2 - MyLA Assignment Planning

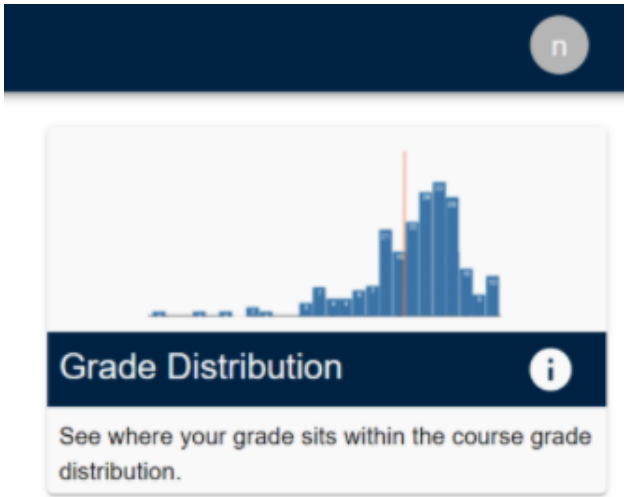

Figure 3.3 - MyLA Grade Distribution

After completion, the journey map on our UI will look similar to the journey map below from Heart of the Customer ("Our Journey Maps Reveal What Customers Do and Why." *Heart of the Customer*), providing a completely different, and more in depth view of the class progression.

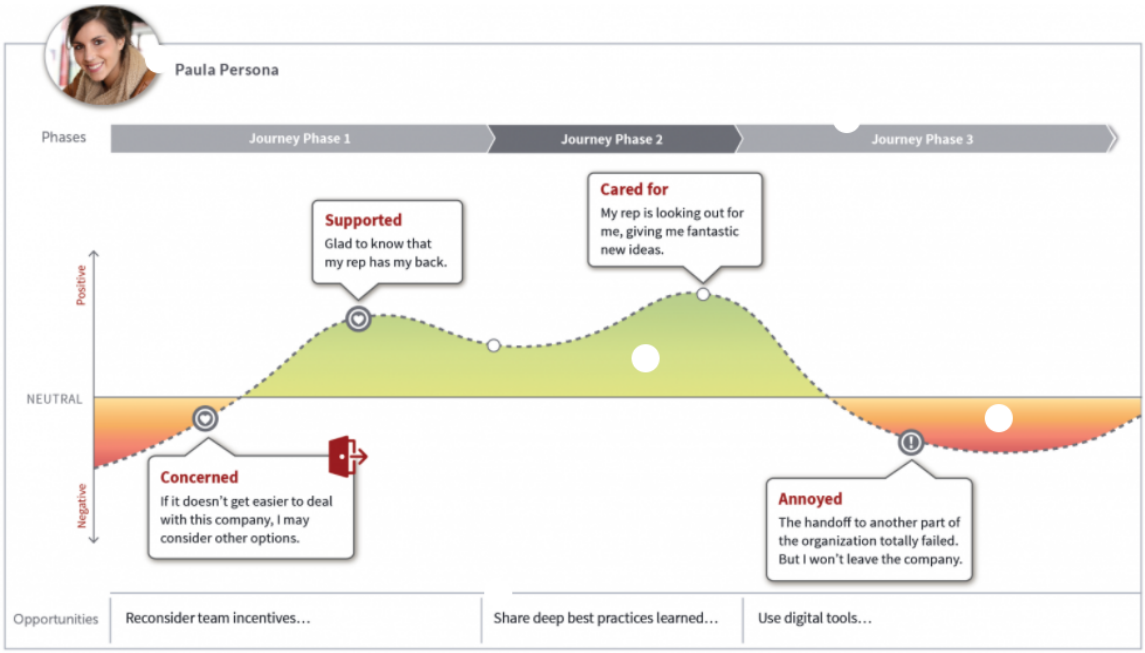

Figure 4. - Example Journey Map

These differences give us our niche between the academic product and the corporate product that we feel very comfortable operating in.

3.2 Design Thinking

There are several defined requirements that shape the project's design. First, there are four identified users with different needs: professors, students, TAs, and administrators. A number of needs exist for these groups.

Students:

- A way to provide feedback to the instructor, so that opportunities to improve the class are created.
- A modification functionality that allows students to edit their feedback.
- An accessible system, so that students with disabilities can use it.

Professors:

- A way to send out surveys so that student feedback and responses can be collected.

Admins:

- Ability to view survey results so that they can gauge class climates in real time.

One of the tools that we used to identify and enumerate our design thinking process was the lotus blossom chart. The way this tool works is that main ideas are gathered and written down into the center of each cluster. Both as a team and with our client and advisor, we each brainstormed possible solutions/key factors for each of the main ideas and brought our individual responses into discussion. This gives an easy way for everyone to have their voices heard on the issues and let everyone evaluate the possibilities. An example of one of the idea clusters is shown below in Figure 5.

