Question Paper Generator System

Prepared by

Refik Erkan Günhan

B1405.010023

June 2019

ABSTRACT

In today’s World people become more and more used to using web and websites. As we all know that time is a very significant thing for us humans. However so many people don’t do any changes in their lifestyle to use that time more efficently. To achieve that we must first decrease the time that we put in our jobs or work.

So the aim of my project is to reduce the time of the teacher’s work that they put in to prepare exams like quizzes, midterms, finals etc. I also aim to calculate the mean of the grades using my project and reduce teh work of the teachers even more. The project also reduces the work of the students because students will just take the exam at the scheduled hour and they will see their grades instantly. Also, it’s hard for them to cheat because the system asks students random questions from a question pool that the teacher prepared and the time that teacher prepared is limited.

To make a project happen I used a lot of diffrent tools like Eclipse Jee Oxygen, Tomcat, Sql, CSS, Ajax etc. In the Eclipse part I wrote my codes in a organized way to keep my work steady. Then I connected the project with the Sql host to store the databases, logins etc. in the servers.On Tomcat part I used the virtual servers to back up my project. On CSS part I designed the projects layout in a smooth way to make it more appealing to people. I also used Ajax to settle the xml part of my project.

At last, I connected all the pieces together in a systematic way and tested the project several amount of times to see that it is working properly. I hope to help the teachers and students to reduce their work time by the help of this project.

Contents of Project

1 Chapter 1: Introduction .........................................................................................................8

* 1. Background ……………………………………………………………………………………………………………….. 8
  2. Project Motivation ……………………………………………………………………………………………………. 8
  3. Problem Statement …………………………………………………………………………………………………… 8
  4. Project Question ………………………………………………………………………………………………………… 9
  5. Project Objectives ……………………………………………………………………………………………………… 9
  6. Project Significance ………………………………………………………………………………………………….... 9
  7. Project Scope …………………………………………………………………………………………………………… 10
  8. First Steps ………………………………………………………………………………………………………….………10

1. Chapter 2:Literatur Review …………………………………………………………………………………………. 12
   1. Question Paper Generator System……………………………………………………………………………. 12
   2. Definition Of Question Paper Generator System ………………………………………………………. 12
   3. Using Sql …………………………………………………………………………………………………………………… 12
   4. Why Sql ……………………………………………………………………………………………………………………. 14
   5. Tomcat ……………………………………………………………………………………………………………………. 15
   6. Relate Work ……………………………………………………………………………………………………………… 15
   7. Automatic Paper Generator System …………………………………………………………………………. 15
   8. Digital Teacher Question Paper Generator ………………………………………………………………. 16
2. Chapter 3:Project Methodology ………………………………………………………………………………….. 19
   1. Adopted Methodology …………………………………………………………………………………………….. 19
   2. Waterfall Methodology ……………………………………………………………………………………………. 20
   3. Why Adopting Waterfall Is Suitable For Question Paper Generator System …………….. 20
   4. Implementation Of Waterfall …………………………………………………………………………………… 20
   5. What I Want From The Project ………………………………………………………………………………… 22
   6. MVC …………….…………………………………………………………………………………………………………… 22

4 Chapter 4:Analysis And Design ………………………………………………………………………………………. 24

4.1 Project Description ……………………………………………………………………………………………………… 24

4.2 User Description ………………………………………………………………………………………………………. 24

4.3 Requirements Development ………………………………………………………………………………………. 24

4.4 System Requirements ………………………………………………………………………………………………… 25

4.5 Functional Requirements …………………………………………………………………………………………… 25

4.6 User Requirements ……………………………………………………………………………………………………. 25

4.7 Non-Functional Requirements …………………………………………………………………………………... 25

4.8 System Analysis …………………………………………………………………………………………………………. 27

4.9 Use-case Diagram …………………………………………………………………………………………………...... 27

4.10 Block Diagram ……………………………………………………………………………………………………….… 28

5 Chapter 5:Implementation ……………………………………………………………………………………………. 30

5.1 Design Phase ………………………………………………………………………………………………………..……. 30

5.2 Building The Back-End …………………………………………………………………………………………..…… 30

5.3 Building The Front-End …………………………………………………………………………………………..….. 30

5.4 Implementation Phase ……………………………………………………………………………………..……… 30

5.5 Design Of The Project ………………………………………………………………………………………………. 31

6 Chapter 6:Testing And Tools ………………………………………………………………………………………….. 42

6.1 Testing ……………………………………………………………………………………………………………………… 42

6.2 Performance Testing ………………………………………………………………………………………………… 42

6.3 Functionality Testing ………………………………………………………………………………………………… 43

6.4 Tools And Techniques …………………………………………………………………………………………………. 44

6.5 Maintance ………………………………………………………………………………………………………………… 45

6.6 Conclusion ………………………………………………………………………………………………………………… 45

References ………………………………………………………………………………………………………………………… 46

Additional Informatıon(Codes) ………………………………………………………………………………………… 48

List of Tables

2 Table Similar Project Comparisson …………………………………………………………………………………. 17

6 Table Tools And Techniques …………………………………………………………………………………………… 44

List of Figures

2 Figure Database Diagram ……………………………………………………………………………………………… 13

2 Figure Layout Of Automatic Paper Generator System …………………………………………………… 15

2 Figure Layout Of Digital Teacher Question Paper Generator …………………………………….…… 16

3 Figure Waterfall Methodology ……………………………………………………………………………………….. 19

3 Figure Waterfall Steps ……………………………………………………………………………………………………. 21

3 Figure MVC ……………………………………………………………………………………………………………………. 22

4 Figure Non-Functional Requirements …………………………………………………………………………….. 26

4 Figure Use-Case Diagrams ……………………………………………………………………………….…………….. 27

4 Figure Block Diagram ……………………………………………………………………………………………………… 28

5 Figure Designs Of The Project ……………………………………………………………………………………..…. 31



**CHAPTER 1: INTRODUCTION**



1 CHAPTER 1: INTRODUCTION

In today’s World time is very important and essantial for human life. As a result of that people started to use and create various kind of things like websites, smartphones and etc. With the help of these things people started to work faster and started to use thier time more efficiently. This project also aims to increase the efficiency in the work of teachers as well as students. With this project teachers can prepare an online exam very much more quickly than a normal exam and attend the students who will take the exam as they want on the system. Students then will attend the exam which the questions come from a pool randomly and they will see their results immediately on the system. The students can also see their past exam and the answers of that exam questions on the system as they like. Another thing the system does for the teacher’s is that the system can calculate the gpa of the class instantly due to the results of the students of that class. The other thing teacher’s can do is that they can add the course they give to the class and start preparing exams instantly for the students who take that course. So by the help of this project I assume that we can increase the efficiency of the work and decrease the time given to the work.

* 1. BACKGROUND

Time is the most unique and important thing in human’s life. Human race adjust almost everything according to the time. So they want to spend that time doing their hobbies. However, most people work in a job that they either don’t like or very time consuming that they don’t have enough time to spend for thier hobbies.So with the help of this project I expect to reduce the work time of the teacher’s and student. To make a project happen, I used a lot of web programming languages like JavaScript, Jsp, Eclipse Jee Oxygen, Tomcat, Microsoft Sql Server Management, CSS, Html, etc. I constructed the main part of the project using Eclipse Jee Oxgygen. Then, I started to work on the visual part of the project using CSS, Html and etc. to maket he web page more appeling to the human eye. After finishing the coding and styling part of the project I needed a localhost server and management of this server. Hence, I used Tomcat and Microsoft Sql Server Management to see and test if the web page is working properly. At a result of these steps and tests, the project is now working properly fort he teachers and students.

* 1. PROJECT MOTIVATION

This project stand for decreasing the time spent while preparing an exam for teachers and help students to see their exam results and where they did wrong more easily. The main motivation that encouraged me to make this project happen is to reduce the spent while working and prevent students from cheating. This is the main thing that motivated me to make that project happen.

* 1. PROBLEM STATEMENT

As long as people work in a job that they don’t like or in a very time consuming job the problems related with time will never go away. Other problem is that most of the students try to cheat in th exams. However, we as programmers can reduce the time spent on works and prevent students from cheating.

* 1. PROJECT QUESTION

The focus questions of developing this project was:

* How to combine eclipse with tomcat server?
* How the main layout of the site should look?
* How should I manage the servers on sql database?
* How to create a question pool on sql database?
* How to randomly select question from sql database?
* How to combine all the parts together in one piece?
  1. PROJECT OBJECTIVES

The main objectives of this project can be listed by follows:

* Succesfully combining the tomcat localhost server with eclipse.
* Succesfully managing the servers on sql.
* Making the layout of the website appealing to the human eye.
* Making the website user friendly for teachers and for students.
* Make people save their time using this project.
* Maket he exam time enough for students to prevent them cheating.
* Randomly asking questions from a pool.
* Prevent students from cheating.
* Combine all the pieces of the project together successfully without any major or minor errors.
  1. PROJECT SIGNIFICANCE

This project’s significance is very important for techers and students because of the fact that it can decrease the time spent while preparing or taking an exam as well as as it can prevent cheating in the exams. This project signifance also can be listed by these matters:

* Linking the tomcat server with eclipse.
* Making teachers and students save time while using this project.
* Prevent students from cheating.
* Make people spend more time with their hobbies instead of work.
* Using the project without any major error.
  1. PROJECT SCOPE

The scope of this project is for teachers of any school and the students of any school. This project aim to reduce the spent while preparing and taking an exam and prevent students from cheating. So with this project I want to decrease the time spent for teachers while they are preparing an exam. Also I want to decrease cheaters in an exam as well as the time spent for students while taking an exam and students will be seeing their points and mistakes instantly. Due to this matters I want to insrease the efficiency while taking and preparing an exam.

* 1. FIRSTSTEPS

At first, I did not much about managing sql servers. So I studied sql and started to learn about managign servers using sql from web as well as from my friends who has quite a knowledge about sql servers. After leraning about sql I started to write the main codes and created the main classes of the project using eclipse. To test the code part I used tomcat for the use of a local host server however at this part of the project I didn’t started to work on the layout of the webpage.

After the realization of that I have to start the layout of the project, I consulted some of my friends and made some researches about how a website should look. After doing the researches, I started to work on layout of the project using many different web promming languages like JavaScript, Html, CSS and etc. Then I decided to test the project progressively with the help of localhost server tomcat.



**Chapter 2: LıTERATURE REVIEW**



1. CHAPTER 2: LITERATURE REVIEW

This chapter includes a literature review of Qestion Paper Generator System ,concepts and benefits for users.

* 1. QUESTION PAPER GENERATOR SYSTEM

Question Paper Generator System is consist of three main parts. These parts are the coding part of the project on eclipse, designing the web page layout of the project using various web programming languages, last but not least connecting the project to a server with the use of tomcat and sql.

* 1. DEFINITION OF QUESTION PAPER GENERATOR SYSTEM

It is a web project consisting of two main parts. These main parts are: Handling the main coding part of the project using eclipse and doing the web layout and style using several web programming languages. After that connecting the project to a local host server with tomcat.

* 1. USING SQL

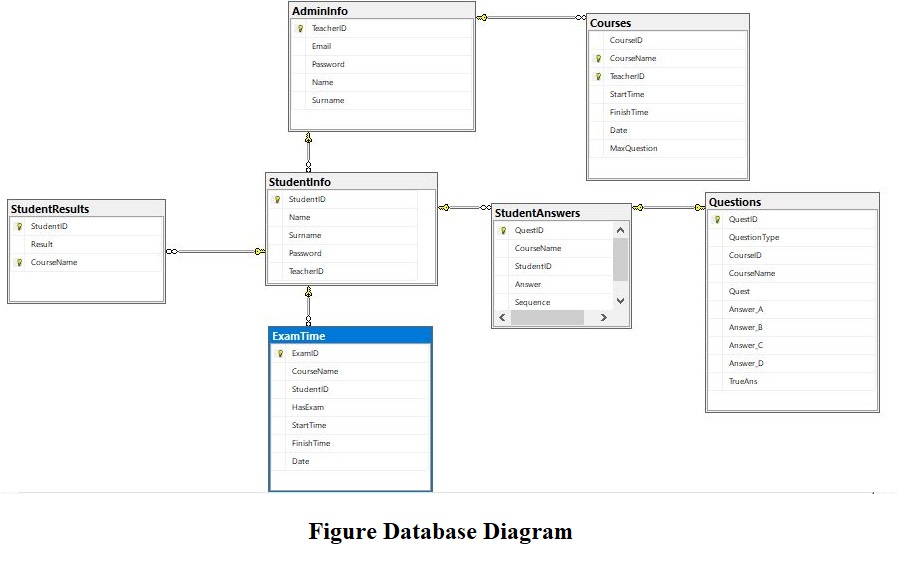
Structured Query Language (SQL) is a standard computer language for relational database management and data manipulation. SQL is used to query, insert, update and modify data. Most relational databases support SQL, which is an added benefit for database administrators, as they are often required to support databases across several different platforms.

First developed in 1970s at IBM by Raymond Boyce and Donald Chamberlin, SQL was commercially released by Relational Software Inc. (now known as Oracle Corporation) in 1979. The current standard SQL version is spontaneous, dealer-compliant and monitored by the American National Standards Institute. Most major dealers also have proprietary versions that are incorporated and built on ANSI SQL. SQLPlus (Oracle), and Transact-SQL (T-SQL) (Microsoft).

One of the most fundamental DBA rites of passage is learning SQL, which begins with writing the first SELECT statement or SQL script without a GUI. Increasingly, relational databases use GUIs for easier database management, and queries can now be simplified with graphical tools like drag-and-drop wizards. However, learning SQL is essential because such tools are never as powerful as SQL.

SQL code is divided into four main parts:

* Queries are performed using the omnipresent yet familiar SELECT statement, which is further divided into clauses, including SELECT, FROM, WHERE and ORDER BY.
* Data Manipulation Language (DML) is used to update, delete or add data and is actually a SELECT statement subset and is comprised of the DELETE, INSERT and UPDATE statements as well as control statements like BEGIN TRANSACTION, SAVEPOINT, COMMIT and ROLLBACK.
* Data Definition Language (DDL) is used for managing tables and index structures. Examples of DDL statements include ALTER,CREATE,TRUNCATE and DROP.
* Data Control Language (DCL) is used to assign and revoke database rights and permissions. Its main statements are GRANT and REVOKE.



* 1. WHY SQL

There are two essential purposes of a database language: to create and edit the database and to perform queries. The simple fact is that these tasks are accomplished by users of SQL and SQL powered applications with generally greater ease and efficiency, and that noone has yet come up with a better way of working with data in a relational database.

Though competitors have come and gone over the years, SQL has remained the standardbearing database language, working behind the scenes on GUI database interfaces and used by those who need customwritten queries and datamanipulations.

Perhaps the first reason for this is the mix ofpower, functionality and relative ease of use that SQL provides. No competing language has been able to combine high functionality with ease of use quite as efficient.

Key to avoide of unnecessary complexity is the fact that SQL’s command structure is restricted to definitional and manipulative commands. While a lack of ‘flow of control’ commands makes SQL somewhat computationally incomplete, the payoff is a language with a much simpler structure, and fewer commands and conventions.

Furthermore, the commands themselves are comprised of simple English words like Create Table, Insert Into etc.and essentially freeformat.

Thus SQL is one of the easiest languages used today both to learn and to use, and is likely to acquire significantly lower staff training costs than more complex languages.

The intuitive commandstructure allows users at any level of an organisation the option to acquaint themselves with some of the fundamentals of the language easily, even apart from database administration staff.

And, whether managers or applicationdevelopers, whether for querying, datamanipulation or other uses it is likely to be of use to many users throughout any organisation interested in formalising relational database usage, or in enabling a wider variety of workers to handle queries or manipulate data independent of GUI applications.

For all it‘s accessibility, SQL is powerful enough and wide enough in scope to satisfy the needs of datahandling workers from endusing management to database administrators who must design and edit the database.

Near universal recognition as the foremost database language is another compelling reason to use SQL. Virtually every database application is currently powered by SQL, with developers customising the language to their own ends like Oracle’s PL/SQL. These variations are based on the standard ANSI-SQL.However, do not differ greatly from the ANSI standard.

This universality breeds a variety of competing solutions to buyers, and simplifies the market considerably.

An unhappy customer of one database application can thus switch to another with less disruption, as the backend workings remain essentially the same, and the underlying database structures don’t need to be disturbed.

Going forward, the fact that all major vendors work with SQL also ensures a continuing investment in SQLbased applications. And with no major competitors to the standard beating at the door, the future of relational database implementation and management in one respect looks a lot like the past.

