SAFE: Smart Authenticated Fast Exams

M. Tech Project Stage I Report

Submitted in partial fulfillment of the requirements for the degree of
Master of Technology

by

Ayush Singh
163050014

under the guidance of

Prof. Kameswari Chebrolu

Department of
Computer Science & Engineering
Indian Institute of Technology, Bombay
Mumbai 400 076
October, 2017
I extend my sincere gratitude to Prof. Kameswari Chebrolu and Prof. Bhaskar Raman for their guidance and supervision. I would like to thank Vicky Gupta who worked with me in this project and helped me with server related issues. I would also like to thank the department and the office staff for their help. Last but not least, I would like to thank all my friends and family for their moral support.
Abstract

SAFE: Smart Authenticated Fast Exams[1] is a tool that helps in continuous assessment in the form of regular quizzes in classes. SAFE is based on a BYOD (bring your own device) model that uses student smart-phones to conduct auto-graded, cheating-free exams in a classroom. It has 3 components: a smart-phone app, a web server and WiFi infrastructure to enable app-server communication.

There are 2 versions of SAFE. Both are under continuous development or bug fixes. Version 1 is stable version. It uses PHP as a back-end for server. It is currently being used in Computer Science & Engineering Department of IIT Bombay for various courses and in some other colleges too.

Version 2 of SAFE is complete remake of its older version, consisting all the feature from version 1 along with some new features. Version 2 is build upon python based Django framework as back-end. This version is still under development and rigorous testing mode to make it stable and bring into usage.

Aim of this project was to develop new features in version 2 of the SAFE android app. The features that came forward as requirement after using SAFE version 1 of the app.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td></td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>v</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Reducing TCP Connections</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Problem Statement</td>
<td>2</td>
</tr>
<tr>
<td>2.2</td>
<td>Why do we need this?</td>
<td>3</td>
</tr>
<tr>
<td>2.3</td>
<td>Implementation</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Quiz Navigation Screen</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>Problem Statement</td>
<td>4</td>
</tr>
<tr>
<td>3.2</td>
<td>Why do we need this?</td>
<td>4</td>
</tr>
<tr>
<td>3.3</td>
<td>Design</td>
<td>4</td>
</tr>
<tr>
<td>3.4</td>
<td>Implementation</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Quiz Download</td>
<td>6</td>
</tr>
<tr>
<td>4.1</td>
<td>Problem Statement</td>
<td>6</td>
</tr>
<tr>
<td>4.2</td>
<td>Why do we need this?</td>
<td>6</td>
</tr>
<tr>
<td>4.3</td>
<td>Design</td>
<td>6</td>
</tr>
<tr>
<td>4.4</td>
<td>Implementation</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Image Pass-code</td>
<td>8</td>
</tr>
<tr>
<td>5.1</td>
<td>Problem Statement</td>
<td>8</td>
</tr>
<tr>
<td>5.2</td>
<td>Why do we need this?</td>
<td>8</td>
</tr>
<tr>
<td>5.3</td>
<td>Design</td>
<td>8</td>
</tr>
<tr>
<td>5.4</td>
<td>Implementation</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Change Account Password</td>
<td>10</td>
</tr>
<tr>
<td>6.1</td>
<td>Problem Statement</td>
<td>10</td>
</tr>
<tr>
<td>6.2</td>
<td>Why do we need this?</td>
<td>10</td>
</tr>
<tr>
<td>6.3</td>
<td>Design</td>
<td>10</td>
</tr>
<tr>
<td>6.4</td>
<td>Implementation</td>
<td>10</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Wireshark Trace before ................................................. 2
2.2 Wireshark Trace after .................................................. 3
3.1 Quiz Navigation Activity .............................................. 5
5.1 PassCode Activity ..................................................... 9
7.1 Previous Submission Screens ........................................ 12
8.1 JMeter test plan for video page ................................. 14
8.2 Request Made to the server ...................................... 15
8.3 Response from server ................................................ 16
1. Introduction

SAFE is used to conduct regular quizzes for courses in a classroom which is invigilated by the instructor. SAFE's android app is the front end for the students through which they can interact and give exams on their own android devices. Version 1 of the SAFE android app is currently being used as it is stable.

After the assuring feedback from V1, new version was planned to for better scalability and easy to use. So V2 of the app was also developed along with its back-end having all the features of V1 and some new essential features. Still there are many requirements left on both server-side and app-side which is to be developed. In this project some of the major feature or request has been implemented/developed on the android side of the SAFE V2.

Some of the important work done in V2 of SAFE App:

- Reduced the number of connections made to server from a single client.
- Implemented feature to download the quiz.
- Implemented image pass-code for Quiz validation.
- New quiz navigation screen.
- Password change feature for user.
- Group the previous submissions by courses.