Represented by a "main idea" that encompasses a wider group, avoids having a persona for each student (exaggerated)*	Role in context	Key challenges (course, curriculum, general)	List of students in each persona (weighted)**
Represent a persona by a vector living in n-space. Similarity can be determined from functions (Cosine similarity?) or from classifiers (SVM?)*	Representation and aspects	Motivations (course, career, personal) ****	Attributes and sentiment tokens (observable behaviors on specific assignments, activities) *
Represent a persona as a normalized vector in n+1 space so that it lies on a sphere. Similarity can be determined from functions (Cosine similarity?) or from classifiers (SVM?)	Represent a persona as a set of attributes. Intersection of these sets above/below a certain threshold can determine similarity (Jaccard Similarity?) *****	Common behaviors (third-person), experiences (first-person)*	Cluster Centroids**

Figure 5. Lotus Blossom cluster

This was one of many clusters that the team worked on. The information that we gathered using this tool helped immensely with our design thinking process and we were able to fully understand all of the goals that the client had in his vision of a final application, most notably in key areas such as persona representation, methods of predicting resonance, and linking students to personas.

3.3 Proposed Design

The completed program will operate as both a local application on the user's computer (for sensitive student data processing) and as a distributed system (for where we can offload tasks/data storage). The application connects with the Canvas API and requires successful user authentication

and correct authorization to access student data. This decision is projected to be more secure, cost effective, and inline with FERPA standards than direct Canvas integration or hosting an external database.

We choose to go with a microservice-based architecture for the following key reasons:

- Our application consists of many small, core functionalities that can easily be broken down into independent subtasks.
- Microservice architectures are very easy to extend, which is important for this software's adopted use across different colleges and departments.
- Scalability becomes very easy if many professors use our application at the same time.
- Microservices are easily fit into containers which are, by construction, operating system agnostic and ephemeral.

This proposed design is visualized using a system diagram and functional breakdown, seen in Figure 6.

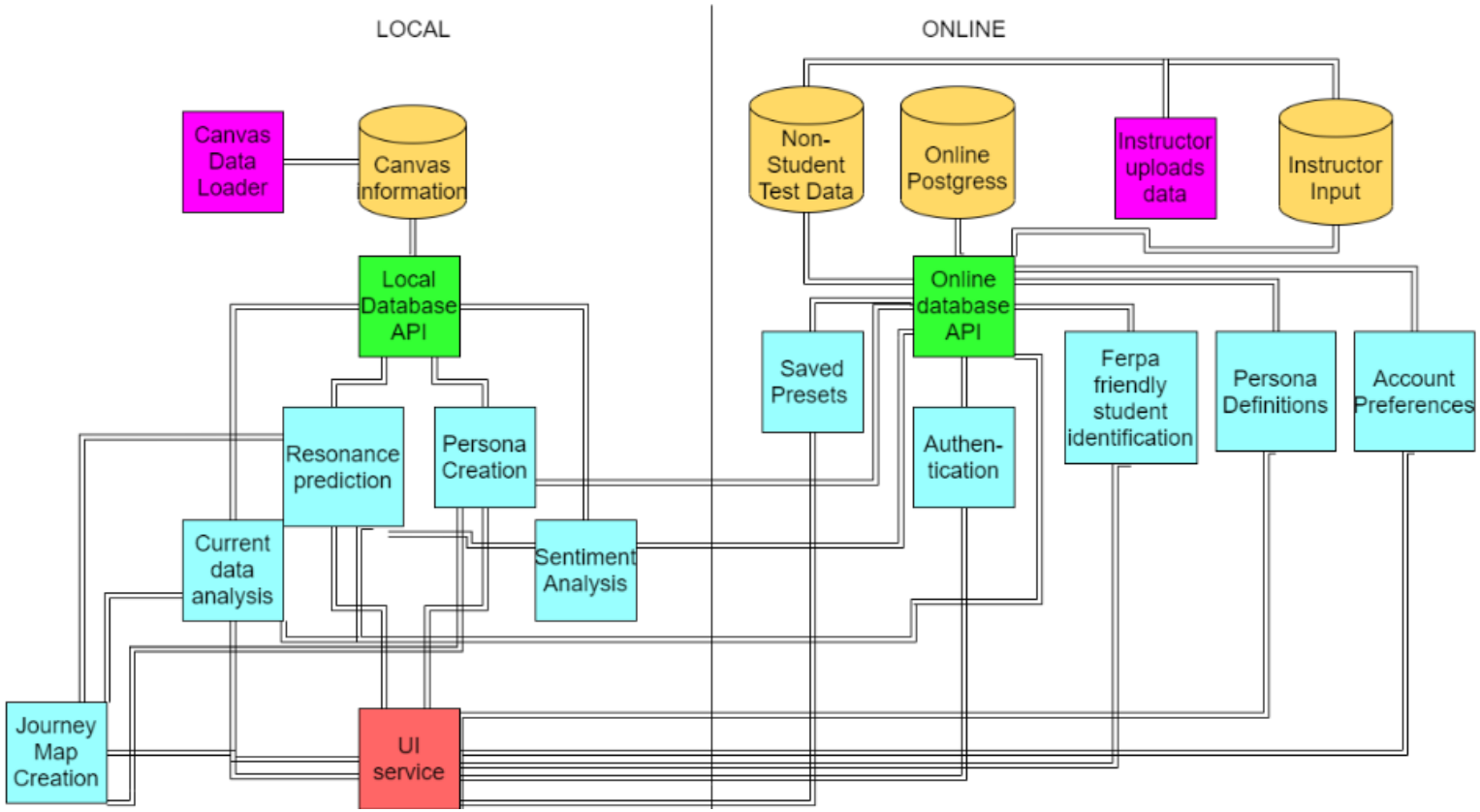

Figure 6. System Diagram/Functional Decomposition

A further breakdown of each type of module found in Figure 6:

- Gold cylinders are databases; each database holds specific information and is stored either online or locally based on the information stored.
- Purple blocks are input sources to the databases. The Canvas information database gets information from the Canvas loader API, while the instructor input goes to either the test data or the instructor input database.
- Green blocks are database APIs. These will act as the in-between for the databases and the data modules.
- Blue blocks are data processing modules. Each block requests data from either other data modules or the green API blocks to perform its task.
- The red module is the final step in the operation, and compiles all data into a viewable UI format to present to the user.