* 1. TOMCAT

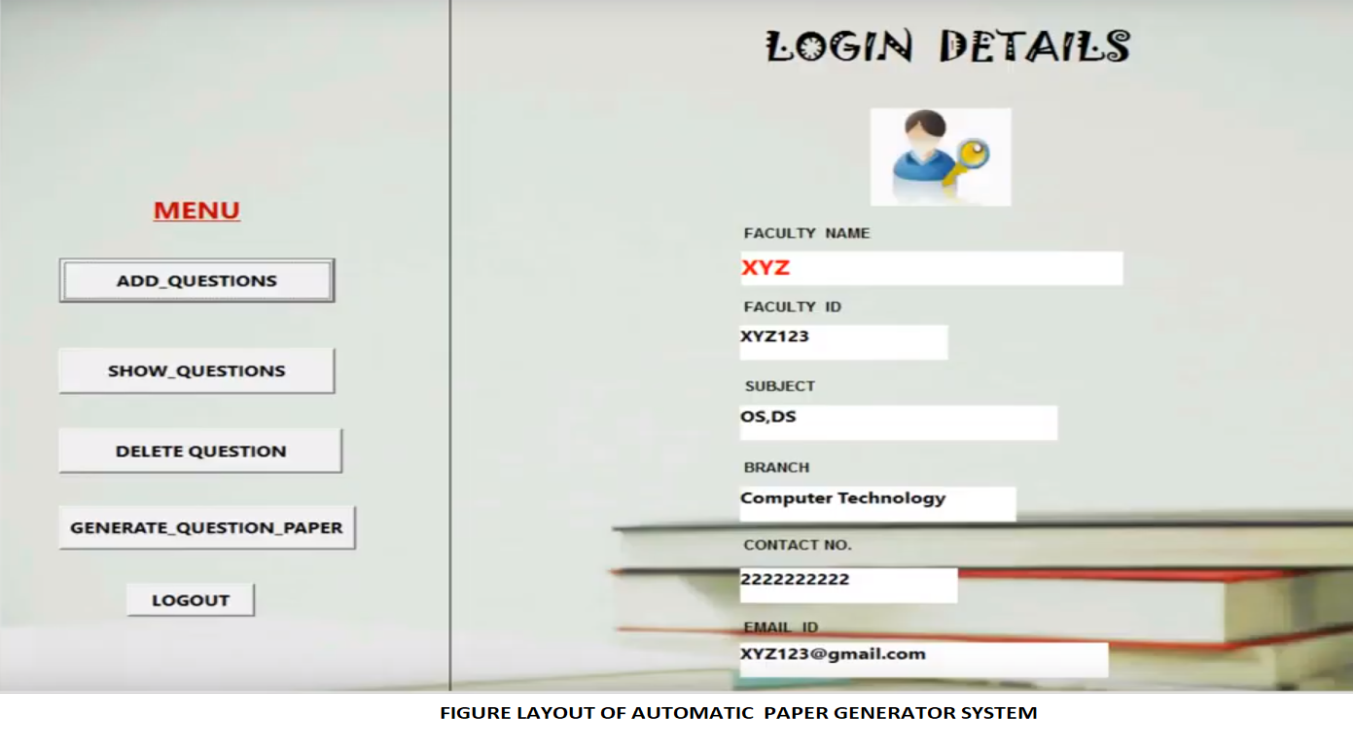
Tomcat is an [application server](https://searchsqlserver.techtarget.com/definition/application-server) from the Apache Software Foundation that executes Java [servlet](https://searchmicroservices.techtarget.com/definition/servlet)s and renders Web pages that consist of [Java Server Page](https://www.theserverside.com/definition/Java-Server-Page-JSP) coding. Described as a "reference implementation" of the Java Servlet and the Java Server Page specifications, Tomcat is the result of an open collaboration of developers and is available from the [Apache](https://whatis.techtarget.com/definition/Apache)Web site in both binary and source versions. Tomcat can be used as either a standalone product with its own internal [Web server](https://whatis.techtarget.com/definition/Web-server) or together with other Web servers, including [Apache](https://whatis.techtarget.com/definition/Apache), Netscape Enterprise Server, Microsoft Internet Information Server ([IIS](https://searchwindowsserver.techtarget.com/definition/IIS)), and Microsoft [Personal Web Server](https://searchmicroservices.techtarget.com/definition/Personal-Web-Server-PWS). Tomcat requires a Java Runtime Enterprise Environment that conforms to JRE 1.1 or later.Tomcat is also one of several open source collaborations that are collectively known as Jakarta.

* 1. RELATE WORKS

There are quite a few similar projects which uses similar components to the questin paper generator system. This part I will explain some of them in a short way.

* 1. AUTOMATIC PAPERGENERATOR SYSTEM

Automatic Paper Generation System helps to efficiently design question paper for any exam.The main objective of Automatic Paper Generation System is to reduce the work of teachers in the examination cell ,the software performs all tasks related to paper setting, starting from preparing question bank to printing paper. However, this project doesn’t help students in a way. Also it didn’t prevent students from cheating in any way.

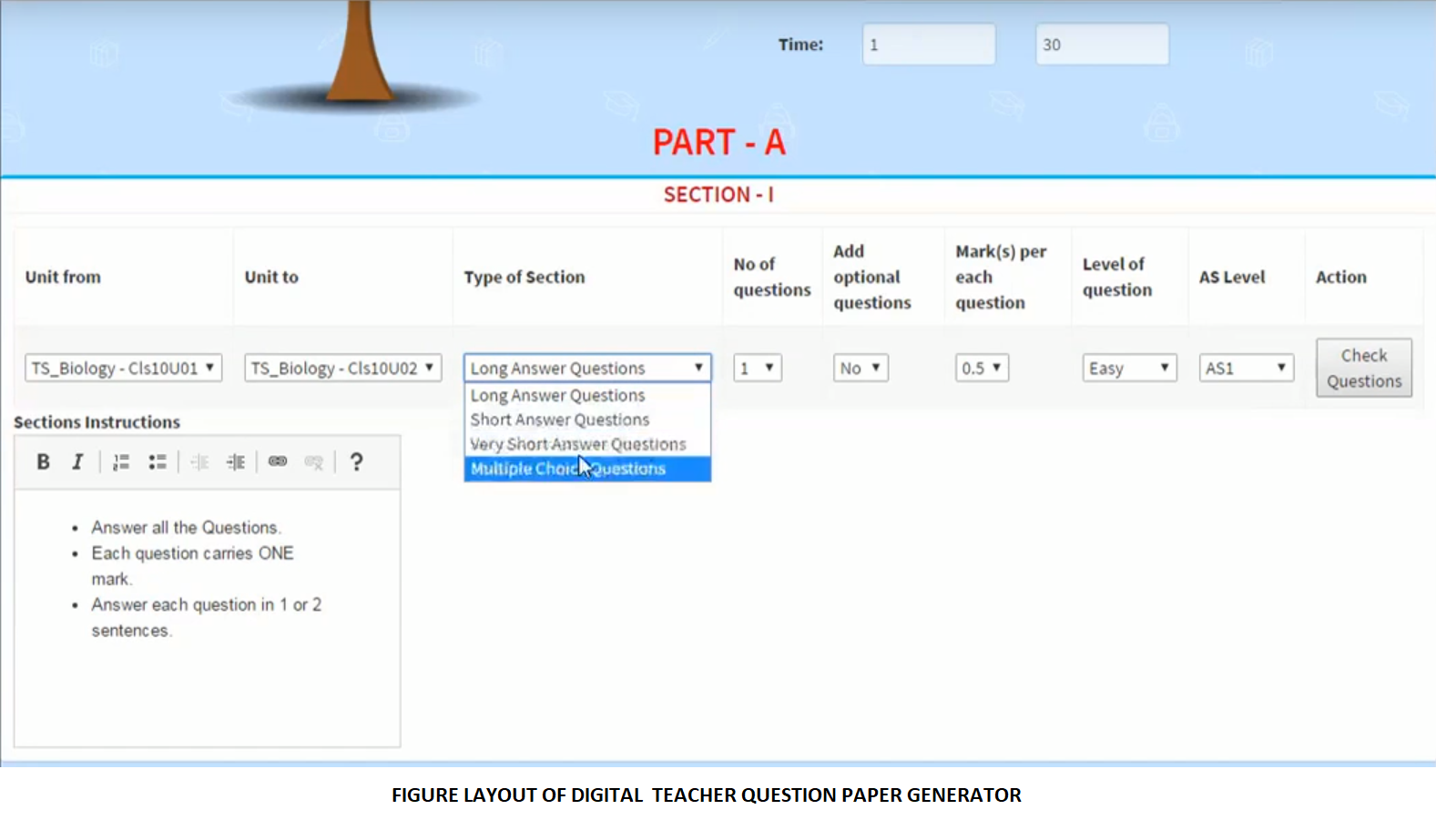


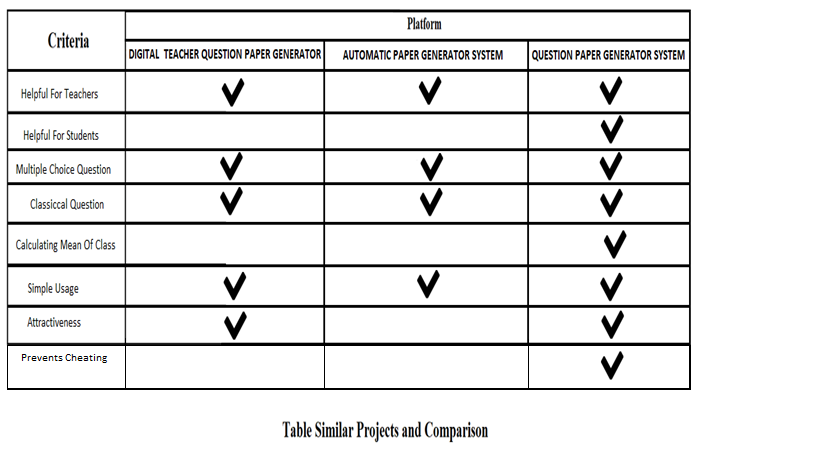
* 1. DIGITAL TEACHERQUESTION PAPER GENERATOR

Digital teacher question paper generator is a software, which prepares different types of question papers based on CCE pattern for the new Telugu states board syllabus.

Now-a-days, preparing an effective question paper is a complex, challenging job, whereas question paper generator makes the task much easier in 3 simple steps. All typical questions are formed by subject experts as per Academic standards.

However this project too doesn’t prevent students from cheating so I just took some hints from this project. But in the end I didn’t find it very helpfull for students.







**Chapter 3: PROJECT METHODOLOGY**

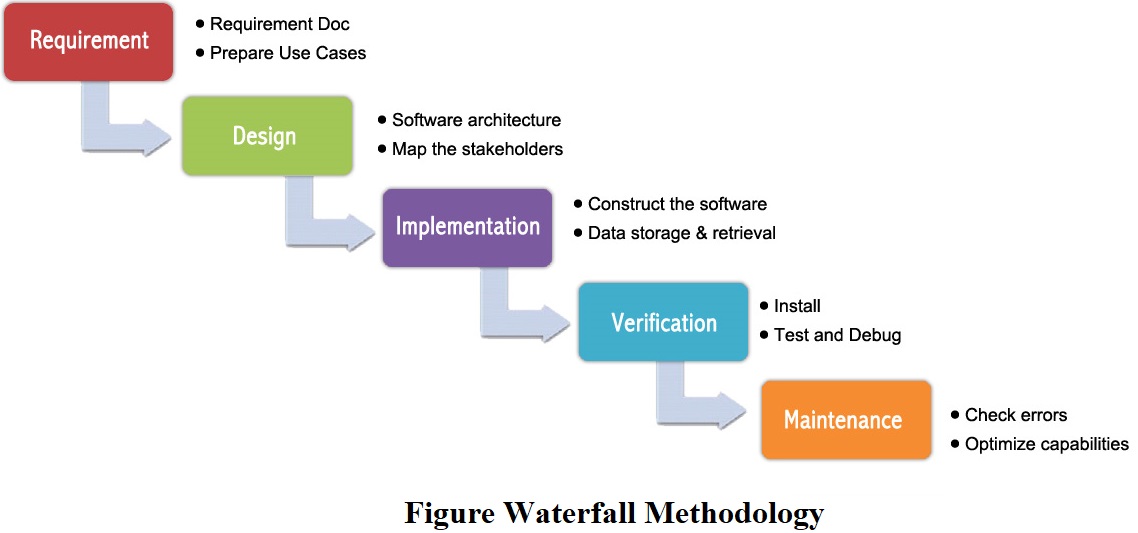


1. CHAPTER 3: PROJECT METHODOLOGY

In this chapter I will describe the adopted methodology that I used in the project. I will describe the methodology phases that I will use to accomplish the project and the phases that the methodology consist.

* 1. ADOPTED METHODOLOGY

The adopted methodology during the development of this project is the waterfall methodology because it was the most suitable methodology for a project like Question Paper Generator System. There are five steps in waterfall methodology starts with requirement analyses, followed by design, implementation, verification and finishes with testing and maintenance.



* 1. WATERFALL METHODOLOGY

The waterfall methodology was first introduced by Dr.Winston W. Royce in a paper published in 1970. The waterfall methodology emphasizes that a logical progression of steps be taken throughout the software development life cycle. It is now the most used methodology and it gets its name from the analogy of water falling downward. When the waterfall methodology was first introduced it was difficult for usage because it was incomplete in its

original framework and structure. Today, the most used version available includes a

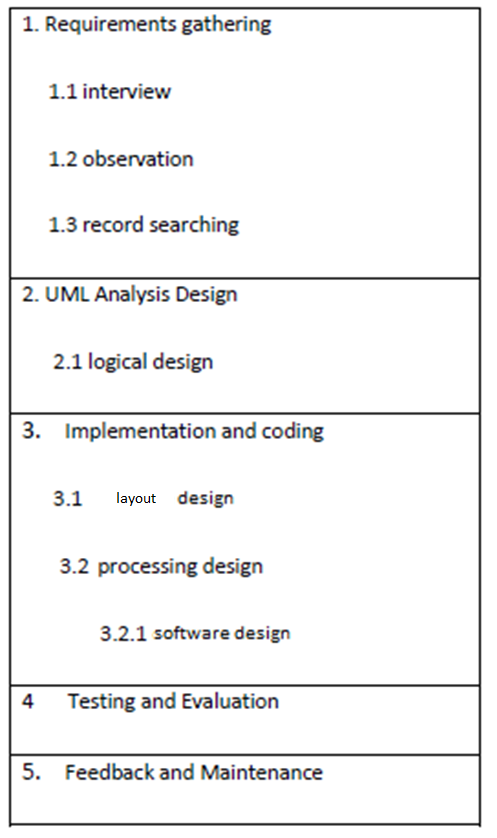
corrective feedback mechanism

* 1. WHY ADOPTING WATERFALL IS SUITABLE FOR QUESTION PAPER GENERATOR SYSTEM?

I adopted this methodology for the project because:

1. The requirements are clear, simple, and well understood for the project.
2. It can be developed step by step.
3. I understood the requirements of the project very well.
4. It was very suitable for this project.
   1. IMPLEMENTATION OF WATERFALL

The steps of the waterfall will be implemented to achieve this project. Each step has its own functions like in implementation and coding. I will make the layout design, processing design and software design.



**Figure Waterfall Steps**

* 1. WHAT I WANT FROM THE PROJECT

What I want from the Project is that, I want the Project to prevent students from cheating, making the time usemore efficeintly both for teachers and students. To maket hat happen I used a lot of web programming languages. However, I coded the main part of the Project on Java. I connected the sql databases to the java and used apache tomcat local-host server. The methods that I used were mssql for databse, tomcat on server and mvc method in general. After implementing the methods, I tested the Project with two of my friend. They were unable to cheat during the exam because of the time limit on the exam and the questions came randomly to them. So just one or maybe two questions were the same in both exams. After they took the exam, they saw their mistakes and the average score in exam instantly. Lastly, when ı was preparing the question I saw that it is time consuming at first. However after the preparation of the questions, I saw that I just need to add 10 or maybe 20 questions more to the pool because the questions are stored in the database and you can use them for another exam.

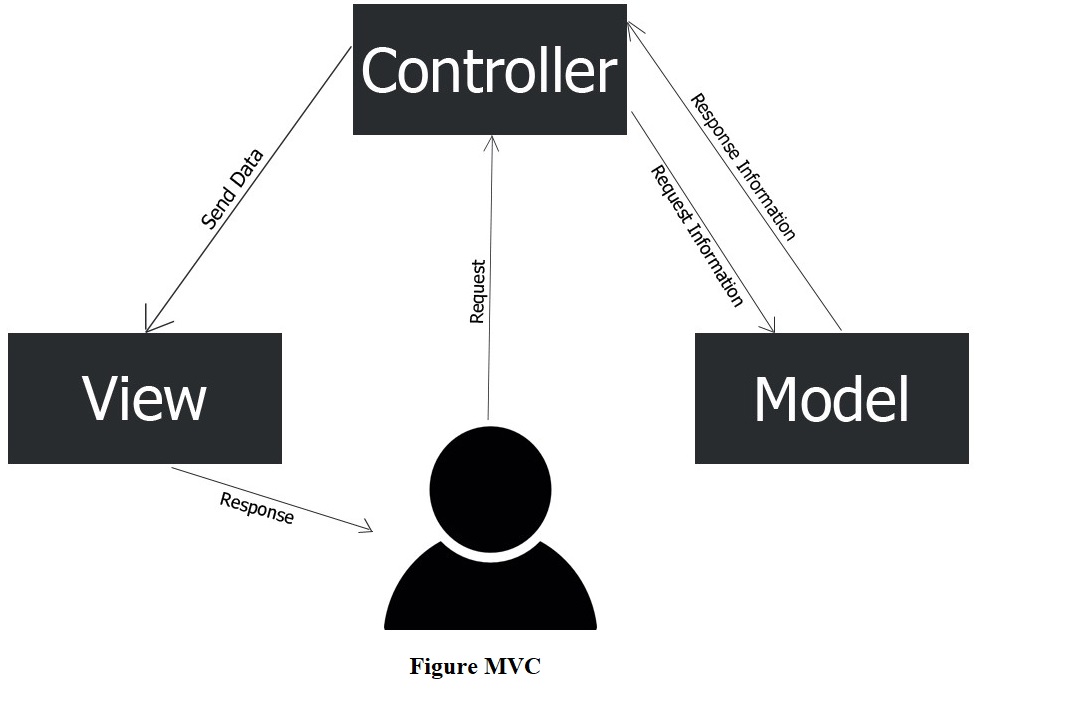
* 1. MVC

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

**Model:** The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.

**View:** The View component is used for all the UI logic of the application. For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

**Controller:** Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model. The same controller will be used to view the Customer data.





**CHAPTER 4: ANALYSIS AND DESIGN**



1. CHAPTER 4: ANALYSIS AND DESIGN

This chapter will provide a full description of the project and its users. Then this chapter will provide functional and non-functional requirements that have been collected using various methods. After the determination of the requirements, requirement analysis was adopted using various tools like use-case diagram, block diagram.

* 1. PROJECT DESCRIPTION

Question paper generator system is a unique project that consist of a lot of web programming languages, SQL, Tomcat, Eclipse Jee development. Some of the web programmin languages are JavaScript, Ajax, CSS, HTML etc. The purpose of the project is to help teachers prepare an exam more efficiently and more quickly. Another reason is to help students see thier mistakes in exam and see thier result immediately after the exam.

At first, I started to develop the back-end part of the project. After that, I started to build the front-end part of the project using JavaScript, Ajax, CSS, HTML etc. Then I used SQL to manage the datbases. Lastly, I connected the project to Tomcat local host server and tested it enough times. After the testing the project was done.

* 1. USER DESCRIPTION

There is two main user for Question Paper Generator Sytem (Teachers and Students). Teachers must first sign up for the site. After signing up, teachers should login and add the courses to the sites which they teach and start preparing questions for the courses. Another step is that adding the students to the site. Lastly, they should assign the students who are affiliated with that courses and prepare the exams.

On the other hand, students just need to login the site and take their exam. After taking their exam, they can see their exam results with the average of the class. Another thing they can do is to see the old questions and thier answer of each exam.

* 1. REQUIREMENTS DEVELOPMENT

To develop the functional and non-functional requirements of the project, I made a lot of researches from internet, talked with my friends and made brainstorming with them. With the help of my friends and the knowledge I got from my researches I generated the requirements according to the previous models I mentioned earlier.

