California State University, San Bernardino
CSUSB ScholarWorks

Electronic Theses, Projects, and Dissertations | Office of Graduate Studies

6-2020

BUBBLE-IN DIGITAL TESTING SYSTEM

Chaz Hampton

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd

 Part of the Software Engineering Commons

Recommended Citation

Hampton, Chaz, "BUBBLE-IN DIGITAL TESTING SYSTEM" (2020). *Electronic Theses, Projects, and Dissertations*. 996.
https://scholarworks.lib.csusb.edu/etd/996

This Project is brought to you for free and open access by the Office of Graduate Studies at CSUSB ScholarWorks. It has been accepted for inclusion in Electronic Theses, Projects, and Dissertations by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

BUBBLE-IN DIGITAL TESTING SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

In

Computer Science

---

by

Chaz Timothy Hampton

June 2020

BUBBLE-IN DIGITAL TESTING SYSTEM

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Chaz Timothy Hampton

June 2020

Approved by:

Dr. Arturo I. Concepcion, Committee Chair, Computer Science

Dr. Owen J. Murphy, Committee Member

Dr. Ernesto Gomez, Committee Member

© 2020 Chaz Timothy Hampton

ABSTRACT

Bubble-In is a cloud-based test-taking system build for students and teachers. The Bubble-In system is a test-taking application that interfaces with a cloud server. The mobile applications have been built for Android and Apple devices and the webserver is hosted on Digital Ocean VPS run with Nginx. The Bubble-In application is equipped with anti-cheating mechanisms such as question-answer key scrambling, not allowing screenshots, screen recording, or leaving the application. The tests students take are sent to the webserver to be graded and have statistics calculated and displayed in easy to use format for the test creator. Instructors can use the webserver to create exams or modify existing exams. This application was developed in Android Studio and XCode. These features were built using Java, Swift, Obj-C, PHP, HTTP requests, and MySQL. The application interacts with the database through the Nginx web server.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Arturo Concepcion, all of my previous professors who have mentored me, the CSUSB university, my family, and my friends.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION AND BACKGROUND

1.1 Intro and Background

Classical distributed computing systems were the best in their time and offered many advantages to business and personal developers alike. With distributed systems, many computers are able to support the same service or systems and provide the opportunity to upscale. In 1960 during a speech at MIT, John McCarthy eluded “like water and electricity, computing can also be sold like a utility.” Later in 1999 the company Salesforce began distribution of its applications to the end users though a convent website. Later around 2002 Amazon Web Services was founded and a few years later it was joined by Google, Microsoft, HP and other large companies. The face of the IT industry was never the same from the inception of distributed cloud computing services. This is not only benefiting large corporations but also individual developers and freelancers today.

In a typical working day, the average computer professional is only utilizing a fraction of their computational resources. Cloud computing provides a way for users and developers to better manage what resources they want, and they then, only pay for what they need instead of needing to own all the hardware

themselves. In this research paper we will evaluate Software as a Service (SaaS) several deployment models such as private, public and community cloud environments. The differences in these models and in contract with conventional distributed computing will be explored. The time of classical grid computing is becoming an outdated feature to be replaced by a more dependable one, cloud computing. In order to be considered a cloud computing platform five key characteristics must be established. In this paper we will examine these characteristics and what advantages and disadvantages are associated with them. One issue is security in cloud computing and the debate around if it is truly more vulnerable or not.

1.2 Cloud Services Architecture

In cloud computing the users only have to pay for the resources they have or plan on consuming. This workload can be adjusted in the future to upscale or downscale accordingly. Classically the user or organization would have to purchase the resources they needed or thought they might need out right. It is common for products to have what is called peak hours or critical execution times where the most amount of demand is placed on the product or system. In the classical method the user or organization would have to purchase far more resources than they would need for regular day to day work if the peak performance was sufficiently high. With cloud computing this massive overhead can be avoided and all the user or organization needs is a web browser and

internet connection to use cloud computing even for massive projects. Cloud computing provides its end users with resource pooling and elasticity or the ability to upscale or downscale [1]. In fact, most of today's social media outlets and websites use a cloud computing service. This compelling fact hints that cloud computing has something to offer to many different types of clients. In general, the reason for this massive adoption is the scalability, agility and simplicity involved with cloud systems. It is just too convenient for businesses and developers alike to use a cloud service to host their platform on that anything else seems archaic.

1.3 Interactive Bubble-In Applications

When using cloud and RESTful services to dynamically present and load data on mobile applications, how does the application receive input from the user and have the resulting input computed and then displayed back to the device? By having the UI or User Interface remain fully on the physical device, the rest of the application can run remotely on the cloud service. By having the applications UI deliver the input events to the remote server for execution they can then be computed safely and securely [14]. This cloud-based server is able to be dynamically scaled up or down depending on the demand placed on the server. This works in the Bubble In application where an agent monitors occurrences of user input then executing on the virtual cloud environment see Fig 1 for more detail. After some time has passed, a window will display prompting for input.

The remote cloud server detects this input and runs this data through several algorithms, the first of which check the validity of the students provided credentials for the desired test. The student who is authenticated will then be presented with a dynamically loaded UI from the cloud server onto the test page of the application with the exam's contents displayed. This process of having the mobile device interact with the server can be seen in Figure 1. In Figure 1, mobile devices are connecting to the Digital Ocean cloud server where the computations for the exams take place.

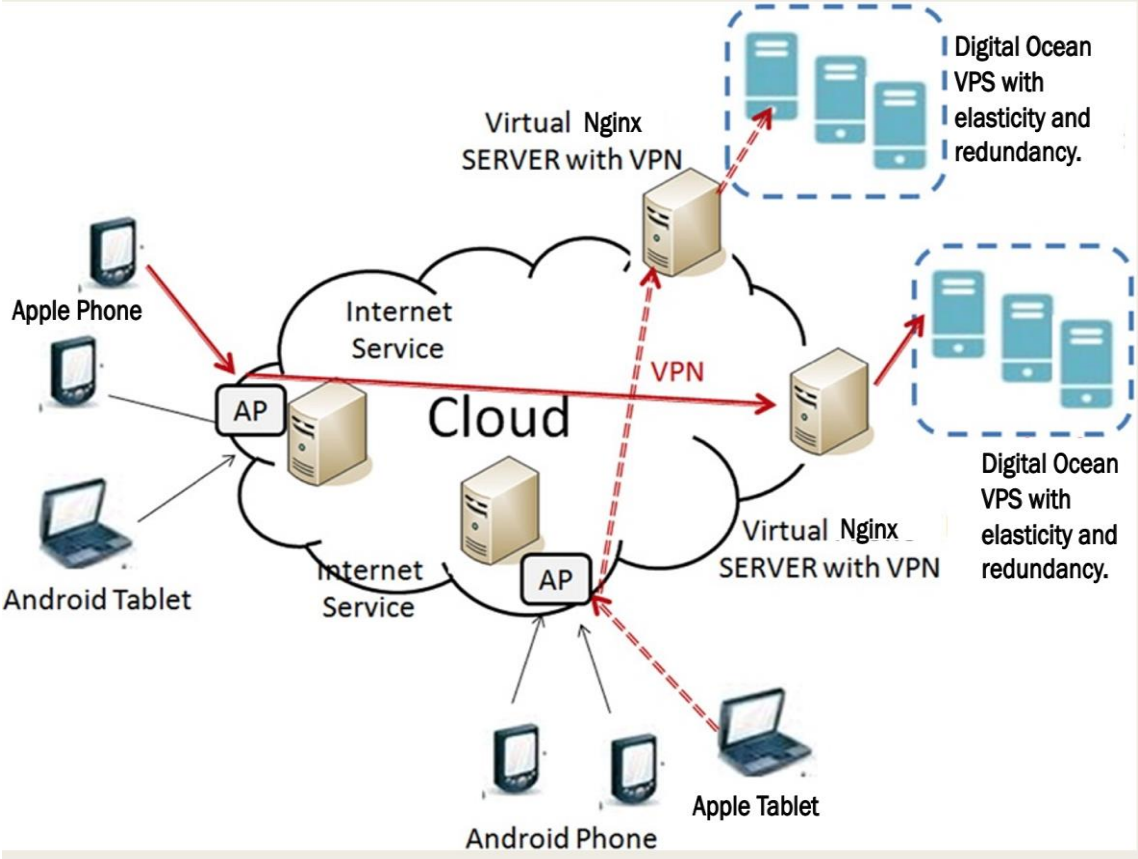

Fig. 1. A VPN Cloud Connection to a Mobile Device [8]. (Edited)

1.4 Cloud Implementation Overview

Consider an average workday in your office or workstation, how much of your machine's resources do you expect are being utilized? Well, as stated by Marston in the article "Cloud Computing a Business Perspective" the average for most users is about 10 percent of the processor, 60 percent of the memory and 20 percent of the bandwidth During peak usage hours [14]. Regardless the user paid 100 percent of the cost of those resources up front when they purchased the computer. Now imagine a very large corporation with many computers going underutilized. If these computers could be used to their potential or the company invested in less resources, they could optimize their performance. This is not the case however since many products need to be flexible and scalable on demand. It is not conceivable to buy and sell computers on the fly, this is where cloud computing saves the day. Web servers, application servers and database servers are commonly under minimal usage when they could be taking advantage of pooled resources.

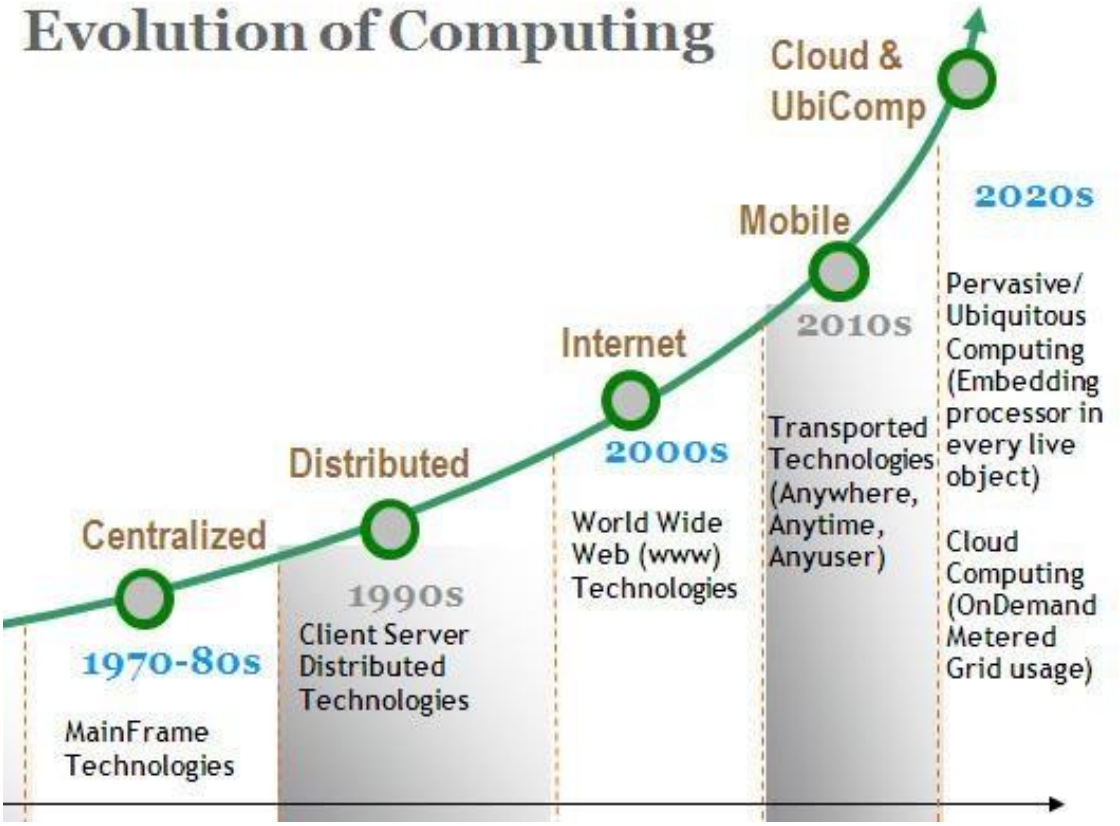

Fig. 2. Evolution of Cloud Computing [8].

