|  |
| --- |
| Stumped  Submitted for partial fulfillment of award of  **B. Sc in Information Technology**        By  Shammar Langaigne  A00938192    Department of Computers and Technology  School of Arts & Sciences  St. George’s University  Grenada  2018 |

**Certificate**

Certified that **Shammar Langaigne** has carried out the project work presented in this project report titled **Stumped** for the award of B.Sc in Information Technology from the Department of Computers and Technology, The St. George’s University, Grenada under my supervision. The thesis embodies result of original work and studies carried out by Student him/herself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else.

Signature:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Designation:

Date: 26/11/2018

Executive Summary

This purpose of the project is to solve the problem of communication that most international students struggle with in the classroom. This project is aimed at breaking down the barriers of communication regardless the language the members of the classroom are familiar with. To achieve this goal, we set out to create a mobile app that is capable of creating classes with teachers and students, for the purpose of students directing those questions to the teacher or any student in the class. The app is also capable of translating the content that users post based on the default language settings in the app. The mobile app was created using the Ionic Cordova framework. To connect to the backend database the app makes use of a cloud-based API which it connects to once an internet connection is provided

Acknowledgements

I would like to first give thanks to God, who gave me the strength and ability to undertake this project. I will also like to thank my advisor Mr. Keston Bhola for the counsel that was provided for the duration of this project. My thanks must also be extended the project director, Mr. Michael Roberts for his guidance and continuous understanding. I must also thank those who were able to help me resolve key issues that I ran into along the way; Mr. Kimanii Daniel and Mr. Gregor Richardson. I was able to learn a lot from them. I must also thank my family for supporting me through this journey and my friends for helping gather key information and for helping me at certain bumps in the road.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SHAMMAR LANGAIGNE

Table of Contents

[CHAPTER 1. INTRODUCTION 8](#_Toc530953730)

[1.1. Introduction 8](#_Toc530953731)

[1.2. Problem Statement 9](#_Toc530953732)

[CHAPTER 2. Objectives 10](#_Toc530953733)

[2.1. Needs statement 10](#_Toc530953734)

[2.2. Objectives 10](#_Toc530953735)

[CHAPTER 3. Requirement Analysis 11](#_Toc530953736)

[3.1. Requirement analysis 11](#_Toc530953737)

[3.2. Functional Objectives 12](#_Toc530953738)

[3.3. Functional specifications 13](#_Toc530953739)

[3.3.1. Classes 13](#_Toc530953740)

[3.3.2. Questions 13](#_Toc530953741)

[3.3.3. Comments 13](#_Toc530953742)

[3.3.4. Joining and Creating Classes 13](#_Toc530953743)

[3.4. Hardware and Software requirements 14](#_Toc530953744)

[3.4.1. Ionic 3 and Cardova framework 14](#_Toc530953745)

[3.4.2. Visual studio code 14](#_Toc530953746)

[3.4.3. Android Studio 14](#_Toc530953747)

[3.4.4. MongoDB: 14](#_Toc530953748)

[3.4.5. Node.js 14](#_Toc530953749)

[3.4.6. GitHub 14](#_Toc530953750)

[3.4.7. Chrome DevTools 15](#_Toc530953751)

[3.4.8. Amazon EC2 15](#_Toc530953752)

[3.4.9. Computer 15](#_Toc530953753)

[3.4.10. Phone 15](#_Toc530953754)

[3.5. OPERATING ENVIRONMENT 15](#_Toc530953755)

[CHAPTER 4. Feasibility study 16](#_Toc530953756)

[4.1. Cost estimation 16](#_Toc530953757)

[4.2. Time estimation 16](#_Toc530953758)

[4.3. Feasibility Report 16](#_Toc530953759)

[CHAPTER 5. Alternatives 17](#_Toc530953760)

[5.1. Desktop application 17](#_Toc530953761)

[5.2. The web-based application 17](#_Toc530953762)

[5.3. Mobile application 17](#_Toc530953763)

[CHAPTER 6. System design 18](#_Toc530953764)

[6.1. Architecture design 18](#_Toc530953765)

[6.2. Application Program Interface (API) 19](#_Toc530953766)

[6.3. Data Flow Diagram 20](#_Toc530953767)

[6.3.1. Conceptual Level 20](#_Toc530953768)

[6.3.2. Level 1 Diagram 20](#_Toc530953769)

[CHAPTER 7. Coding 21](#_Toc530953770)

[CHAPTER 8. Testing 28](#_Toc530953771)

[8.1. Test Cases 28](#_Toc530953772)

[8.1.1. Users should be able to post questions to a class. 28](#_Toc530953773)

[8.1.2. Each class should be split into answered and unanswered questions. 30](#_Toc530953774)

[8.1.3. The user who posted the question should be able to indicate that their question has been answered. 31](#_Toc530953775)

[8.1.4. Users should be able to respond to questions. 33](#_Toc530953776)

[8.1.5. The app should be able to pull information from the backend to populate itself. 35](#_Toc530953777)

[8.1.6. Users should be able to communicate which questions and comments they think are important so that more users can see them. 37](#_Toc530953778)

[8.1.7. Users of the app should be able to sign up and sign in to access their user account and each user should be verified and authenticated upon login. 38](#_Toc530953779)

[8.1.8. Users should be able to create classes 39](#_Toc530953780)

[8.1.9. Users should be able to join classes created by other users. 40](#_Toc530953781)

[8.1.10. Users should be able to change the language of the content from a list of languages available to them 41](#_Toc530953782)

[8.1.11. The teacher of the class should be able to edit the class if they please. 43](#_Toc530953783)

[8.1.12. The creator of the class should be able to add new users to the class if they please. 45](#_Toc530953784)

[8.1.13. Users should be able to edit their user information. 46](#_Toc530953785)

[CHAPTER 9. Implementation 47](#_Toc530953786)

[CHAPTER 10. Appraisal 48](#_Toc530953787)

[CHAPTER 11. User guide 49](#_Toc530953788)

[11.1. Login page 49](#_Toc530953789)

[11.2. Home page 50](#_Toc530953790)

[11.2.1. Selecting your class 50](#_Toc530953791)

[11.3. Class page 50](#_Toc530953792)

[11.3.1. Selecting a question 50](#_Toc530953793)

[11.4. Add class page 51](#_Toc530953794)

[11.4.1. Creating a class 51](#_Toc530953795)

[11.4.2. Joining a Class 51](#_Toc530953796)

[11.4.3. Posting a question 51](#_Toc530953797)

[11.4.4. Posting a comment 52](#_Toc530953798)

[11.5. Question page 52](#_Toc530953799)

[11.5.1. Using the answered button 52](#_Toc530953800)

[11.5.2. Using the unanswered button 53](#_Toc530953801)

[11.5.3. Editing a class 53](#_Toc530953802)

[11.5.4. Adding students to a class 53](#_Toc530953803)

[11.5.5. Changing languages 53](#_Toc530953804)

[11.5.6. Translating dynamic content. 53](#_Toc530953805)

[11.6. Deleting a class 54](#_Toc530953806)

[11.7. Deleting a question 54](#_Toc530953807)

[11.8. Deleting a comment 54](#_Toc530953808)

[11.9. Leaving a class 54](#_Toc530953809)

[11.10. Class information page 55](#_Toc530953810)

[11.11. Logging Out 55](#_Toc530953811)

[CHAPTER 12. Appendix 56](#_Toc530953812)

[CHAPTER 13. References 65](#_Toc530953813)

# INTRODUCTION

## Introduction

This project aims to create a prototype mobile app to reduce the difficulties that most international students may encounter in a traditional classroom setting. Namely, this project seeks to better the communication between students and lecturers towards course related content, both in and outside of the classroom. The app aims to accomplish this by providing teachers and students with an interactive and easy to use multilingual platform that will allow them to raise key questions and concerns they may have with any of the course material. The app will be created by using free open source software to avoid any copyright and licensing issues. The current version of the app will be released in a first beta to be tested. This mean that currently the app will be free to use. This may change in later versions to support either a subscription business model or may remain free by adopting an advertising revenue model. By providing international students with this new platform, we can hope to promote the use of technology in education and give these struggling students a better chance of success in their studies, by giving them a direct line to the aid they need without forcing teachers and lectures to give up their personal contact information.

