Headway

Tom McKernan mckernant1@udayton.edu
Alex Jennison jennisona1@udayton.edu
Tyler Berkshire berkshiret1@udayton.edu
Corey Reichel reichelc1@udayton.edu

University of Dayton

Department of Computer Science

CPS 491 - Capstone II, Spring 2019

Instructor: Dr. Phu Phung

Project Management Information

Trello: https://trello.com/b/8iRguS52/headway-season-2
Web App Source: https://bitbucket.org/4bitboys/capstone-ii-web-app
Api Source: https://bitbucket.org/4bitboys/capstone-ii-classroom-api
Website: https://headway.dev

Revision History

Date	Version	Description
9/05/2019	0.0	Init draft
9/27/2019	0.1	Sprint1 draft - adding axios mocking and api services, updated to vuetify 2.0
10/25/2019	0.2	Sprint2 draft - adding live quizzes! Additional styling updates
11/27/2019	1.0	Sprint3 draft - added stats api! Added live test api

Overview

Headway is an easy to use educational web application which allows users to test their knowledge of subject material in a classroom setting and can facilitate discussion amongst users for deeper understanding of concepts. This application can also be used as a study tool by users to quiz themselves on classroom knowledge. This function of knowledge solidification is paired with the ability for users to review their progress on a given concept, lecture, or subject, through the use of Headway's visual analytics. The goal of Headway is to foster discovery and discussion in an educational manner congruent to neurologically advantageous learning methods.

Project Context and Scope

Our project consists of developing a web application that is a tool for students and educators which provides a simple and easy to use environment where quizzes can be taken and viewed. This will require customers to create accounts to be able to form a structured grouping of quizzes, that we will refer to as a class. Every user can participate as both a “student” in some classrooms and as a “teacher” in others. These classes can be shared with other user accounts directly with a variety of available settings of permissions to the members of the classrooms.

Quizzes are divided into two main categories: live and static. Live quizzes are those which must be taken by students in a limited time period, presumably while a lecture is in session. The class creator will open the quiz for completion, often one question at a time, and give their students a unique code to enter the classroom and complete the assessment. This method is useful for traditional lecture discussion amongst a classroom and allows for flexibility from the instructor to add or remove questions on the fly. Static quizzes are those which are posted by the class creator to be taken by students over an extended period of time. These are more akin to the traditional online quiz, where the instructor posts the entire quiz and students have several days to complete the assessment. This method is useful for prolonging beneficial study habits and encouraging extensive review of subject material.

The class creator can add live quizzes to specific class sessions or static quizzes to specific classrooms. All quiz results can be reviewed by any students who have taken them and can also be retaken for further study and review. Each classroom is identified by a unique code which allows only members of the classroom to view its quizzes. While the class creator can view all quizzes and student results of a classroom, students can only view quizzes and results of assessments they have participated in.

System Analysis

This project is split into two distinct development parts. Firstly, there is a front-end web application in Vue.js that is highly interactive for user experience. The main functions of these webpages is to allow students and teachers to create accounts, setup classes, develop quizzes, and review previous materials. Vue.js’s component framework allows us to split these functions easily between our team while maintaining a seamless development cycle. The web application is responsible for acting as a interface for the user to communicate with Headway’s REST API.

The REST API is our main back-end system in combination with its attached database. The API utilizes Spring and Kotlin to serve as a seamless connection between user requests and database accesses. The database serves to hold all account information and any quizzes created or taken by each account. It is constructed in a non-relational scheme, specifically using JSON string objects. All together, the user will make a request, the REST API will access the database, and the web-application will display the results of such a request.

Another function provided by the Headway web service is a data visualization engine for users to view their quiz reports over a period of time. This involves creating graphs and charts for the user to view and customize based on theme, class, quiz, subject, etc. The graphical engine is built into the web application with short JavaScript and Python scripts. Another smaller REST API is utilized to channel data between the database and this simple visual engine, but it is implemented in a similar manner to the core back-end system.