In the second part we have also done server setup and designed some testing plan for scalability testing for the Bodhitree & SAFE server.
2. Reducing TCP Connections

2.1 Problem Statement

The access points which were being created for conducting the SAFE quizzes were causing bottleneck resulting in latency and ‘No Network Connection’ issues in app.

We used Wireshark[2] to observe the traffic on access point. Wireshark is free and open source software to analyze any type of packet which is passing through an interface. It can be used for network troubleshooting and analysis of network interface. It lets the user see all the traffic visible on a particular interface.

Below is a wireshark trace recorded during execution of app[Fig 2.1]. The packets with green color represent HTTP request/response from device/server. Packets with grey color represent TCP SYN/ACK packets. It was observed through Wireshark trace that for every HTTP request which was made through app, there was new TCP connection being made to the server.

![Wireshark Trace before](image-url)

Figure 2.1: Wireshark Trace before
2.2 Why do we need this?

Since every time a new connection is made for each http request, resulting in each user to have many connections thus making network resources scarce. So after some time when new user is trying to make connection or a new request is made to server, no connection error is showed as access point may have reached its limit.

2.3 Implementation

In android OKHttp[3] library is being used for all its network calls. So for making HTTP request and reading their response an instance of OKHttpClient is required. It was observed that for every request a new instance was being created resulting in new TCP connection. So changes were made such that OKHttpClient is instantiated in singleton pattern. In singleton design pattern it is ensured that only one instance of class exist which is used whenever required in entire program. Thus resulting in single connection being made for any number of request from a single user[Fig 2.2].

![Wireshark Trace after](image)

As it can be observed from new trace, now for multiple request made from same device only 1 TCP connection is made as there is only one SYN/ACK cycle and that same connection is reused for all the new HTTP request and response.
3. Quiz Navigation Screen

3.1 Problem Statement
Implement a quiz navigation screen on android side.

3.2 Why do we need this?
Navigating to specific question during a quiz quickly and easily, a user friendly interface must be provided for student who are using the app. Appropriate information must be provided related to each question so user can easily jump to particular question which he want to solve.

3.3 Design
In this screen all the question related to particular quiz will be listed. Each question is assigned a color code according to whether it has been attempted, marked for review or not attempted yet. Marks for each question will also be displayed. For each question some part of the question can be viewed along with its tag which is used to easily identify questions to which topic it belong.

3.4 Implementation
In base class of question a new data member is added ”isReview” to check whether the question has been marked for review. So according to whether question is marked for review or it is attempted by user blue or green color is assigned to question respectively, mark for review being given higher priority. If question is large then only first 30 characters of question’s description is displayed.
Figure 3.1: Quiz Navigation Activity

Question tag will also be displayed. Through question tag user can easily identify the question and get to know what it is related to. This is yet to be implemented from server side but has been implemented from android side.
4. Quiz Download

4.1 Problem Statement
Implement an option to download the quiz before coming for quiz in class.

4.2 Why do we need this?
During a quiz when many user try to download the quiz simultaneously it may create load on server and network interface. So allowing the user to download the quiz beforehand will reduce the load on the server and access point during the quiz. It will also reduce the straddle duration for user, allowing them to start quiz quickly.

4.3 Design
Users can download the quiz with its unique Quiz ID which they can get from instructor. The downloaded quiz will be encrypted compressed file which will contain all the data related to that quiz. It can only be decrypted by entering pass-code. The pass-code for that quiz will be disclosed by instructor during scheduled quiz time.

4.4 Implementation
Since OKHttp library has been used in the app for all network related calls & request. Download of quiz is also implemented using that to maintain uniformity and reduce the use of extra library for download. Below is code snippet that is used to create a connection to server for downloading the file.

```java
Response response = ServerInterface.getHttpResponse(url,
                                                    null,
                                                    null);
ResponseBody body = response.body();
long quizSize = body.contentLength();
```
OKio library is also used with Okhttp for easy download of file. OKio\[4\] is library which complements java.io library which makes easier to access, store or process data. It has its own stream type ‘source’ which work as InputStream and ‘sink’ which is OutputStream. So after opening connection we created a stream to read the file save it locally in android device. Below is code snippet that shows how the file being read from connection and saved in a file using OKio library easily.

```java
quizZip = new File(getApplicationContext()
    .getFilesDir(),
    quizId + "\.zip");
BufferedSource source = body.source();
BufferedSink sink = Okio.buffer(Okio.sink(quizZip));
Buffer sinkBuffer = sink.buffer();
long totalBytesRead = 0;
int bufferSize = 8 * 1024;

for (long bytesRead;
    (bytesRead = source.read(sinkBuffer, bufferSize)) != -1;
    )
    {
    sink.emit();
    totalBytesRead += bytesRead;
    int progress = (int)((totalBytesRead * 100) / quizSize);
    publishProgress(progress);
    }
```