3.3.1 Detailed Design

3.3.1.1 System Description

The application consists of many modules, each providing their own function. Each module is independently designed, but can depend on the output from other modules. Expanding from the high level system description given in the beginning of the document, we will have two databases, each with their own data access API module. These modules' sole purpose is to perform queries on the database/schemas to return results.

Modules that interact directly with these database APIs alone, such as the Saved Presets, Authentication, FERPA Friendly Student Identification, Persona Definitions, and Account Preferences modules, are a part of the Data Retrieval layer that was aforementioned in section 2.2.1. These modules deal with the retrieval of specific data and are broken up into components for modularity, scaling, and updating on-the-fly purposes.

The Data Processing modules consist of the Resonance Prediction, Persona Creation, Sentiment Analysis, and Current Data Analysis modules also straddle the data interface layer line as they interact with both database APIs. The key here is that each module will consist of multiple parts, with the module itself being broken down into an interface layer and processing layer. This is done so that each module can be updated independently of the others.

Finally the Journey Map module and UI Service transparently map to the Journey Map creator and User Interface respectively. Each only performs one overreaching function and so are kept in their own separate, but single, module. All actions by the professor are driven through this UI module. The majority of the display - the Journey Map - is constructed for the UI in the Journey Map module, which directly pulls upon the Resonance Prediction, Persona Creation, and Sentiment Analysis Modules. Only the Canvas Data Loader module is kept out of this dependency chain, operating on its own to keep the Canvas Data up to date by polling periodically the Canvas API.

3.3.1.2 Functional Decomposition & High-Level Design

Because we went with the microservice architectural approach, each functional module in Figure 4. also corresponds to one of our functional decompositions (in fact, the functional decomposition came before we created the diagram). The full functional decomposition list is given below, with their functionality described shortly after.

- Canvas Data Loader - On initialization, pulls the entire set of relevant data (statically set) from the Canvas API and pushes it to the local cache.

- Local Database API - Provides a RESTful interface for the cached database.
- Resonance Prediction - Predicts a set of personas' resonance with a class outline. Because this is a complex module, the high level steps this module will take are:
    - First, requests the filter and pipeline information from the online database for the logged in user.
    - Then, requests Canvas Data from the local cache API and Instructor notes from the online database API
    - Next, the pipeline instructions are applied to the retrieved dataset and returned as a score for each persona.
- Persona Creation - Maps a group of students to a set of predefined personas given their Canvas data and instructor notes. The option to automatically define a persona will be an option and implemented as a stretch goal.
- Sentiment Analysis - Takes in a string of text and produces a sentiment score for the group of text. Additionally, this module will return the key words that were used in scoring the text string.
- Current Data Analysis - Compiles the different types of current class data (grades, turn-in times, etc) and maps it to different personas based on user-defined preferences.
- Journey Map Creation - Takes in a set of personas, resonance score sets, and a class outline to produce a JSON description of the Journey Map in the UI module's standard.
- UI Service - The modules that serves the webpage for the user interface.
- Saved Presets - Provides a RESTful interface for the saved presets stored in the Online Database for each account.
- Authentication - Provides a RESTful interface for the authentication credentials stored in the Online Database for each account.
- FERPA Friendly Student Identification - Provides an interface to an anonymous mapping function from Student ID's in Canvas to Instructor notes in the Online Database for each class.
- Persona Definitions - Provides an interface to the persona definitions stored in the Online Database for each class.
- Account Preferences - Provides an interface to the account preferences stored in the Online Database for each account.

3.3.1.3 Functional Module Design

The communication between the different modules will be done through JSON producers and consumers, and the communication between the databases and the APIs will be done through TCP/IP SQL statements. An example of this communication between the functional modules is shown below in a use case diagram of the resonance prediction module in Figure 7.