The Live Quiz buffering system is a Socket.io app that creates rooms for each live quiz. Professors can choose a question to display and have the students respond to that question. That question will display on the students phones and laptops for them to answer. Professors can change the visibility of a question at any point in time.

SECTION ABOUT HOW OUR STATS API WORKS

High level requirements Requirements

It is imperative that Headway have functional classroom, this inculdes the ability to:

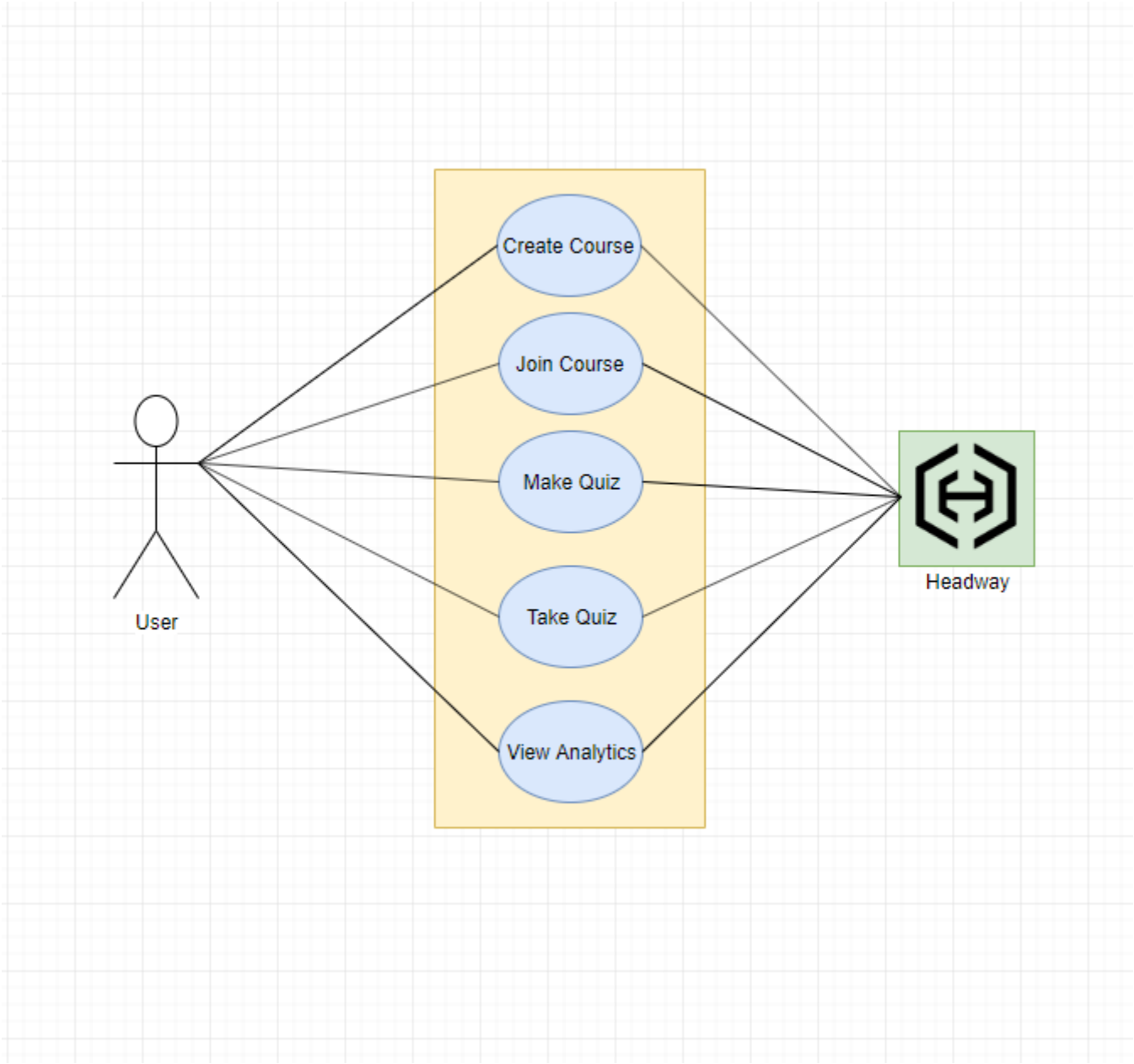
A. Add new students

B. Manage test and quizzes

C. Upload additional resources

D. Provide in depth performance analytics to owners and students alike.

Use cases



Use Case Name:	Create Course	
Scenario:	User wants to create a course.	
Triggering Event:	User presses “Create Course” button.	
Actors:	User	
Related Use Cases:	Course Settings	
Preconditions:	User has an account and is logged in.	
Post Conditions:	User owns a new course in their account.	
Flow of activities:	User presses “Create Course” button.	System displays new course settings.
	User fills out course settings.	System prompts for submission.
	User submits.	
Exception Conditions:	1. Incomplete course settings information. 2. User does not submit.	

Use Case Name:	Join Course	
Scenario:	User wants to join a course to participate in quizzes.	
Triggering Event:	User presses “Join Course” button.	
Actors:	User	
Related Use Cases:		
Preconditions:	User has an account, is logged in, and a course exists in the system.	
Post Conditions:	User joins a course.	
Flow of activities:	User presses “Join Course” button.	System prompts for Course ID.
	User enters Course ID.	System adds user to course.
Exception Conditions:	1. Invalid course code.	

Use Case Name:	Make Quiz	
Scenario:	User wants to make a quiz.	
Triggering Event:	User presses the “Make Quiz” button.	
Actors:	Admin User	
Related Use Cases:		
Preconditions:	User has made an account and is logged in and has sufficient permissions.	
Post Conditions:	A new quiz has been created in the user’s classroom.	
Flow of activities:	User presses button to make a quiz.	System displays the type of questions
	User fills in questions and answers.	System saves input to database.
	User can now take or share the quiz.	
Exception Conditions:	1. User fails to provide complete information. 2. User does not save. 3. User does not have sufficient permissions.	