When using a cloud computing platform your company would not have to invest up front as a capital expenditure, in buying the hardware but simply signs up with a service provider in a pay as you go or pay what you use billing model [8]. This model means that companies can change from a capital expenditure model (CapEx) to an operating expenditure model (OpEx) just for meeting the computational needs.

1.5 Three Basic Cloud Computing

Client computers (the end user can interact with the cloud using the client computers), Distributed computers (the servers are distributed among the different places but acts like they as working with each other) and data center's (the compilation of the servers) [8]. There exist three main services cloud computing provides which are: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). In a public cloud system, which is the system the Bubble In application utilizes, access is available to anyone with an internet connection and this system may exist anywhere worldwide. This type of service has the problem of data integrity which arises partly from regulatory regulations. Some corporations for example those based in the United States are not allowed to store consumer data in other countries. This issue is so detrimental for some that they opt for private cloud solutions [8]. A private cloud solution is one that provides services to a single entity. This single entity maybe a government, corporation or anyone else willing to pay for the resources. While this service is available to anyone, it is expensive and is usually reserved only for large enterprises and governments [4]. Lastly, the community cloud solution can be decently described as a middle ground for private and public cloud services. However, community cloud systems come with pros and cons of both implementations. In this implementation entities with common interests can pool their resources to create what is called a hybrid cloud.

1.6 RESTful Protocol

REST (REpresentational State Transfer) is an architectural style based on transferring representations of resources from a server to a client. It is the style that underlies the web as a whole and has been used as a much simpler method than SOAP/WSDL for implementing web services. A RESTful web service is identified by its URI (Universal Resource identifier) and communicates using the HTML protocol. It responds to HTML methods GET, PUT, POST, and DELETE and returns a resource representation to the client. Simplistically, POST means create, GET means read, PUT means update, and DELETE means delete. RESTFul services involve a lower overhead than so-called ‘big web services’ and are used by many organizations implementing service-based systems that do not rely on externally provided services [2].

1.7 RESTful Cloud Pitfalls

REST is a flexible and powerful technique used to communicate in distributed systems. HTTP provides direct stateless access to web operations. As stated in the paper Managing Authorization with RESTful XML, “The reason for using REST as interaction technique lies in the supported operations going along with the HTTP-verbs, the lack of com- plex handling of states and the variability regarding resource characteristics.” REST is able to work with many types of resources even when they are represented in custom formats or JSON [3]. In Figure 3 we can see how the Bubble-In system is implementing RESTful protocols. The Laravel framework handles requests which are resources inside of the project. These resources are created in code files such as course, class and HTML files. Outside of the figure we can see a campus with multiple classrooms utilizing the system.

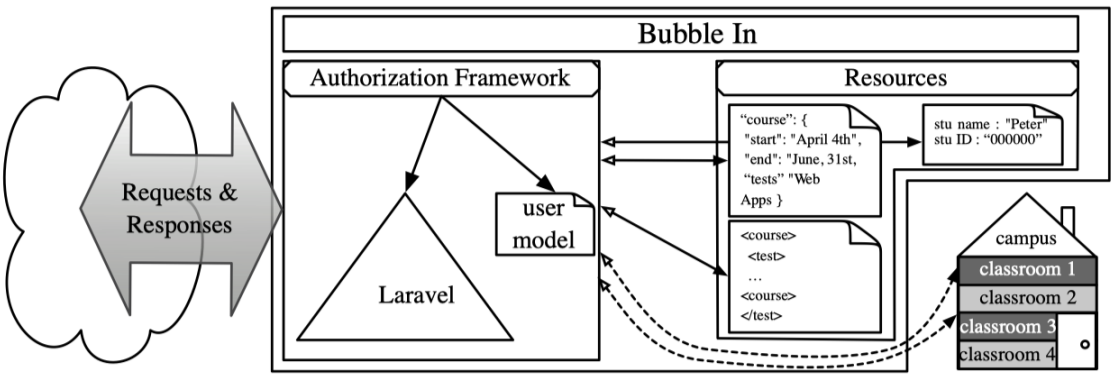

Fig. 3 Overview of RESTful Architecture [8]. (Edited)

This is why REST services are used in the Bubble In application where the user data communicated are a mix of string and integer values as well as formatted JSON data types. While the widespread adoption of cloud computing seems to benefit every party, this may not be the case when considering security [8]. Security is comprehensive, not only the providers use of the system, but also yours and any middle user or application. When evaluating a systems security, we must do so in an end-to-end approach. From the end user to the data center many regulation laws and security issues may be present and provide issues in data integrity. If the data sent by the user to the data center or data center to the user can be intercepted and read, our data integrity has been compromised. The only way to ensure maximal data integrity seems to be to have secured end points which are the user and data center [8]. Further, the data channel which information is transferred must be secure at all times. This data to be transferred must also be in an encrypted state to avoid man in the middle attacks where some entity intercepts the message and can use it to their liking.

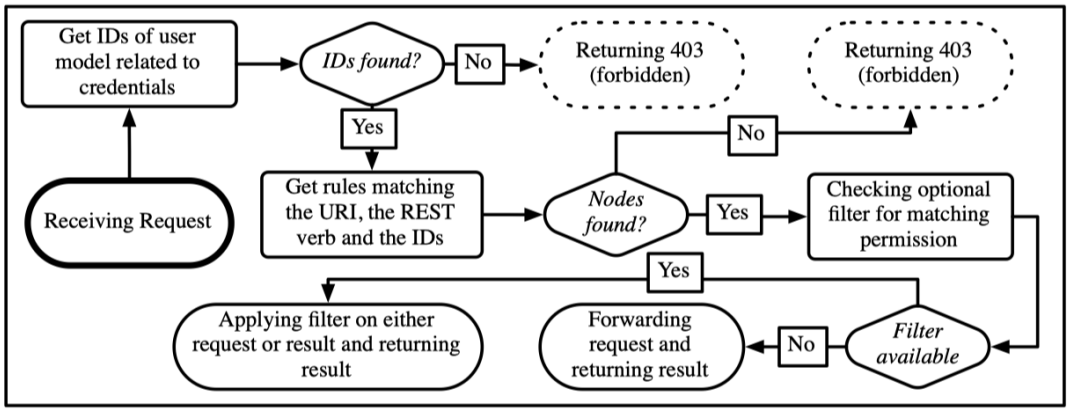

Fig. 4 Authentication Workflow [8].

What is accomplished by a secured and encrypted channel is third party software is unable to compromise data integrity of the cloud system. Most cloud systems use encryption keys when encrypting a channel. Another method is to use a public key chain such as a certificate authority to verify this public chain (CA), in this method both parties must agree on the protocols and keys to be verified valid. In Figure 4 we can see how the Laravel authentication framework handles requests. This framework comes with multiple loops of error checking upon receiving a request from a mobile device.

1.8 Security and Privacy

Since a virtual cloud environment will usually operate in a public cloud, it is critical that the user's data will communicate over a secured channel and all end points are also secured [1]. In addition to this is it is important to choose a reputable IaaS provider when deploying the virtual cloud environment. In the

case of the Bubble In system Digital Ocean VPS is used to host the cloud server which is a small scale but also reliable cloud service provider. Digital Ocean provides state of the art security measures from on premises security, backup generators, full CCTV, Role Bases Access Control to mitigate unrestricted access to any instance. The Bubble In instance hosted on Digital Ocean also provides security monitoring as well as instance backup and recovery options. A verbose authentication and encryption are required to secure and establish the communication channel. Storing SSH exchange keys for connection is utilized in Bubble In to provide this encryption and authentication to the cloud server. For a stronger level of security, multiple user accounts with the minimum required permissions are provided for admin and developer accounts alike.

1.9 System in Summary

Cloud computing is a catalyst for the future to come in which social, professional and political life are affected. Its benefits are a tremendous improvement over conventional distributed grid style computing service. Cloud computing's services being offered at such a reasonable set of pricing options is truly the best choice for most companies in the modern world. The startup costs for this service when compared to traditional methods are vastly less and now bringing a new product or idea to fruition is simpler, cheaper and better than ever before. While there are certain security concerns, with practical and responsible security measure they can be effectively dealt with. We have discussed many

practical approached to implementing safe and secured communication channels over RESTful service communication. By taking greater care of the increased number of end points added by the cloud system it can remain secured. In general, the benefits cloud computing offers dwarf its drawbacks making it a very logical component in the future of industry and education for the coming years. In applications such as Bubble In the benefits of RESTful services combined with Cloud Computing are critical to operational success. The flexibility of both REST services and dynamic cloud services are key should the applications resources need to be quickly upscaled. The same issues found in this paper were addressed in the creation and management of the Bubble In cloud system where security measures and restrictions are placed throughout the system at the application and server level as well as authentication on each component and user account. By taking carful considerations for the security and integrity of the system it is now able to operate in public cloud system without the worry of data a breach due to negligence.

2.0 System Implementation

These previously discussed topics will be implemented into the Bubble In system with Service-Orientated Architecture. The Bubble In PHP Laravel Web Server will be hosted on a cloud server provided by Digital Ocean. Digital Ocean is a reputable cloud computing company with reasonable pricing and flexible plans. Laravel is made for the development of web applications following the

model–view–controller (MVC) architectural pattern. Features of Laravel include: a modular packaging system with a dedicated dependency manager, different ways for accessing relational databases, utilities that aid in application deployment and maintenance, and its orientation toward easy to understand syntax. Most of the Bubble In test taking system will be computed on the cloud to minimize the workload on mobile devices. The student's individual grades, statistics for each exam such as most and least chosen question and printable PDF documents will be computed on the cloud. The mobile devices will interface with this cloud system via the RESTful paradigm. In accordance with Representational State Transfer (REST) protocols our web application will use HTTP looks ups such as GET, PUT, POST and DELETE requests to communicate with devices. Service-Oriented Architecture in Bubble In allows the ability to: combine a large number of facilities from existing services to form applications, encompass a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system, computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains. With Bubble In we will combine multiple web server programing standards such as HTML, CSS, PHP, and some JavaScript. System development will be decentralized and stored on a git repository. The web server will be built using visual studio and the command terminal. The server will run on Nginx and use a MySQL database for storage.

The cloud server will also be outfitted with an SSL certificate to allow secure and trusted communication. The mobile applications will be developed in XCode and Android studio with native code. When the full system is in place, it could be used by many different organizations at the same time with no issue needing to add new code.