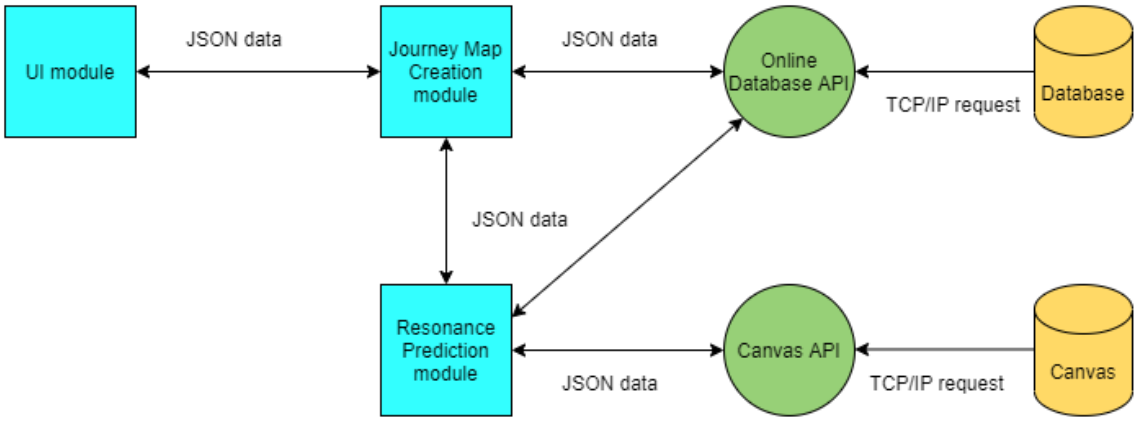

Figure 7. Use Case Diagram with Data Flow

3.3.2 Application Considerations and Focuses

The main element of focus for our application is charting how the course resonates with various groups of students. The application will provide instructors with student feedback and other indicative data already available in Canvas. Instructors can then create user personas with their own descriptions and exert additional control on how data is evaluated on a empathy map: control over axes, what data sets are currently evaluated, identify important statistics (maxima, minima, delta and other trends), and the ability to manipulate data with custom scripts/functions. A hierarchy of empathy maps may be useful for viewing how students responded to different aspects of the course in addition to the course as a whole. Some of these features, such as the ability to write a function for evaluating data, may be inaccessible or overwhelming to some instructors. The application will keep advanced analysis features separate from simpler tools.

The program's empathy maps, personas, and data analysis features must connect instructors to the students' experience in class and provide insight for future course planning. Being able to compare empathy maps across semesters and possibly years is important, and should allow instructors to track how course changes over time have affected student resonance.

3.3.2.1 System Requirements

As noted in the operational environment section, we hope to be able to migrate all modules over to the virtual machine pending the investigation into detailed FERPA requirements. Our design stays the same either way, only which modules are run locally and which on the VM would change. Thus we have two cases: 1) Where we have both the local and VM systems to worry about; 2) We only need to worry about the VM.

<u>If we have two systems that we have to worry about, then we have two sets of system requirements:</u>

Locally, the user must have Docker Installed on their machine. This introduces additional requirements that, if on Windows, the user be running Windows 10 1903 or higher and that the machine must have a 64-bit processor with Second Level Address Translation. For Mac Users you must be running the OS of version 10.14 or newer. All users must have at least 4 GB of system RAM on their machine.

The VM must be able to run Kubernetes. This requires the system to have at least 2GB of RAM and a minimum of 1.5 CPU cores. These requirements are satisfied by the VM we have checked out.

If we can migrate all to the VM:

We no longer need to worry about the local machine, and the VM requirements stay the same. These are repeated here for completeness: The VM must be able to run Kubernetes. This requires the system to have at least 2GB of RAM and a minimum of 1.5 CPU cores. These requirements are satisfied by the VM we have checked out.

3.3.3 User Interface Description

The user interface will, after logging in, consist mostly of a journey map, with various menus and drop downs available for the user to customize what they see. Below is our conceptual sketch of this page. In the center, the graphs with blue and red lines are two example journey maps with the x-axis being time and the y-axis representing persona resonance with the class. Each of the grey circles represents an event - assignment, exam, lab - that has affected the persona's resonance score, and can be clicked on to expand into more detail. For (a very simple) example, if a quiz had a free-response question "What are your thoughts about this class after taking this quiz", clicking on the grey circle would return the average sentiment scores for this persona along with the keywords that determined the sentiment score.