The scope of this project is very large. Therefore, the project will only be done on the scale of a small prototype that will be released as a beta. This prototype will allow users to create and edit classes and post questions to those classes. The prototype will also allow users to translate information posted to these classes to their selected language.

## Problem Statement

Communication is an element in the classroom that can affect a student’s quality of learning. A lack of communication or an issue with disseminating information can sometimes be the driving force in a student’s downfall. Achieving proper communication is not always easy however, as an array of factors may be difficult to work around. Factors such as teaching styles or individual mannerisms may disincentivize students from seeking the help they need from their lecturers and though teachers and lecturers may make an effort to work around these issues with one on one sessions, the schedules of both teachers and students may not be always able to accommodate for that. Out of all these factors the most notable factor would be a language barrier. Whether it is as simple as a heavy accent or whether either party is struggling to communicate and understand effectively using the same language, this can lead to discrepancies in the classroom.

This could affect the student’s learning experience and create a stagnant environment in the classroom. It would easy for a student to become frustrated and disinterested in the content of the class.

As a result, a student may never achieve their full potential in that class and their overall grade may not be a true reflection their ability.

To solve this issue, I propose that a platform be built that specifically addresses these issues. A convenient platform that gives students a direct channel to the help they need without forcing lecturers to give up their personal contact information. A multilingual platform that can attempt to break down the language barrier between students and teachers of different backgrounds. A platform where other students can see if other members of the class encountered similar problems and how they were overcome. A platform that is easy and quick for users to access and use. This way teachers and lectures can address questions and problems easily without having to sacrifice too much of their time. Teachers and lecturers may also be able to better understand where students are struggling so they can put more focus to these problematic areas in class sessions. This platform can also make it easier for users to understand each other by translating the content to whatever language the user is more comfortable with.

# Objectives

## Needs statement

To be effective the solution must allow for effective communication between users of a classroom. The proposed solution must also be able to handle communication between users of different languages. The solution must be user friendly and simple enough to avoid confusing the user. The solution should be able to be accessed on the go and user activity should be rewarded. Upon completion of the project we hope to create a prototype mobile application that can be used by teachers and students to help break down the barriers of communication in the classroom. When this platform is completed our objective is make it easier for students to grasp concepts in the classroom so that they can succeed in their studies.

## Objectives

• To provide a mobile application that students can use to ask questions to other members of the classroom.

• The tool must be interactive.

• The tool should not use any copyrighted assets.

• This platform should support as many languages as possible.

• A user-friendly interface should be provided for each user.

• Proper security measures must be in place to keep all user information secure.

# Requirement Analysis

## Requirement analysis

Along with the appropriate documentation I am expected to deliver an operational prototype mobile app. The documentation should be clear so that the project can be developed further after the final hand up. The prototype should be able to:

* Have a database backend to store information.
* Have user login functionality.
* Allow users to create classes.
* Allow users to post questions to classes.
* Allow uses to post comments to questions.
* Feature a user-friendly interface.
* Allow users to choose from a selection of languages.
* Have a cloud-based API to handle backend operations.

## Functional Objectives

• Users should be able to post questions to a class.

• Each class should be split into answered and unanswered questions.

• The user who posted the question should be able to indicate that their question has been answered.

• Users should be able to respond to questions.

• The app should be able to pull information from the backend to populate itself.

• Users should be able to communicate which questions and comments they think are important so that more users can see them.

• Users of the app should be able to sign up and sign in to access their user account.

• Each user should be verified and authenticated upon login.

• Users should be able to create classes.

• Users should be able to join classes created by other users.

• Users should be able to change the language of the content from a list of languages available to them

• The teacher of the class should be able to edit the class if they please.

• The creator of the class should be able to add new users to the class if they please.

• Users should be able to edit their user information.

## Functional specifications

The mobile app should allow user login. The app should also allow users to sign up with their login information, along with their default language.

### Classes

Each class should have a class id, a name, a subject and one teacher user. The teacher should be able to add and remove other users to the class and edit the class information. The other users are the students of the class. Users should be able to ask questions on the class page and have the option to leave the class.

### Questions

Each question should have a title and content. Students in the class should be able to respond to questions. The user who posted the question should be able to indicate if a user already answered their question.

### Comments

Each comment should be a response to a question. Each comment consists of the text response as well as the user.

### Joining and Creating Classes

Users should be able to join classes by searching the class ID. Users should be able to create classes and add students to the class by searching for their student ID.

## Hardware and Software requirements

### Ionic 3 and Cardova framework

Ionic 3 and Cardova framework is an open source mobile app framework that will be used for front end development of the mobile app. Ionic provides tools and services for developing hybrid mobile apps using Web technologies like CSS, HTML5, and Sass.

### Visual studio code

Visual studio code (source code editing software HTML, CSS, JavaScript, JSON): Will also be used for development of the app alongside other applications to view, edit, build, test and debug the code.

### Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development (developer.android.com, 2018). This program gives us the ability to build versions of app with the android gradle. This program will not be used for coding since the functions we would be using are already built into visual studio code.

### MongoDB:

MongoDB is a free and open-source cross-platform document-oriented database program. This is the platform that will be used to create the database used by the front end.

### Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. This is a crucial component in the architecture of the app. Node.js allows the components of the app to be deployed locally, to make testing easier. With Node.js the development and testing process can take place in the browser as opposed to using a device or an android studio AVD. Node.js can also be used to deploy any backend APIs the app may need.

### GitHub

GitHub is a web-based hosting service for version control using Git. This service will be used to store all the relevant source code and resources used during the development process. This is not just used for storage, but it also makes it easy to distribute the project to others.

### Chrome DevTools

Chrome DevTools are a set of web developer tools built directly into the Google Chrome browser. DevTools can help you edit pages on-the-fly and diagnose problems quickly, which ultimately helps you build better websites, faster (developers.google.com, 2018). Using this I can debug and resolve issues that may arise during the development process.

### Amazon EC2

Amazon EC2 (Amazon Elastic Compute Cloud) Ubuntu server is a cloud-based server provided by amazon web services. This server handles HTTP requests that are sent to a specific port. These requests will come from the front end of the app and

### Computer

A HP Laptop is the system that will run the software needed to build this project. This is system where all the coding and testing will take place. Specifications include: intel i3 2.4Ghtz, HDD, windows 10 with 4 gigabytes of DDR4 SDRAM.

### Phone

This phone will be used to run the app on a mobile environment LG Risio 2 [Android Phone]: Marshmallow OS, Qualcomm® MSM8909 1.1GHz Quad-Core processor and 15 gigabytes of RAM.

## OPERATING ENVIRONMENT

The app will operate on the android operating system. The app will require a stable internet connection for the best experience. The app will be compatible with android API levels 16 and upwards (Jelly bean). This is to ensure that a great majority of android users will be able to run the app on their phone. According to the android developer’s API distribution dashboard, 0.5% of android users use API levels that are lower than Jelly Bean OS. Since we are using the Cordova to build the app this ensures that 99.5% of android users can use the app effectively.

# Feasibility study

## Cost estimation

Currently the cost to develop and application with the specifications outlined for me should come with very little upfront cost. Due to the open source frameworks and services outlined in the requirements and the fact that some of these resources were already available to me, the cost price for a project like this should be very low.

## Time estimation

What comes at a high cost in this case would be the amount of time that should be dedicated to a project like this. Based on the requirements outlined below, a prototype should take four months to complete at my current skill level.

## Feasibility Report

Based on my findings it is possible to produce the prototype within the time frame given.