* 1. SYSTEM REQUIREMENTS

Before the creation of the project, it is a must to anticipate the model and design of the project. How the user will intereact with the website. After thinking these in mind we can get the necessary requirements for the project. These requirements will determine the quality and chracteristic of the project to make a easy usage for users. After the reqirements are ready we can develop other technical works like system design, development of the project, testing, implemantation and operation.

Like in every system there are functional and non-functional reqirements to be considered to determine the reqirements of the project. With the non-functional requirements we can describe how the system work like reliability and maintainability.

* 1. FUNCTIONAL REQUIREMENTS

The functional requirements were developed by reviewing literature review, comparing similar circuits, interviewing and brainstorming with friends and experts.

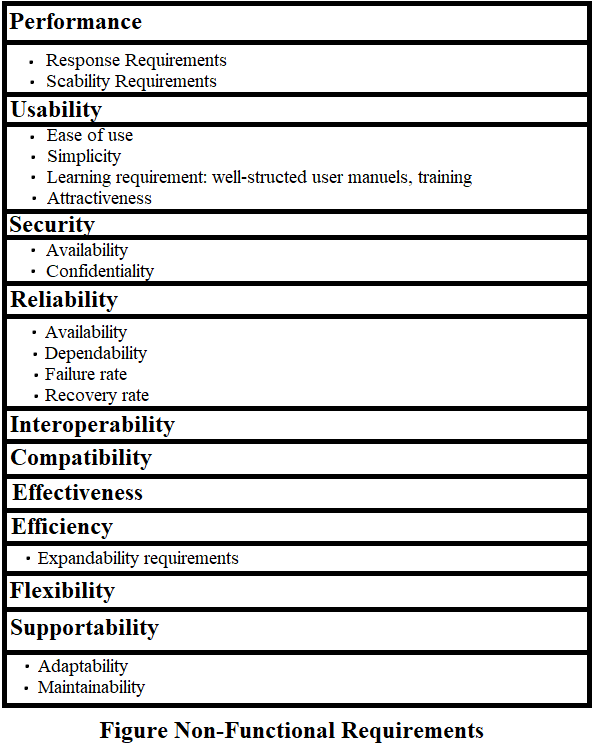
The user who will use this project will have specific requirements in order to use this project.

* 1. USER REQUIREMENTS

The main reqiurement of the project is of course a computer which the teachers will work on and students will take their exams.

1. Teachers must sign up and then sign in for the site.
2. Teachers must add the courses that they teach.
3. Teachers must prepare questions for the courses.
4. Teachers must add the students that they teach to and than assign the courses to these students.
5. Teachers must prepare the questions affiliated with the lessons.
6. Students must login the site and take the exam.
   1. NON-FUNCTIONAL REQUIREMENTS

So many non-functional requirements can be considered in order to develop this project. For example, the project must be simple to achieve its goal, it must be understandable, easy to learn and use. To make that happen I brainstormed with a lot people mostly my friends who have more knowledge than me in this particular subject and adopted non-functional requirements to the project.

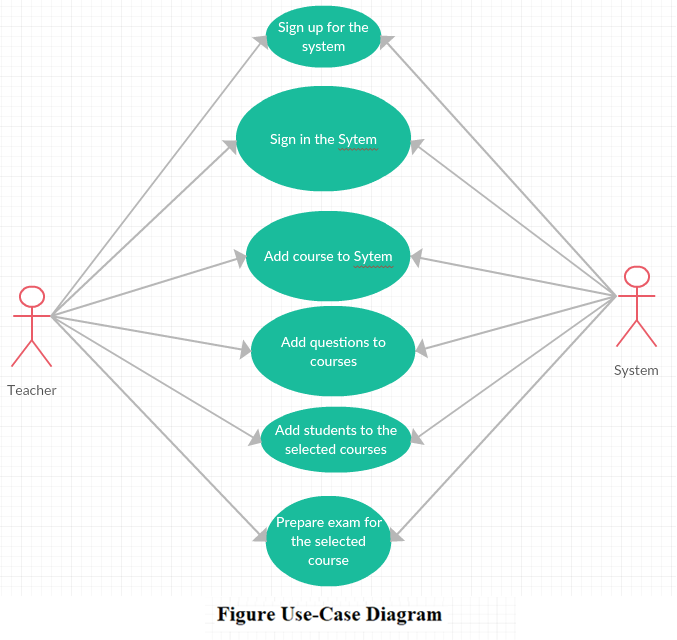


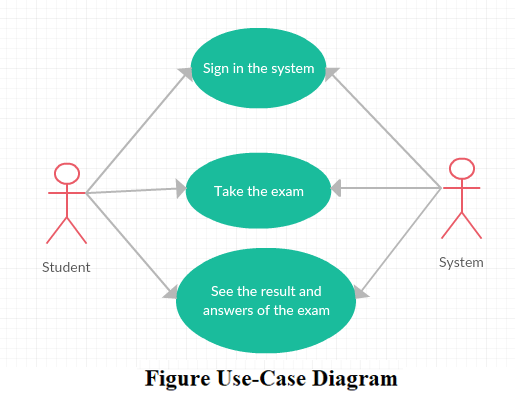
* 1. SYSTEM ANALYSIS

This section will contain the analysis of funtional and non-functional requirements of the project using use-case diagram and activities will be analized using a block diagram.

* 1. USE-CASE DIAGRAM

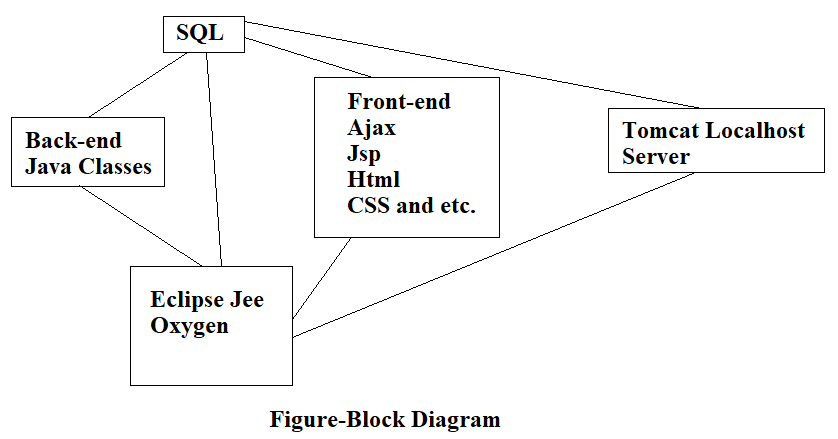
To see the main aspects of the project I developed a use-case diagram. The diagram shows the interaction between the user and the system. The use-case diagram was developed in order to clarify, identify and organize system requirements.





* 1. BLOCK DIAGRAM

To see the main aspects of the project I developed a block diagram. The whole front-end and backend part is putted inside of Eclipse Jee Oxygen. After that, I designed a SQL database to keep track of datasets. Last but not least, I connected the project yo Tomcat localhost server.





**CHAPTER 5: IMPLEMENTATION**



1. CHAPTER 5: DESIGNAND IMPLEMENTATION

This chapter will cover the design and implementation of the project. The implementaion phase combines the design phase outputs, requirements and the technology for usage.

* 1. DESIGN PHASE

Designing phase can be explanied by connecting every part of the system together. The back-end part, front-end part, tomcat and sql part.

The front-end part is the layout part of the project which is appealing to the human eye. The back-end part is the main part that does several operations of project. Tomcat is the local host server. Lastly, SQL is the part that manages datasets.

* 1. BUILDING THE BACK-END

It took me more than two months to finish the back-end part of the project. The reason fort hat is, I first did a lot of researches during the process of the back-end part. I consulted with my friends who have more knowledge than me about this subject. After that, I started coding in Eclipse and coded like 1900 lines of code.

* 1. BUILDING THE FRONT-END

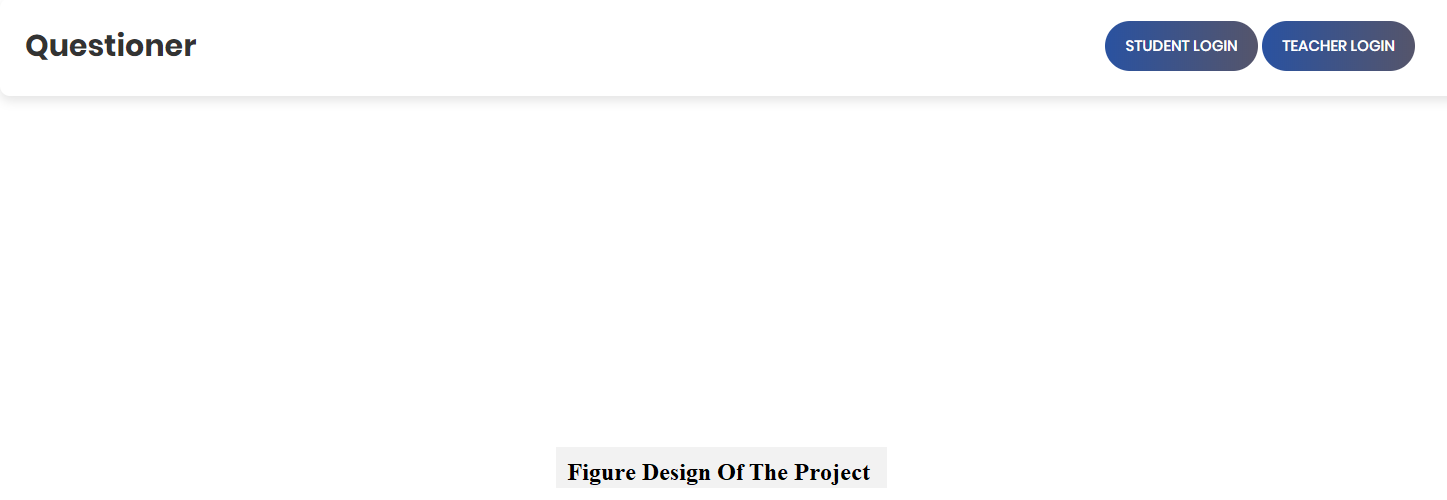
After I finished the back-end part of the project, I started building the front-end part of the project. Firstly, I consulted with my friends and did my reseacrhes. After doing that, I started designing the front-end part using various languages. I will list them all here: TTF, OTF, WOFF, WOFF2, SVG, HTML, Ajax, CSS, JSP, Java-Script and LESS. I coded like 2000 maybe more lines of code in the front-end part.

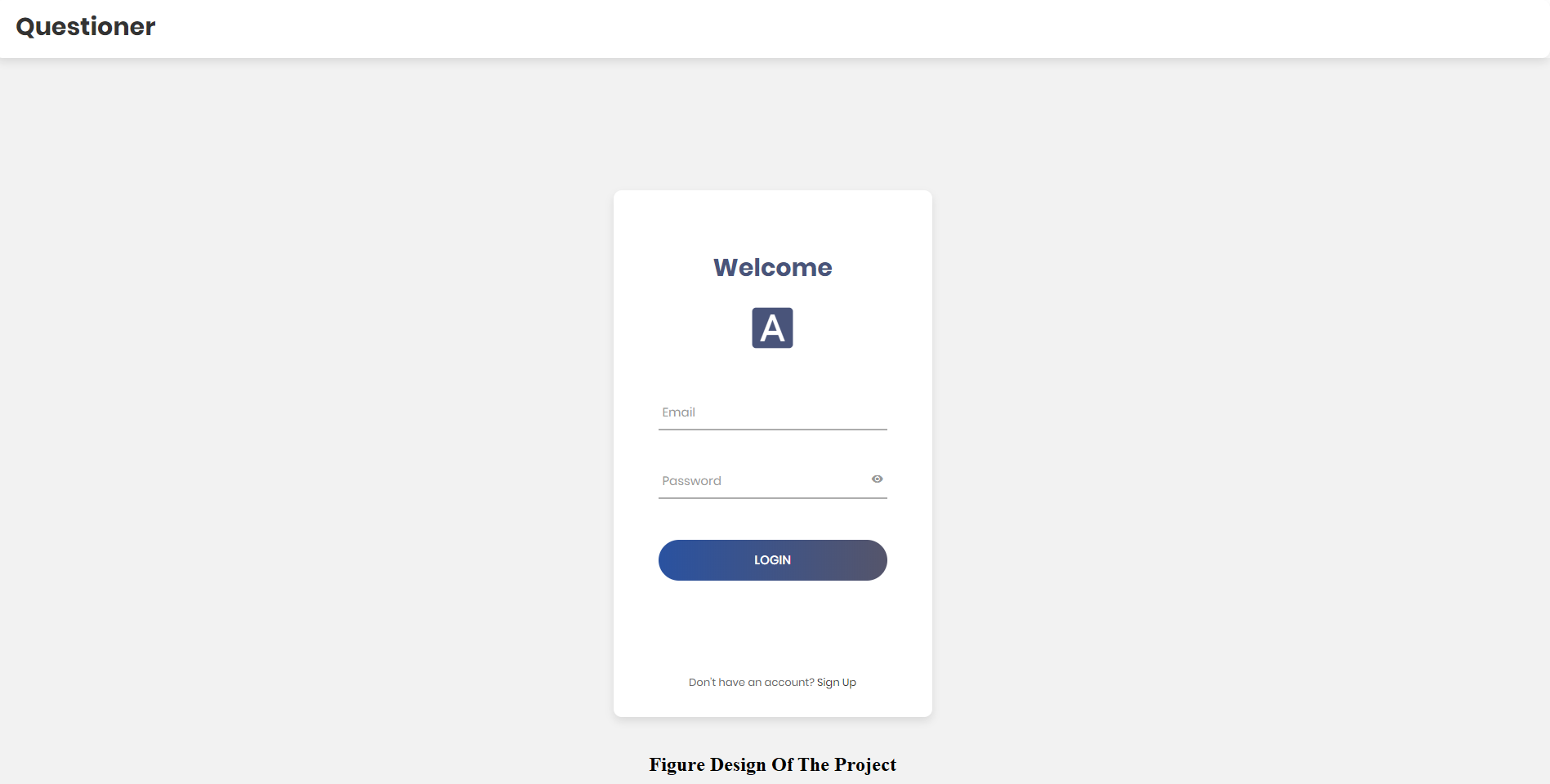
* 1. IMPLEMENTATION PHASE

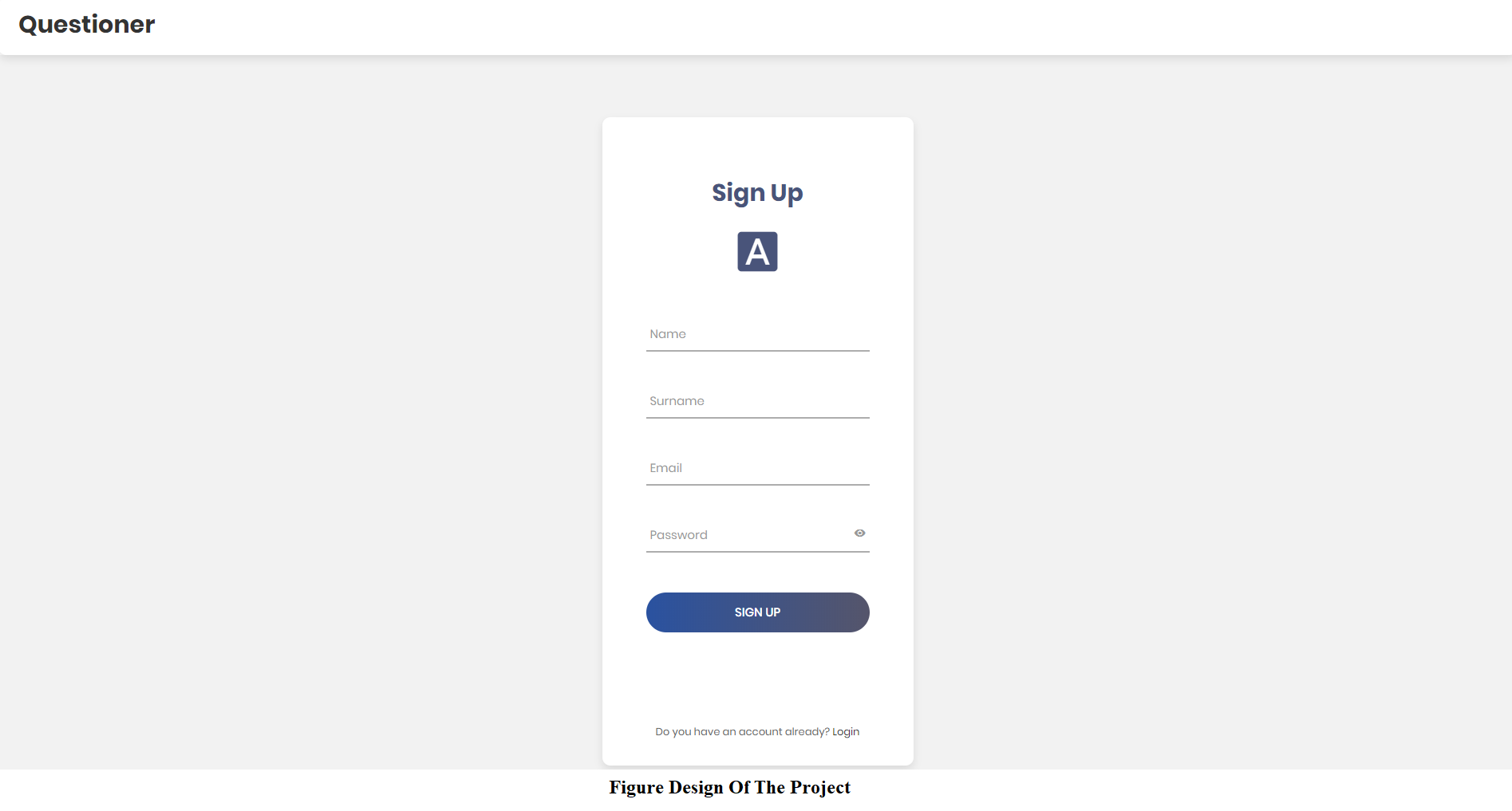
After the development of both front-end and back-end parts of the project, I started to combine them together using Eclpse Jee Oxygen. After that, I installed SQL to manage the datasets within my project. Lastly, I connected the project to a Tomcat local-host server and combined all the parts together.