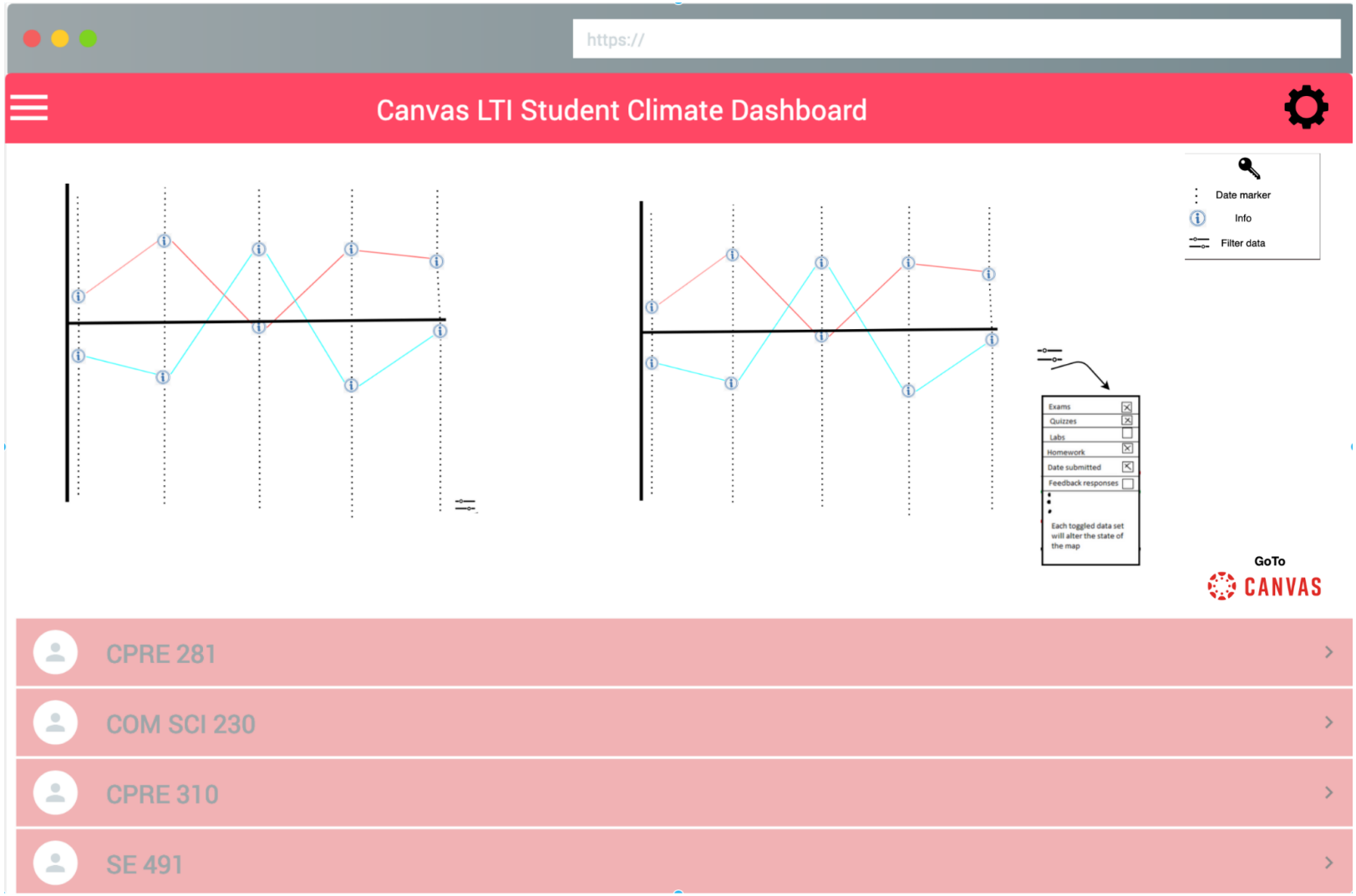

Figure 8. Conceptual UI Sketch

Below is an expanded view of the checklist in the bottom right hand corner of the main page. This checklist filters out the types of assignments that are included in the resonance score for each persona. For example, in Figure 9. below, the *Labs* box is not checked, indicating that the Lab data from the class should not be included in the resonance score for the personas. This box can be subsequently checked by the instructor, which would refresh the journey map to include the new data.

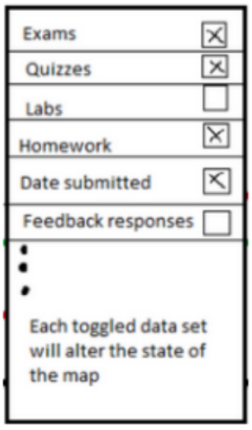

Figure 9. Conceptual UI Sketch - Enlarged Checklist

In the top left corner of the main page (Figure 8.), there exists a hamburger menu button. When clicked on and expanded, the options below appear.

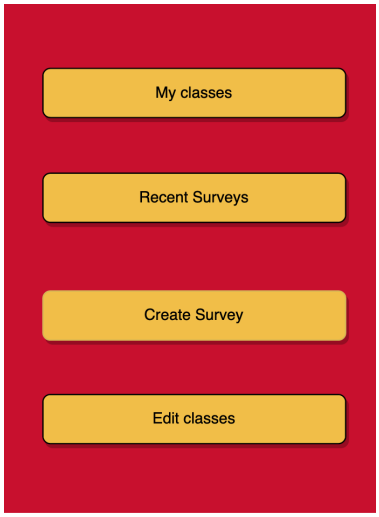

Figure 10. Conceptual UI Sketch - Enlarged Hamburger Button

Each button corresponds exactly as expected with respect to the text appearing on it. For example, the *Recent Surveys* button will show a list of surveys ordered most recent to least recent in time. The figure on the next page relates the functionality of each UI component shown above to the Conceptual Sketch talked about in section 2.2. In this figure, the arrows show the flow of data and the solid lines without arrows indicate the area is being enlarged.