# Alternatives

## Desktop application

This project could be done as a standalone application that users can download. This option would work quite similarly to the web-based option however it would still to have online functionality to work. The reason this option wasn’t chosen is because it is not as convenient or easily portable as a mobile application solution. It takes less effort to check your phone or mobile device that it takes to check your laptop. Though this solution may have its advantages in the potential for better performance, the current scope of the project focuses on ease of use and connectivity. It is common knowledge that most users check their phone more frequently than their laptop.

## The web-based application

This method was chosen at first because I felt the most comfortable with web development compared to other options of development. This has since changed however due to the existence of the Cordova-Ionic framework. Using ionic I was able to apply similar techniques and knowledge used in web development to mobile app development. Once this became a reality the drawbacks of the mobile app alternative seemed less than they were before making it the ideal choice.

## Mobile application

This project will work quite similarly to the web-based application but instead run on mobile devices. This alterative is the most ideal for this project due to the availability and convenience. With this solution users can access the platform much easier and taking less time to resolve issues than if I were to use the other options. I am hoping that by taking this approach using the platform would be more rewarding to users.

# System design

## Architecture design

The architecture of the project itself consists of 3 crucial components;

1. The front end, which is the actual mobile app itself. This is the component that the user will install on their phone.
2. The Application programming interface (which will be referred to as the API), that handles communication between the front end (the user) and the database. I may also refer to the API as the backend. The API is hosted in the cloud on a Ubuntu OS server. Using the address of the server the app connects to the IP of the server to post HTTP request, which the server responds to.
3. The mongo database, which stores user and class information. This database is hosted through a service known as mLabs, that allows the API to communicate with the front end and the database. Only the API has the credentials to pull information from the database.

The app itself pulls information the backend and When a user interacts with the app the front end posts these changes to the database through the API. The mobile app itself needs to have a working internet connection to function. In addition, the app will not work if the API is not deployed for the front end to connect to.

## Application Program Interface (API)

As previously stated, the API handles requests that come from the user interface. These requests consist of: user login and authentication, creating new users, creating editing and removing all class information. Upon initialization of the front end the API handles the request that provides the front end with the data that will populate the app. The API also handles search requests, for example if a user searches for another user or a class these search results will be provided by the API. It is important to note, that a part of the sign-up process includes hashing the password the user enters. By using this method, we are adding an extra layer of security to our user’s information making difficult for even the database administrator to login as another user. This also secures user information in the case of a database security breach. The API is the only component in the system that is able to decrypt the password hash making it a crucial component in the login process.

## Data Flow Diagram

### Conceptual Level

A screenshot of a cell phone

Description generated with high confidence

### Level 1 Diagram

\*Refer to the diagram in the appendix labeled “Level 0 Data Flow Diagram” (code.visualstudio.com, n.d.)

# Coding

The following is the code for the data service provider of the app. This is an essential component and is responsible for handling tasks such, as login, logout and http requests.

public frizbe: any;//parsing data to pages

public saucer: any;//parsing data to pages

public lang: string;//language the user set

// rootUrl : string = "https://shammarlangaignecom.000webhostapp.com";

rootUrl: string = "http://localhost:3000"; //Local API Connection For testing purposes

// rootUrl: string = "http://ec2-3-16-165-84.us-east-2.compute.amazonaws.com:9000";//The Url to Connect to the API

file: string = 'classes';

private currentUser: any;

public cred: any;//user credentials

// file : string = 'pages';

constructor(public http: Http, public toastCtrl: ToastController, public event: Events, public storage: Storage, public translate: TranslateService) {

console.log('Hello DataServiceProvider Provider');

}

//retrieving data from the backend

getData() {

try {

return this.http.get(`${this.rootUrl}/${this.file}`).map(res => res.json()).take(1)

} catch (e) {

this.showToast(e)

}

}

//functions posts new class to the backend

createClass(dass) {

let headers = new Headers();

headers.append('Content-Type', 'application/json');

try {

return this.http.post(`${this.rootUrl}/${this.file}`, JSON.stringify(dass), { headers: headers }).map(res => res.json()).subscribe(

res => {console.log('success')},

err => {

this.refuse()

}

);

} catch (e) {

this.showToast(e)

}

}

//find specific user

getUser(id) {

try {

return this.http.get(`${this.rootUrl}/users/${id}`).map(res => res.json()).take(1)

} catch (e) {

this.showToast(e)

}

}

//get specific class

getClass(id) {

try {

return this.http.get(`${this.rootUrl}/${this.file}/${id}`).map(res => res.json()).take(1)

} catch (e) {

this.showToast(e)

}

}

//update a specific class

joinClass(id, body) {

let lang = body.lang;

\_.forEach(body.questions, value => {

this.bandies(value.topic, lang).then(val => {

value.topic = val

});

this.bandies(value.description, lang).then(val => {

value.description = val

});

//Attempt at like button for comments

// \_.forEach(body.comment, comm =>{

// this.yandex(comm., )

// })

})

let headers = new Headers();

headers.append('Content-Type', 'application/json');

try {

return this.http.put(`${this.rootUrl}/${this.file}/${id}`, body, { headers: headers }).map(res => res.json()).take(1)

} catch (e) {

this.showToast(e)

}

}

//login function

login(cred) {

try {

let headers = new Headers();

headers.append('Content-Type', 'application/json');

return this.http.post(`${this.rootUrl}/users/login`, JSON.stringify(cred), { headers: headers }).map(res => res.json())

} catch (e) { Promise.reject(e) }

}

//function that handles population and caching login information after login

preLogin(body) {

try {

this.login(body).subscribe(allowed => {

if (allowed.success) {

this.cred = allowed;

allowed.userInfo.password = body.password

this.publish(allowed.userInfo);

this.storage.set('cred', allowed);

} else {

this.showToast(allowed.msg);

}

},err => {

this.refuse()

})

} catch (e) {

this.showToast(e);

}

}

//register function

public register(cred) {

let body = {

id: '21',

userName: cred.value.userName,

email: cred.value.email,

fName: cred.value.fName,

lName: cred.value.lName,

password: cred.value.password,

Classes: [],

};

this.postRegister(body).subscribe(allowed => {

if (allowed.success) {

this.cred = allowed;

allowed.userInfo.password = body.password

this.publish(allowed.userInfo);

this.storage.set('cred', allowed);

} else {

this.showToast(allowed.msg);

}

},

error => {

this.showToast(error);

});

}

//function that posts the login function to the backend

postRegister(body) {

let headers = new Headers();

headers.append('Content-Type', 'application/json');

try {

return this.http.post(`${this.rootUrl}/users/signup`, JSON.stringify(body), { headers: headers }).map(res => res.json())

} catch (e) {

this.showToast(e)

}

}

public logout() {

return Observable.create(observer => {

this.currentUser = null;

observer.next(true);

observer.complete();

});

}

//This function catches and displays app erros as a toast message at the bottom of the screen

showToast(message: string, timeout: number = 8000, position: string = 'buttom') {

this.toastCtrl.create({

message: message,

duration: timeout,

position: 'bottom'

}).present();

}

//The function that

publish(user) {

this.event.publish('user : logged in', user)

}

//update a specific user

updateUser(user) {

let body = {

Classes: this.cred.userInfo.Classes,

\_id: this.cred.userInfo.\_id,

id: this.cred.userInfo.id,

userName: user.value.userName,

email: user.value.email,

password: this.cred.userInfo.password,

fName: user.value.fName,

lName: user.value.lName,

}

let headers = new Headers();

headers.append('Content-Type', 'application/json');

try {

return this.http.put(`${this.rootUrl}/users/${body.\_id}`, JSON.stringify(body), { headers: headers }).map(res => res.json())

} catch (e) {

this.showToast(e)

}

}

//stores user credentials to the backend

getStuff() {

try {

return this.storage.get('cred')

} catch (e) {

this.showToast(e)

}

}

//removes a class from the backend

delete(body) {

try {

return this.http.delete(`${this.rootUrl}/${this.file}/${body.\_id}`, JSON.stringify(body)).map(res => res.json())