OKio is component of OKHttp, so it is easy and efficient to use both of them together in android app. While downloading Quiz it has been taken into care that sufficient memory is present in device to download and extract the files afterwards. If there is not enough memory in device it is handled and proper message will be displayed to user. If some error occurs during downloading of files the partially downloaded file is deleted from the server.
5. Image Pass-code

5.1 Problem Statement
Implement image pass-code to validate users for quiz

5.2 Why do we need this?
Allowing the users to give quiz based solely on Quiz ID is not secured as student inside the class room may communicate it to their friend who are not in class. So those student may also be able to give the quiz. By implementing image pass-code it will be difficult for students to convey the code to others.

5.3 Design
Before every quiz begins the instructor will display the pass-code on screen in class. Students have to enter that pass-code to get a validation from server that he is eligible to take that particular quiz. If student enters the wrong code they will not be given access to give quiz. So event if Quiz ID is known they can’t take that exam.

5.4 Implementation
User will be given a set of images. They have to enter pass-code i.e. correct subset of images in correct order to be get access to the quiz. Pass-code is set by instructor for each quiz from server side and will be displayed to student during the exam. The pass-code then enter will be sent to server for validation. Once the sever gives the validation to user, he can begin the test.
Every image has certain byte pattern associated with it which is not known to user. So on clicking images in order specific byte pattern is created, which then is validated with byte pattern from server.
6. Change Account Password

6.1 Problem Statement

Provide facility for user to change password for their account.

6.2 Why do we need this?

When bulk registration of new student is done for a course initially a random password is set for their account and they are sent in their respective mails. So to allow the students to change the password once the account has been created.

It may also happen that student forgets the password. So he can request the respective instructor/TA to reset the password and after resetting they can set their new password.

6.3 Design

Once the user logs in with his current credentials and the server has authenticated him, he then can change his current password for some new password.

6.4 Implementation

While entering the new password some local validation is done from android side also viz. user have to enter new password two times and they have to be same. The password field and password confirmation field cannot be left blank. The respective error message is displayed if validation is not successful.

Then the new password is sent to the server and server side validation is also done. After validation the success/error message is sent to device. If the password is validated a success message is displayed to user. If there were some validation error they are displayed the error and asked to reenter the new password.
7. Course-wise Previous Submission

7.1 Problem Statement

Display the previously submitted quizzes course-wise.

7.2 Why do we need this?

As the user keep giving quizzes, his previous submission will keep on getting increasing. So on displaying previous submission the list will become very long and user will not be able to easily find the quiz submission which he will be looking for.

7.3 Implementation

To display list of course an API is called from server which returns the list of course to which student is enrolled. The list of all this course is displayed in screen. Then user can select the course of which he want to see his previous submission.

To display the list of courses in android activity there are two options either to use 'List View' or 'Recycler View'. We have used 'Recycler View' as it has many advantages over 'List View' when handling dynamic data i.e we don’t know how many item will be present in the list. Recycler view helps in saving lot of memory as it reuses the old views to display the new data this helps in smooth running of android app.
<table>
<thead>
<tr>
<th>Enrolled Courses</th>
<th>Submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Graphics Basics</td>
<td>BQMG1076</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>T47FLCMT</td>
</tr>
<tr>
<td>Foundation only</td>
<td>12-6-2017</td>
</tr>
<tr>
<td>Introduction to Computer</td>
<td>Binary Search Tree</td>
</tr>
<tr>
<td>Science Basic</td>
<td>Mark: 2.0 (2.0)</td>
</tr>
<tr>
<td></td>
<td>XIRA40M1</td>
</tr>
<tr>
<td></td>
<td>12-6-2017</td>
</tr>
<tr>
<td></td>
<td>Binary Search Tree</td>
</tr>
<tr>
<td></td>
<td>Mark: 2.0 (2.0)</td>
</tr>
<tr>
<td></td>
<td>YTCZBS5M</td>
</tr>
<tr>
<td></td>
<td>12-6-2017</td>
</tr>
<tr>
<td></td>
<td>Binary Search Tree</td>
</tr>
<tr>
<td></td>
<td>Mark: 0.0 (2.0)</td>
</tr>
<tr>
<td></td>
<td>1AN4WMKC</td>
</tr>
<tr>
<td></td>
<td>12-9-2017</td>
</tr>
<tr>
<td></td>
<td>Binary Search Tree</td>
</tr>
<tr>
<td></td>
<td>Mark: NA (2.0)</td>
</tr>
<tr>
<td></td>
<td>BH56VWC8</td>
</tr>
<tr>
<td></td>
<td>12-6-2017</td>
</tr>
<tr>
<td></td>
<td>Binary Search Tree</td>
</tr>
<tr>
<td></td>
<td>Mark: NA (2.0)</td>
</tr>
<tr>
<td></td>
<td>6F8YP5E8</td>
</tr>
<tr>
<td></td>
<td>12-6-2017</td>
</tr>
<tr>
<td></td>
<td>Binary Search Tree</td>
</tr>
</tbody>
</table>