CHAPTER TWO:

SCANTRON VS. BUBBLE-IN

2.1 Scantron

Scantron sheets grew in popularity as a test taking aid in the mid 1970's. A small Scantron reader can cost $4000 [5], the type of readers used by large organizations can cost up to $85000 [6]. Scantron readers can be unreliable and if broken may take days to fix at best by a certified technician (More cost) Readers can be temperamental and produce errors during grading Students must purchase their forums and may forget them under stress such as on exam day. Instructors must manually enter grades for each student Students could cheat by copying very easily since randomizing questions is tedious for the teacher and makes the grading process much longer and complicated.

2.2 Bubble-In

The Bubble-In application implements the following:

- Anti-cheating mechanisms such as disabling screenshots, screen recording reading messages or leaving the application.
- Automatic question-key scrambling
- Automatic exam statistic calculation
- Immediate test exam feedback

- Low overhead to begin using the system
- Low-no maintenance cost
- Portable grading sheet (PDF, CSV) available for download and print via single click
- Saves money for students and schools
- Customizable test layout design accommodating text based and bubble in only formats
- Easily accessible to download on mobile devices

2.3 Bubble-In vs. Scantron

Scantron Pros:

- Integrated into schools
- Familiar

  Cons:

  - Expensive overhead to start
  - Expensive maintenance
  - Unreliable
  - Lacking in grading features
  - Lacking in anti-cheating features

Bubble-in Pros:

- Feature rich
- Anti-cheating mechanisms

- Automatic question-key scrambling
- Automatic exam statistic calculation
- Immediate feedback
- Low overhead
- Low-no maintenance cost
- Portable grading sheet (PDF, CSV)
- Saves money for students and schools

Cons:

- Instructor must be willing to utilize a webpage to create exams

From this comparison we can see compelling reason to use the Bubble-In system instead of the classical Scantron for taking exams.

CHAPTER THREE:

SOFTWARE REQUIREMENT SPECIFICATIONS (SRS)

3.1 Purpose

This document defines the requirements for the Bubble-In mobile application. The mobile application is being developed primarily for teachers, but with the students use as well.

The aim is to satisfy the requirements given by Dr. Helena Addae, Associate Professor in the Department of Management. Dr. Addae wants an application that allows students to be able to use the application to take a test, and for that test to be graded without the use of a typical ScanTron reader. The goal is to focus on security of the application, implement a optimized server side application, and add functionality to the application.

3.2 Scope

The first task is to disable screenshots within the application. I want to keep a running total of how many times a student leaves the application while taking a test. Should the student exit the application more than once, the test will be submitted. The teacher will have access to these records and can then take disciplinary action if necessary. A time feature will also be implemented, this will

not allow students to submit the exam outside of designated time. Others are a delete test function for the teacher and improved server programming and a method to scramble test questions for each student. In addition, applications will be made on Android and IOS to support most phones in the class. Loaner tablets may also be in each class in case of a forgotten device or another brand of phone not supported. The Server must be hosted on a cloud-based solution to ensure it is scalable, reliable and dependable during exam time.

3.2.1 Definitions, Acronyms, and Abbreviations

Definitions, Acronyms, and Abbreviations	
Android	Google's mobile operating system
Android Studio	IDE for software development
Apache Web Server	Web server software
App	Application
HTML	Language used to create webpages
HTTP	HyperText Transfer Protocol. Used to send signals to server
IDE	Integrated Development Environment
Java	Language used on Android Studio and promotes functionality to mobile app
MySQL	Database language that interacts with a server to query data
PHP	Server-side scripting language
SRS	Software requirement specifications

SDK	Software development kit
Wifi	Wireless internet for devices
XML	Language that produces layout and design
ScanTron	A form that requires penciled-in answers and machines check its correctness
Test Id	A specific generated identification number that uniquely identifies a singles test key
Xcode	Apples IDE for software development
IOS	Apple's brand of mobile devices

3.2.2 System Interfaces (DEPLOYMENT DIAGRAM)

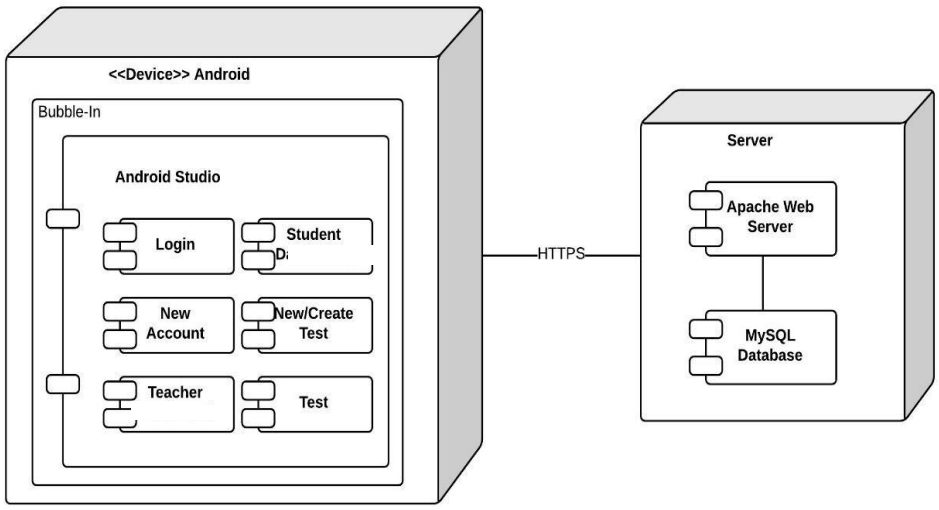

Fig. 5. Deployment Diagram.

This application has been developed in Android Studio and XCode. These features have been built with the use of: Java, Swift, Obj-C, PHP, HTTP requests and MySQL. The application interacts with the database through a Nginx web server.

3.2.3 User Interfaces

- Login Page – Allows an existing user to sign into his/her account and provides a link to the new account page. Successful login links to the Teacher's Dashboard page. Students require no login and will continue as students.
- New Account Page – Allows a teacher to create an account to begin managing tests and grades. Teacher's Dashboard Page – This page lists all tests the teacher has created and allows the teacher to view student grades and the key for each test. This page also allows the teacher to create a new test.
- Create Test Page – This page allows a teacher to create a new test. The teacher will be able to generate a random test ID and specify the number of questions.
- Student Page – Used to keep all info about one particular student in the course. Enrolled courses and all previous scores on exams will be found in this file.

- Teacher Page – Used to store all data on teacher accounts. The courses teachers have created are stores here along with all exams made by the teachers.
- Test Page – Used to store exams used by teachers currently in use.
- Apache Web Server- Engine used in hosting the Bubble-In cloud server.

3.2.4 Software Interfaces

- Java – The primary language for developing any functions that will take place on Android devices.
- Swift – The primary language for developing any functions that will take place on Apple devices.
- PHP – Used to communicate with the server and push/pull data from the database.
- HTTP – Communicates with the web server. Also uses JSON requests to get/push to the database.
- MySQL – Used to create the database and tables in which the data will be stored as well as provide any necessary queries.

3.2.5 Communication Interfaces

This application will use either cellular (3G, 4G, 5G) data or Wi-Fi to request and submit data from the server.

3.2.6 Memory

This application is built for all android systems that run Android 4.4 KitKat (API 19) and above. Apple devices running IOS 10.3 and above are supported. This legacy support ensures 90% of devices support the application.

3.2.7 Operation

Should the application need changes these can be made offline, tested and then published for a new app store / google play store update.

3.2.8 Product Functions (USE CASE DIAGRAM)

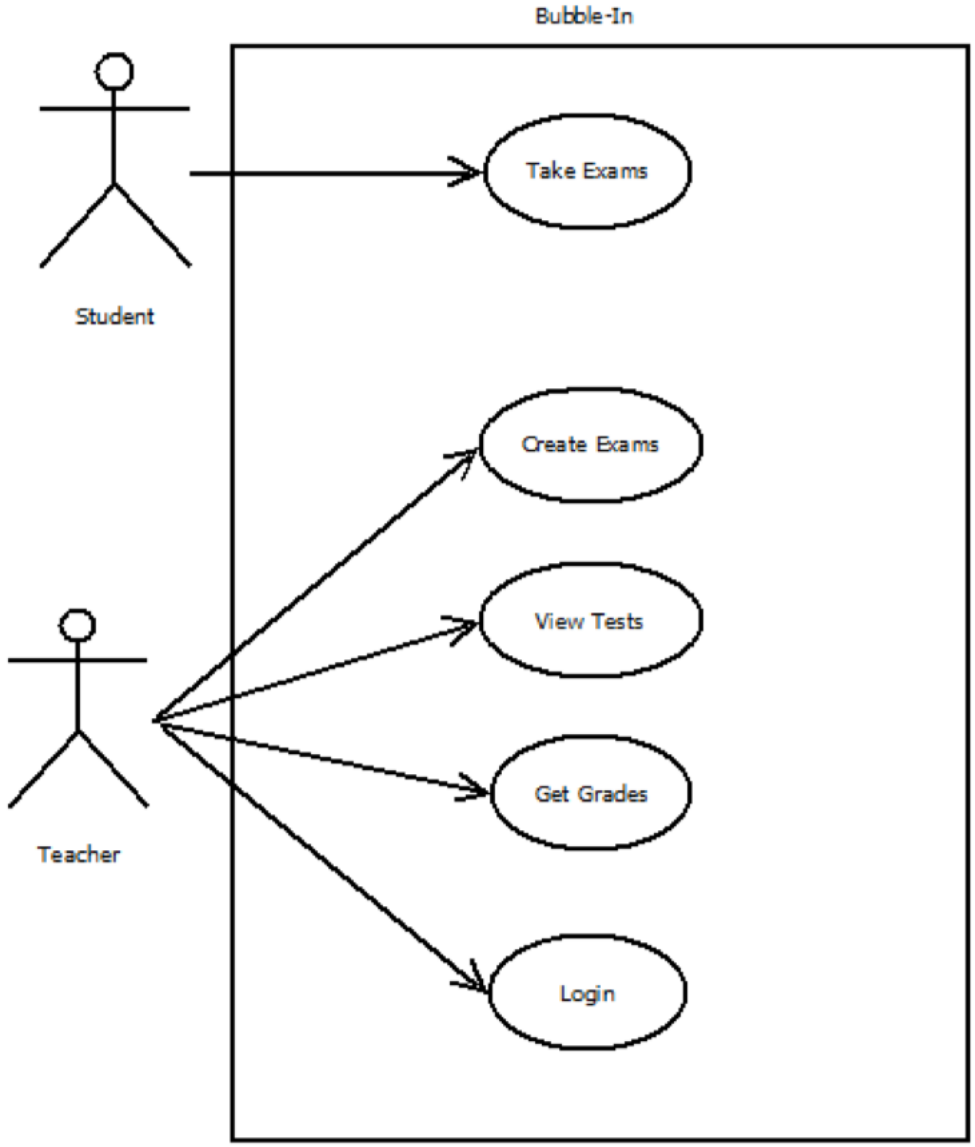

Fig. 6. Use Case Diagram.

3.3 User Characteristics

A user can be either a student or a teacher. Both the student and teacher can access the application on their Android/Apple device.

3.3.1 Constraints

Operating system constraint - The user must have Android 4.4 KitKat (API 19) and above. Apple devices running IOS 10.3 or later installed on his/her mobile device.

3.3.2 Assumptions and Dependencies

- The user must have Internet access on his/her mobile device in order to use the application.
- There must be a server to store the database which contains student and teacher data.

3.3.3 User Interfaces

This section shows the screens that the user interacts with in the application.