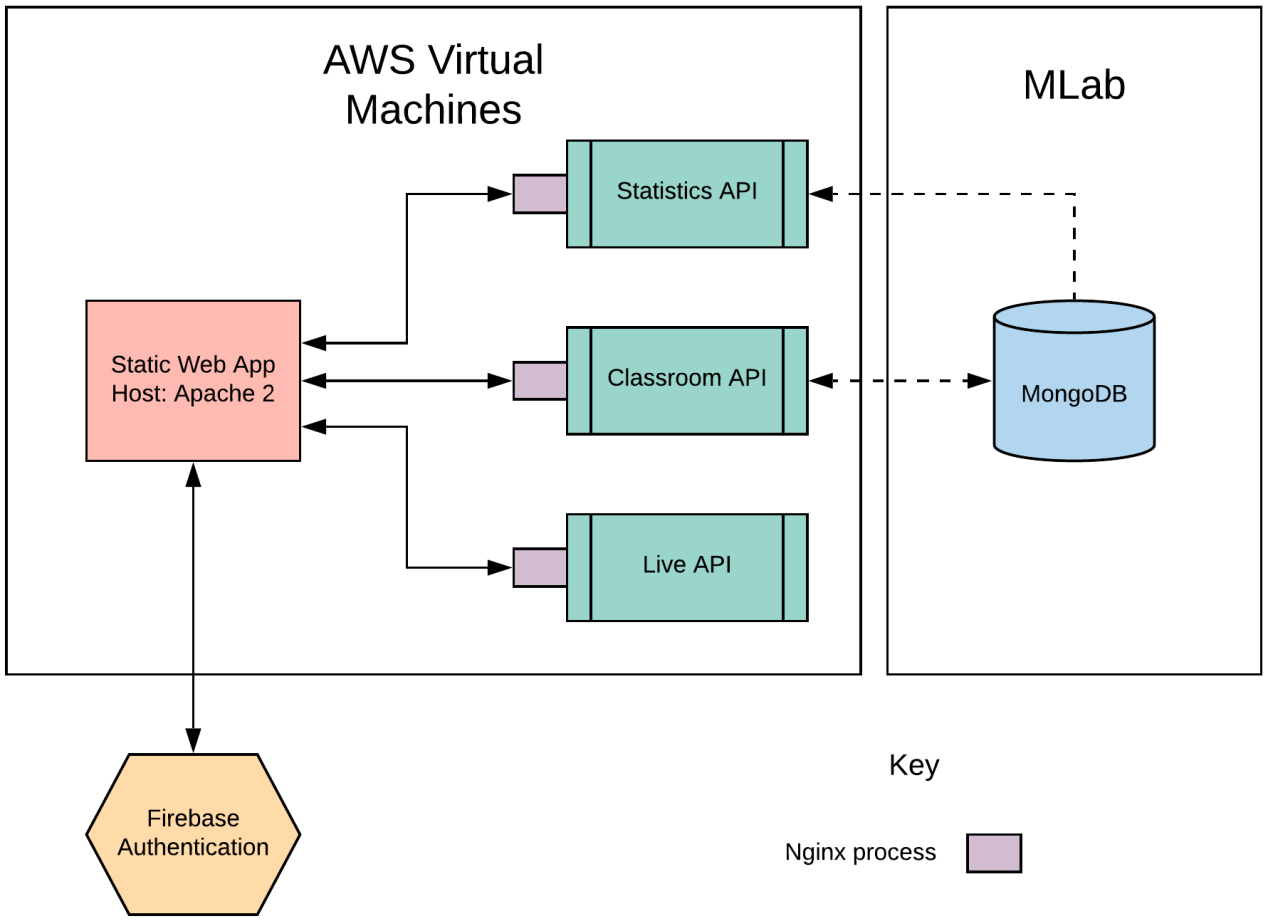
Use Case Name:	Take quiz
Scenario:	User wants to take a quiz.
Triggering Event:	User presses the button to take a quiz.
Actors:	User
Related Use Cases:	
Preconditions:	User is logged in.
Post Conditions	Quiz results are recorded.
Flow of activities:	User presses button to take quiz. System gives the user a quiz to take. User takes quiz. System saves results.
Exception Conditions:	1. Actor does not answer all questions.

Use Case Name:	View Course Analytics (Teacher)
Scenario:	User wants to see various Statista about the class.
Triggering Event:	User selects Class analytics.
Actors:	User
Related Use Cases:	View quizzes, Class Overview
Preconditions:	User is logged in and a classroom admin for the given course.
Post Conditions:	User will be prompted further for what type of specifics they are looking for.
Flow of activities:	User selects Class Analytics. System Prompts for Various domains in Class. User browses further for further selections. System provides charts etc. to reflect data.
Exception Conditions:	1. User is not logged in. 2. User does not have proper permissions.

Use Case Name:	View Quiz Analytics (Student)
Scenario:	A user wishes to view data and analytics about a specific quiz taken in class
Triggering Event:	Student Role User selects View Class Analytics.
Actors:	Student role user, Class admin
Related Use Cases:	View Class Analytics
Preconditions:	User is logged in and has taken the selected quiz.
Post Conditions:	User will have more specific options of types of analytics to view.
Flow of activities:	User selects Class Analytics. System Prompts for Various domains in Class. User browses further for further selections. System provides charts etc. to reflect data.
Exception Conditions:	1. User is not logged in. 2. User is no longer registered in class.