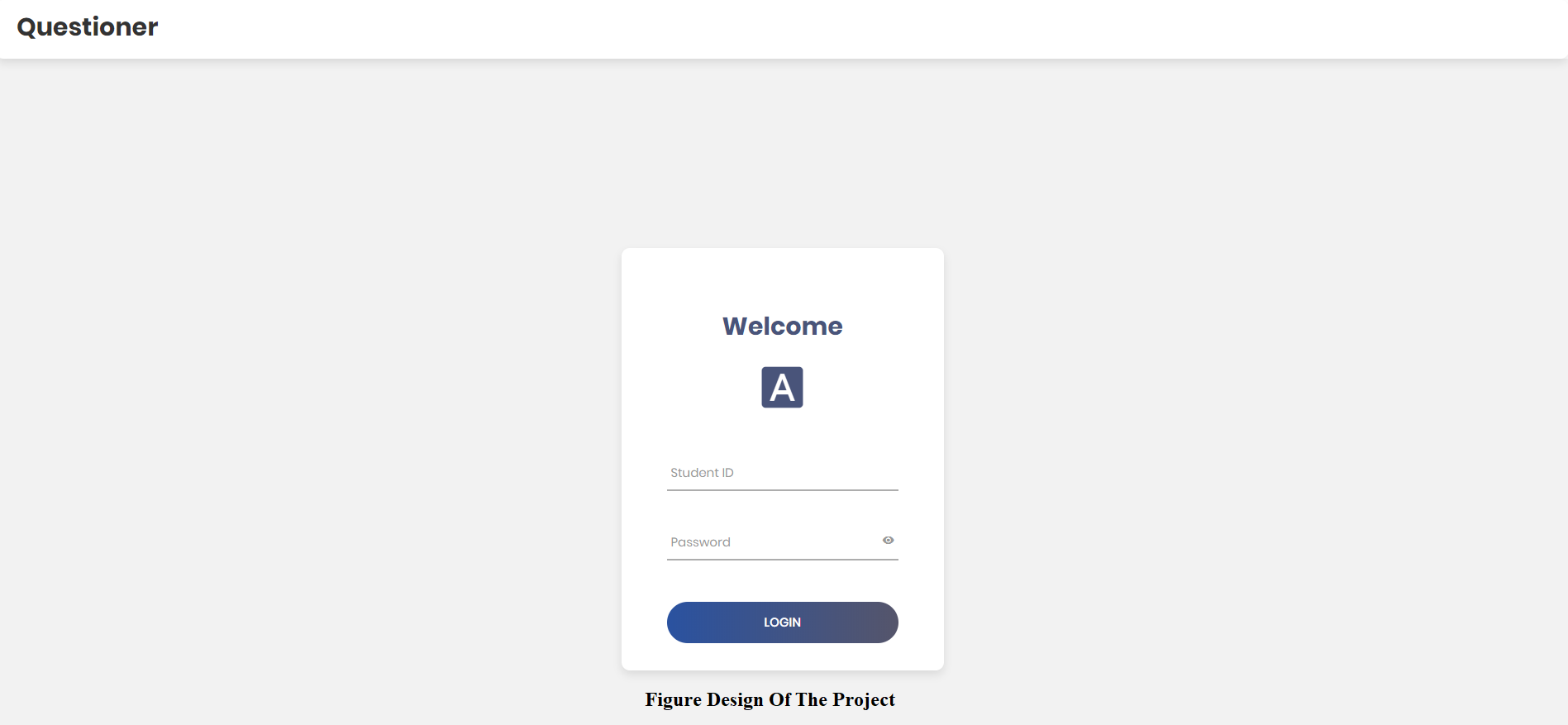
* 1. DESIGN OF THE PROJECT

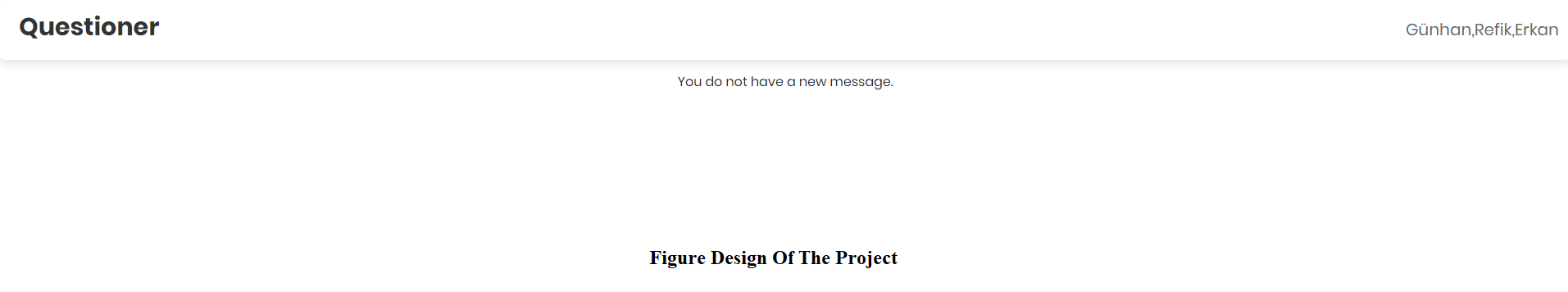
I took the screenshots of the design of the project to show all the details about the design of the project.