3.3.4 Login Page

When the user enters the app, he/she will encounter a login page.

3.3.5 Test Page

This page is the actual test. When the students arrive at this page, they will be given the test that the teacher has generated.

3.3.6 Thank You Page

This page confirms submission of test. As long as the test was submitted to the webserver successfully this will display to the user.

3.3.7 Hardware Interfaces

This application supports on Android 4.4 or IOS 10.3 or higher.

3.4 Software Interfaces

This application uses MySQL for its database and has been developed using Android Studio and XCode.

3.4.1 Communication Interfaces

This application will require an internet connection via Wi-Fi or mobile data for access to the database.

3.4.2 Functional requirements

In software engineering, functional requirements define a function of a system or its component.

3.4.3 Login Screen

Teachers will be able to create an account if they don't already have one and log in with their credentials. Students will have to continue as students to use the app.

3.4.4 Test Page

A student user will be able to answer questions and submit them to the database. A teacher user will use this test page to create test keys.

3.4.5 Performance Requirements

The application will require a stable internet connection to communicate with the database. The response for the individual grades for a student should be within a few minutes from the time they submitted their test answers.

3.4.6 Design Constraints

This application is designed for Android and Apple mobile devices and therefore constrained to those devices.

3.4.7 Software System Attributes

The requirements in this section specify the required reliability, availability, and security.

3.4.8 Reliability

This application can be used on most Android and Apple mobile devices.

3.4.9 Availability

The application will always be available. However, the functionality will be gained through access that professors will grant.

3.5 Security

Ideas implemented in order to protect software from malicious or accidental harm.

3.5.1 System

Students will only be able to take the test within the open test date and time. Suspicious activities such as when the app is put into a background process or take screen shots will be logged. The Laravel framework hides the source code and by scrambling test questions we ensure no two students may copy off each other.

3.5.2 Testing

Is an evaluation of the quality of the software produced and consists of several types of testing.

3.5.3 Unit Testing

Each page will be tested separately for correct functionality.

3.5.4 Integration Testing

All the pages will be integrated into the mobile application, which then they are going to be tested in different sequences.

3.5.5 Acceptance Testing

After the testing of unit and integration, this testing will make sure the product is ready to be used when it is released.

3.5.6 Asset List

CHAPTER FOUR:

SOFTWARE ARCHITECTURE AND DESIGN (SAD)

4.1 Individual Architecture Design

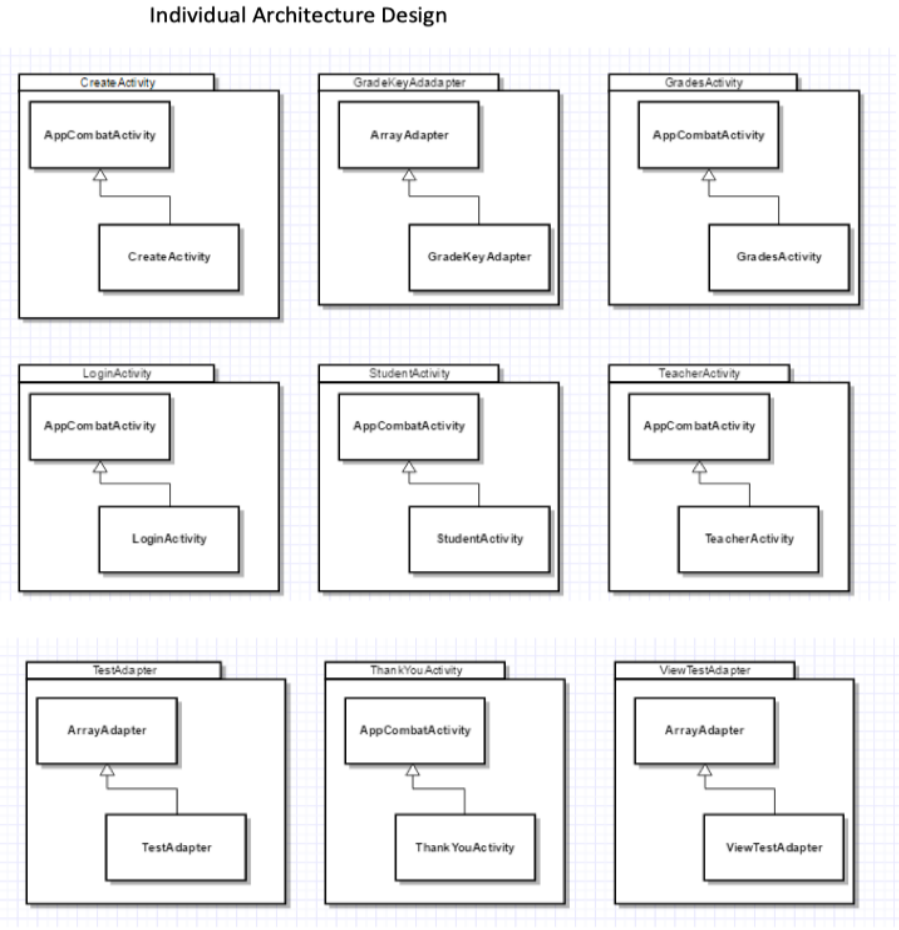

Fig. 7. Individual Architecture Design.

- LoginActivity – When the user enters the app, he/she will encounter a login page. If the user is a teacher, he/she will login using his/her username and password. If the user is a student, there is a button that allows the user to continue as a student. There is also a “Create account” button if the teacher does not yet have an account
- WarningActivity – Displays a warning text box to inform the student that the application is using anti-cheating mechanisms. The app prevents screenshots, leaving the application and other features.
- TeacherActivity – After a teacher selects a test key that has already been created, he or she will be directed to this page. This page will have three different columns that the user can choose from. One will contain the grades the students for that particular test. The middle column will contain the answers to the test which can be modified as well as the individual weights associated. Lastly, the right column will contain settings for that particular test.

4.2 Application Architecture Design

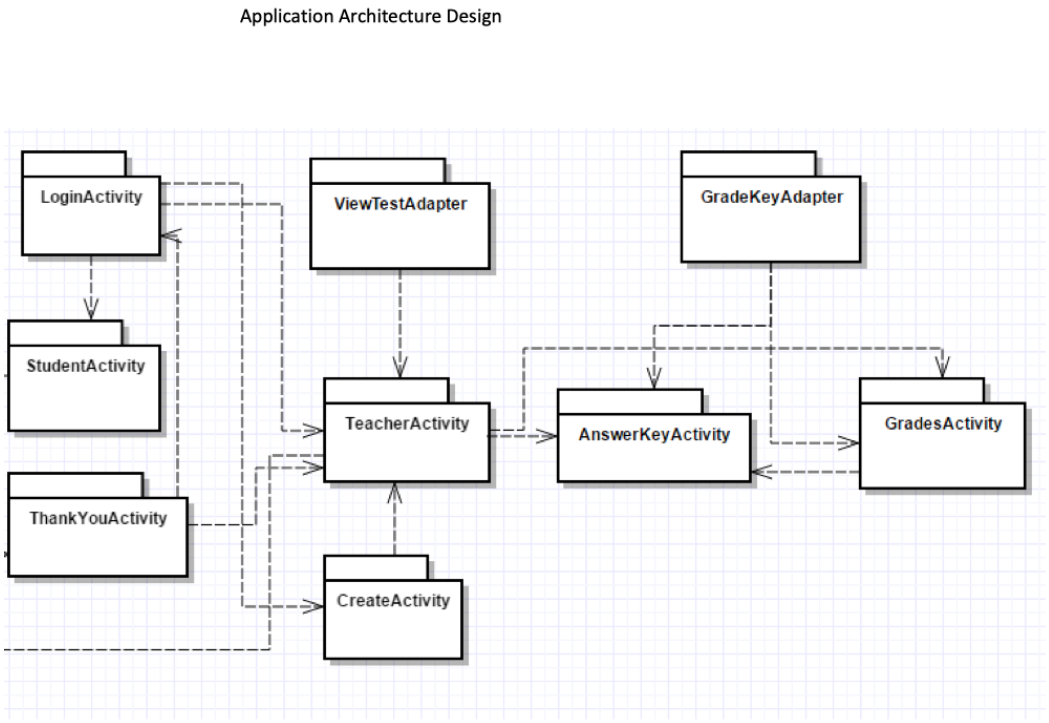

Fig. 8. Application Architecture Design.

- LoginActivity – When the user enters the app, he/she will encounter a login page. If the user is a teacher, he/she will login using his/her username and password. If the user is a student, there is a button that allows the user to continue as a student. There is also a “Create account” button if the teacher does not yet have an account

- WarningActivity – Displays a warning text box to inform the student that the application is using anti-cheating mechanisms. The app prevents screenshots, leaving the application and other features.
- TeacherActivity – After a teacher selects a test key that has already been created, he or she will be directed to this page. This page will have three different columns that the user can choose from. One will contain the grades the students for that particular test. The middle column will contain the answers to the test which can be modified as well as the individual weights associated. Lastly, the right column will contain settings for that particular test.
- ThankyouActivity – Displays to the user that their test has been successfully submitted.

CHAPTER FIVE:

IMPLEMENTATION AND CODING

5.1Test Controller Source Code IOS

This code file is described as TestActivity in the previous chapter.

```
3   // Displays the Answer Sheet
4   class TestViewController: UIViewController, UITableViewDelegate,
    UITableViewDataSource
5   {
6       var OptionBText = ["-":"-"]
7       var OptionCText = ["-":"-"]
8       var OptionDText = ["-":"-"]
9       var OptionEText = ["-":"-"]
10      var onResumeCounter = 1         // amount of times the user is able to leave
    the app "accidently"
11      var allowedTimeOutsideApp = 5.0 //Amount of time outside the app that is
    aloud
12      var TestTime = 0
13      var timer = Timer()
14      var dateLeft = Date()
```

```
15      var ReasonForSubmission = ""
16      var SubmissionID = 0
17      var TestCode = 0
18      var notificationCenter = NotificationCenter.default
19
20          //set up a short and long tap gesture, must hold the submit button to
    submit answersheet
21          let tapGesture = UITapGestureRecognizer(target: self, action:
    #selector(normalTap))
22
23          let longGesture = UILongPressGestureRecognizer(target: self, action:
    #selector(longTap))
24
25          tapGesture.numberOfTapsRequired = 1
26          SubmitBtn.addGestureRecognizer(tapGesture)
27          SubmitBtn.addGestureRecognizer(longGesture)
28
29          timer = Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:
    #selector(UpdateTimer), userInfo: nil, repeats: true)
30      override func didReceiveMemoryWarning() {
31          super.didReceiveMemoryWarning()
32          // Dispose of any resources that can be recreated.
```

```
33      }
34
35
36      @objc func normalTap(sender: UITapGestureRecognizer)
37      {
38          SubmitBtn.setTitle("Hold to Submit", for: .normal)
39      }
40
41      //called by notificationCenter whenever app moves to the background
42      @objc func movedToBackground ()
43      {
44          self.onResumeCounter -= 1
45          self.dateLeft = Date() // Sets the dateLeft var to the time when the app
    was but intot the backgroud
46      }
47      // called bt the noticationCenter whenever the app resumes from a
    background state
48      @objc func startApp()
49      {
50          let TimeSpentOutside = abs(self.dateLeft.timeIntervalSinceNow)
51          print(TimeSpentOutside)
52
```

```
53        if onResumeCounter < 0 || TimeSpentOutside > allowedTimeOutsideApp
54        {
55            var method = ""
56            let StudentAnswers = getAnswerString()
57            if onResumeCounter < 0
58            {
59                method = "Exited"
60            }
61            else if onResumeCounter > 0
62            {
63                method = "Time Exited"
64            }
65            doInBackGround(method, Answers: StudentAnswers)
66        }
67    }
68
69    @objc func ScreenShot()
70    {
71        let StudentAnswers = getAnswerString()
72        doInBackGround("Screen Shot", Answers: StudentAnswers)
73    }
74  @IBAction func radioSelected(_ sender: DLRadioButton) {
```

```
75        answerArray[sender.tag] = (sender.titleLabel?.text)!
76    }
77
78    //the following function will create an array of strings for numbering our
      questions
79    //it also initializes our answerArray with 0 representing blank
80    func generateDataSet (_ questions: Int) -> [String] {
81        var dataSet = [String]()
82        var count = 0
83
84        while (count < questions && count < 9) {
85            count += 1
86            let tempString = "0\(count)"
87            dataSet.append(tempString)
88            answerArray.append("0")
89        }
90
91        }
92
93        return dataSet
94    }
95
```

```
96      //this function converts our array of strings into a single string
97      func getAnswerString () ->String {
98
99          var answerString = ""
100             noAnswer = ""
101
102             //convert answers array to a string
103             for (index, answer) in answerArray.enumerated() {
104
105                 //while building the string, if we come across an unanswered
    question
106                 //notify the user.
107                 if answer == "0" {
108                     noAnswer += "\(index+1), "
109
110                 }
111
112                 answerString += answer
113             }
114             return answerString
115         }
116             @objc func UpdateTimer()
```

```
117         {
118             if (TestTime > 0)
119             {
120                 TestTime -= 1
121                 print(TestTime)
122             }
123             else
124             {
125                 let StudentAnswers = getAnswerString()
126                 doInBackGround("OutOfTime", Answers: StudentAnswers)
127             }
128         }
129
130         func getReasonForSubmission(reason: String) -> String {
131             switch reason {
132             case "Exited":
133                 return "You have exited the test too many times and your Test was
    submitted."
134             case "Time Exited":
135                  return "You have left the test for too long and your Test was
    submitted."
136             case "OutOfTime":
```

```
137             return "Time has expired and your Test was submitted."
138         case "Screen Shot":
139             return "Screen Shot taken and your test was submitted."
140         default:
141             //should not be called ever
142              return "Something Went Wrong and your Test was Submitted"
143         }
144     }
145
146     override func prepare(for segue: UIStoryboardSegue, sender: Any?)
147     {
148         if segue.identifier == "TestToBad"
149         {
150             let badVC: BadViewController = segue.destination as!
    BadViewController
151             badVC.reason = getReasonForSubmission(reason:
    ReasonForSubmission)
152         }
153
154     }
155
156
```

```
157     }
```

5.2 Test Controller Source Code Android

This code file is described as TestActivity in the previous chapter.

```
1.
    public class TestViewActivity extends AppCompatActivity
2.  {
3.      //Test questions and Time variables
4.      int numberofQs = 0;
5.      int TestTime = 0 ;
6.      int SubmissionID = 0;
7.      String TestName = "";
8.      String TestCode = "";
9.
10.     //json object for the database
11.     JSONObject jsonObject;
12.     boolean canSumbit = false;
13.
14.     //timers for the exiting of the app
15.     public long pausedTime;
```

```
16.  public long allowedTimeOutsideApp = 10000; //This is a time representation in
     milliseconds
17.  public int onResumeCounter = 0; // number of times you have left the app
18.  public boolean exited = false;
19.
20.  //Views for disabling the status bar
21.  private WindowManager manager;
22.  private WindowManager.LayoutParams localLayoutParams;
23.  private customViewGroup view;
24.
25.
26.  @Override
27.  protected void onCreate(Bundle savedInstanceState)
28.  {
29.      super.onCreate(savedInstanceState);
30.
31.      DisableStatusBar();
32.
33.      setContentView(R.layout.activity_test);
34.      try
35.      {
36.
```

```
37.         //Populate Arrays
38.         //Number of elements of the TestViewActivity = numbers variable that was
    sent form the server.
39.         // makes the number double digits
40.         for (int i = 1; i <= numberofQs; i++)
41.         {
42.             if (i < 10 )
43.             {
44.                 numbers.add("0" + String.valueOf(i) + ".");
45.             }
46.             else
47.             {
48.                 numbers.add(String.valueOf(i) + ".");
49.             }
50.             //fills the array that will be sumbited with all 0s
51.             submitted.add("0");
52.         }
53.
54.         //pass submited through the adapter page
55.         Intent sub = getIntent();
56.         sub.putStringArrayListExtra("sub",submitted);
57.
```

```
58.         //init listview
59.         scantron = (ListView) findViewById(R.id.listView);
60.
61.         //Add the header at the top of the list
62.         header = new TextView(this);
63.         header.setBackgroundColor(Color.parseColor("#EF5D16"));
64.         header.setTextColor(Color.parseColor("#FFFFFF"));// White writing
65.         header.setTextSize(30);
66.         header.setGravity(Gravity.CENTER_VERTICAL |
    Gravity.CENTER_HORIZONTAL);
67.         scantron.addHeaderView(header);
68.
69.         // set the TestViewActivity of header to the TestViewActivity name
70.         header.setText(TestName);
71.
72.         scantron.addFooterView(submit);
73.
74.         //Set the adapter and populate rows
75.         //Inputting this activity, layout, two string arrays.
76.         TestAdapter testadapter = new
    TestAdapter(this,R.layout.activity_test_layout,numbers,submitted);
77.         scantron.setAdapter(testadapter);
```

```
78.
79.         //sets up a countdown timer Needs alot of testing
80.         Log.i("Test Time" , Integer.toString(TestTime));
81.         Log.i("Test Time" , Integer.toString(TestTime*1000));
82.         if(TestTime <= 86400)
83.         {
84.             new CountDownTimer(TestTime * 1000, 1000)
85.             {
86.                 @Override
87.                 public void onTick(long millisUntilFinished)
88.                 {
89.                     if (millisUntilFinished % (60 * 1000) == 0) {
90.                         if (millisUntilFinished / (60 * 1000) < 5) {
91.                             header.setText(TestName + Long.toString(millisUntilFinished /
    (60 * 1000)) + " minutes Remaining");
92.                         }
93.                     }
94.                 }
95.
96.                 public void onFinish()
97.                 {
98.                     for (int i = 0; i < submitted.size(); i++) {
```

```
99.                 studentAnswers += submitted.get(i);
100.                }
101.                try {
102.                    Submission("OutOFTime", studentAnswers,
    SubmissionID);
103.                } catch (JSONException e) {
104.                    e.printStackTrace();
105.                }
106.            }
107.        }.start();
108.    }
109.
110.    //Sets he button to change the text on press and not holding down
111.    submit.setOnClickListener(new View.OnClickListener()
112.    {
113.        @Override
114.        public void onClick(View v)
115.        {
116.            Log.i("STUDENT ANSWERS", studentAnswers);
117.            if(submit.getText() == "Submit Test" )
118.            {
119.                canSumbit = true;
```

```
120.                        int i = 1;
121.                        for(String answer : submitted)
122.
123.                //Onlongclick - requires the button to be held down and it will submit
124.                submit.setOnLongClickListener(new View.OnLongClickListener()
125.                {
126.                    @Override
127.                    public boolean onLongClick(View v)
128.                    {
129.                        Log.i("SUBMIT", "Attempting to submit the test");
130.                        //submits the test
131.                        if(canSumbit)
132.                        {
133.                            Log.i("Student Answers", studentAnswers);
134.                            try
135.                            {
136.                                Submission("Student Submit", studentAnswers,
    SubmissionID);
137.
138.                //Back button function
139.                public void onBackPressed() {
140.                    //do nothing
```

```
141.                //disables back button
142.            }
143.
144.            public void onPause() {
145.                super.onPause();
146.                pausedTime = System.currentTimeMillis() + allowedTimeOutsideApp;
147.                Log.i("Time paused", String.valueOf(System.currentTimeMillis()));
148.                Log.i("Acceptable reentry time", String.valueOf(pausedTime));
149.                manager.removeView(view);
150.            }
151.
152.            // // TODO: Think of way to handle a spilt screen
153.            public void onResume() {
154.                super.onResume();
155.                //onResume function is called on load, which sets the counter to 2 so
    this is there one free exit from the app as long as its under 10 seconds
156.                onResumeCounter ++;
157.                if ((System.currentTimeMillis() > pausedTime) &&
    (onResumeCounter > 1)) {
158.                    exited = true;
159.                    for (int i = 0; i < submitted.size(); i++)
160.                    {
```

```
161.                     studentAnswers += submitted.get(i);
162.                 }
163.                 try {
164.                     Submission("Time Exited",studentAnswers,SubmissionID);
165.                 } catch (JSONException e) {
166.                     e.printStackTrace();
167.                 }
168.             }
169.             @Override
170.             protected void onPreExecute()
171.             {
172.                 super.onPreExecute();
173.             }
174.             //gets the result from the server, this will include result, number of
    TestViewActivity questions, TestViewActivity name, and time
175.             //result will be checked for errors each possible error will need to be
    checked
176.             //if result is good will put the rest of the information into the exsisting
    json object and continue
177.             protected void onPostExecute(JSONObject method)
178.             {
179.                 if (Responsecode == 200) {
```

```
180.                    String temp = "";
181.                    try {
182.                        temp = method.getString("Method");
183.                    } catch (JSONException e) {
184.                        e.printStackTrace();
185.                    }
186.
187.                    if (temp.equals("Student Submit")) {
188.                        Intent i = new Intent(TestViewActivity.this,
    ThankYouActivity.class);
189.                        i.putExtra("json", method.toString());
190.                        Log.i("TEST BEFORE START", method.toString());
191.                        startActivity(i);
192.                    } else {
193.                        Intent i = new Intent(TestViewActivity.this,
    BadSubmissionActivity.class);
194.                        i.putExtra("json", method.toString());
195.                        Log.i("TEST BEFORE START", method.toString());
196.                        startActivity(i);
197.                    }
198.                }else{
199.                    // no connection to the server
```

```
200.                    //Todo change to a different screen
201.                    Intent i = new Intent(TestViewActivity.this, LoginActivity.class);
202.                    startActivity(i);
203.                }
204.            }
205.            @Override
206.            protected void onProgressUpdate(Void... values)
207.            {
208.                super.onProgressUpdate(values);
209.            }
210.        }
211.
212.    }
```