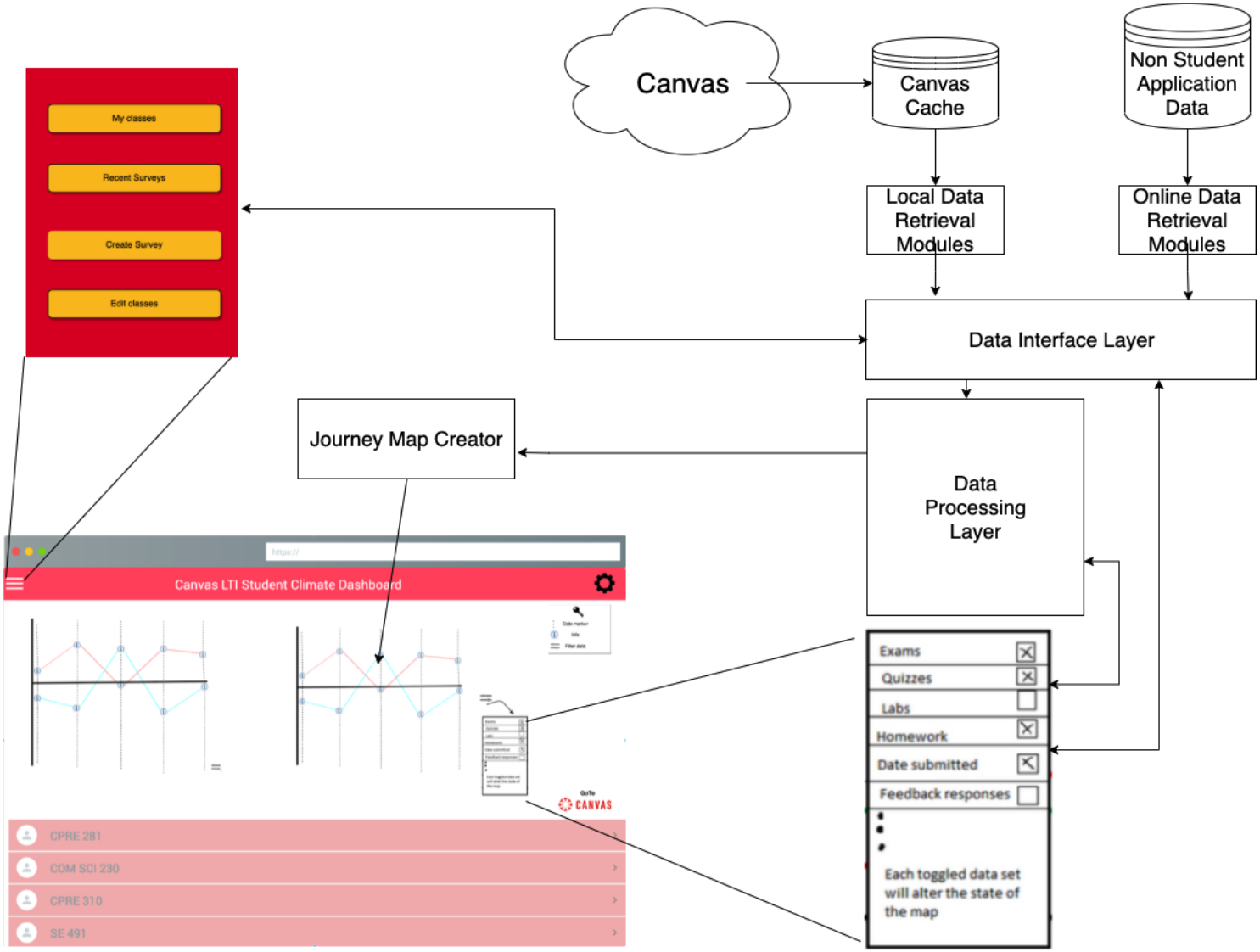

Figure 11. - Conceptual UI Interactions

3.4 Technology Considerations

3.4.1 Technical Constraints

The client imposed no programming language, operating system, libraries, or frameworks that we had to use. Instead, our technical constraints are derived from the non-functional requirements that we discovered while talking to the client. Specifically, in our non-functional requirements we had six key items that drove our constraints:

- Data integration should be modular for future extensions
- The system should be modular
- The system should be easily extendible
- Data Storage should not violate FERPA
- A journey map should be cold constructed within 60 seconds.
- A journey map should be warm constructed within 5 seconds.

In addition, we took into account the programming language experience of the team for speed of execution. The requirements that our system be modular and extendible naturally drove us to choose microservices for our architecture; although, in our decision making process, we heavily considered MVC and Microkernel as well. In addition, our team is familiar with a wide range of languages: C, C++, C#, Python, Javascript, Java, the .NET framework just to name a few. Using microservices allows us to take advantage of this, since each microservice can be written in a different language. The space in our conceptual design where the microservice architecture becomes really apparent is boxed in blue in Figure 10. below.

The other big driver was the security concern for FERPA. This means that we cannot just pull all of the student data and store it in our online server with our application data. However, the bottom two of our listed non-functional requirements deal with speed, so we cannot pull from Canvas every time we wish to update or perform a calculation. Instead, we opt to do a full pull of the data we will use at the start and cache if for later, giving us the red box in Figure 10. above..

3.4.2 Design Decisions

The most significant trade-off may be that a locally-hosted program would most likely be built from scratch, although there are templates we can use for local web pages in Docker Hub. Additionally, an important concern is the lifetime of the software application. Because security is of high importance, the application will be structured in a way that any dependencies it uses are easy to manage and update without much intervention.

Our choice of microservices as main architecture allows for the leverage of exciting technologies such as containers (specifically Docker containers), container orchestrators such as Kubernetes, and Infrastructure as Code solutions (IaC) such as Ansible and Istio. The underlying attractiveness of these technologies to us boils down to three large categories:

1. IaC allows our system as a whole to be highly portable, testable, and more secure.
2. Containers allow our modules to be (mostly) operating system agnostic for high portability and easy testing.
3. Container Ephemerality allows us to spin up and delete entire databases quickly and securely so that sensitive data does not live on any computer longer than it needs to.

In addition to these three categories and as mentioned in the previous paragraph for web pages specifically, we can leverage existing containers to build upon so each component does not necessarily need to be built from scratch. Good example of this is the Postgresql databases readily available on Docker Hub.