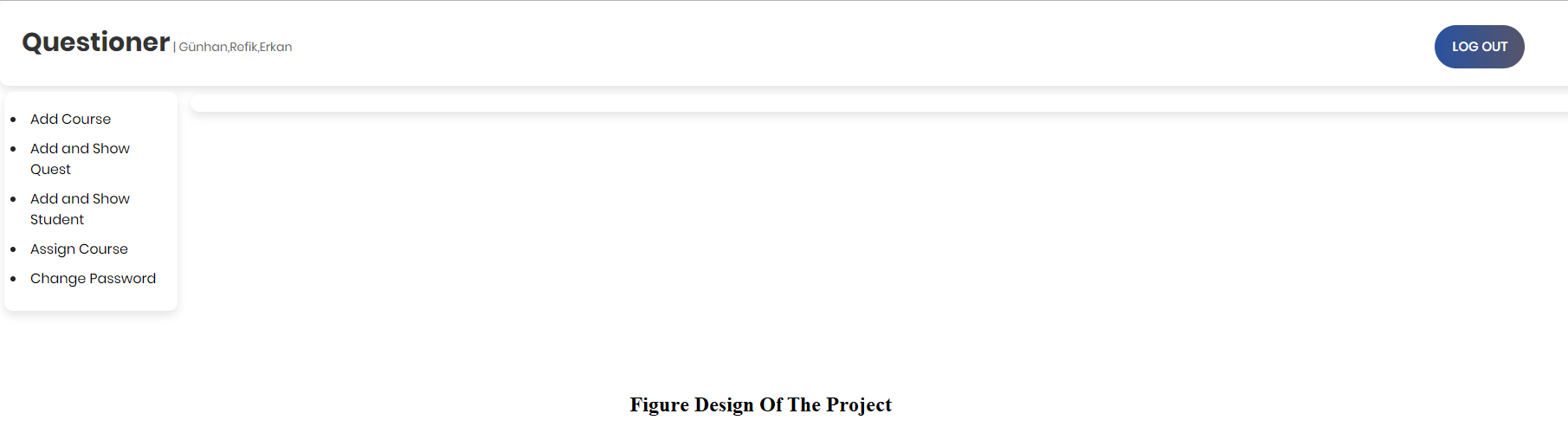


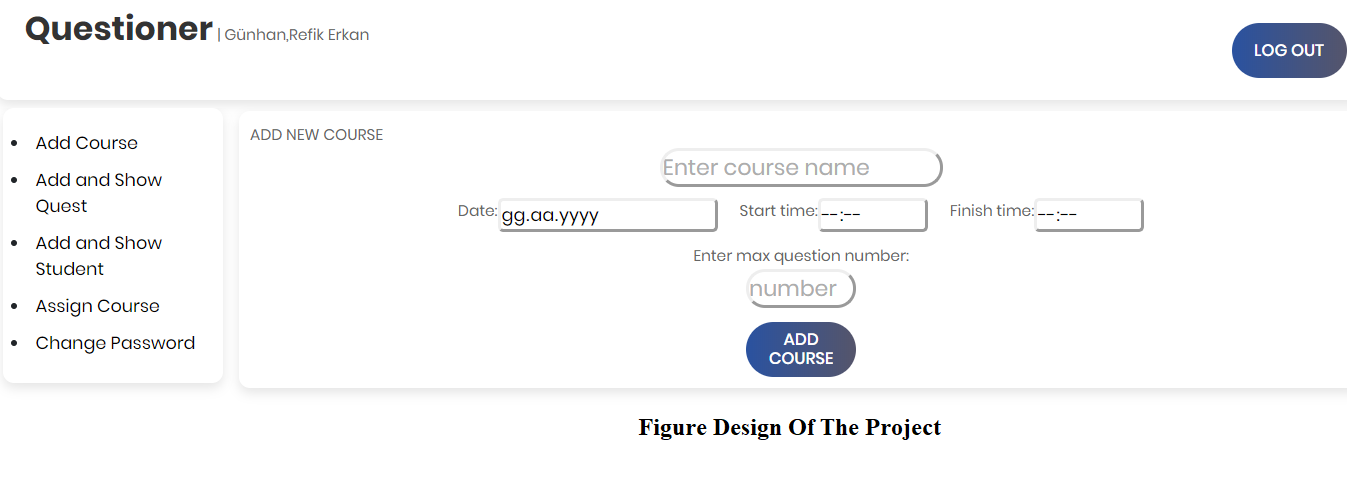


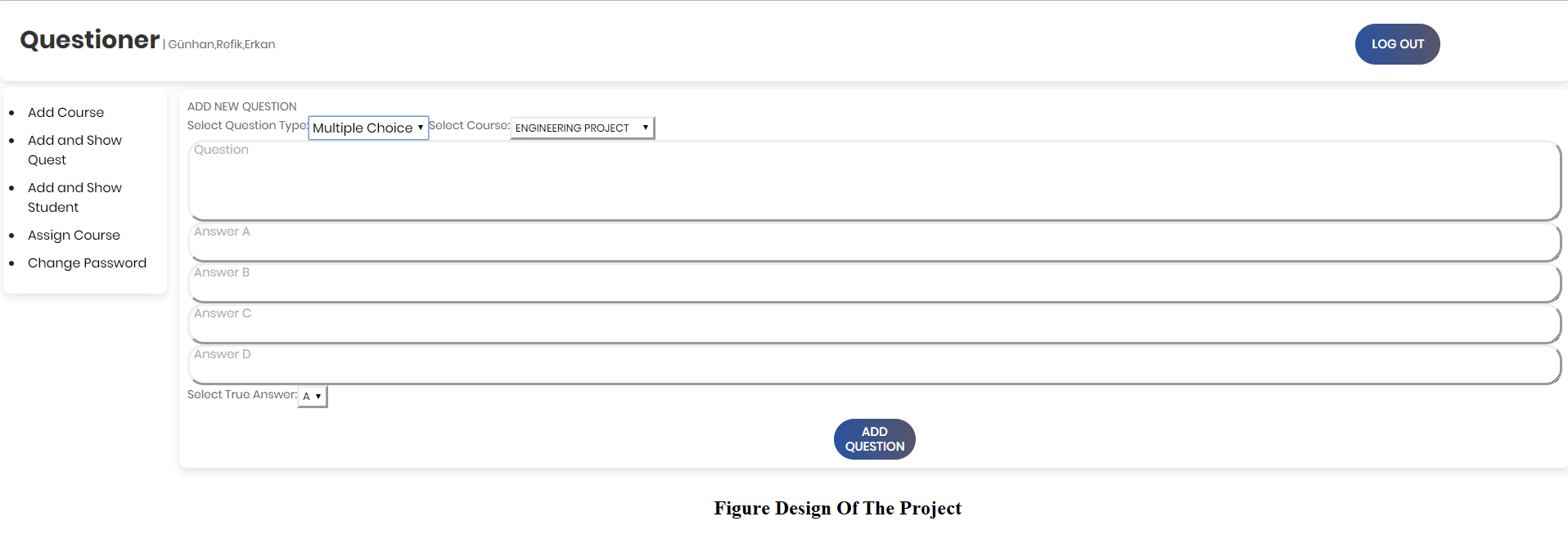


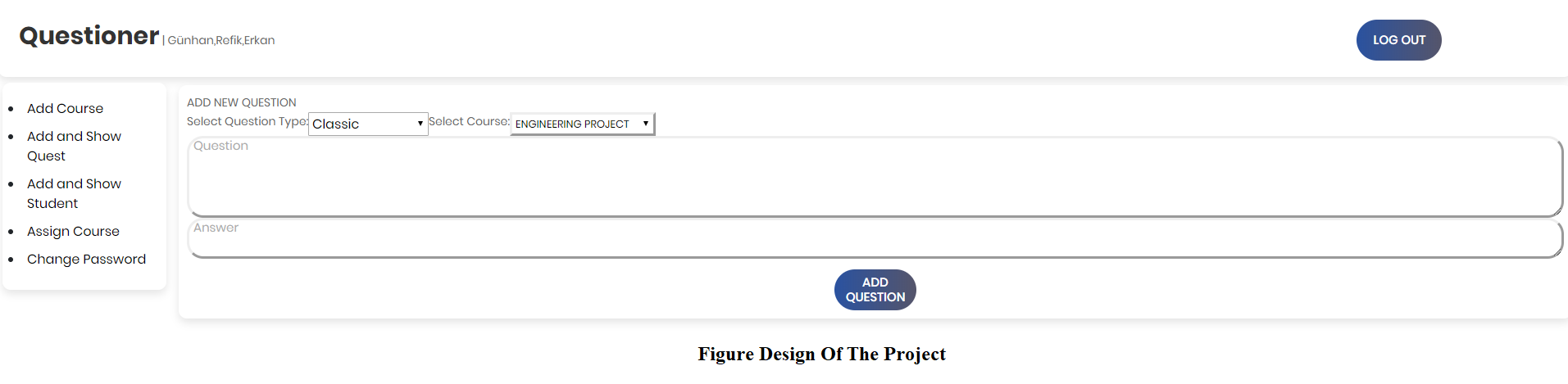


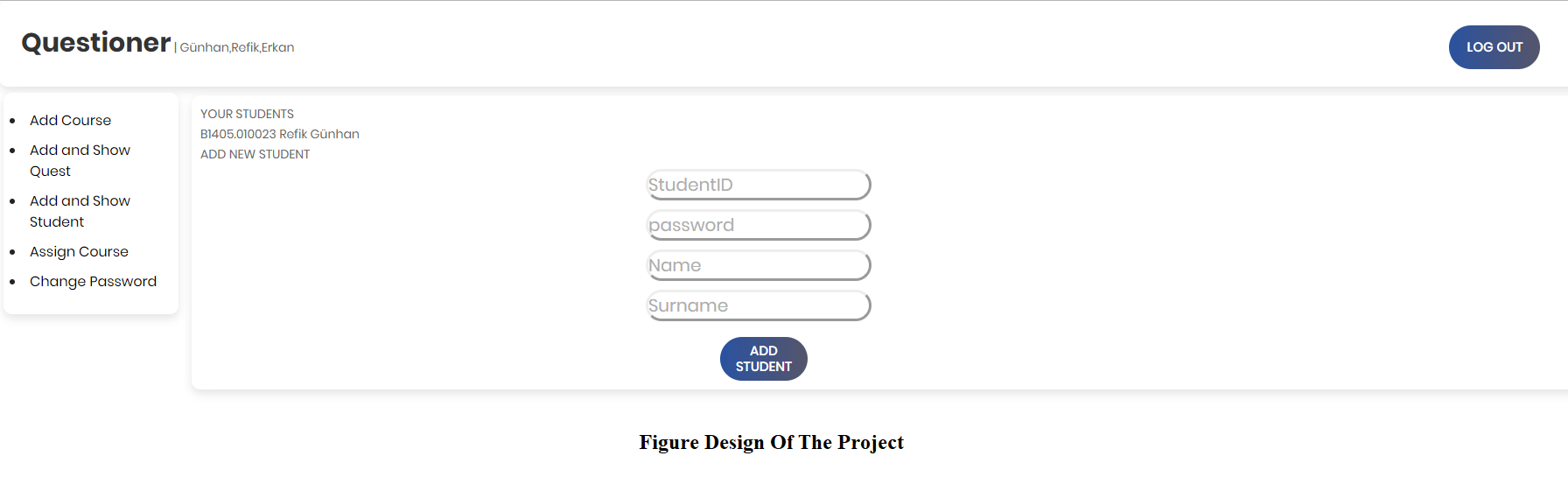


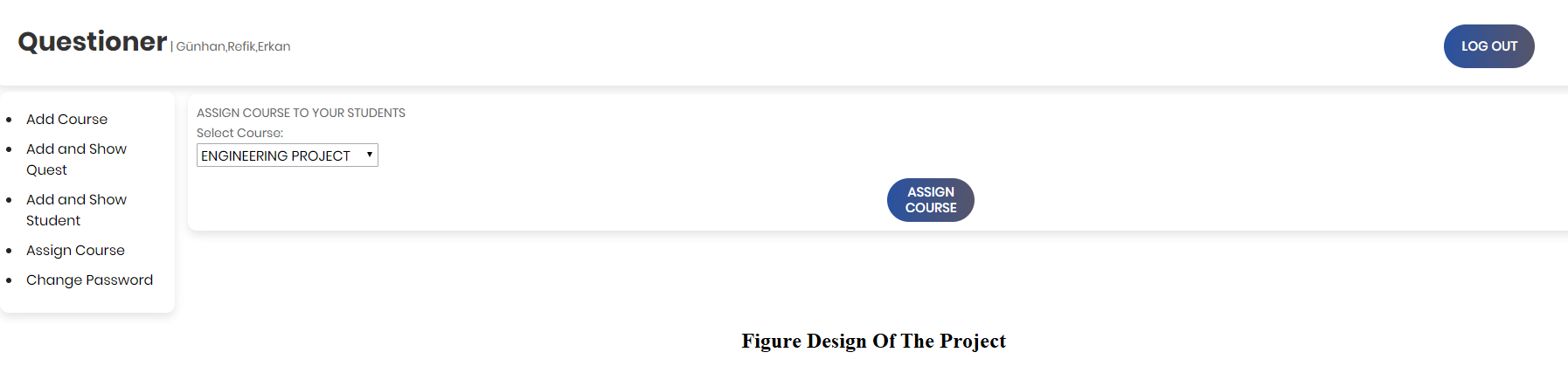


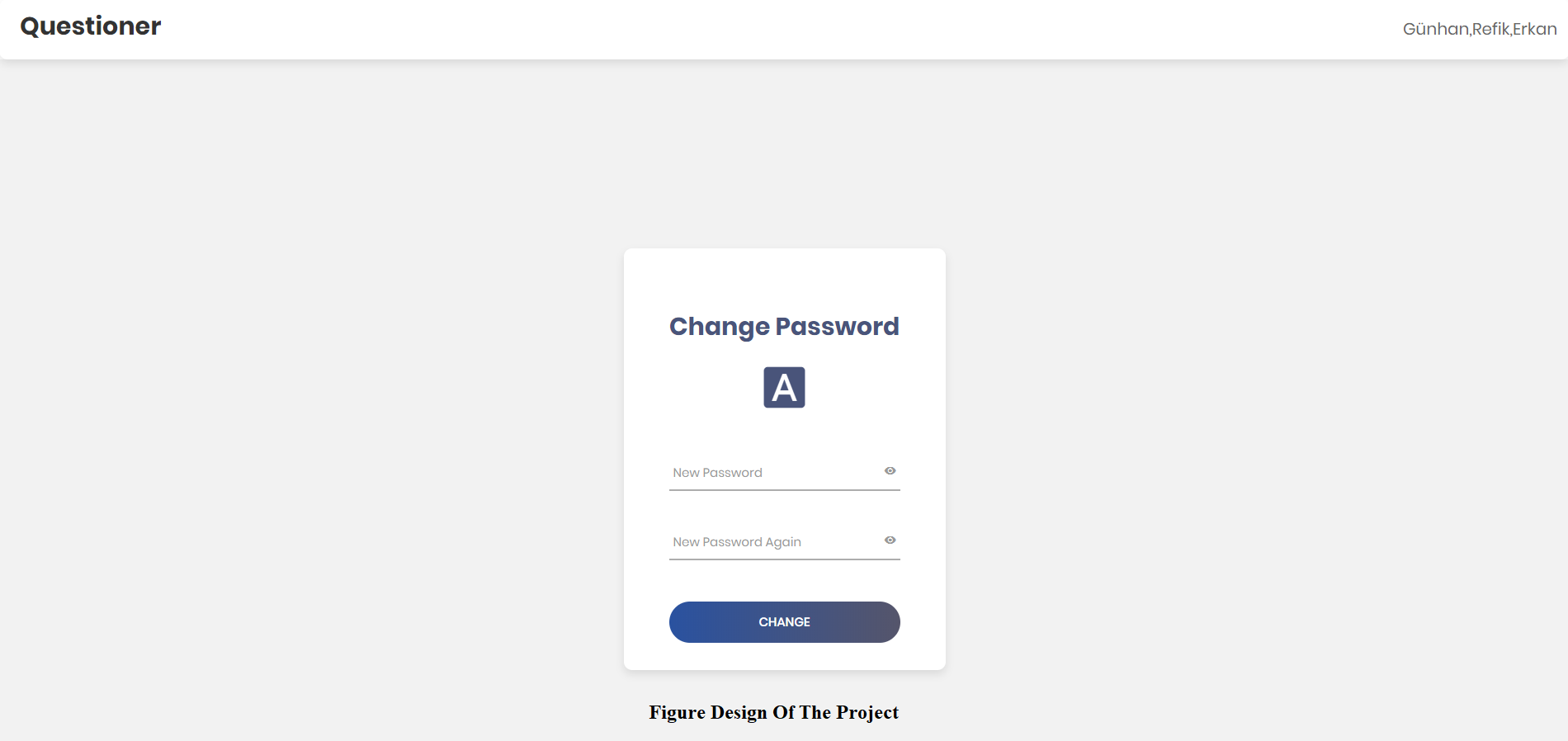


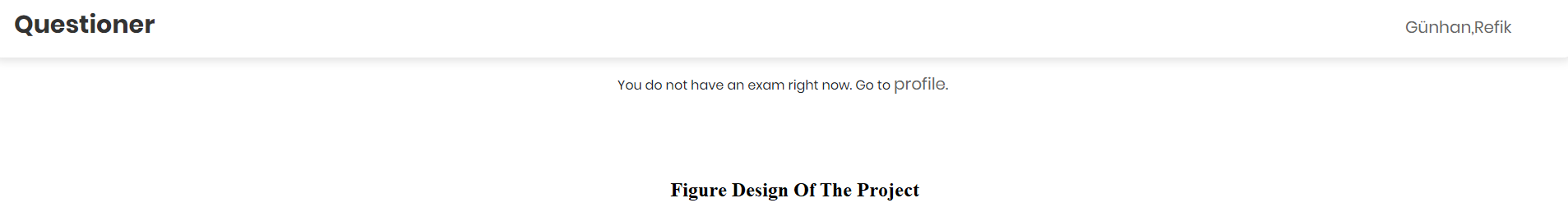


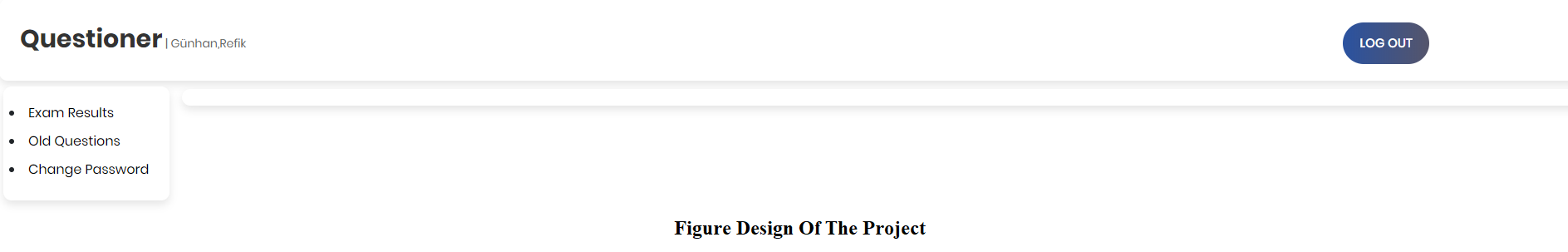


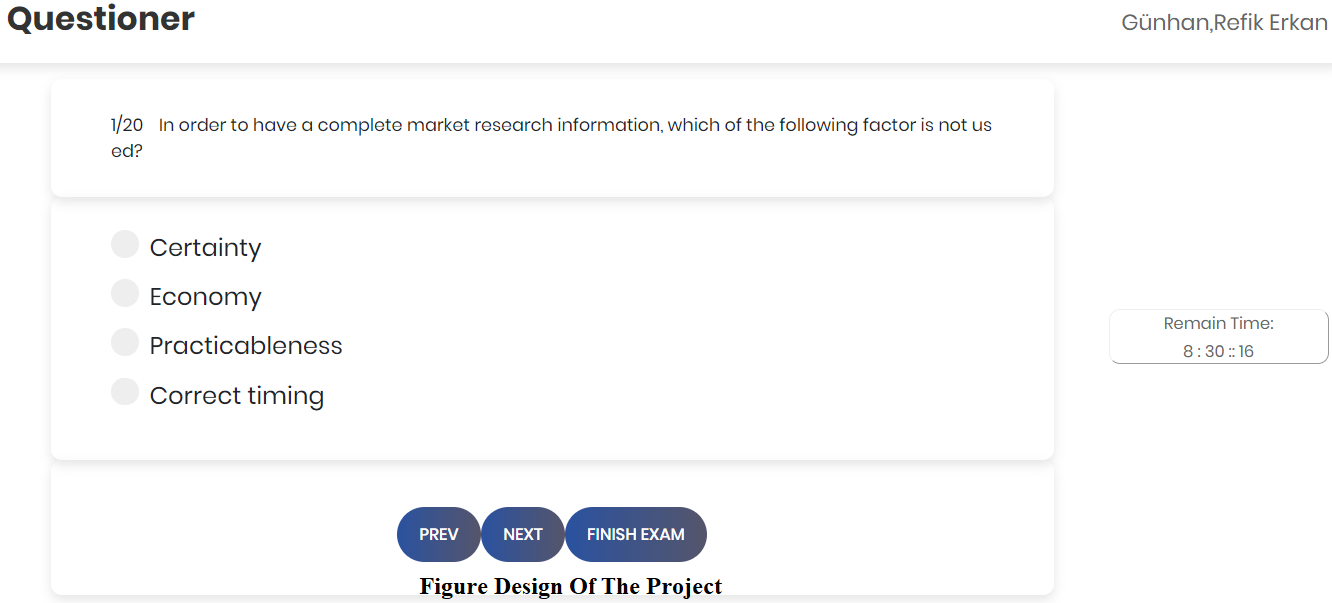


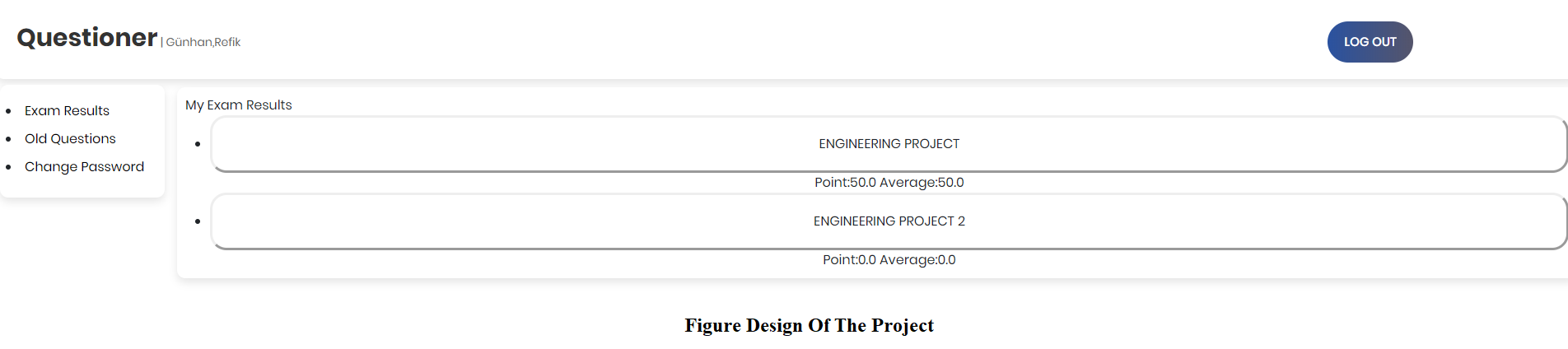


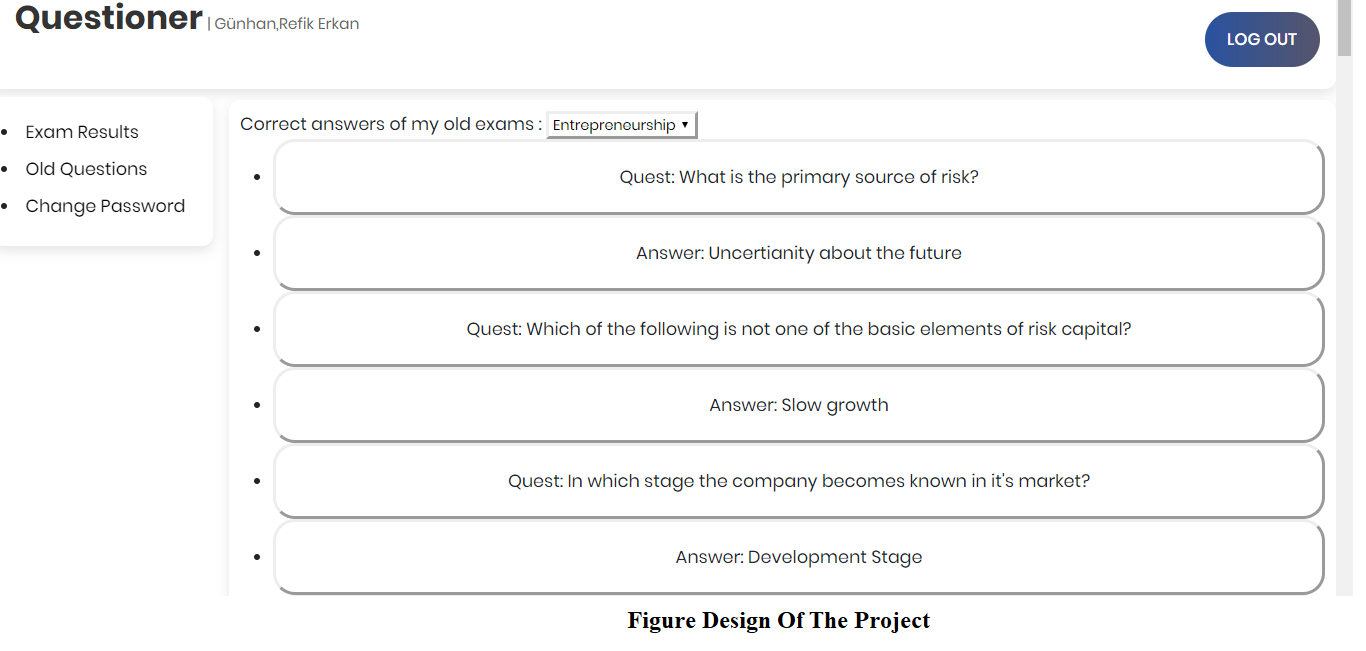


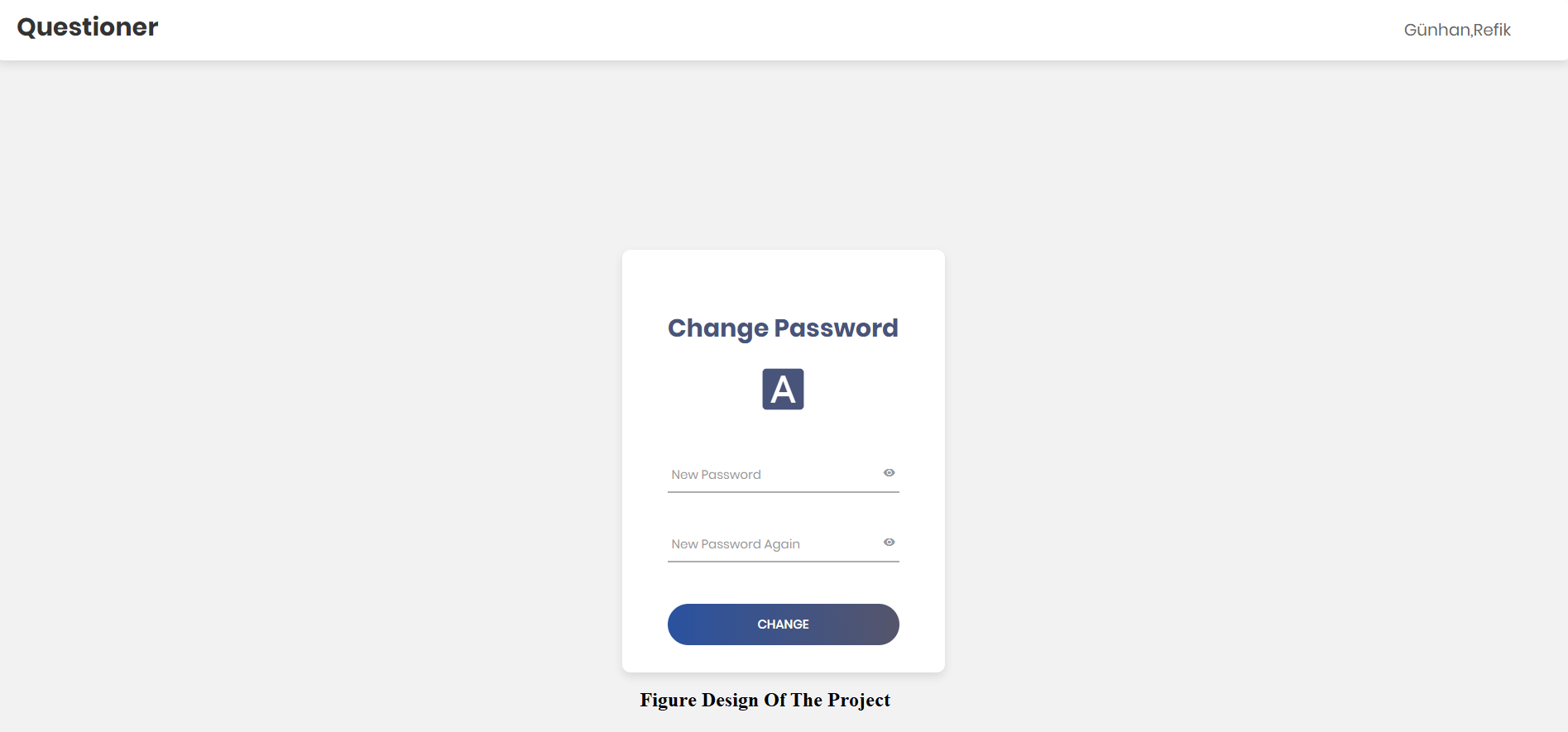














**CHAPTER 6: TESTING AND TOOLS**



1. CHAPTER 6: TESTINGAND TOOLS

This chapter will contain the last phase of this project, which is testing and tools. In the testing part, performance and functionality testings were performed.