} catch (e) {

this.showToast(e)

}

}

//translates strings in the app based on the parameters sent and the language files in assets

translateFunc(param: string) {

try {

var yeild

this.translate.get(param).subscribe((res: string) => {

console.log(res);

yeild = res

});

return yeild

} catch (e) {

this.showToast(e)

}

}

//used to translate dynamic content

async yandex(param, lang) {

try {

let value = await translation(param, { from: lang, to: this.lang, engine: 'yandex', key: 'trnsl.1.1.20181114T194518Z.640236167701ee19.296338b36aa783c4b9a4aee7b5155955c41493ca' }

).then(text => {

console.log(text)

return text

});

return value

} catch (e) {

this.showToast(e)

}

}

//fix this

async bandies(param, lang) {

try {

let value = await translation(param, { from: this.lang, to: lang, engine: 'yandex', key: 'trnsl.1.1.20181114T194518Z.640236167701ee19.296338b36aa783c4b9a4aee7b5155955c41493ca' }

).then(text => {

return text

});

return value

} catch (e){

this.showToast(e)

}

}

refuse(){

let toast = this.toastCtrl.create({

message: 'Connection to the API was unsuccessful. Check your internet connection.',

// duration: 3000,

position: 'bottom',

showCloseButton : true,

closeButtonText : 'OK'

});

toast.present();

}

# Testing

## Test Cases

These Test Cases are an evaluation of the functional objectives laid out previously.

### Users should be able to post questions to a class.

Test Case

A student in the computer Ethics class wishes to post a question regarding fair use.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user has selected the computer Ethics class from the list of classes on the home page.

Steps:

* The user must first press the button on the bottom right of the page 
* When the modal page comes up, the user must enter both the subject and the description of the question in the input fields provided.
* Once the user is satisfied with their question they can click the post button

Test Data

Subject: “Fair use”

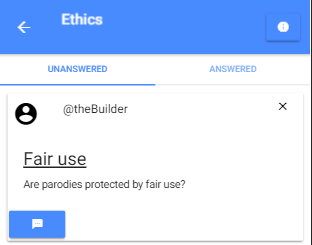
Description: “Are parodies protected by fair use?”

Actual Results

* A new question should appear in the unanswered section of the class with the subject and description the user outlined in the modal page.
* The user who posted this question should also be displayed.

Actual Results

The results of this test are successful.



### Each class should be split into answered and unanswered questions.

Test Case

A student in the computer Ethics wishes to see all the unanswered and answered questions in the class.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user has selected the computer Ethics class from the list of classes on the home page.

Steps:

To see the unanswered questions

* The user should click on the unanswered tab in the class.

To see the answered questions

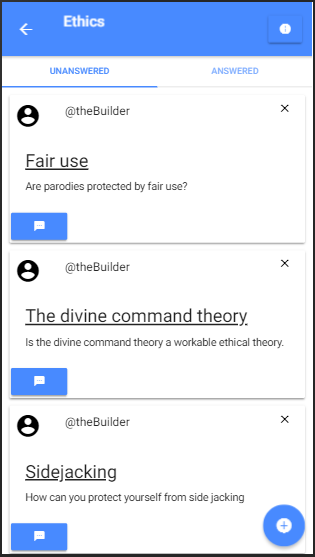
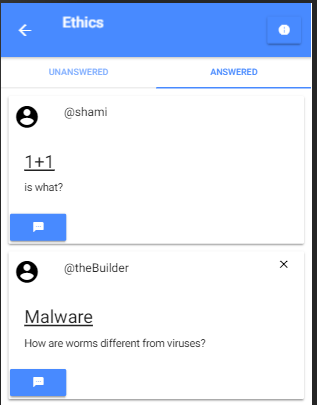
* The user should click on the answered tab in the class.

Actual Results

* The unanswered questions should appear first by default but should also be displayed if the user selects the unanswered tab
* The answered questions should appear if the user clicks the answered tab.

Actual Results

The results of this test are successful.

### The user who posted the question should be able to indicate that their question has been answered.

Test Case

A student who posted a question in the computer Ethics class wishes to indicate that their question has been answered.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user has selected the computer Ethics class from the list of classes on the home page.

Steps:

* The user must first select their question from the list of questions in the class.
* Once navigated to that question the user must press the green button on the question post. 
* The user must click “Yes” to the confirm dialog that should pop up after.

Test Data

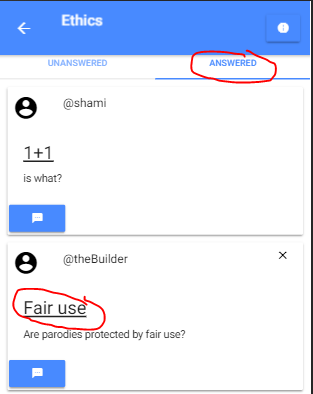
[The fair use question]

Actual Results

* This button should change from green to red 
* Once the user navigates to the class page this question should now be only found in the answered section

Actual Results

The results of this test are successful.



### Users should be able to respond to questions.

Test Case

A student in the computer Ethics class wishes to respond to a question.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user has selected the computer Ethics class from the list of classes on the home page.

Steps:

From the class page.

* The user must click on the blue comment button at the bottom of the question card.
* When the modal page comes up, the user must their response in the input field provided.
* Once the user is satisfied with their question they can click the post button

From the question page

* The user must first select the question from the list of questions in the class.
* The user can then type their response in the text box below.
* The user can click the POST button to post the comment to the question.

Test Data

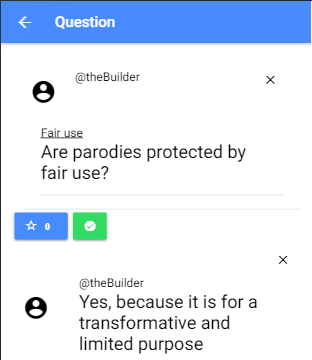
“Yes, because it is for a transformative and limited purpose”

Actual Results

* A new comment should appear below the question with the description the user outlined in the modal page.
* The user who posted this comment should also be displayed.

Actual Results

The results of this test are successful.



### The app should be able to pull information from the backend to populate itself.

Test Case

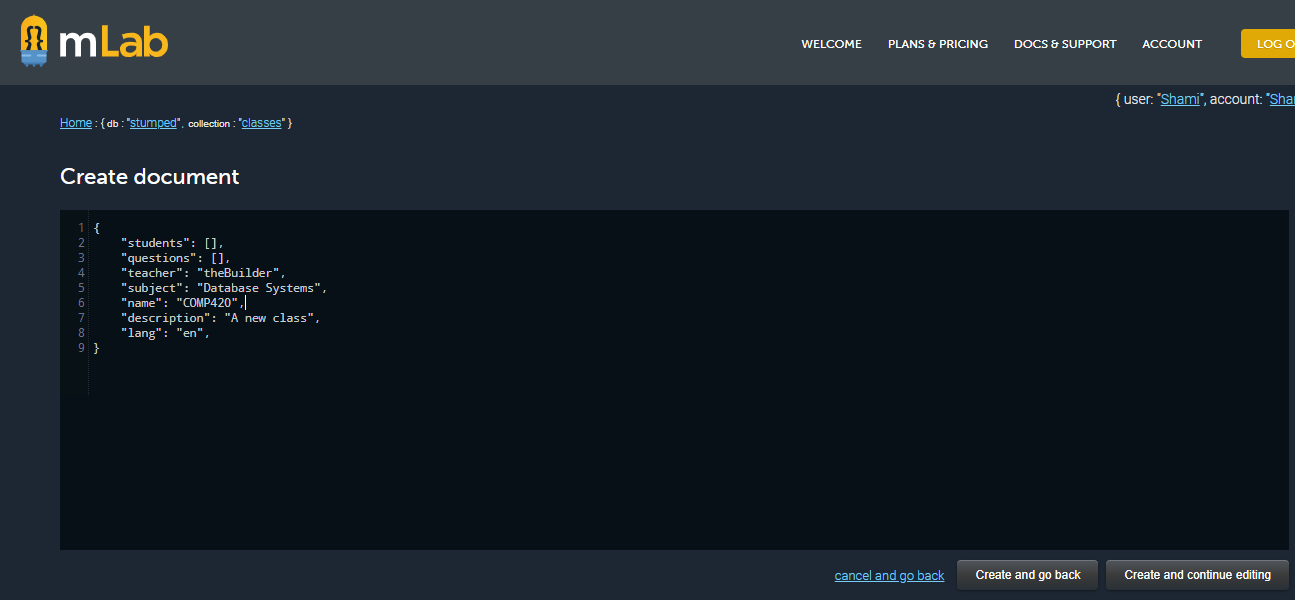
The information in the database should match the information displayed in the app. To test this, we will try to add a new class to the app from the backend. This new class should appear on the app once we refresh the feed.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned.