5.3 Submission Controller Server Source Code

This code file is described as Apache Web Server in the previous chapter.

```
if ($test->simpleT == "no")//Not a simple test, randomize the ordering.
{
    Log::info("NOT A SIMPLE TEST. WILL SHUFFLE");
    array_multisort($OTF, $QTF, $ATF, $BTF, $CTF, $DTF, $ETF, $KTF);
    $string = join($KTF);
    $test->answer_key = $string;
```

```
}
else
{

    Log::info("SIMPLE TEST. NO SHUFFLE");
    //leave the order as it was.
}



$test->questionsss = json_encode($QTF);
$test->optionsA = json_encode($ATF);
$TimeRemaining = $datetime->diffInSeconds($end);

if ($test->answer_key == null) {
    $numQ = $test->questions()->count();
} else {
    Log::info("else");

    $numQ = strlen($test->answer_key);
}

Log::info($numQ);
```

```
        $answer_string = "";
        for ($i = 0; $i < $numQ; $i++) {
            $answer_string = $answer_string . '0';
        }

        if (!($datetime > $test->start && $datetime < $test->end)) {
            $res = "The test you are trying to access is not available to students at this
time.";
            return Response::json(['Reason' => $res], 403);
        }

        Log::info("TE
ST IS AVAILABLE");

        $submission = Submission::where([
            ['test_id', '=', $test->id],
            ['stu_id', '=', $st_id]
        ])->first();

        $subtestCode = Submission::where(['test_id' => $test->id])->get();
        $subStuId = Submission::where(['stu_id' => $st_id])->get();
```

```
        $vtest = Test::where('code', $testCode)->get();
        if ($submission != null) {
            Log::info("This user already submitted a test!");

            //if  0 >= 1
            if ($submission->submission_number >= $test->subs) {
                $res = "You have already taken this exam the maximum number of times";
                return Response::json(['Reason' => $res], 403);
            }

            $subID = $submission->id;
        } else {


            $subID = $sub->id;
        }

        $test->save();


        return Response::json([
```

```
            'TestName' => $test->name,
            'optionsD' => $optionsDF,
            'optionsE' => $optionsEF,
            'simpleT' => $test->simpleT
        ], 200);

    }
```

5.4 Test Controller Server Source Code

```
public function store(Request $request)
   {
       Log::info($request);

          $questions = Bank::find($request->banks)->questions()->with('options')-
>inRandomOrder()->take($request->questions)->get();
       }
       $gid = Test::all()->max('group');
       $gid++;
       for ($i = 0; $i < $request->tests; $i++)//for ($i = 0; $i < $request->tests; $i++)
        {
          $code = HomeController::randomString();
          $test = new Test();
```

```
$test->code = $code;
$test->name = $request->name;

if($request->simpleT == "")//box is unchecked. (real string value is "")
{
    $test->simpleT = "no";
}
else //box is checked. (real string value is "on")
{
    $test->simpleT = "yes";
}

//if the professor did not select a test bank to use, allow them to generate an answer key
if ($request->banks == null) {

    $key = "";
    for ($i = 0; $i < $request->questions; $i++) {
        $key = $key . "0";
    }

    $test->answer_key = $key;
```

```
            $test->save();
            return redirect()->action('TestController@edit', ['course' => $test->course_id, 'id' => $test->id ]);
        }

        //otherwise generate the test from the bank
            }
        }
        //if all students get the same test, only generate one
        if ($request->sameT == 'yes') {
            $i = $request->tests;
        }

        if ($request->simpleT == 'yes') {
            $test->simpleT = $request->simpleT;
        }
        if ($request->simpleT != 'yes') {
            $test->simpleT = $request->simpleT;
        }

    }
```

```
        $test = Test::where('group', $gid)->first();
        $bank = Bank::find($request->banks)->first()->name;

        return view('print_test', ['test' => $test, 'bank' => $bank, 'number' => $request->tests]);
    }
```

CHAPTER SIX:

TESTING AND DEMO

6.1 Student Demo

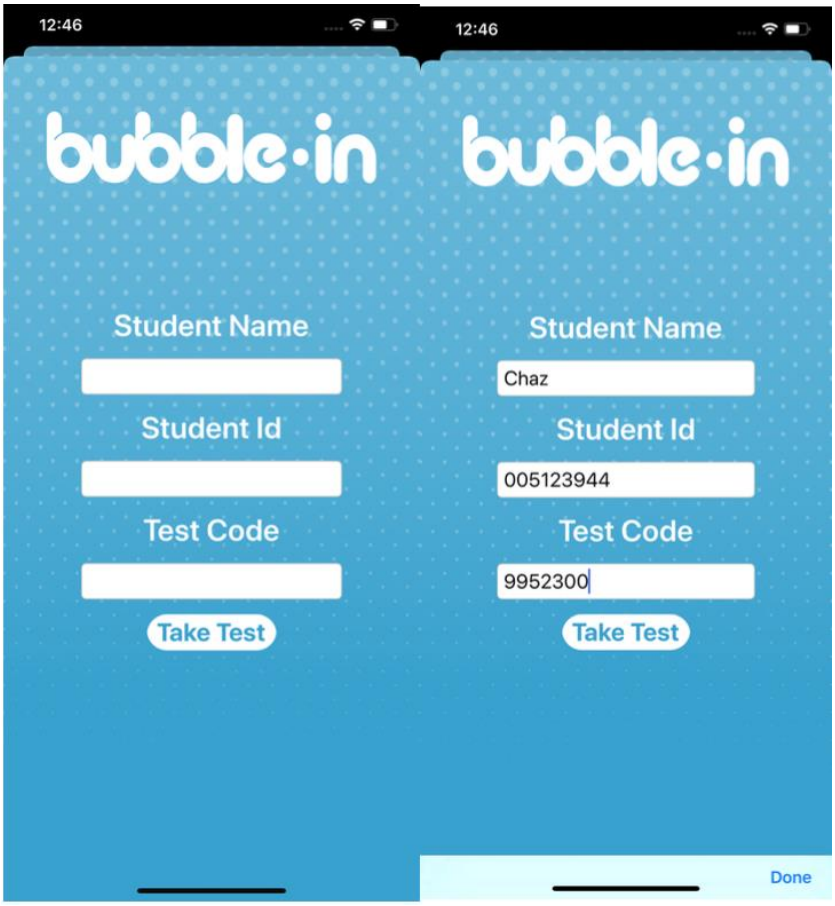

Fig. 9. Student Login Page.

Before student's test, the Bubble-In application is equipped with anti-cheating mechanisms such as question-answer key scrambling, not allowing screen shots, screen recording or leaving the application.

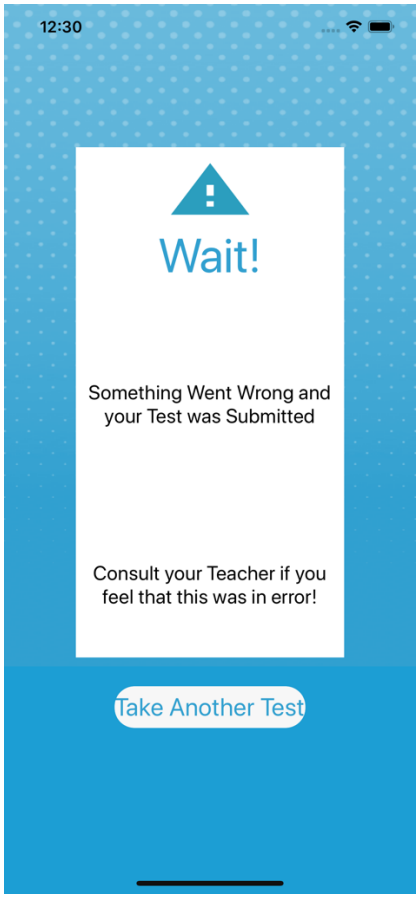

Fig. 10. Student Cheat Page.

This is the screen displayed on the application when the student leaves the application for any reason. If the application leaves the foreground, that event causing it to do so is logged and reported.

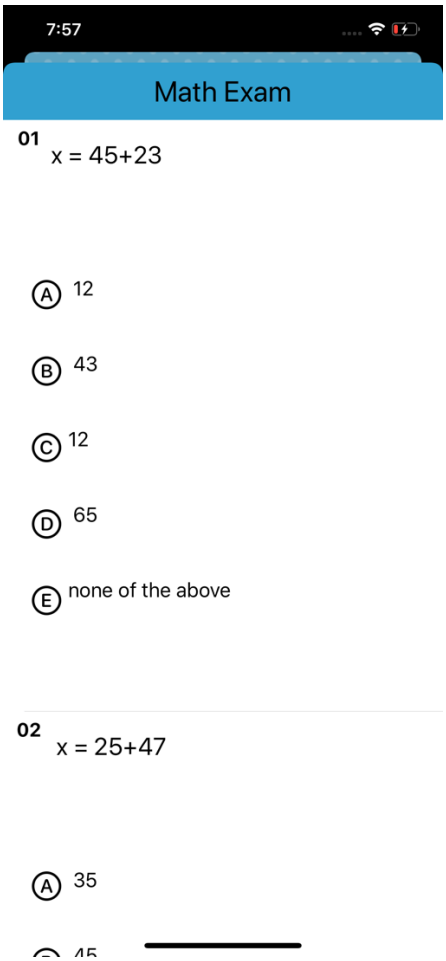

Fig. 11. Student Test Page.

6.2 Teacher Demo

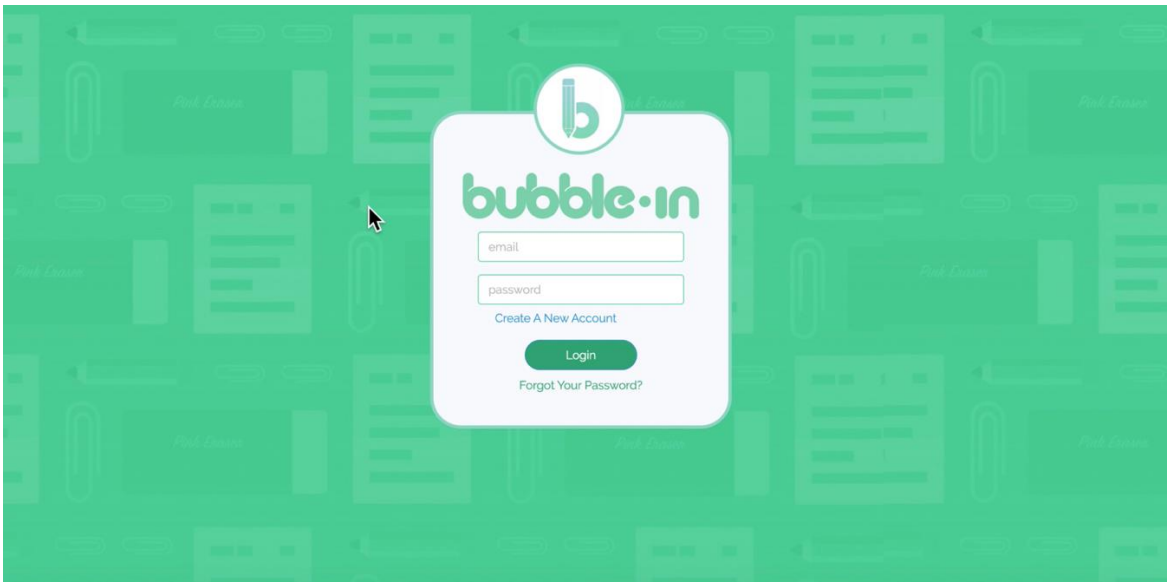

Fig. 12. Login Page.