(a) Course List          (b) Submissions of a Course

Figure 7.1: Previous Submission Screens
8. Scalability Testing

Scalability testing is like performance testing of an application that is focused on how application scales as it is deployed on a larger system or as more load is applied to the current system. In this project, we did initial setup, created a basic plan, and did some initial testing for SAFE and Bodhitree.

Bodhitree is an e-learning platform developed in the Computer Science & Engineering Department of IIT Bombay. SAFE is a tool that is used to conduct regular quizzes in classes. They both are currently being used in many courses in IIT Bombay. The backend design for both is relatively the same with some slight variations. They both use Django, a Python-based framework, as their server. The frontend for both is different, i.e., HTML, CSS, JQuery, etc., for Bodhitree and Android for SAFE.

8.1 Server Setup

For testing purposes, we setup an instance of Bodhitree and Safe on different servers rather than using the original server on which courses were running. The testing server also had the same specification as that of the original, so there was no difference in performance, and we can properly simulate the original server. A proper update of the testing server and upgrade of software was also done so that no compatibility issues should appear during setup.

Nginx

Nginx is a reverse proxy server. All the request comes first to Nginx; it sits behind the firewall in the server. According to the request which have come from the client and its URL, Nginx directs the request to the respective backend server. It serves static files directly from the cache. All these settings can be configured in the nginx.conf file.

We configured the nginx configuration files for both the backend so that incoming request can be handled respectively. There is a common configuration file for all request and for each backend server, we can keep different nginx.conf files to perform differently.

Nginx also keeps the log of all the request that comes to the server. This logging feature doesn’t affect the performance in responding to request as it is done after the request has been served. We configured the nginx logging to our requirements, so that all the essential attributes would be logged, which will be required for analysis.
8.2 Designing Test Plan

JMeter

JMeter is a open source java application which is used for testing and performance measurement. It is designed to test web application. It has ability to extract data from responses format like HTML, JSON, XML or any textual data. It can be used to simulate how through a browser, a typical user explore the web pages.

We wanted to load test important features of Bodhitree and SAFE for some urls. JMeter can send request to the server at specified rate, it can send multiple request also simulating multiple parallel users. It then gets the response from server which can be used to calculate various server parameters such as throughput and CPU utilization. Below is an example of JMeter test plan.

![Figure 8.1: JMeter test plan for video page](image)
BlazeMeter

BlazeMeter is a tool that can be integrated with a browser to create a test plan for JMeter. This tool enables us to easily record the test plan by simply visiting the pages as a normal user would do with a browser. The recorded test can then be exported as a 'jmx' file which is a JMeter readable file.

So we used this tool for creating a basic test plan like visiting a series of web pages. This has an advantage over creating a test plan from scratch in JMeter. Since there are many extra components to a web page than just HTML which are also being called from server like javascript, JSON etc. These all are recorded with BlazeMeter so a correct test plan is generated.

After creating a basic test plan, we can export it to JMeter and tweak it to our needs. Through JMeter we can manage cookies and session, but we have to manually configure it. We can then set the ramp up period for the test. The ramp up period defines how much time it should take to reach the full number of threads. Example, if we are simulating 100 users and ramp up time is 50 seconds, then it would take 50 seconds to get all 100 threads up and running.

8.3 Running Test

To test the server, we created a simple test plan for Bodhitree. We wanted to check the current throughput of the server when a video is being played.

So test plan consists of navigating to a video page in a course in Bodhitree and playing the video. Total duration of the test was 140 seconds including the video's length. Below are the graphs which were obtained from the test plan.

![Figure 8.2: Request Made to the server](image)
Fig 8.2 represents the number of request being made from user and Fig 8.3 represents the corresponding response that is being given by the server. We can see that initially the request/response is less and later it increase, it is due to the fact that video was streaming and it require lot more components from server.

In response graph we can see that some error has also occurred, blue and magenta lines in graph. The magenta line represent 403:Forbidden error which means we are not authorized to access that component. It may have occur due to the improper session management done during making of the test plan in JMeter. Blue line is for Non HTTP response, it depends on what error may have occurred while processing user’s request.
Bibliography