Steps:

* Add a new user following the JSON structure format outlined before.



* Pull down the list of classes on the home page to refresh the content.

Test Data

{

"students": [],

"questions": [],

"teacher": "theBuilder",

"subject": "Database Systems",

"name": "COMP420",

"description": "A new class",

"lang": "en"

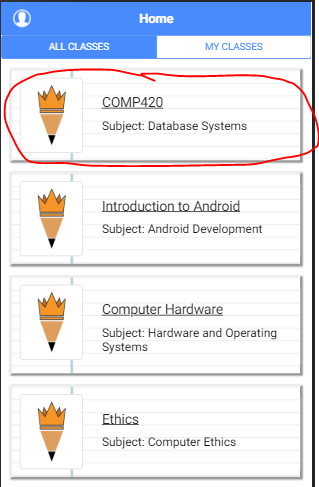
}

Actual Results

* A new class should appear on the home page matching the specifications outlined in the Test Data

Actual Results

The results of this test are successful.



### Users should be able to communicate which questions and comments they think are important so that more users can see them.

Test Case

A student in the computer Ethics class wishes to like a question regarding fair use.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user has selected the computer Ethics class from the list of classes on the home page.

Steps:

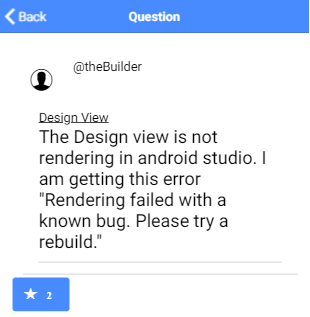
* The user must select the question they wish to like
* Once on the question page the user must press the star button. 

Actual Results

* The number label next to the star should increase by 1
* The star icon should become filled in. 

Actual Results

The results of this test are successful.



### Users of the app should be able to sign up and sign in to access their user account and each user should be verified and authenticated upon login.

Test Case

A User wishes to login to their account with valid login information.

Preconditions

A user has installed the app and launched it.

* The user must enter their username and password in the fields provided
* The user must press the login button that should now be available.

Test Data

User Name: “shami”

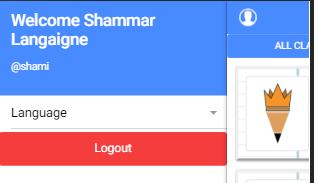
Password: “password”

Actual Results

* The user should be navigated to their home page.
* The side bar should be populated with the user name and first and last name of the user.

Actual Results

The results of this test are successful.



### Users should be able to create classes

Test Case

A user should be able to create and post a class to the app database.

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned.

Steps:

* The user must first press the button on the bottom right of the page 
* The create class tab should appear by default.
* When the page comes up, the user must enter both the class information in the input fields provided.
* Once the user is satisfied with their question they can click the “create class” button

Test Data

Name “Math220”

Subject: “Statistics”

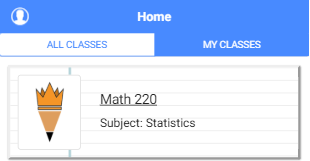
Description: “A very hard class”

Language “en”

Actual Results

* Students and teachers in the class should be able to access the class from their home page from now on.

Actual Results

The results of this test are successful. 

### Users should be able to join classes created by other users.

Test Case

The user mrDoe wants to join the Math220 class

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned.

Steps:

* The user must first press the button on the bottom right of the page 
* When the page comes up, the user must select the join class tab below the navbar of the page
* The user must enter the class ID in the search bar provided below.
* Once the ID in the search-bar matches an ID in the database the user can click on the class icon that should now appear.

Test Data

Class ID: 5bfadfc881a56537bce43b78

Actual Results

* The user mrDoe should be able to access the class from their home page from now on.

Actual Results

The results of this test are successful.

### Users should be able to change the language of the content from a list of languages available to them

Test Case

The user wants to change the language from English to Spanish

Preconditions

A user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned.

Steps:

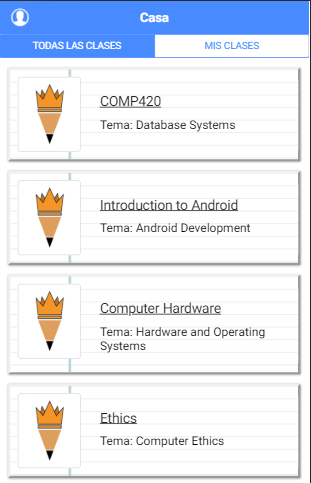
* The user can either click on the user icon on the left side of the navbar, or swipe right across the screen.
* Once the user side bar appears the user can click on the language select.
* Once the select bar comes up the user can select whichever language they want translated (in this case the user selects language).

Test Data

language: “es”

Actual Results

* All the static content in the app should now appear in Spanish

Actual Results

The results of this test are successful.

### The teacher of the class should be able to edit the class if they please.

Test Case

The teacher of Introduction to android wants to change the name of the class to: “Intro to Android Development”

Preconditions

The user in this case is the teacher of the Introduction to Android class. The user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user navigates to their Introduction to Android class.

Steps:

* The Teacher can click on the “i” icon located at the right corner of the navbar. 
* A popup should appear over the class page and the teacher should select the text that says “Edit Class”
* A new page (class information page) should appear with a form. The fields should all be filled in with the name, subject and description of the class.
* The teacher can update the changes made in the form by clicking the “Edit Class button”

Test Data

Name: “Intro to Android Development”

Subject: “Android Development”

Description: “This course is the first in a series of courses that will impart to students the requisite knowledge to design, develop, deploy and maintain marketable android applications.”

Actual Results

* The name of the class should change to reflect the changes in the form on the home page.
* The name of the class should change to reflect the changes in the form on the class page.
* The name of the class should change to reflect the changes in the form on the class information page.

Actual Results

The results of this test are successful.

### The creator of the class should be able to add new users to the class if they please.

Test Case

The teacher of Introduction to android wants to add mrDoe to the class.

Preconditions

The user in this case is the teacher of the Introduction to Android class. The user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned. The user navigates to their Introduction to Android class.

Steps:

* The Teacher can click on the “i” icon located at the right corner of the navbar. 
* A popup should appear over the class page and the teacher should select the text that says “Add Students”
* A new page (search users page) should appear with a search-bar. The user must enter the username of the student they wish to add in the search bar provided.
* Once the username in the search-bar matches a username in the database the user can click on their icon that should now appear. After clicking on the user icon, they will now be added to the list of students that will be added upon completion of the process
* Once the user has added all the names they wish to the list, they can click the “Add student” button to complete the process.

Test Data

mrDoe

Actual Results

* mrDoe should now be able to access the class

Actual Results

The results of this test are successful.

### Users should be able to edit their user information.

Test Case

The user “theBuilder”, wants to change his username to “bobolicious”.

Preconditions

The user has installed the app and launched it. The user is logged in to the app and the app has a stable internet connection to the Stumped API previously mentioned.