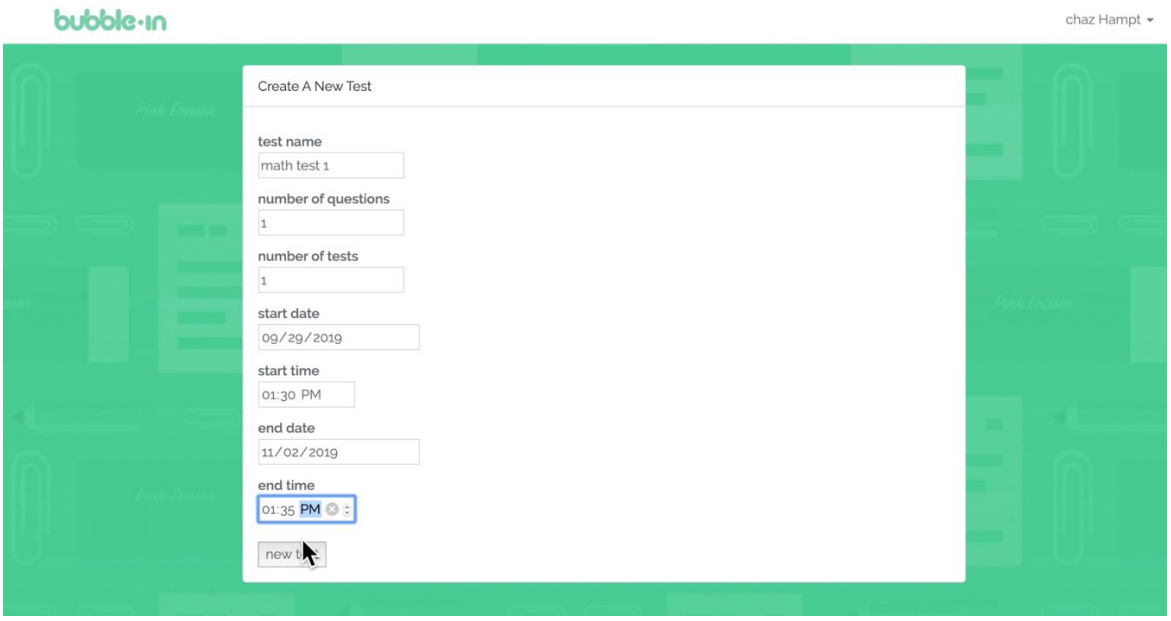

Fig. 13. Test Creation Page.

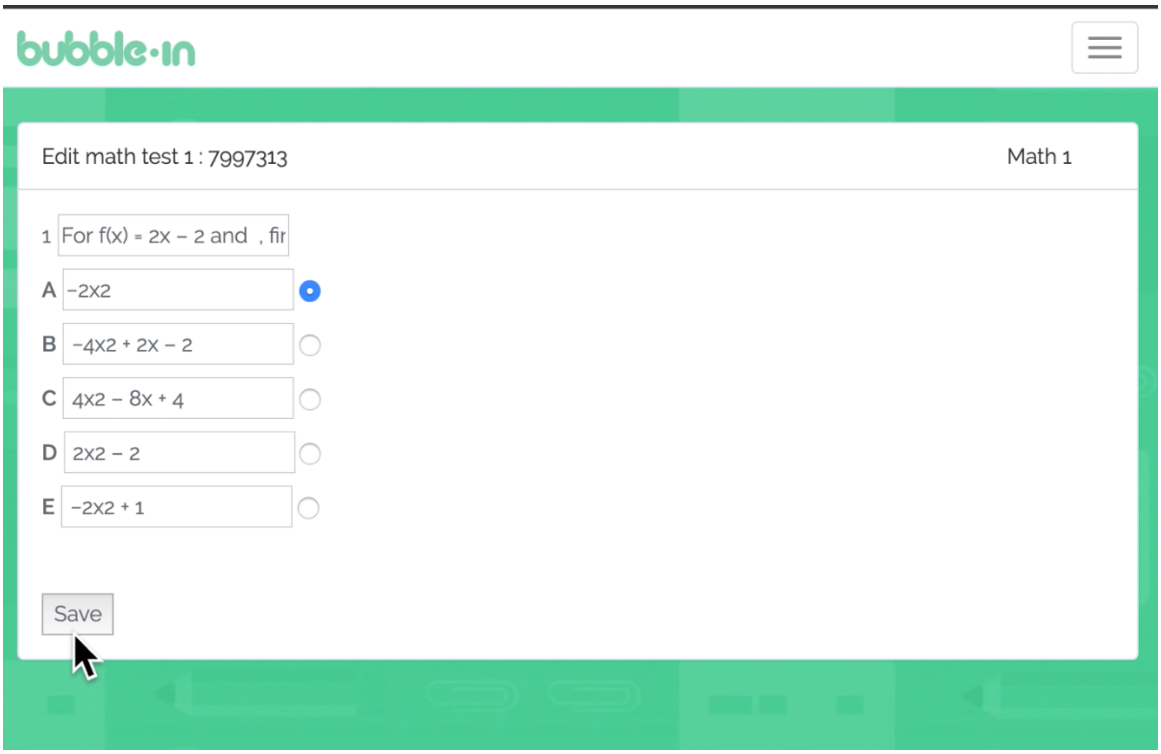

Fig. 14. Test Edit Page.

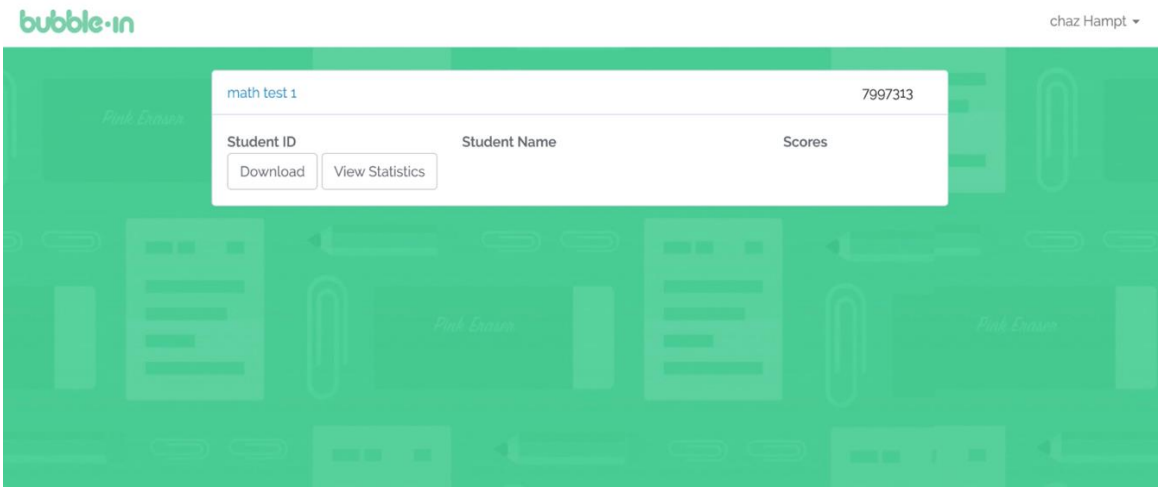

Fig. 15. Scores Page.

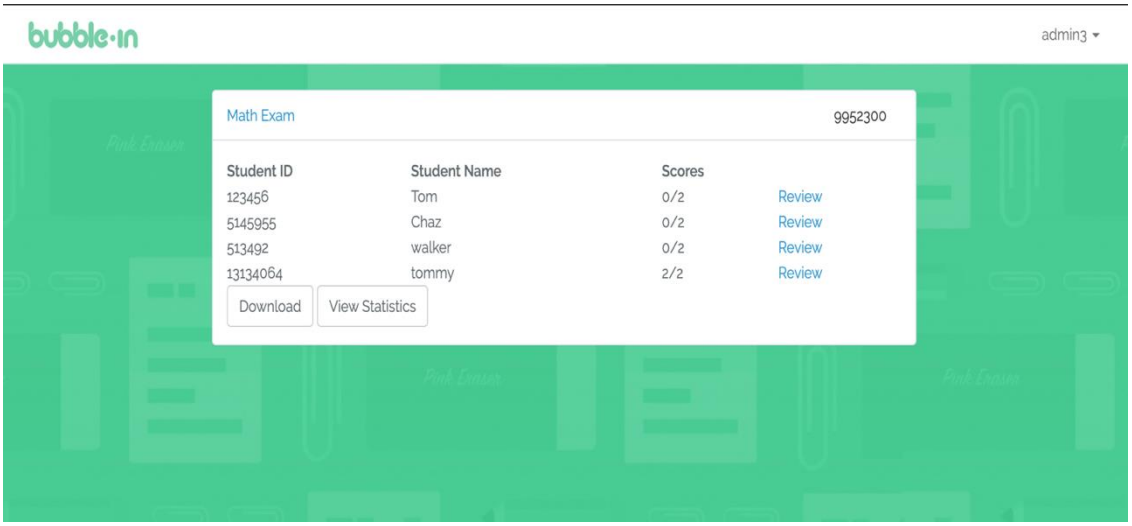

Fig. 16. Scores Page with Data.

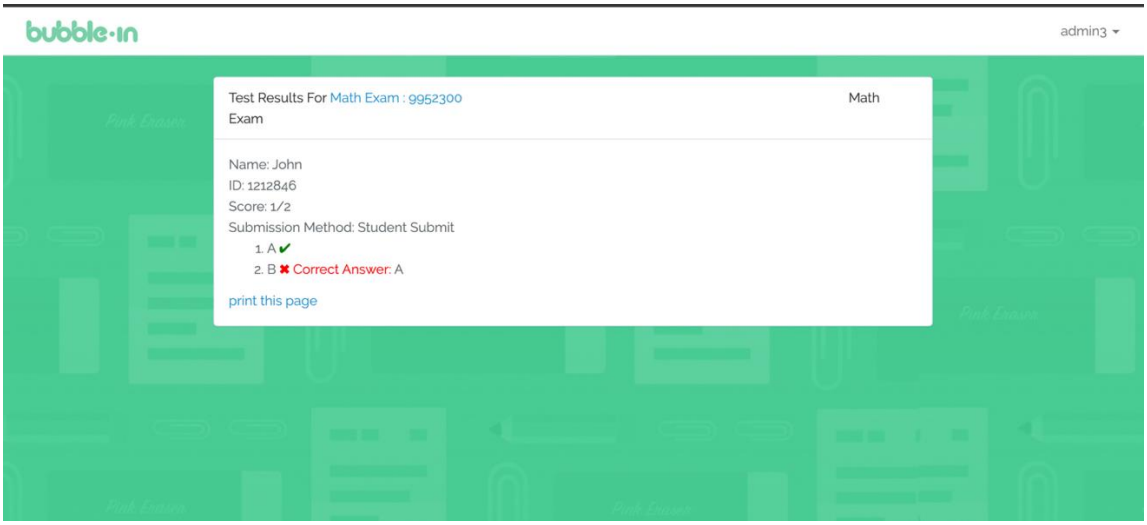

Fig. 17. Student Review Page.