Some limitations that these technologies impose are:

- Relatively hard to get a UI out of a container without creating the application as a local web page. This limits us to web-page technology instead of a Matlab/simulink type application that is very well suited for heavy data processing.
- Microservices, because the intercommunication is message based (TCP/UDP) instead of function calls (RAM/memory based), incur an unavoidable overhead when many separate modules are involved. If we are running into runtime constraints this will need to be mitigated, however the overhead is usually minimal enough to only be noticeable in real-time systems.
- Security for microservice based systems is sometimes a concern because of the heavy traffic flow between all of the modules. This allows for network sniffers to get their hands on data and creates many endpoints for possible breaches. However, recent technologies - if taken advantage of - reduce, if not completely mitigate, these concerns. For example:
    - The Service Mesh IaC solution Istio encrypts ALL messages (by means of Lets Encrypt) between modules and keeps the keys safe in lockdown. This makes the communication between modules as secure as normal web traffic can get.
    - Istio also uses a centralized database to ensure 1) any Module A has the right to request the said data from Module B before the request is sent out to B 2) any Module B that receives a request from Module A checks to make sure it can handle requests from Module A. This secures the endpoints.

3.5 Design Analysis

Our proposed design, although some refinement down the road should be expected (as any healthy project has) will be successful if implemented as designed.What we have has been discussed thoroughly with the advisors and the team many times and follows practices used by many top tech companies (think Netflix).

As an Agile process is to be followed, we will have a refinement at the end of every sprint. During this refinement we will discuss what can be repointed or what needs to be done / issues we didn't see when originally starting. In addition, as we go along implementing modules, we will have weekly discussions with our client consisting of short demos to create a short-circuit feedback loop. This is the best way to go about modifying / iterating over our application and has been shown in the literature (L. Chen, "Continuous Delivery: Huge Benefits, but Challenges Too,").

3.6 Development Process

We will be following the Agile approach for our software development. This is the best way to go for our project for two reasons: 1) We have the opportunity every week to talk with our client 2) The client is still in the process of figuring out exactly what kind of end product is desired. . This short feedback loop, combined with our microservice architecture, will allow us to utilize TDD in a CI/CD

pipeline to ensure the correct functionality and limit integration woes down the road. All of our sprints will be planned out and story-pointed in Trello where we can view and assign story cards. We will be following the classic Agile format of two week sprints with a design, build, test, review, and launch cycle as shown in Figure 12.

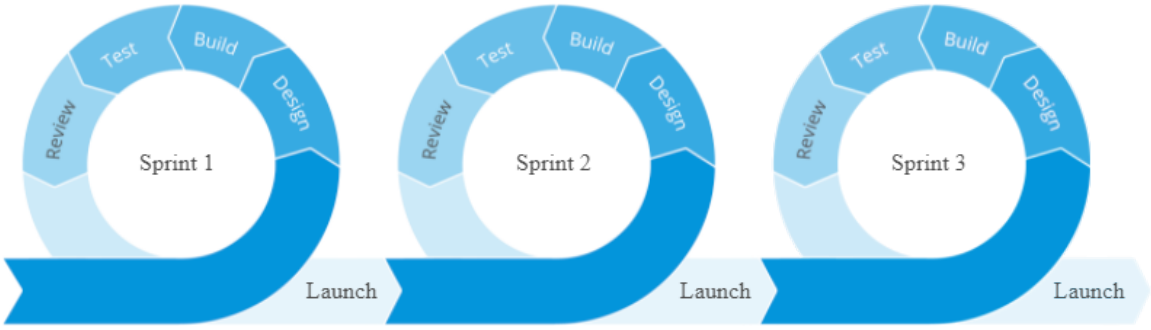

Figure 12. Agile Development Cycle

3.7 Design Plan

We plan to have two sections in our microservice architecture: 1) Locally run modules 2) Distributedly run modules. Each portion will have their own standalone database to house its respective data and a web of interconnecting modules to process requests made by the user through the UI. In addition, the locally hosted portion will have modules specifically for interacting with the Canvas API to pull/scrape data, push comments/quizzes, and any other desired functionality discovered down the road.

4 Testing

While we have not yet started development on the project, it is still good practice to look ahead and make plans for types of testing tools and methods of testing that will be implemented for the life cycle of the project.

With the current plans, it makes the most sense to employ unit testing on all of the back-end code that makes the application run, and GUI testing on the web interfaces that the customer will be using. These tests will be run in a regression-testing style, with a CI/CD pipeline through git to make sure that all code that is published to mainline is working, bug-free code. Acceptance testing will also be a regular facet of the project to make sure that the continued development of the project stays close to the original vision from the customer, and to make any changes necessary before unusable code gets propagated too far into the project.

Since the development has not yet started, there are no current concrete test cases for any component of the project. This will be one of the first things we do when development starts to follow a test-driven development cycle as closely as possible.

4.1 Unit Testing

The main section of the project that will undergo unit testing is the back-end driving code for the project. The specific type of unit testing framework, like JUnit, will be decided once the language for the project is decided. The goal is to have this section of the project be modular so that future development and integration is straightforward without changing large parts of pre-existing code. Unit testing is important in this case so that when new microservices are implemented into the project, both new and old unit tests can be run on the entire project to ensure that nothing has been broken with the addition of new code. With test-driven development, this process gets even easier since the test cases are created before the code is, so the code is much more likely to be correct on the first iteration.