Steps:

* The user can either click on the user icon on the left side of the navbar, or swipe right across the screen.
* The user can click on the “Edit User” button which will bring up a page with the user’s information.
* The user can make whatever changes they like (change the username from theBuilder to bobolicious)
* Click the save Changes button to post those changes back to the database.

Test Data

Username: “bobolicious”

First name: “Bob”

Last name: “Becket”

Email: “yeah@he.can”

Actual Results

* The username on the sidebar should change from theBuilder to bobolicious

Actual Results

The results of this test are unsuccessful.

Though the app was able to complete this successfully adding this functionality to the app caused a bug with the login functions, as passwords were not being encrypted properly after a user edited their information. This functionality was then later removed to avoid this bug.

# Implementation

# Appraisal

After consulting with an app development company in Grenada, I was able to get an appraisal for the value of the app if it was to be released in a fully functional state. After reviewing the requirement and system design specifications if the app were to be released for android, with approximately 10 pages, database and mobile API functionality and the features outlined in the design specifications, the app is valued at $20,000 EC.

# User guide

## Login page

This page interacts with the user like any most standard login pages. The most notable components here are the two inputs for username and password and the login button. The underlined text at the bottom of the page is also noteworthy. Both fields for the login page are required. If a field is left empty it will be underlined in red and fields that are filled will be underlined in green. If the user is focused on a field, that field is underlined in blue. The login button is faded out by default and will only be clickable when the user fills both fields. The password field does not show what the user types, but only shows characters as asterisks. When the user clicks the login button either one of two things happens. If the login is successful, the user will be directed to the home page. If the login is unsuccessful a brief popup will appear at the bottom of the screen indicating the reason. If the login is unsuccessful all fields will be clear, and the user can login again.

The login page also works as a signup page. The signup form will only be visible once the user clicks the underlined text: “create an account”. Once the signup form is visible the login form is no longer visible and vice versa. The login form will reappear when the user clicks the underlined text “sign-in”.

The signup form consists of the same inputs from the login form with the addition of an Email, First and Last name and an additional password field. Like the previous form the email and password confirm fields are required as well however the first and last name fields are not required. Just like the last form, the sign-up button will only be clickable once all the required fields are filled, following the same underline color scheme for filled, empty and focused. Even after the button becomes clickable, the app will not post the request to the API until both of the password fields are identical. If the password fields do not match, a popup message will appear at the bottom of the page to advise the user to revise their password. Finally, similar to the login form if the sign-up is successful the user will be logged in. If an error occurs a popup message will indicate the error to the user.

## Home page

The home page consists of a navbar, where the use icon will be located to the left corner. If the user clicks their icon a tab will slide over the home page to display the user information. The content of the page is divided into two sections by divider tabs. The default tabs show all the classes the user is a part of and the “My Classes” tab, shows all the classes the user created. There is a floating button at the bottom right corner of the page, that will navigate the user to a page where they can add another class to their list.

### Selecting your class

Each class is presented to the user as a card with the title and the description of the class as the header. Each card also has the class thumbnail which is set by the creator of the class. If no thumbnail is set the default thumbnail is displayed. When a user clicks a card, the app navigates to that class.

## Class page

The class page consists of a header with an information icon at the right corner. By clicking this button, the user will be presented with a popup that shows them general information about the class and gives them certain options.

If a class has no questions a message will appear alongside a button that will pull up a prompt to post a question.

When the class is populated with questions however, each question is organized into its own card. There is a divider directly under the navbar that separates the questions into two categories; answered and unanswered. Each card contains the icon of the user that posted the question the subject the tagline of the question.

### Selecting a question

The whole question will be displayed once the user clicks on the card. Users can also comment below this question from a button placed at the bottom of the card. A floating button, located at the bottom right of the page can be used to post new questions to the class.

## Add class page

This page allows the user to create their own class or join an already existing class. These functions a separated by a divider located directly below the navbar.

### Creating a class

The form to create a class contains a form with the fields name description, language, students and subject. The fields name, subject and description are required fields and the “create class” button below will only become clickable when both these fields are filled. By clicking on the language tab, the user will bring up a select prompt to enter the default language of the class. The default value for this field is English. These fields follow the exact same color code as the forms on the login page. The form also contains a search bar. By using the search bar, a user can add students to their class by searching for their user name. Once the search finds a result the student will be displayed in a card below the search bar. If the user clicks on this card that student will be added to the students field. Like the login page if there is ever an error when the user clicks the button this error message will display in a popup below. If the class is created successfully however, the user will be prompted with the option of navigating to this class.

### Joining a Class

The join class segment only has a search bar. The user can search for a class using the class ID. Once a user finds the class they want to join they will be prompted with the option of navigating to this class.

### Posting a question

By clicking on the fab button at the bottom right of the class page a prompt form should appear with a subject and a description field. The same prompt can be accessed by clicking the “Be the first” button, that should be visible if the class has no questions.

To post the question, press the post button located at the navbar of the page.

The post button will not be clickable however, if a field is left empty.

To cancel the post, press the red button at the left corner of the navbar.

### Posting a comment

There are two ways this can be done.

By clicking the blue dialog box at the bottom corner of a question card. Once this is done a prompt should appear with a text box.

* To post the comment, press the post button located at the navbar of the page.
* The post button will not be clickable however, if the field is left empty.
* To cancel the post, press the red button at the left corner of the navbar.

By clicking on the question and bringing up the question page.

## Question page

This page has the question the user asked at the very top of the page. Below this post are all the questions users have asked. The question post should also feature a blue star button, users can use to “like” that post.   
At the bottom of the page, is a text box with a post button below the text box. The user can add comments to the question as well by typing into this text box and clicking the post button. This post button is only clickable however when the text box is holding a value. The user can navigate back to the class by clicking the back button at the left side of the navbar.

### Using the answered button

If the user that asked a question wants to indicate that it has been answered, they should navigate to the answered question from the tab on the class page. Once there a green button with a checkmark should appear.

If the user clicks that button a confirmation prompt will appear and if they click yes, the question will be moved from the unanswered to answered section. At this point that green button will now be changed to a red button.

### Using the unanswered button

If for whatever reason you want to reopen a question that has been vaulted to the answered section, navigate to the answered question from the tab on the class page.

Once there a red button with a question mark should appear.

If the user clicks that button a confirmation prompt will appear and if they click yes, the question will be moved from the answered to unanswered section. At this point that red button will now be changed to a green button.

### Editing a class

If the teacher of the class wants to edit the class information, they must select the “Edit Class” option located on the class information popup. A new page should appear with a form. The fields should all be filled in with the name, subject and description of the class. The teacher can update the changes made in the form by clicking the “Edit Class button”.

### Adding students to a class

If the teacher of the class wants to edit the class information, they must select the “Add Students” option located on the class information popup. A new page (search users page) should appear with a search-bar. The user must enter the username of the student they wish to add in the search bar provided. Once the username in the search-bar matches a username in the database the user can click on their icon that should now appear. After clicking on the user icon, they will now be added to the list of students that will be added upon completion of the process. Once the user has added all the names they wish to the list, they can click the “Add student” button to complete the process.

### Changing languages

If a user wishes to change the language of the app they can do once they have logged in by opening the side menu on the home page and switching the language, select tab from English to any of the other 3 languages currently available.

### Translating dynamic content.

If you want to change the language of a class simply, change the language settings in your sidebar and navigate to that class again. You should be prompted with an option to try to translate the dynamic content in the class, if yes is selected this content will now be translated in the language selected in the side bar.

## Deleting a class

To remove a class completely from the database you first have to be the teacher of the class. If you are navigating to the class and pull up the class information popup. Once you are the teacher of that class there should be an option that says, “Delete Class”. After selecting that option, you will be prompted with a confirm dialog. Once yes is selected the class will be removed from the database entirely and the user will be navigated back to the home page.

**NB Once this is done there is no way to reverse it.**

## Deleting a question

To be eligible to remove a question you must have one of either two requirements.