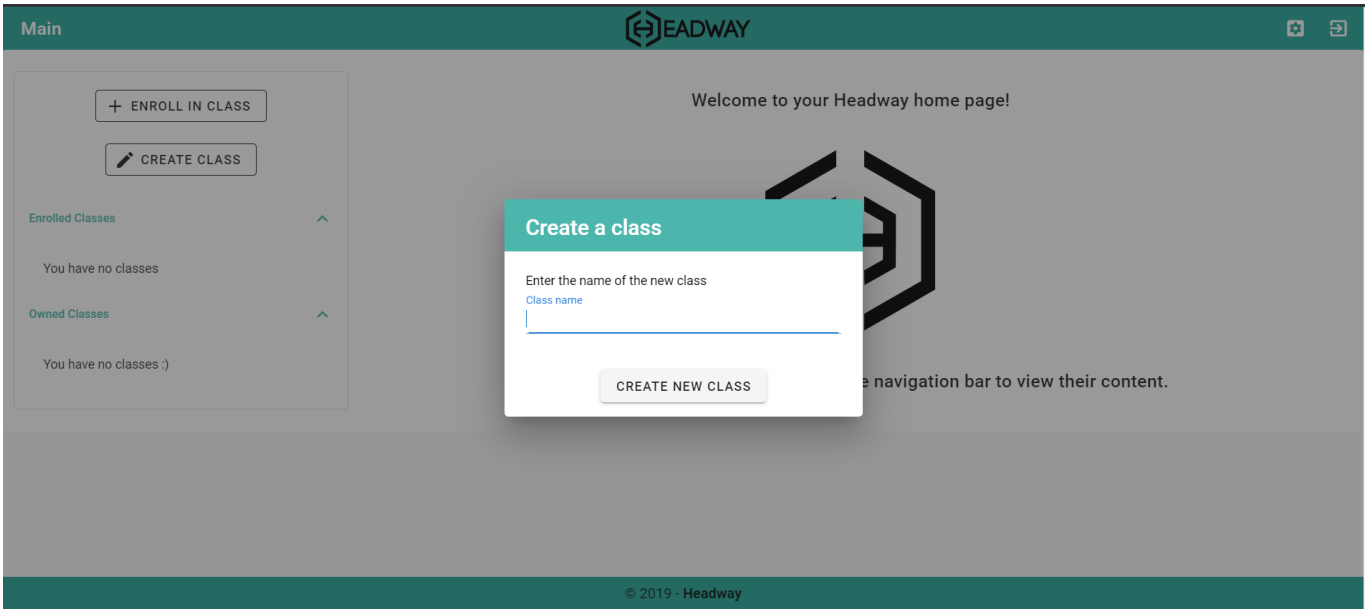
System Design

Infrastructure



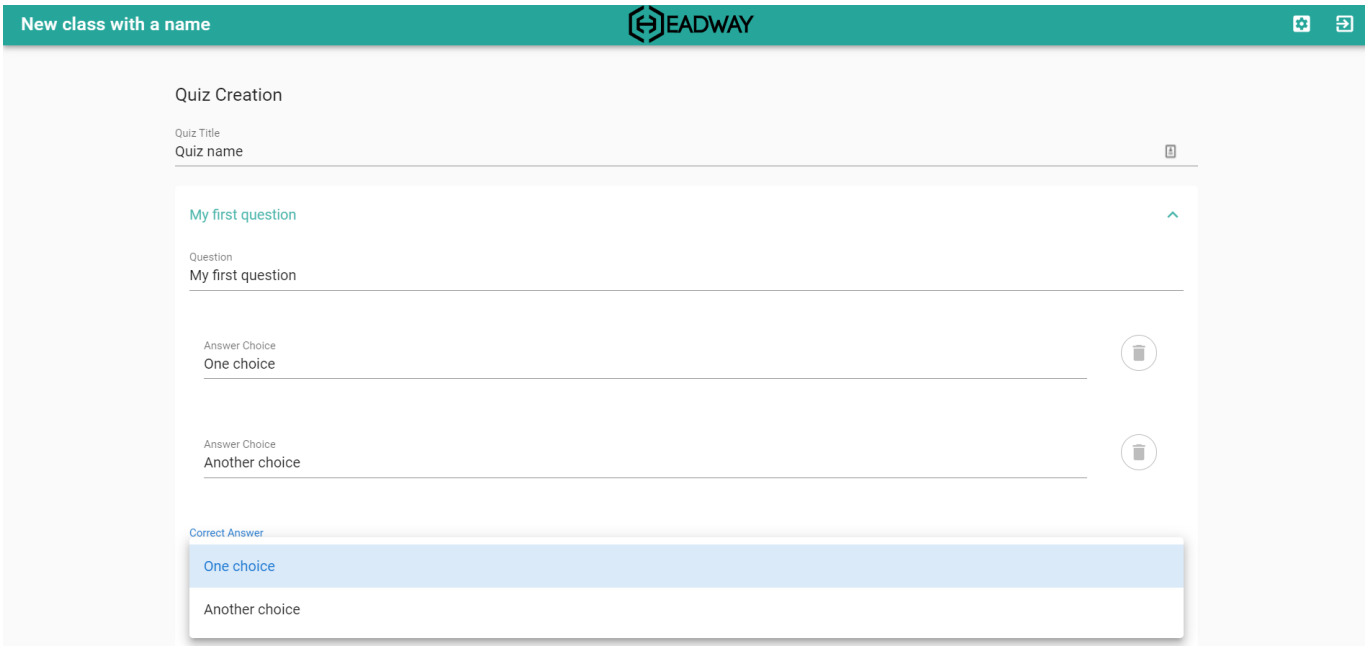
Use-Case Realization

Use cases were only used initially to help visualize how critical system components would be developed. After the intial implementation, the SCRUM process was the primary drive for creating new aspects of the Headway project. Below are the realizations of several of the intial use-cases: Create Course:



Here a user able to create a course from the main page only if they are logged in and afterwards they will be able to access the course.

Create Quiz:



Here a user is able to create a quiz from within the course they created, this is dependent on them being logged in and being the owner of the class. After this creation, the user will be able to modify the quiz and publish it so that students will be able to access it.