* 1. TESTING

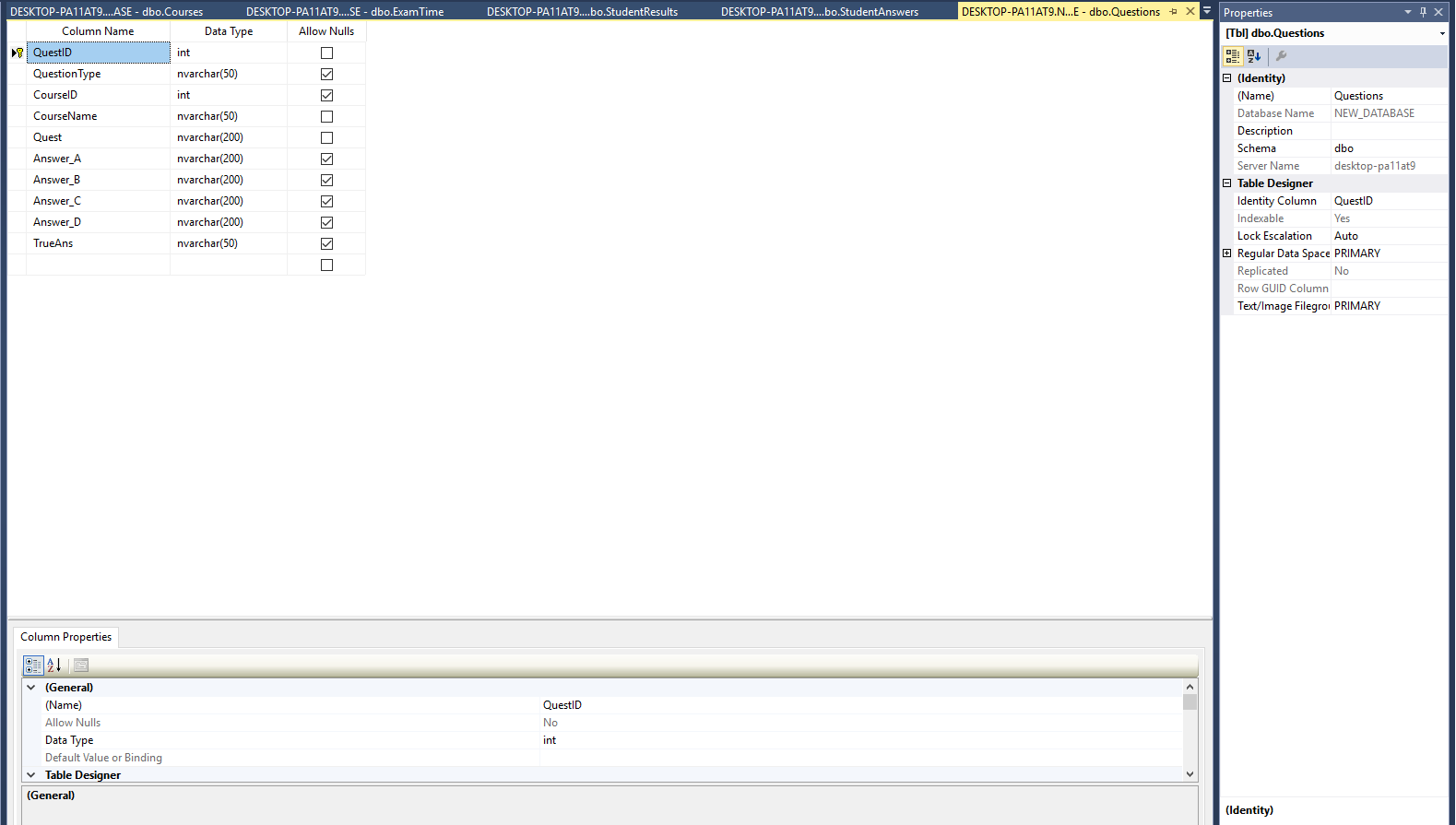
Two types of testing was done to test the Question Paper Generator System. These tests were performance and functionality testing.

* 1. PERFORMANCE TESTING

After combining all the parts together in my project I began testing it. One major problem is that at first I couldn’t enter a classical type of question to the system. This was major beacuse I didn’t want teachers just make multiple choice type of question. After consulting with my friends and with the help of internet researches I solved that problem and designed it using Eclipse and various web programming languages.

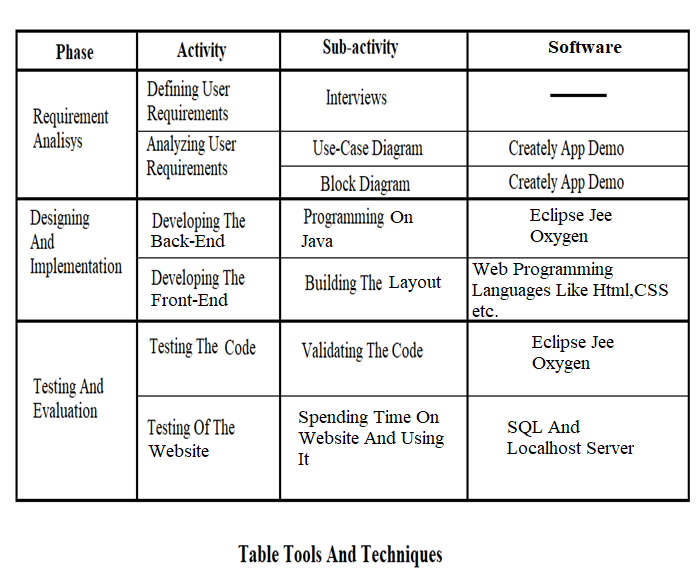
* 1. FUNCTIONALITY TESTING

After solving classical qustion problem I started to spend time on the site to see if there is something I don’t want. Luckily everything was working acording to the plan. The datas I enter was storing in SQL. Also Tomcat was working properly too.



As we can see that there is nothing wrong about data storage and everything is working properly on SQL.

* 1. TOOLS AND TECHNIQUES



* 1. MAINTANCE

This is the final stage of this project. At this stage, maintaining, updating and enhancing the system will take place, which contains all the problems and errors that are met through the usage of the completed system which resulted in a updated and enchanced system.

I am maintaining both front-end and back-end parts of the project because if I decide that something will change in fron-end I may need to change it in back-end too. This must be done because front-end and back-end part are connected to each other. So if I change something on both part, the other will be affexted from it. This can lead the need of experience to maintain the components whenever something changes.

* 1. CONCLUSION

In today’s World, with the improvement of the technological development, people have become more demanding in terms of quality of life. However, today they don’t have enough free time to spend for their hobbies.

So, I decided to maket his project for at least lower the amount of time that teachers work by preparing an exams. I also thought about students too and made the project helpfull for them. For example, students will be seeing their exam results instantly and see their mistakes on the exam instantly too.

However, this project doesn’t encourage people to be hardworking or doesn’t encourage them to spend more time with their hobbies. If people want to spemd more time with their hobbies, they need have a planned lifestyle and be hardworking.

REFERENCES

* To learn about eclipse adn downloading eclipse. [www.eclipse.org](http://www.eclipse.org) (18.03.2019 )
* To learn about webdesigning and coding. [www.tutorialspoint.com](http://www.tutorialspoint.com) (19.03.2019)
* To see about the importance of EclipseJee. [www.slideshare.net](http://www.slideshare.net) (19.03.2019)
* To learn about how to manage servers on eclipse using tomcat. [www.jazz.net](http://www.jazz.net) (19.03.2019)
* To learn about coding in eclipse. [www.jdoodle.com](http://www.jdoodle.com) (20.03.2019)
* To learn about Java IDE. [www.compilejava.net](http://www.compilejava.net) (20.03.2019)
* To maket he diagrams. [www.creately.com](http://www.creately.com)(25.03.2019)
* To learn about synonyms of english words. [www.thesaurus.com](http://www.thesaurus.com)(27.03.2019)
* To learn about Tomcat. [www.theserverside.com](http://www.theserverside.com) (31.03.2019)
* To learn about using SQL. [www.databasedev.co.uk](http://www.databasedev.co.uk)(04.04.2019)
* To learn about SQL and web coding. [www.codingsight.com](http://www.codingsight.com)(07.04.2019)
* To learn about SQL. [www.sqlcourse.com](http://www.sqlcourse.com)(09.04.2019)
* To learn about SQL and Operators. [www.w3schools.com](http://www.w3schools.com)(12.04.2019)
* To learn about SQL commands. [www.mode.com](http://www.mode.com)(15.04.2019)
* To learn about SQL and Tomcat. en.wikipedia.org (17.04.2019)
* To leran about what is SQL. [www.techopedia.com](http://www.techopedia.com)(19.04.2019)
* To learn about why I should use SQL and why is it important. [www.quora.com](http://www.quora.com)(23.04.2019)
* To learn about english words. [www.tureng.com](http://www.tureng.com) (26.04.2019)
* To leran about web programming. [http://voidcanvas.com](http://voidcanvas.com/) (28.04.2019)
* To learn about CSS. [www.purelybranded.com](http://www.purelybranded.com) (03.05.2019)
* To learn about web designing. [www.networksolutions.com](http://www.networksolutions.com) (05.05.2019)
* To learn about HTML. [www.educba.com](http://www.educba.com) (07.05.2019)
* To learn about web-fonts. [www.andrewsun.com](http://www.andrewsun.com) (09.05.2019)
* To learn about web-font formats. [www.transfonter.org](http://www.transfonter.org) (10.05.2019)
* To learn about the importance of txt files. [www.seositecheckup.com](http://www.seositecheckup.com) (11.05.2019)
* To learn about Xml files. whatis.techtarget.com (13.05.2019)
* To learn how to use xml files. [www.makeuseof.com](http://www.makeuseof.com) (14.05.2019)
* To learn about how to prepare a question paper generator system. [www.researchgate.net](http://www.researchgate.net) (02.06.2019)
* To learn about how to prepare a question paper generator system. [www.ijtra.com](http://www.ijtra.com) (02.06.2019)
* To see how other people prepared question paper generator system. [www.ieeexplore.ieee.org](http://www.ieeexplore.ieee.org) (03.06.2019)
* To see how other people prepared question paper generator system. [www.digitalteacher.in](http://www.digitalteacher.in) (03.06.2019)
* To see other aspects about question paper generator system. [www.semanticscholar.org](http://www.semanticscholar.org) (04.06.2019)
* To see an implementation about question paper generator system. [www.sciencedirect.com](http://www.sciencedirect.com) (05.06.2019)
* To see how algoritms work on question paper generator system. [www.ccsenet.org](http://www.ccsenet.org) (05.06.2019)
* To compare my Project with other similar projects. [www.gtu-info.com](http://www.gtu-info.com) (06.06.2019)
* To see the differences between my design and other designs. [www.brpaper.com](http://www.brpaper.com) (07.06.2019)
* To learn about how virtual question paper generator system works. [www.pediawikiblog.com](http://www.pediawikiblog.com) (07.06.2019)

ADDITIONAL INFORMATION (CODES)

\*

\* ALL OF THE DATABASE TRANSACTIONS ARE PERFORMED IN THIS CLASS.

\* THE TRANSACTIONS THAT ARE PERFORMED IN FRONT-END ARE TOOK FROM DATABASE, ADDED, EDITED OR DELETED.

\*

\* \*/

**public** **class** Data

{

Connection conn = **null**;

String dbURL = "jdbc:sqlserver://localhost";

String user = "sa";

String pass = "123456";

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CHECKS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**public** **boolean** IsAdmin(String name){

//The function that checks if the login is made by a student or a teacher.

**boolean** check = **false**;

Pattern VALID\_EMAIL\_ADDRESS\_REGEX =

Pattern.*compile*("^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$", Pattern.***CASE\_INSENSITIVE***);

Matcher m = VALID\_EMAIL\_ADDRESS\_REGEX.matcher(name);

**if**(m.find())

check = **true**;

**return** check;

}

@SuppressWarnings("deprecation")

**public** **boolean** IsExamInterval(Date current, Date exam, Date finish){

//The function that checks if the students has exam or not.

**boolean** check = **false**;

**if**( (current.~~getDay~~() == exam.~~getDay~~() && current.~~getMonth~~() == exam.~~getMonth~~() && current.~~getYear~~() == exam.~~getYear~~()) &&

((current.~~getHours~~() >= exam.~~getHours~~() && current.~~getMinutes~~() >= exam.~~getMinutes~~()) || ( current.~~getHours~~() >= exam.~~getHours~~() && current.~~getMinutes~~() < exam.~~getMinutes~~())) &&

((current.~~getHours~~() <= finish.~~getHours~~() && current.~~getMinutes~~() <= finish.~~getMinutes~~()) || (current.~~getHours~~() < finish.~~getHours~~() && current.~~getMinutes~~() > finish.~~getMinutes~~()) ) )

check = **true**;

System.***out***.println("Current: "+current.~~getDay~~()+"-"+current.~~getMonth~~()+"-"+current.~~getYear~~()+"::"+current.~~getHours~~()+":"+current.~~getMinutes~~());

System.***out***.println("Exam: "+exam.~~getDay~~()+"-"+exam.~~getMonth~~()+"-"+exam.~~getYear~~()+"::"+exam.~~getHours~~()+":"+exam.~~getMinutes~~());

System.***out***.println("Finish: "+finish.~~getHours~~()+":"+finish.~~getMinutes~~()+"\n\n\n");

**return** check;

}

**public** **boolean** CheckUserNamePassword(String name,String password) **throws** SQLException {

//A function that controls whether the user is held in the database according to the input information entered on the front-end.

DriverManager.*registerDriver*(**new** com.microsoft.sqlserver.jdbc.SQLServerDriver());

**boolean** Check = **false**;

conn = DriverManager.*getConnection*(dbURL, user, pass);

**if** (conn != **null**)

{

Statement stmt = conn.createStatement();

String sql = **null**;

**if**(IsAdmin(name))

sql = "SELECT Password FROM NEW\_DATABASE.dbo.AdminInfo WHERE Email = '"

+name+"'";

**else**

sql = "SELECT Password FROM NEW\_DATABASE.dbo.StudentInfo WHERE StudentID = '"

+name+"'";

ResultSet rs = stmt.executeQuery(sql);

**if**(rs.next())

**if**(password.equals(rs.getString("Password")))

Check = **true**;

rs.close();

stmt.close();

conn.close();

}

**return** Check;

}

**public** **boolean** CheckEmail(String email){

//Checks if the user who is registered will be registered via the previously specified mail.

**boolean** check = **false**;

**try**

{

DriverManager.*registerDriver*(**new** com.microsoft.sqlserver.jdbc.SQLServerDriver());

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT Email FROM NEW\_DATABASE.dbo.AdminInfo WHERE Email = '"+email+"'");

**if**(!rs.next())

check = **true**;

stmt.close();

conn.close();

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** check;

}

**public** **boolean** CheckExam(String ID){

//Checks if the student has an exam.

Date current = **new** Date(System.*currentTimeMillis*());

**boolean** check = **false**;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT HasExam,Date,StartTime,FinishTime FROM NEW\_DATABASE.dbo.ExamTime WHERE StudentID = '"+ID+"' AND HasExam = 1 ORDER BY Date,StartTime ASC";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next()) {

Date examDate;

Date finishTime;

**try** {

examDate = (Date) **new** SimpleDateFormat("yy-MM-dd HH:mm").parse(rs.getString("Date")+" "+rs.getString("StartTime"));

finishTime = (Date) **new** SimpleDateFormat("HH:mm").parse(rs.getString("FinishTime"));

**if**(IsExamInterval(current,examDate,finishTime))

check = **true**;

} **catch** (ParseException e) {e.printStackTrace();}

}

}

**catch**(SQLException e){e.printStackTrace();}

**return** check;

}

**public** **boolean** CheckHasExamQuestion(String course){

//Checks if the teacher has assigned questions to a given course.

**boolean** check = **false**;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT CourseName FROM NEW\_DATABASE.dbo.Questions WHERE CourseName = '"+course+"'";

ResultSet rs = stmt.executeQuery(sql);

**if**(rs.next())

check = **true**;

}

**catch**(SQLException e){e.printStackTrace();}

**return** check;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GETS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**public** **int** GetTeacherID(String email){

//Takes the ID number of the teacher from database who logged in.

**int** ID = 0;

**try** {

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT TeacherID FROM NEW\_DATABASE.dbo.AdminInfo WHERE Email = '"+email+"'");

**if**(rs.next())

ID = rs.getInt("TeacherID");

}

**catch**(SQLException e){e.printStackTrace();}

**return** ID;

}

**public** String GetName(String ID){

//Takes the name of the user(teacher or student) from database who logged in.

String name = **null**;

**try** {

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = **null**;

**if**(IsAdmin(ID))

sql = "SELECT Name FROM NEW\_DATABASE.dbo.AdminInfo WHERE Email = '"+ID+"'";

**else**

sql = "SELECT Name FROM NEW\_DATABASE.dbo.StudentInfo WHERE StudentID = '"+ID+"'";

ResultSet rs = stmt.executeQuery(sql);

**if**(rs.next())

name = rs.getString("Name");

}

**catch**(SQLException e){e.printStackTrace();}

**return** name;

}

**public** String GetSurName(String ID){

//Takes the surname of the user(teacher or student) from database who logged in..

String surname = **null**;

**try** {

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = **null**;

**if**(IsAdmin(ID))

sql = "SELECT Surname FROM NEW\_DATABASE.dbo.AdminInfo WHERE Email = '"+ID+"'";

**else**

sql = "SELECT Surname FROM NEW\_DATABASE.dbo.StudentInfo WHERE StudentID = '"+ID+"'";

ResultSet rs = stmt.executeQuery(sql);

**if**(rs.next())

surname = rs.getString("Surname");

}

**catch**(SQLException e){e.printStackTrace();}

**return** surname;

}

**public** CourseContainer GetExamCourse(String ID) {

//Takes the course information from the logged in student's student number.

Date current = **new** Date(System.*currentTimeMillis*());

CourseContainer co = **new** CourseContainer();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT E.CourseName,E.Date,E.StartTime,E.FinishTime,C.MaxQuestion\r\n" +

"FROM NEW\_DATABASE.dbo.ExamTime E, NEW\_DATABASE.dbo.Courses C\r\n" +

"WHERE E.StudentID = '"+ID+"' AND E.HasExam = 1 AND E.CourseName = C.CourseName ORDER BY E.Date,E.StartTime ASC";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next()) {

Date examDate;

Date finishTime;

**try** {

examDate = (Date) **new** SimpleDateFormat("yy-MM-dd HH:mm").parse(rs.getString("Date")+" "+rs.getString("StartTime"));

finishTime = (Date) **new** SimpleDateFormat("HH:mm").parse(rs.getString("FinishTime"));

**if**(IsExamInterval(current,examDate,finishTime)) {

co.setCourseName(rs.getString("CourseName"));

co.setDate(rs.getString("Date"));

co.setStartTime(rs.getString("StartTime"));

co.setFinishTime(rs.getString("FinishTime"));

co.setMaxQuestion(rs.getInt("MaxQuestion"));

**return** co;

}

} **catch** (ParseException e) {e.printStackTrace();}

}

}

**catch**(SQLException e) {e.printStackTrace();}

**return** co;

}

**public** ExamContainer GetQuestionInformations(String ID, String exam,**int** max){

//Takes the exam information from the student who looged in from student number.

ExamContainer examCont = **new** ExamContainer();

ArrayList<String> id = **new** ArrayList<>();

ArrayList<String> quest = **new** ArrayList<>();

ArrayList<String> type = **new** ArrayList<>();

ArrayList<String> a = **new** ArrayList<>();

ArrayList<String> b = **new** ArrayList<>();

ArrayList<String> c = **new** ArrayList<>();

ArrayList<String> d = **new** ArrayList<>();

ArrayList<String> t = **new** ArrayList<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT TOP "+max+" Q.QuestID,Q.QuestionType,Q.Quest,Q.Answer\_A,Q.Answer\_B,Q.Answer\_C,Q.Answer\_D,Q.TrueAns\r\n" +

"FROM NEW\_DATABASE.dbo.Questions Q, NEW\_DATABASE.dbo.ExamTime E\r\n" +

"WHERE E.StudentID = '"+ID+"' AND E.CourseName = Q.CourseName AND E.CourseName = '"+exam+"' "+

"ORDER BY NEWID()";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next()){

id.add(rs.getString("QuestID"));

type.add(rs.getString("QuestionType"));

quest.add(rs.getString("Quest"));

a.add(rs.getString("Answer\_A"));

b.add(rs.getString("Answer\_B"));

c.add(rs.getString("Answer\_C"));

d.add(rs.getString("Answer\_D"));

t.add(rs.getString("TrueAns"));

}

examCont.SetQuestion(quest);

examCont.SetQuestionType(type);

examCont.SetQuestionID(id);

examCont.SetAnswer\_A(a);

examCont.SetAnswer\_B(b);

examCont.SetAnswer\_C(c);

examCont.SetAnswer\_D(d);

examCont.SetAnswer\_T(t);

}

**catch**(SQLException e){e.printStackTrace();}

**return** examCont;

}

**public** HashMap<Integer,String> GetStudentAnswers(String ID,String examClass){

//Takes the answered questions from the student who logged in.