Be the teacher of the class where the question was posted or be the user who posted the question. If you meet any of these requirements there should be a X that appears at the top right corner of the question card. Once this X is clicked you will be prompted with a confirm dialog. Once yes is selected the question will be removed from the class.

## Deleting a comment

To be eligible to remove a question you must have one of either three requirements.   
Be the teacher of the class where the comment was posted, be the user who posted the question the comment is posted under or be the user who posted the comment. If you meet any of these requirements there should be a X that appears at the top right corner of the comment. Once this X is clicked you will be prompted with a confirm dialog. Once yes is selected the comment will be removed from the question.

## Leaving a class

If you wish to leave a class, you should navigate to the class and bring up the class information popup. Once you are just a student in that class the only option available to you should be, “Leave Class.” Once this option is selected you will be prompted with a confirm dialog. Once yes is selected the class will be removed from your list of classes.

## Class information page

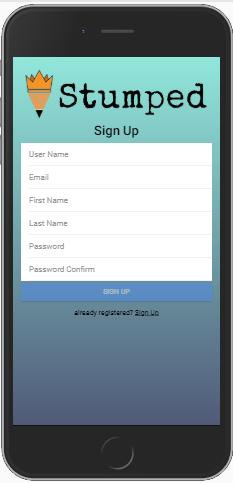
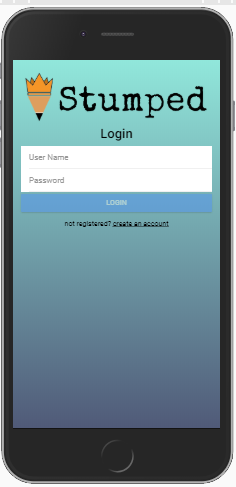
This page contains the information about the class.

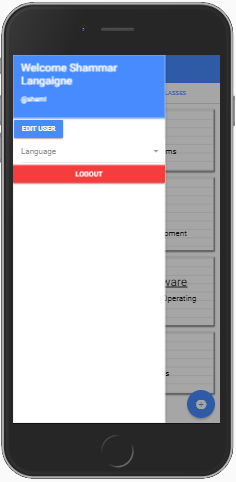
Here users are able to see the name, subject, description and ID of the class as well as the username of the teacher. If the teacher navigates to this popup they should have access to three options: Edit Class, Add Students and Delete Class. If a student of the class navigates to this page however, the only option they can choose is Leave Class.

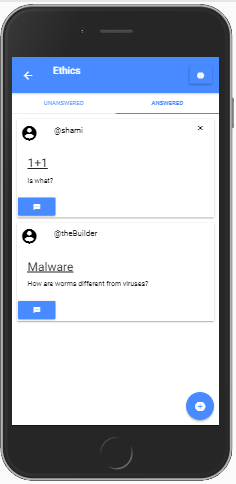
## Logging Out

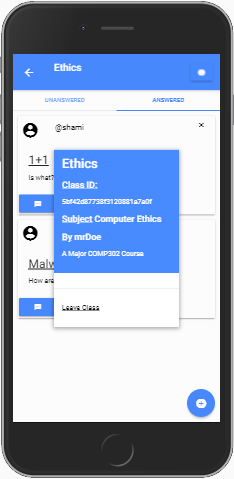
To logout access the sidebar at the home page and click the red logout button.   
This should bring the user back to the log-in screen.

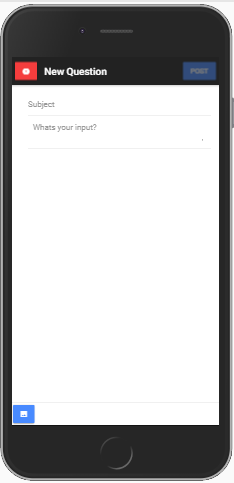
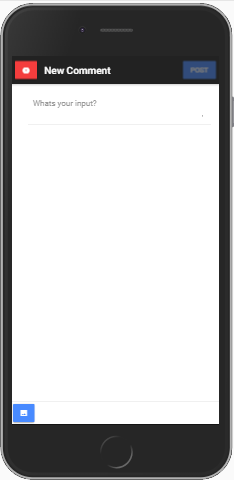
# Appendix

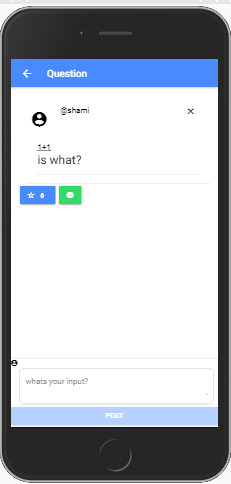
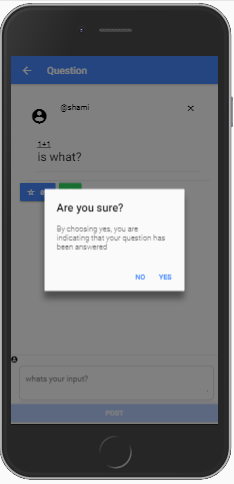


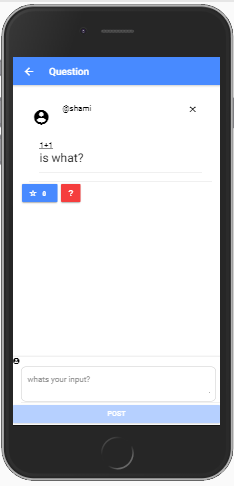
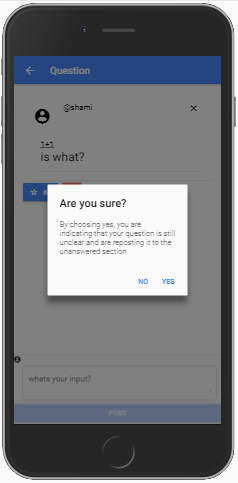
 

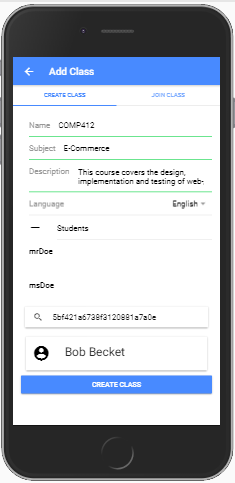
 

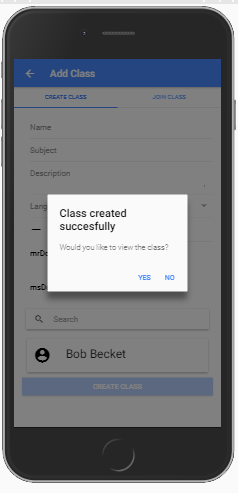
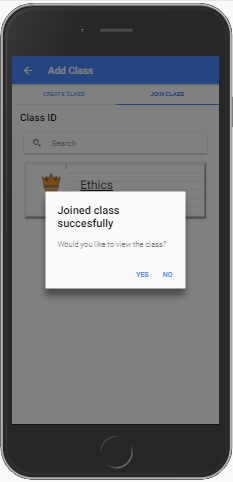


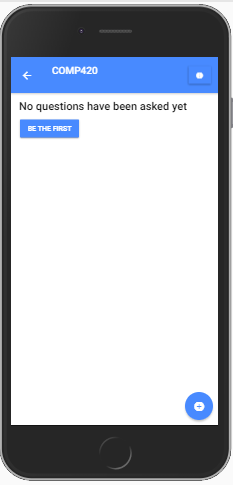
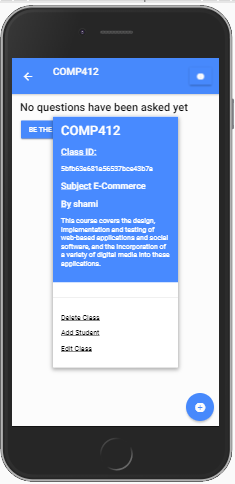
 