4.2 Interface Testing

Our project has a large emphasis on the user interface; therefore, interface testing will need to be a main focus of the project. With a powerful graphic user interface testing software like Selenium, we will be able to create iterative interface tests on our GUIs to make sure that the intended functionality of specific interfaces is met, and that deploying new interfaces to the project does not interrupt the functionality of previous interfaces. Test-driven development and CI/CD is harder to achieve with GUI testing, but it will still be important to create a comprehensive suite of interface tests that are run on a regular basis for the project. At this point it is impossible to enumerate all of the interfaces that our project will need since it is a very interface heavy project.

4.3 Acceptance Testing

In order to achieve successful acceptance testing, we will need to present new working builds to the client whenever possible to ensure that the project is staying in line with the original vision. This is most easily accomplished through scheduled meetings with clients to show them new builds and record feedback for aspects of the project that will need to be changed for the next iteration. Our team has already established a regular weekly meeting schedule with our clients, so acceptance testing should be able to run smoothly once the project has started development.

4.4 Mutation Testing

In order to make sure that we are truly testing the code and not just writing trivial tests for code coverage, we will be adding mutation testing to our codebase. The mutation testing will mutate the codebase and make sure that our tests are truly testing the codebase. This will significantly improve the test quality as well as making our codebase less error prone.

4.5 Performance Metrics

Most of the tests must be run automatically to stop us from having to allocate QA resources to the project for manual testing.

Another major performance metric will be our total code coverage. After passing mutation testing this should be right around 100% to make sure all the codebase is covered.

The main metric we will need to test for is ensuring data integrity. We need to make sure that the mock data that is coming in is processed exactly as expected. There will be lots of data in our application and we need to make sure that it's correct for generating our persona maps.

One final major metric we will track is for test efficiency in defect finding. We want to make sure we aren't just writing tests to keep code coverage up, but also find bugs in the code before the code is deployed.

4.5 Results

At this time, the only testing that has been done is the creation of mock user interfaces for the project. These ideas were presented to the clients, and feedback was recorded to make sure that the team has a clear idea of the vision that the clients have for the final project. Using this feedback, we have been able to further refine our design process, which will make future decisions about the direction of the project much easier.

5 Implementation

5.1 Basic Building Blocks

Our basic building blocks of our application are our modules which are shown, in detail, in our System Diagram (Figure 2.). Each module will be housed in its own container and spun up on Kubernetes to facilitate the interpod communication. The main modules that will take the most work and require the most attention will be:

- Resonance Prediction Module
- Sentiment Analysis Module
- Persona Creation Module
- Web-Hosted UI Module

Currently, we have a working base prototype of the sentiment analysis pipeline with core functionality.

5.2 Familiarity with Tools

We are fortunate to have members of the team that are very familiar with containers (and their orchestration) and new, more complete, testing procedures. Our team as a whole has extreme familiarity with:

- Docker
- Kubernetes
- Istio
- Mutation Testing
- Python
- Javascript
- Postman
- Git
- CI/CD pipelines
- Linux CLI

all of which will be heavily used throughout the duration of the project. Specifically, for previous internships, we've had multiple members work with or on CI/CD pipelines, Git, Docker, Linux CLI systems, and Javascript. In addition, we've had at least one member have extensive experience with Docker, Kubernetes, Istio, mutation testing, and Python.

6 Closing Material

6.1 Conclusion

We've done all of the design work for our system. In addition to all of the work this document shows, we have also already set up our Kubernetes cluster, implemented a rudimentary version of our sentiment analysis pipeline, and have done extensive research into the technologies we are going to use to implement our system. By the end of next semester we will have implemented and be handing off to our client:

- A Functional UI that displays an interactive journey map that details a set of personas' resonance scores and reactions to class events (quizzes, assignments, exams, etc).
- A module that can group students into a set of personas based on their Canvas data and instructor notes.
- A data processing module that can predict, client satisfaction accuracy, how certain personas react to class events.
- A customizable set of data analytic pipelines that the instructor can choose to use or modify to predict persona resonance.

To do this, we will use the Agile methodology and Test Driven Development approach beginning immediately at the end of this semester following through the end of the Fall semester. Each task that we will be completing via the Agile methodology will be timeboxed by the time shown in the Gantt chart in Figure 2. This is the best plan of action because much of our effort will be spent on the user interface and custom functionalities, which heavily depend on the client's opinion to consider complete. Using the Agile approach will ensure that we receive constant feedback and enable consistent communication throughout the development of the application.

6.2 References

UXPressia. [UXPressia]. (2020, May 21). UXPressia - Customer Journey Platform Demo (2020) [Video]. Youtube. https://www.youtube.com/watch?v=YXyLJisQOxc

L. Chen, "Continuous Delivery: Huge Benefits, but Challenges Too," in IEEE Software, vol. 32, no. 2, pp. 50-54, Mar.-Apr. 2015, doi: 10.1109/MS.2015.27.

M. Nasir, "A Survey of Software Estimation Techniques and Project Planning Practices," Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06), 2006, pp. 305-310, doi: 10.1109/SNPD-SAWN.2006.11.

Wang, Rui, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. "StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones." In Proceedings of the ACM Conference on Ubiquitous Computing. 2014.

My Learning Analytics Support, University of Michigan, 12 Jan. 2021, 3:44 P.M., its.umich.edu/academics-research/teaching-learning/my-learning-analytics/support.

"Our Journey Maps Reveal What Customers Do and Why." Heart of the Customer, heartofthecustomer.com/customer-experience-journey-maps/.