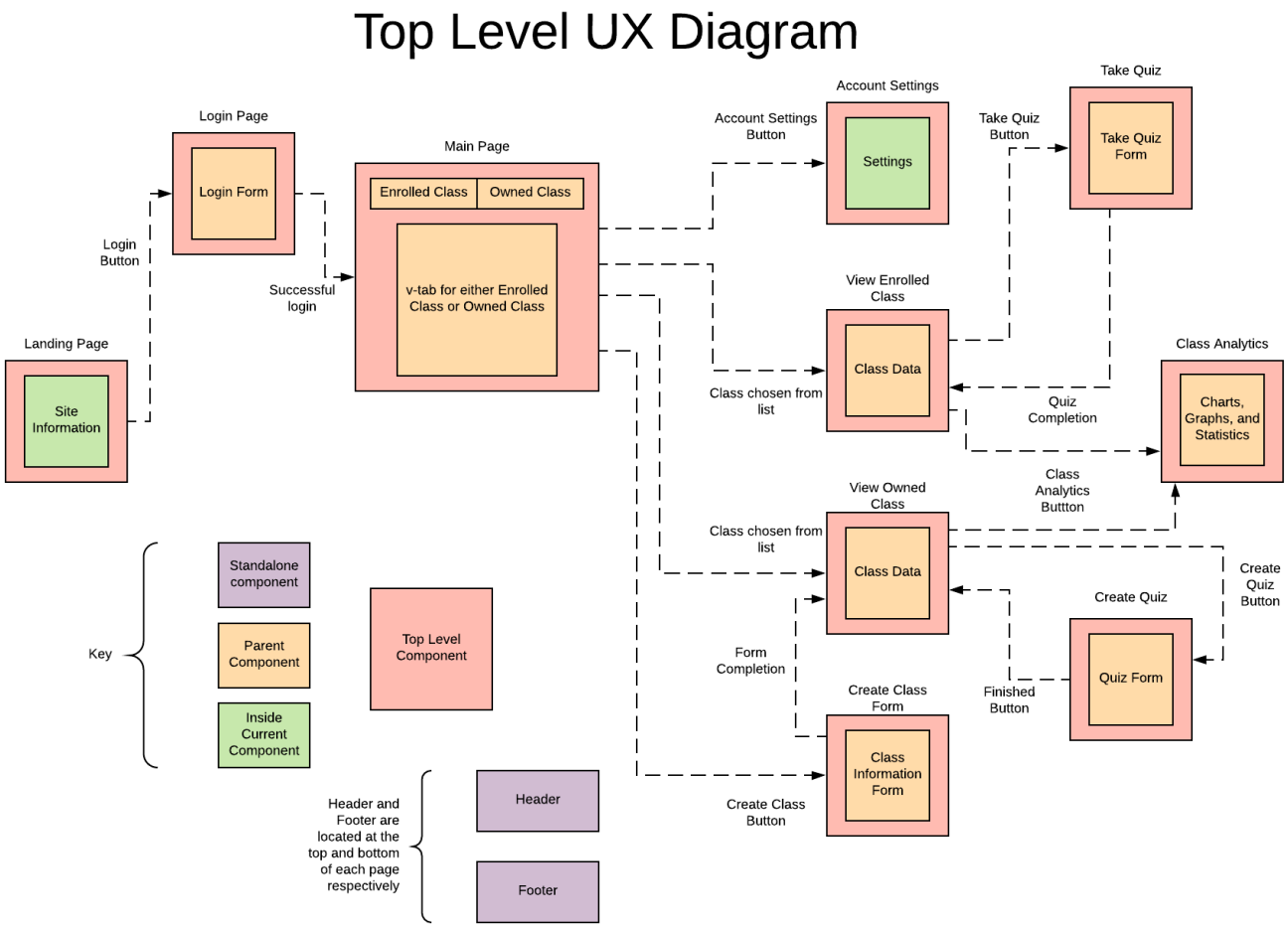
Database

MongoDB structure

```
{
classrooms: {
  classroomID: {
    name: "MyClassroom",
    students: {
      studentID: {
        quizID: {
          questionID: {
            type: "", // Question Type
            answer: [""] // Student Answers
          }
        }
      }
    },
    quizzes: {
      quizID: {
        name: "MyQuiz",
        state: "UNPUBLISHED",
        questions: {
          questionID: {
            questionString: "",
            correctAnswer: [""],
            answerOptions: [""]
```

```
            }
          }
        }
      }
    }
  },
    accounts: {
      ID: {
        AcctSettings: {
          email: "",
          darkTheme: true //boolean
        },
        myAcctClassrooms: {
          classroomID: {
            classroomID:  "",
            owner: true, // boolean
          },
        },
      },
    },
  }
```

User Interface



Implementation

The web app is built with Vue (javascript) using Vuetify and Typescript.

The classroom api is built in Kotlin and interfaces with Firebase. The stats is api is built in Python and using a Flask framework. The live api is built using [Node.js] and Socket.io. Live site: headway.dev

Deployment

This application is deployed using Bitbucket pipelines.
The applications are built and copied to AWS Lightsail VMs.
The Web-App is run statically off apache2, and the api is a Spring Application behind an Nginx Proxy.

Software Process Management

Scrum process

Every week the team gets together to discuss the upcoming challenges for the week. All the tasks for the week are decided and allocated on the trello board.

User Guide/Demo

Clone the repo. Then in repo directory do:

```
git checkout <current-release-branch>
git pull
npm install
npm run serve
```

Then go to: `localhost:8080`

Acknowledgments

Thank you Dr. Phung for providing inspiration for new features!