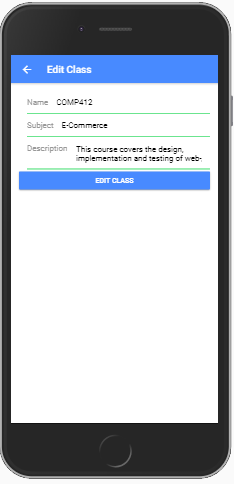
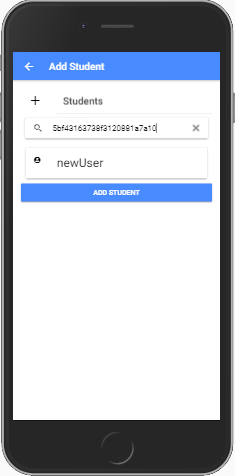
 

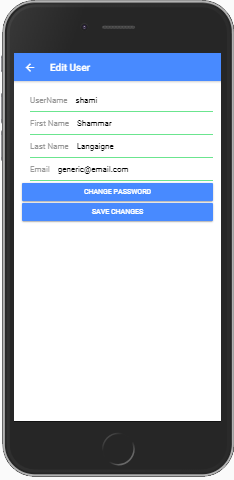
 

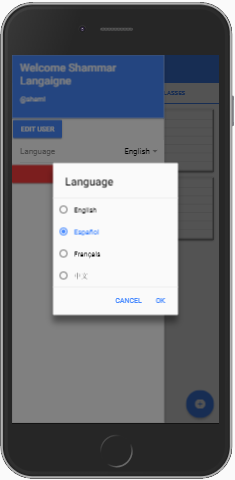
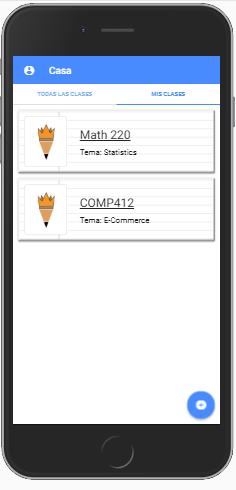
 

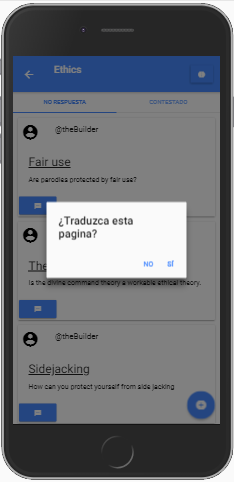
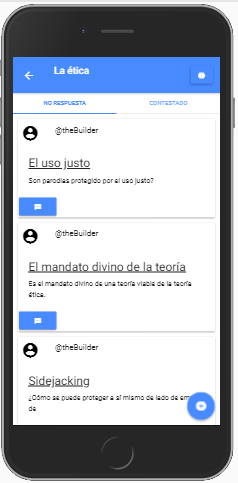
 

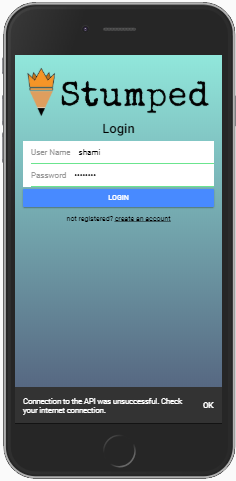
 

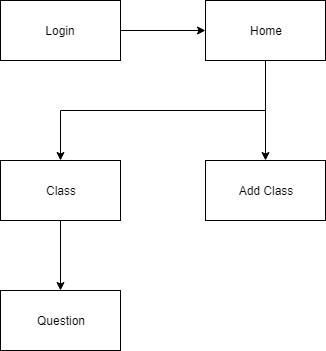


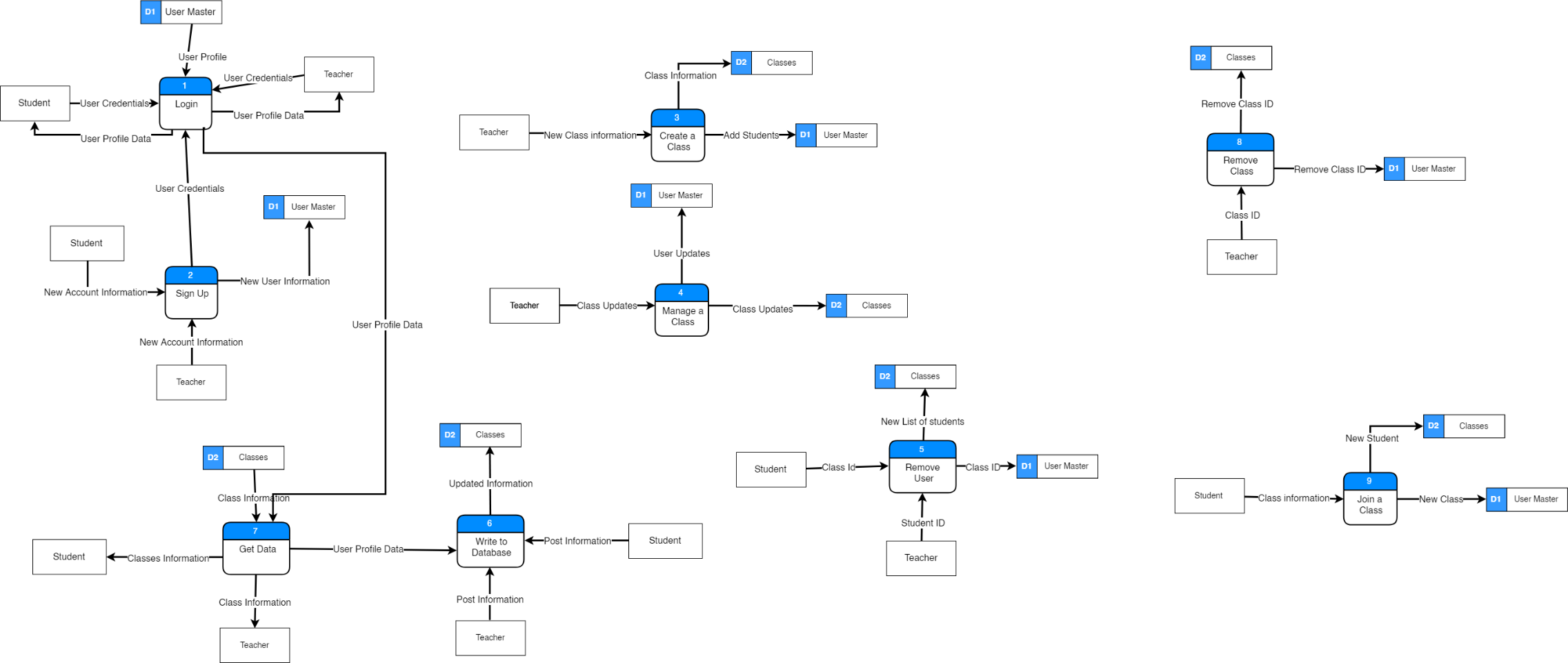
 



Site Map



Level 0 Data Flow Diagram

# References

aws.amazon.com. (n.d.). *Amazon EC2*. From aws.amazon.com: https://aws.amazon.com/ec2/

code.visualstudio.com. (n.d.). *Code editing redefined*. From code.visualstudio.com: https://code.visualstudio.com/

developer.android.com. (2018, September 24). *Meet Android Studio*. From developer.android.com: https://developer.android.com/studio/intro/

developers.google.com. (2018, September 21). *Chrome DevTools*. From developers.google.com: https://developers.google.com/web/tools/chrome-devtools/

github.com. (n.d.). *Learn Git and GitHub without any code!* From https://github.com/: https://github.com/

ionicframework.com/. (n.d.). From ionicframework.com: https://ionicframework.com/docs/

nodejs.org. (n.d.). *Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.* From nodejs.org: https://nodejs.org/en/

www.mongodb.comm. (n.d.). *MongoDB Atlas Deploy a fully managed cloud database in minutes*. From www.mongodb.com: https://www.mongodb.com/