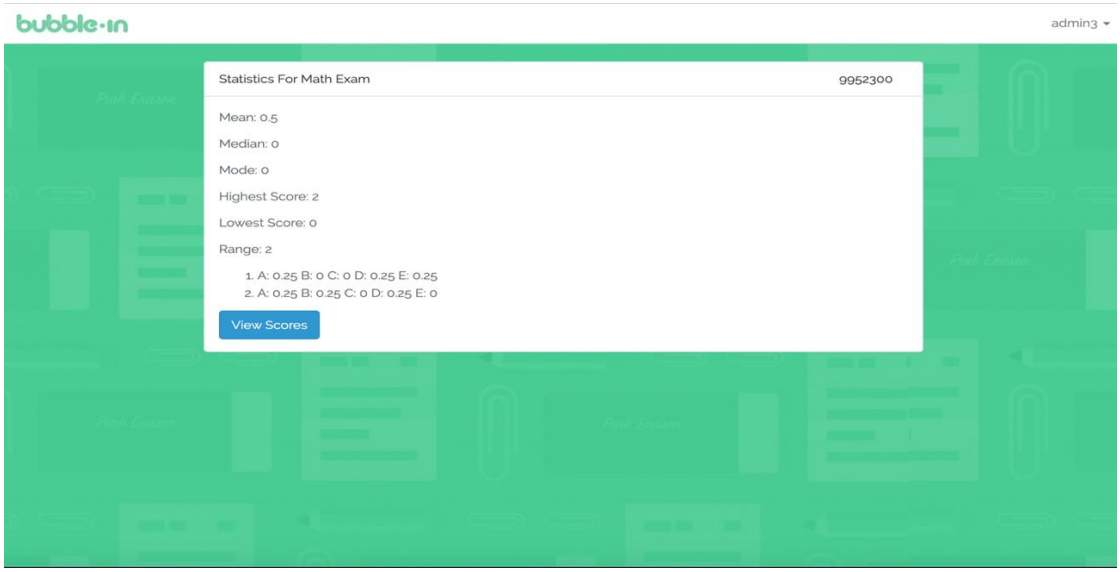

Fig. 18. Statistics Page.

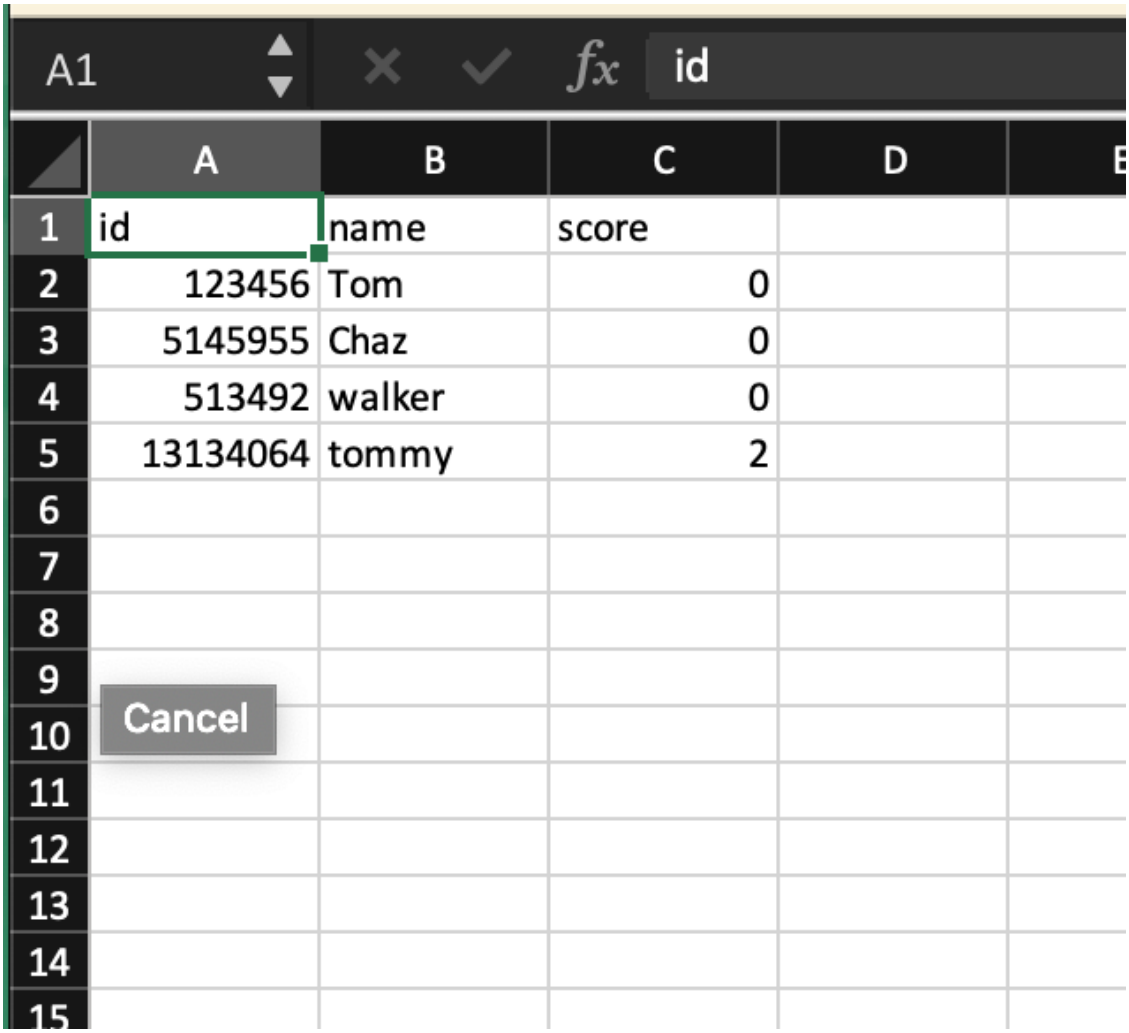

Fig. 19. Statistics .CSV Excel Format.

Here we can see the .csv file generated by the website. This is the results from a specific exam. Teachers can upload this file type directly to blackboard with confidence that student have not cheated due to various anti-cheating mechanisms in the applications in addition to question order scrambling and answer scrambling on top of the question scrambling.

CHAPTER SEVEN:

CONCLUSION

7.1 Next Steps

The next foreseeable steps for the Bubble-In system are adding and improving the anti-cheating mechanisms in the mobile applications of Android and Apple devices. Another feature that needs to be addresses is taking attendance. Making sure that a student is not taking an examination from home needs to be taken into account. The Bubble-In application is a competitor to the iClicker brand of products and should be updated with features to beat those offered by iClicker. The last feature which needs to be added is a new method for scrambling test questions and answers. Currently both are scrambled for each student when they are making the request to the server. A perhaps more efficient solution would be to scramble these ahead to time and store them on the server. This needs to be evaluated however since storing the exams on the server will take up more storage, which is cheap, but scrambling them on the fly uses more RAM, which is also currently at a low utilization due to the optimization of the Bubble-in system. These metrics can all be found on the Digital Ocean dashboard and should be logged and kept track of to compare both implementations.

7.2 What I Learned

In this project I have worked on cloud computing servers where I have implemented multiple instances of the Bubble In system. Cloud computing is a catalyst of many new systems and technologies so by understanding its intricacies I am well informed and educated on the topic for job opportunities. By not only being familiar with cloud computing but having implemented a running web server I have no doubt this or any other future products I work on can be made more effective with this gained knowledge.

In addition to cloud computing technologies, this project gives me in depth and hands on experience with mobile application development using native code. By implementing the Bubble In application on Android and IOS I am well versed in both Java and Swift as well as the SDK. Further, when developing the cloud server, I became experienced with PHP in the Laravel framework. I also had the opportunity to work in JavaScript and HTML hyper script. I became much more proficient with MySQL when developing the database which involved the RESTful protocols when communicating with mobile devices running the Bubble-In application including GET, POST, PUT. Simplistically, POST means create, GET means read, PUT means update, and DELETE means delete [2]. The web server Bubble-In and the web server provider, Digital Ocean, operate with the Service-Oriented Software Engineering (SOSE) paradigm of building software today. This paradigm is the latest development in building software with a service-oriented approach.

7.3 Accomplishments

I would like to conclude my project proposal by expressing my gratitude for this opportunity and challenge. For a short background on myself, I transferred to CSUSB from Victor Valley Community College, completed a bachelor's degree and I couldn't enjoy the university more. The Campus is beautiful and loaded with every resource you would need to be successful. The help and guidance I have received from faculty and instructors during my degree has been instrumental.

This masters project directly assisted me in my long term educational and career goals. I hope to attain my master's degree in Computer Science and eventually move into industry involved with software development. Once a master's degree is achieved, I also have the option to pursue my doctorate degree or begin teaching at the college level immediately with an M.S. which I would also enjoy. I have a special interest in mobile app development and find the creative process very fulfilling where I truly enjoy working many hours on my projects.

By completing this project, I have advanced towards my career goals. First, my knowledge of the subject has strengthened and my professional marketability and earning potential has increased. I have also obtained a strong foundation for graduate study. My long-term career goals include finishing my 4-year commitment in the United States Air Force serving as a Cyber Security

Officer. I hope to move into industry after my service in the Air Force which I believe this project has helped me in immensely.

Putting in this extra time is necessary to stay knowledgeable and ensure you are solving problems in the most optimal way. I have a special interest in helping and teaching my peers and in mobile application development. I find that the more useful my work is to someone the happier I am with the time I have invested, and the products created. I find the creative process very fulfilling and I truly enjoy working many hours on my projects. As the late Apple CEO Steve Jobs once said, "Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do." These few sentences perfectly summarize my desire and drive in my development and education.

I understand that not everyone is fortunate enough or always in a position to complete a master's degree so with this in mind, I have made a conscious effort to study diligently, work smarter, put in extra time and to encourage others to do the same.

Very Respectfully,

Chaz T. Hampton

REFERENCES

[1] AlShahwan, F., Faisal, M. & Ansa, G. J Ambient Intell Human Comput (2016)

[2] “Chapter 19 Service-Oriented Architecture.” *Software Engineering*, by Ian Sommerville, Pearson Education Asia Limited, 2014, pp. 511–513.

[3] Garrison, G., Kim, S., Wakefield, R.L.: Success Factors for Deploying Cloud Computing. Commun. ACM. 55, 62–68 (2012).

[4] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing — The Business Perspective. Decis. Support Syst. 51, 176–189 (2011).

[5] "Panasonic KV-S4065CL, Panasonic KV-S4065CL Scanner, New (KV-S4065CL)". Esupplybox.Com, 2020, https://www.esupplybox.com/index.php/panasonic-kv-s4065cl-new-scanner-kv-s4065cl-e014-202g001.html?gclid=CjwKCAiAhc7yBRAdEiwAplGxX62ey4LzZ3L0oNkAp6w8Rz6Ng_xvBy2iZxEJPz2_zcwvcm4YVbXdjBoCBmEQAvD_BwE. Accessed 24 Feb 2020.

[6] Sebastian Graf & Vyacheslav Zholudev. Managing Authorization with RESTful XML (2017)

[7] "Product". Insight, 2020, https://www.insight.com/en_US/shop/product/1615962/Kodak%20Alaris/1615962/Kodaki5850-documentscanner/?platform=google-

shopping&partner=smb&campaign=smb&sku=1615962&gclid=CjwKCAiAhc7yBRAdEiwAplGxXwZUhBWlJ4y-beXXlF4w2b87tuM7K-oEbkUmjf0M8BPQtG7uke8OwRoCHrYQAvD_BwE. Accessed 24 Feb 2020.

[8] Srivastava, Priyanshu & Khan, Rizwan. A Review Paper on Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering. 8. 17. 10.23956/ijarcsse.v8i6.711 (2018)

[9] Venters, W., Whitley, E.A.: A Critical Review of Cloud Computing: Researching Desires and Realities. J. Inf. Technol. 27, 179–197 (2012).

[10] Yang, H., Tate, M.: A Descriptive Literature Review and Classification of Cloud Computing Research. Commun. Assoc. Inf. Syst. 31 (2012).

[11] Shih-Hao Hung and Chi-Sheng Shih, Executing mobile applications on the cloud: Framework and issues. National Taiwan University, Taipei 106, Taiwan (2011)