HashMap<Integer,String> quest = **new** HashMap<Integer,String>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT Answer,Sequence\r\n" +

"FROM NEW\_DATABASE.dbo.StudentAnswers\r\n" +

"WHERE StudentID = '"+ID+"' AND CourseName = '"+examClass+"'";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next())

{

**int** questId = rs.getInt("Sequence");

String Ans = rs.getString("Answer");

quest.put(questId, Ans);

}

}

**catch**(SQLException e){e.printStackTrace();}

**return** quest;

}

**public** List<String> GetExamResult(String ID){

//The student receives the exam results that he entered from the database via the profile page.

List<String> list = **new** ArrayList<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CourseName,Result FROM NEW\_DATABASE.dbo.StudentResults WHERE StudentID = '"+ID+"'");

**while**(rs.next())

{

String class\_name = rs.getString("CourseName");

**float** avg = GetExamAverage(class\_name);

list.add(class\_name);

list.add("Point:"+Float.*toString*(rs.getFloat("Result"))+" Average:"+Float.*toString*(avg));

}

}

**catch**(SQLException e) {e.printStackTrace();}

**return** list;

}

**public** **float** GetExamAverage(String class\_name){

//The student receives the average of his / her previous exam results from the database via the profile page.

**float** Avg = 0;

**int** size = 0;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement(ResultSet.***TYPE\_SCROLL\_INSENSITIVE***, ResultSet.***CONCUR\_READ\_ONLY***);

ResultSet rs = stmt.executeQuery("SELECT Result FROM NEW\_DATABASE.dbo.StudentResults WHERE CourseName = '"+class\_name+"'");

rs.last();

size = rs.getRow();

rs.beforeFirst();

**while**(rs.next())

Avg = Avg+ rs.getFloat("Result");

}

**catch**(SQLException e) {e.printStackTrace();}

**return** (Avg/(**float**)size);

}

**public** List<String> GetOldExamCourse(String ID) {

//Takes the name of the exams that student has entered before from the database.

List<String> list = **new** ArrayList<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT CourseName\r\n" +

"FROM NEW\_DATABASE.dbo.ExamTime\r\n" +

"WHERE StudentID = '"+ID+"' AND HasExam = 0 ORDER BY ExamID ASC";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next())

list.add(rs.getString("CourseName"));

}

**catch**(SQLException e) {e.printStackTrace();}

**return** list;

}

**public** List<String> GetOldExamQuestions(String name){

//Takes the questions of the exams that student has entered before from the database.

List<String> list = **new** ArrayList<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT QuestID,Quest,TrueAns\r\n" +

"FROM NEW\_DATABASE.dbo.Questions\r\n" +

"WHERE CourseName = '"+name+"' ORDER BY QuestID";

ResultSet rs = stmt.executeQuery(sql);

**while**(rs.next())

{

**int** ID = rs.getInt("QuestID");

String TrueAns = rs.getString("TrueAns");

list.add("Quest: "+rs.getString("Quest"));

list.add(" Answer: "+GetOldExamAnswer(TrueAns,ID));

}

}

**catch**(SQLException e) {e.printStackTrace();}

**return** list;

}

**public** String GetOldExamAnswer(String TrueAns,**int** ID){

//Receives the answers to the questions of the exams that student has previously entered from the database.

String answer = **null**;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "SELECT Answer\_" +TrueAns+" FROM NEW\_DATABASE.dbo.Questions WHERE QuestID = "+ID;

ResultSet rs = stmt.executeQuery(sql);

**if**(rs.next())

answer = rs.getString("Answer\_"+TrueAns);

}

**catch**(SQLException e) {e.printStackTrace();}

**return** answer;

}

**public** ArrayList<String> GetTeacherCourses(**int** ID){

//Takes the course names added by teacher from database.

ArrayList<String> list = **new** ArrayList<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CourseName FROM NEW\_DATABASE.dbo.Courses WHERE TeacherID = "+ID+"ORDER BY CourseID");

**while**(rs.next())

list.add(rs.getString("CourseName"));

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** list;

}

**public** HashMap<String,String> GetTeacherCourseDate(**int** ID){

//Takes the course dates adde by teacher from database.

HashMap<String,String> list = **new** HashMap<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CourseName, Date FROM NEW\_DATABASE.dbo.Courses WHERE TeacherID = "+ID+"ORDER BY CourseID");

**while**(rs.next())

list.put(rs.getString("CourseName"),rs.getString("Date"));

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** list;

}

**public** HashMap<String,String> GetTeacherCourseStart(**int** ID){

//Takes the course's start time added by teacher from database.

HashMap<String,String> list = **new** HashMap<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CourseName, StartTime FROM NEW\_DATABASE.dbo.Courses WHERE TeacherID = "+ID+"ORDER BY CourseID");

**while**(rs.next())

list.put(rs.getString("CourseName"),rs.getString("StartTime"));

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** list;

}

**public** HashMap<String,String> GetTeacherCourseEnd(**int** ID){

//Takes the course's end time added by teacher from database.

HashMap<String,String> list = **new** HashMap<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CourseName, FinishTime FROM NEW\_DATABASE.dbo.Courses WHERE TeacherID = "+ID+"ORDER BY CourseID");

**while**(rs.next())

list.put(rs.getString("CourseName"),rs.getString("FinishTime"));

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** list;

}

**public** **int** GetCourseID(**int** teacher\_id,String course\_name){

//Takes the ID number of courses added by teacher from database.

**int** ID = 0;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT CourseID FROM NEW\_DATABASE.dbo.Courses WHERE TeacherID = "+teacher\_id+

"AND CourseName = '"+course\_name+"'");

**if**(rs.next())

ID = rs.getInt("CourseID");

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** ID;

}

**public** ArrayList<String> GetNoExamStudents(**int** t\_id,String course){

//Takes the students who are not registered for the student registration for the course added by teacher from the database.

ArrayList<String> students = **new** ArrayList<>();

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT StudentID, Name,Surname FROM NEW\_DATABASE.dbo.StudentInfo WHERE "

+ "TeacherID = "+t\_id+" AND StudentID NOT IN (SELECT StudentID FROM NEW\_DATABASE.dbo.ExamTime WHERE CourseName = '"+

course+"')");

**while**(rs.next())

students.add(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** students;

}

**public** ArrayList<String> GetTeacherStudents(**int** id){

ArrayList<String> students = **new** ArrayList<>();

//Takes the student information added by teacher from database.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT StudentID,Name,Surname FROM NEW\_DATABASE.dbo.StudentInfo WHERE "+"TeacherID = "+id);

**while**(rs.next())

students.add(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** students;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*SETS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

**public** **void** SetStudentAnswers(HashMap <Integer,String> map, ArrayList<String> QuestID, String ID,String examClass){

//Saves the answers of the exam on database.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

//HashMap<Integer,String> old\_answers = new HashMap<Integer,String>();

//old\_answers = GetStudentAnswers(ID,examClass);

**for**(**int** i = 0; i< map.size(); i++ )

{

**if**(map.containsKey(i))

{

stmt.executeUpdate("INSERT INTO NEW\_DATABASE.dbo.StudentAnswers (QuestID,CourseName,Sequence,StudentID,Answer)"+

"VALUES('"+QuestID.get(i)+"','"+examClass+"',"+i+",'"+ID+"','"+map.get(i)+"')");

System.***out***.println(examClass+":"+i+":"+ID+":"+map.get(i));

}

**else**

**continue**;

}

}

**catch**(SQLException e){e.printStackTrace();}

}

**public** **void** SetExamFinish(String ID,String class\_name){

//Edits the student information whose exam's are finished on database.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

stmt.executeUpdate("UPDATE NEW\_DATABASE.dbo.ExamTime SET HasExam = 0 WHERE StudentID = '"+ID+"' AND CourseName = '"+class\_name+"'");

}

**catch**(SQLException e) {e.printStackTrace();}

}

**public** **void** SetExamResult(String ID, String class\_name, **float** result){

//Adds the student points to database.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

stmt.executeUpdate("INSERT INTO NEW\_DATABASE.dbo.StudentResults VALUES('"+ID+"',"+result+",'"+class\_name+"')");

}

**catch**(SQLException e) {e.printStackTrace(); }

}

**public** **void** UpdatePassword(String ID, String new\_pass){

//Edits the user (teacher or student) who wants to renew the password, in the database according to the specified user name and new password.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

**if**(IsAdmin(ID))

stmt.executeUpdate("UPDATE NEW\_DATABASE.dbo.AdminInfo SET Password = '"+new\_pass+"' WHERE Email = '"+ID+"'");

**else**

stmt.executeUpdate("UPDATE NEW\_DATABASE.dbo.StudentInfo SET Password = '"+new\_pass+"' WHERE StudentID = '"+ID+"'");

}

**catch**(SQLException e) {e.printStackTrace(); }

}

**public** **void** SetNewUser(String email,String password,String name,String surname){

//Adds the signedup teacher infromation to database.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

stmt.executeUpdate("INSERT INTO NEW\_DATABASE.dbo.AdminInfo (Email,Password,Name,Surname) VALUES('"+email+"','"+password+"','"+name+"','"+surname+"')");

}

**catch**(SQLException e) {e.printStackTrace(); }

}

**public** **boolean** SetNewStudent(String ID,String name,String surname, String password,**int** TeacherID){

//Add the information of new student added by teacher to database.

**boolean** check = **false**;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

**int** i = stmt.executeUpdate("INSERT INTO NEW\_DATABASE.dbo.StudentInfo (StudentID,Name,Surname,Password,TeacherID) VALUES('"+ID+"','"+name+"','"+surname+"','"+password+"',"+TeacherID+")");

**if**(i == 1)

check = **true**;

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** check;

}

**public** **boolean** SetNewCourse(String class\_name,String date, String start,String finish,String max\_quest, **int** AdminID){

//Adds the new course information which is added by teacher to database.

**boolean** check = **false**;

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

**int** i = stmt.executeUpdate("INSERT INTO NEW\_DATABASE.dbo.Courses (CourseName,Date,StartTime,FinishTime,MaxQuestion,TeacherID) VALUES('"+class\_name+"','"+date+"','"+start+"','"+finish+"',"+max\_quest+","+AdminID+")");

**if**(i == 1)

check = **true**;

}

**catch**(SQLException e) {e.printStackTrace(); }

**return** check;

}

**public** **boolean** SetNewQuestion(**int** ID,String course\_name,ArrayList<String> quest){

//Adds the new lesson question information added by teacher to database.

**boolean** check = **false**;

**int** course\_id = GetCourseID(ID,course\_name);

**if**(course\_id != 0)

{

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

stmt.executeUpdate("INSERT INTO NEW\_DATABASE.dbo.Questions (CourseID,CourseName,Quest,Answer\_A,Answer\_B,Answer\_C,Answer\_D,TrueAns,QuestionType)"+

" VALUES("+course\_id+",'"+course\_name+"','"+

quest.get(0)+"','"+quest.get(1)+"','"+quest.get(2)+"','"+quest.get(3)+"','"+quest.get(4)+"','"+quest.get(5)+"','"+quest.get(6)+"')");

check = **true**;

}

**catch**(SQLException e) {e.printStackTrace(); }

}

**return** check;

}

**public** **void** SetNoExamStudents(String course, ArrayList<String> stu\_id, String start, String end,String date){

//Adds the student exam assignment information added by teacher to database.

**try**

{

conn = DriverManager.*getConnection*(dbURL, user, pass);

Statement stmt = conn.createStatement();

String sql = "INSERT INTO NEW\_DATABASE.dbo.ExamTime (StudentID,HasExam,CourseName,StartTime,FinishTime,Date) VALUES ";

**for**(**int** i = 0; i<stu\_id.size(); i++)

{

sql = sql + "('"+stu\_id.get(i)+"','1','"+course+"',CAST('"+start+"' AS time),CAST('"+end+"' AS time),CAST('"+date+"' AS date))";

**if**(i != stu\_id.size()-1)

sql = sql + ",";

}

stmt.executeUpdate(sql);

}

**catch**(SQLException e) {e.printStackTrace(); }}}

// THE CLASS WHICH CONTAINS COURSE NAME, TIME ,ETC.

**public** **class** CourseContainer {

**private** String courseName,startTime,finishTime,date;

**private** **int** max\_question;

**public** CourseContainer()

{

courseName = **null**;

startTime = **null**;

finishTime = **null**;

date = **null**;

max\_question = 0;

}

///////// SETS

**public** **void** setCourseName(String courseName) { **this**.courseName = courseName; }

**public** **void** setStartTime(String startTime) { **this**.startTime = startTime; }

**public** **void** setFinishTime(String finishTime) {**this**.finishTime = finishTime; }

**public** **void** setDate(String date) { **this**.date = date; }

**public** **void** setMaxQuestion(**int** max\_question) { **this**.max\_question = max\_question;}

//////// GETS

**public** String getCourseName() { **return** **this**.courseName; }

**public** String getStartTime() { **return** **this**.startTime; }

**public** String getFinishTime() { **return** **this**.finishTime; }

**public** String getDate() { **return** **this**.date; }

**public** **int** getMaxQuestion() { **return** **this**.max\_question; }

/////// FUNCTIONS

**public** **void** Clear() {

courseName = **null**;

startTime = **null**;

finishTime = **null**;

date = **null**;

}

}

// THE CLASS WHICH CONTAINS EXAM QUESTION AND ANSWERS

**public** **class** ExamContainer {

**private** ArrayList<String> questions,answer\_a,answer\_b,answer\_c,answer\_d,answer\_true,question\_type,questions\_id;

**public** ExamContainer()

{

questions = **new** ArrayList<>();

question\_type = **new** ArrayList<>();

questions\_id = **new** ArrayList<>();

answer\_a = **new** ArrayList<>();

answer\_b = **new** ArrayList<>();

answer\_c = **new** ArrayList<>();

answer\_d = **new** ArrayList<>();

answer\_true = **new** ArrayList<>();

}

**public** **void** SetQuestion(ArrayList<String> que) { questions = que;}

**public** **void** SetQuestionType(ArrayList<String> type) {question\_type = type;}

**public** **void** SetQuestionID(ArrayList<String> id) {questions\_id = id;}

**public** **void** SetAnswer\_A(ArrayList<String> ans) { answer\_a = ans;}

**public** **void** SetAnswer\_B(ArrayList<String> ans) { answer\_b = ans;}

**public** **void** SetAnswer\_C(ArrayList<String> ans) { answer\_c = ans;}

**public** **void** SetAnswer\_D(ArrayList<String> ans) { answer\_d = ans;}

**public** **void** SetAnswer\_T(ArrayList<String> ans) { answer\_true = ans;}

**public** String GetQuestions(**int** index){**return** questions.get(index);}

**public** ArrayList<String> GetQuestions() {**return** questions;}

**public** String GetQuestionType(**int** index){**return** question\_type.get(index);}

**public** ArrayList<String> GetQuestionType(){**return** question\_type;}

**public** String GetQuestionID(**int** index) {**return** questions\_id.get(index);}

**public** ArrayList<String> GetQuestionID() {**return** questions\_id;}

**public** String GetAnswer\_A(**int** index){**return** answer\_a.get(index);}

**public** String GetAnswer\_B(**int** index){**return** answer\_b.get(index);}

**public** String GetAnswer\_C(**int** index){**return** answer\_c.get(index);}

**public** String GetAnswer\_D(**int** index){**return** answer\_d.get(index);}

**public** String GetAnswer\_T(**int** index){**return** answer\_true.get(index);}

/// FUNCTION

**public** **void** Clear() {

questions.clear();

question\_type.clear();

questions\_id.clear();

answer\_a.clear();

answer\_b.clear();

answer\_c.clear();

answer\_d.clear();

answer\_true.clear();

}

}

// THE CLASS WHICH CONTAINS ALL STUDENT INFORMATION

**public** **class** StudentContainer {

**private** **static** **final** String ***MULTIPLE*** = "multiple";

**private** **static** **final** String ***TEXT*** = "text";

**private** **static** **int** *last\_question*;

**private** **static** String *student\_id*,*student\_name*,*student\_surname*;

**private** **static** **boolean** *has\_exam*;

**private** HashMap<Integer,String> student\_ans;

**private** ExamContainer exam;

**private** CourseContainer course;

**public** StudentContainer(){

exam = **new** ExamContainer();

course = **new** CourseContainer();

student\_ans = **new** HashMap<Integer,String>();

}

/////////// SETS

**public** **void** setExamContainer(ExamContainer exam){**this**.exam = exam;}

**public** **void** setCourseContainer(CourseContainer course) {**this**.course = course;}

**public** **void** SetStudentID(String ID){*student\_id* = ID;}

**public** **void** SetStudentName(String name){*student\_name* = name;}

**public** **void** SetStudentSurname(String surname){*student\_surname* = surname;}

**public** **void** SetLastQuestionNum(**int** last){*last\_question* = last;}

**public** **void** SetHasExam(**boolean** has){*has\_exam* = has;}

**public** **void** SetStudentAnswers(HashMap<Integer,String> map){student\_ans = map;}

**public** **void** ReplaceStudentAnswers(**int** Index,String Value){student\_ans.replace(Index, Value);}

**public** **void** PutStudentAnswers(**int** Index, String Value){student\_ans.put(Index, Value);}

/////////// GETS

**public** String GetStudentID(){**return** *student\_id*;}

**public** String GetStudentName(){**return** *student\_name*;}

**public** String GetStudentSurname(){**return** *student\_surname*;}

**public** String GetExamLesson(){**return** course.getCourseName();}

**public** String GetExamStartTime() {**return** course.getStartTime();}

**public** String GetFinishTime() {**return** course.getFinishTime();}

**public** String GetExamDate() {**return** course.getDate();}

**public** **int** GetLastQuestionNum(){**return** *last\_question*;}

**public** **boolean** HasExam(){**return** *has\_exam*;}

**public** ArrayList<String> GetQuestions(){**return** exam.GetQuestions();}

**public** String GetQuestions(**int** index){**return** exam.GetQuestions(index);}

**public** **int** GetQuestSize(){**return** exam.GetQuestions().size();}

**public** ArrayList<String> GetQuestionID(){**return** exam.GetQuestionID();}

**public** String GetQuestionID(**int** index){**return** exam.GetQuestionID(index);}

**public** String GetAnswer\_A(**int** index){**return** exam.GetAnswer\_A(index);}

**public** String GetAnswer\_B(**int** index){**return** exam.GetAnswer\_B(index);}

**public** String GetAnswer\_C(**int** index){**return** exam.GetAnswer\_C(index);}

**public** String GetAnswer\_D(**int** index){**return** exam.GetAnswer\_D(index);}

**public** String GetAnswer\_T(**int** index){**return** exam.GetAnswer\_T(index);}

**public** ArrayList<String> GetQuestionType(){**return** exam.GetQuestionType();}

**public** String GetQuestionType(**int** index){**return** exam.GetQuestionType(index);}

**public** HashMap<Integer,String> GetStudentAnswers(){**return** student\_ans;}

**public** String GetStudentAnswers(**int** Index){**return** student\_ans.get(Index);}

**public** **boolean** IsStudentAnswerContains(**int** Index){**return** student\_ans.containsKey(Index);}

/////////// FUNCTION

**public** **boolean** HasAnotherExam(){**return** **new** Data().CheckExam(*student\_id*);}

**public** **float** CalculateResult() {

**float** result = 0, point\_per\_ans = 0;

String stu\_ans,true\_ans;

point\_per\_ans = (**float**) (100.0/(**float**)GetQuestSize());

**for**(**int** i = 0; i<GetQuestSize(); i++) {

stu\_ans = GetStudentAnswers(i+1);

true\_ans = GetAnswer\_T(i);

**if**(GetQuestionType(i).equals(***MULTIPLE***)) {

**if**(true\_ans.equals(stu\_ans))

result = result + point\_per\_ans;

}

**else** **if**(GetQuestionType(i).equals(***TEXT***)) {

**if**(GetAnswer\_A(i).equals(stu\_ans))

result = result + point\_per\_ans;

}

**else**

**continue**;

}

**return** result;

}

**public** **void** ClearExamContainer() {exam.Clear();}

**public** **void** ClearCourseContainer() {course.Clear();}

}

// THE CLASS WHİCH CONTAINS ALL TEACHER INFORMATION

**public** **class** TeacherContainer {

**private** String email,name,surname;

**private** **int** ID;

ArrayList<String> courses;

HashMap<String,String> course\_start;

HashMap<String,String> course\_end;

HashMap<String,String> course\_date;

**public** TeacherContainer() {

courses = **new** ArrayList<>();

course\_start = **new** HashMap<>();

course\_end = **new** HashMap<>();

course\_date = **new** HashMap<>();

}

**public** **int** GetAdminID()

{

**return** ID;

}

**public** **void** SetAdminID(**int** newID)

{

ID = newID;

}

**public** String GetAdminEmail()

{

**return** email;

}

**public** **void** SetAdminEmail(String newEmail)

{

email = newEmail;

}

**public** String GetAdminName()

{

**return** name;

}

**public** **void** SetAdminName(String newName)

{

name = newName;

}

**public** String GetAdminSurname()

{

**return** surname;

}

**public** **void** SetAdminSurname(String newSurname)

{

surname = newSurname;

}

**public** ArrayList<String> GetCourses()

{

**return** courses;

}

**public** **void** SetCourses(ArrayList<String> new\_courses)

{

courses = new\_courses;

}

**public** **void** SetCourseDate(HashMap<String, String> date)

{

course\_date = date;

}

**public** **void** SetCourseStart(HashMap<String, String> start)

{

course\_start = start;

}

**public** **void** SetCourseFinish(HashMap<String, String> end)

{

course\_end = end;

}

**public** **void** SetNewCourse(String course,String date, String start, String end)

{

courses.add(course);

course\_date.put(course, date);

course\_start.put(course,start);

course\_end.put(course,end);

}

**public** String GetCourseStart(String course)

{

**return** course\_start.get(course);

}

**public** String GetCourseEnd(String course)

{

**return** course\_end.get(course);

}

**public** String GetCourseDate(String course)

{

**return** course\_date.get(course);

}

}

// THE CLASS WHICH ABORTS THE EXAM IN SESSION

@WebServlet("/abortExam")

**public** **class** AbortExam **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** AbortExam() {

**super**();

}

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

response.getWriter().append("Served at: ").append(request.getContextPath());

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login")!= **null** && session.getAttribute("user\_type") == "Student") {

StudentContainer stu = (StudentContainer)session.getAttribute("login");

BufferedReader reader = req.getReader();

String line = reader.readLine();

System.***out***.println("Line: "+line);

**if**(line != **null**) {

@SuppressWarnings("unchecked")

HashMap<String,String> dataSet = **new** Gson().fromJson(line, HashMap.**class**);

**if**(dataSet.containsKey("checked") && dataSet.get("checked").equals("abortExam")) {

ExamContainer ex = **new** ExamContainer();

CourseContainer co = **new** CourseContainer();

String class\_name = stu.GetExamLesson();

String ID = stu.GetStudentID();

**new** Data().SetExamFinish(ID, class\_name);

**new** Data().SetStudentAnswers(stu.GetStudentAnswers(),stu.GetQuestionID(), ID, class\_name);

**float** result = stu.CalculateResult();

**new** Data().SetExamResult(ID, class\_name, result);

stu.ClearExamContainer();

stu.ClearCourseContainer();

**if**(stu.HasAnotherExam())

{

co = **new** Data().GetExamCourse(ID);

ex = **new** Data().GetQuestionInformations(ID,co.getCourseName(),co.getMaxQuestion());

stu.SetHasExam(**true**);

stu.SetStudentAnswers(**new** Data().GetStudentAnswers(ID, co.getCourseName()));

stu.setExamContainer(ex);

stu.setCourseContainer(co);

stu.SetLastQuestionNum(0);

}

**else**

stu.SetHasExam(**false**);

session.setAttribute("login", stu);

**return**;}}}}}

// THE CLASS WHICH ADDS COURSE DURING A SESSION

@WebServlet("/add\_course")

**public** **class** AddCourse **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@SuppressWarnings("deprecation")

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

TeacherContainer ad = (TeacherContainer)session.getAttribute("login");

PrintWriter out = res.getWriter();

**int** ID = ad.GetAdminID();

String class\_name = req.getParameter("course");

String \_date = req.getParameter("date");

String start\_time = req.getParameter("start\_time");

String finish\_time = req.getParameter("finish\_time");

String max\_question = req.getParameter("max\_question");

System.***out***.println("Course Name: "+class\_name+" Date: "+\_date+" Start: "+start\_time+" Finish: "+finish\_time);

Date start;

Date finish;

Date date;

Date current = **new** Date(System.*currentTimeMillis*());

**if**(class\_name == **null** || \_date == **null** || start\_time == **null** || finish\_time == **null** || max\_question == **null**){

out.println("<script type=\"text/javascript\">");

out.println("alert('Please fill all blanks.');");

out.println("location='profile.jsp';");

out.println("</script>");

**return**;

}

**try** {

start = (Date) **new** SimpleDateFormat("HH:mm").parse(start\_time);

finish = (Date) **new** SimpleDateFormat("HH:mm").parse(finish\_time);

date = (Date) **new** SimpleDateFormat("yy-MM-dd HH:mm").parse(\_date+" "+start\_time);

**if**(finish.compareTo(start) <= 0){

out.println("<script type=\"text/javascript\">");

out.println("alert('Start time cannot bigger than finish time.');");

out.println("location='profile.jsp';");

out.println("</script>");

**return**;

}

**else** **if**(date.compareTo(current) <= -1)

{

System.***out***.println("Current: "+current.~~getDate~~()+" Date: "+date.~~getDate~~());

out.println("<script type=\"text/javascript\">");

out.println("alert('You cannot set passed date.');");

out.println("location='profile.jsp';");

out.println("</script>");

**return**;

}

} **catch** (ParseException e) {e.printStackTrace();}

**if**(**new** Data().SetNewCourse(class\_name,\_date,start\_time,finish\_time,max\_question,ID))

{

ad.SetNewCourse(class\_name,\_date,start\_time,finish\_time);

session.setAttribute("login", ad);

out.println("<script type=\"text/javascript\">");

out.println("alert('The course has been successfully added.');");

out.println("location='profile.jsp';");

out.println("</script>");

}

**else**

{

out.println("<script type=\"text/javascript\">");

out.println("alert('The course has already been created.');");

out.println("location='profile.jsp';");

out.println("</script>");

}

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

doGet(req, res);

**else**

res.sendRedirect("/Web/index.jsp");

}

}

// THE CLASS WHICH ADDS QUESTIONS DURING A SESSION

@WebServlet("/add\_question")

**public** **class** AddQuestion **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

PrintWriter out = res.getWriter();

out.println("<script type=\"text/javascript\">");

out.println("alert('The question has been successfully added.');");

out.println("location='profile.jsp';");

out.println("</script>");

}

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

ArrayList<String> list = **new** ArrayList<>();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

String type;

type = req.getParameter("question\_type");

**if**(type.contains("multiple"))

{

list.add(req.getParameter("quest"));

list.add(req.getParameter("Ans\_A"));

list.add(req.getParameter("Ans\_B"));

list.add(req.getParameter("Ans\_C"));

list.add(req.getParameter("Ans\_D"));

list.add(req.getParameter("true\_ans"));

}

**else** **if**( type.contains("text"))

{

list.add(req.getParameter("quest"));

list.add(req.getParameter("TrueAns"));

list.add("");

list.add("");

list.add("");

list.add("A");

}

list.add(type);

TeacherContainer ad = (TeacherContainer) session.getAttribute("login");

**int** t\_id = ad.GetAdminID();

String course = req.getParameter("course");

**if**(**new** Data().SetNewQuestion(t\_id,course,list))

doGet(req,res);}}}

// THE CLASS WHICH ADDS STUDENTS DURING A SESSION

WebServlet("/add\_student")

**public** **class** AddStudent **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

TeacherContainer ad = (TeacherContainer) session.getAttribute("login");

ArrayList<String> list = **new** ArrayList<String>();

**int** id = ad.GetAdminID();

list = **new** Data().GetTeacherStudents(id);

String json = **new** Gson().toJson(list);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

req.setCharacterEncoding("ISO-8859-9");

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

TeacherContainer ad = (TeacherContainer) session.getAttribute("login");

PrintWriter out = res.getWriter();

**int** T\_ID = ad.GetAdminID();

String S\_ID = req.getParameter("ID");

String name = req.getParameter("name");

String surname = req.getParameter("surname");

String pass = req.getParameter("pass");

**if**(**new** Data().SetNewStudent(S\_ID, name, surname, pass, T\_ID))

{

out.println("<script type=\"text/javascript\">");

out.println("alert('The student has been successfully added.');");

out.println("location='profile.jsp';");

out.println("</script>");

}

**else**

{

out.println("<script type=\"text/javascript\">");

out.println("alert('Failed.This id has been used.');");

out.println("location='profile.jsp';");

out.println("</script>");

}

}

**else**

res.sendRedirect("/Web/index.jsp");

}

}

// THE CLASS WHICH ASSINGS COURSES TO STUDENTS DURING A SESSION

@WebServlet("/assign\_course")

**public** **class** AssignCourse **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** **static** ArrayList<String> *stu\_id* = **new** ArrayList<>();

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(!*stu\_id*.isEmpty())

*stu\_id*.clear();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

String course = req.getParameter("val");

TeacherContainer ad = (TeacherContainer) session.getAttribute("login");

**int** t\_id = ad.GetAdminID();

ArrayList<String> list = **new** Data().GetNoExamStudents(t\_id,course);

**if**(list.size() > 0)

{

**for**(**int** i = 0; i< list.size(); i++)

{

String id = list.get(i).substring(0, 12);

*stu\_id*.add(id);

}

String json = **new** Gson().toJson(list);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

req.setCharacterEncoding("ISO-8859-9");

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

TeacherContainer ad = (TeacherContainer) session.getAttribute("login");

ArrayList<String> list = **new** ArrayList<>();

PrintWriter out = res.getWriter();

String course = req.getParameter("course");

String[] stu = req.getParameterValues("stu\_id");

**for**(**int** i = 0; i<stu.length; i++)

list.add(*stu\_id*.get(Integer.*parseInt*(stu[i])));

**if**(**new** Data().CheckHasExamQuestion(course))

{

String start = ad.GetCourseStart(course);

String end = ad.GetCourseEnd(course);

String date = ad.GetCourseDate(course);

**new** Data().SetNoExamStudents(course,list,start,end,date);

out.println("<script type=\"text/javascript\">");

out.println("alert('Student assignment is successful.');");

out.println("location='profile.jsp';");

out.println("</script>");

}

**else**

{

out.println("<script type=\"text/javascript\">");

out.println("alert('Add questions before assigning students to this course.');");

out.println("location='profile.jsp';");

out.println("</script>");}} }}

// THE CLASS WHICH CHANGES PASSWORD DURING A SESSION

@WebServlet("/change\_password")

**public** **class** ChangePassword **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

HttpSession session = request.getSession(**true**);

**if**(session.getAttribute("login") != **null**)

{

String pass = request.getParameter("pass");

String pass\_confirm = request.getParameter("pass\_confirm");

**if**(!pass.equals(pass\_confirm))

{

request.setAttribute("TextValue","Alert");

response.setCharacterEncoding("ISO-8859-9");

RequestDispatcher reDis = request.getRequestDispatcher("/change\_password.jsp");

reDis.forward(request, response);

}

**else**

{

String ID = **null**;

**if**(session.getAttribute("user\_type") == "Admin")

{

TeacherContainer ad = (TeacherContainer) session.getAttribute("login");

ID = ad.GetAdminEmail();

}

**else**

{

StudentContainer ex = (StudentContainer) session.getAttribute("login");

ID = ex.GetStudentID();

}

**new** Data().UpdatePassword(ID, pass\_confirm);

request.removeAttribute("TextValue");

response.sendRedirect("/Web/profile.jsp");

}

}

**else**

response.sendRedirect("/Web/index.jsp");

}

}

// THE CLASS WHICH FINISHES THE EXAM DURING A SESSION

WebServlet("/FinishExam")

**public** **class** FinishExam **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

PrintWriter out = res.getWriter();

**if**(session.getAttribute("login")!= **null** && session.getAttribute("user\_type") == "Student")

{

StudentContainer stu = (StudentContainer)session.getAttribute("login");

Data Ch = **new** Data();

ExamContainer ex = **new** ExamContainer();

CourseContainer co = **new** CourseContainer();

String class\_name = stu.GetExamLesson();

String ID = stu.GetStudentID();

Ch.SetExamFinish(ID, class\_name);

Ch.SetStudentAnswers(stu.GetStudentAnswers(),stu.GetQuestionID(), ID, class\_name);

**float** result = stu.CalculateResult();

Ch.SetExamResult(ID, class\_name, result);

stu.ClearExamContainer();

stu.ClearCourseContainer();

**if**(stu.HasAnotherExam())

{

co = **new** Data().GetExamCourse(ID);

ex = **new** Data().GetQuestionInformations(ID,co.getCourseName(),co.getMaxQuestion());

stu.SetHasExam(**true**);

stu.SetStudentAnswers(**new** Data().GetStudentAnswers(ID, co.getCourseName()));

stu.setExamContainer(ex);

stu.setCourseContainer(co);

stu.SetLastQuestionNum(0);

}

**else**

stu.SetHasExam(**false**);

session.setAttribute("login", stu);

out.println("SUCCESS");

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login")!= **null** && session.getAttribute("user\_type") == "Student")

{

String check = req.getParameter("checked");

StudentContainer ex = (StudentContainer)session.getAttribute("login");

**int** last = ex.GetLastQuestionNum()+1;

System.***out***.println("Answer is : "+check+" QuestNumber : "+last);

**if**(check != **null**)

{

**if**(!ex.IsStudentAnswerContains(last))

ex.PutStudentAnswers(last, check);

**else** **if**(ex.IsStudentAnswerContains(last) && !ex.GetStudentAnswers(last).contains(check))

ex.ReplaceStudentAnswers(last, check);

session.setAttribute("login", ex);

}

//doGet(req,res);

}

}

}

// THE CLASS WHICH GETS COURSES DURING A SESSION

@WebServlet("/GetCourses")

**public** **class** GetCourses **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Admin")

{

TeacherContainer ad = (TeacherContainer)session.getAttribute("login");

ArrayList<String> list = ad.GetCourses();

String json = **new** Gson().toJson(list);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

doGet(req, res);

}

}

// THE CLASS WHICH GETS NEXT QUESTION ANSWERS DURING A SESSION

@WebServlet("/get\_next\_quest")

**public** **class** GetNextQuestAnswer **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

HashMap<String,String> hash = **new** HashMap<>();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Student")

{

StudentContainer ex = (StudentContainer)session.getAttribute("login");

**int** last = ex.GetLastQuestionNum() +1;

**if**(last< ex.GetQuestSize())

{

hash.put("status", "valid");

hash.put("question", ex.GetQuestions(last));

hash.put("ans\_A",ex.GetAnswer\_A(last));

hash.put("ans\_B",ex.GetAnswer\_B(last));

hash.put("ans\_C",ex.GetAnswer\_C(last));

hash.put("ans\_D",ex.GetAnswer\_D(last));

hash.put("type",ex.GetQuestionType(last));

hash.put("number",String.*valueOf*(last+1));

**if**(ex.IsStudentAnswerContains(last+1))

hash.put("checked",ex.GetStudentAnswers(last+1));

**else**

hash.put("checked","false");

String json = **new** Gson().toJson(hash);

ex.SetLastQuestionNum(last);

session.setAttribute("login", ex);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

**else**

{

hash.put("status", "invalid");

String json = **new** Gson().toJson(hash);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Student")

{

String myJsonData = req.getParameter("checked");

StudentContainer ex = (StudentContainer)session.getAttribute("login");

**int** last = ex.GetLastQuestionNum()+1;

System.***out***.println("Answer is : "+myJsonData+" QuestNumber : "+last);

**if**(!ex.IsStudentAnswerContains(last) && !myJsonData.contains("false") && myJsonData != **null**)

ex.PutStudentAnswers(last, myJsonData);

**else** **if**(ex.IsStudentAnswerContains(last) && !ex.GetStudentAnswers(last).contains(myJsonData))

ex.ReplaceStudentAnswers(last, myJsonData);

session.setAttribute("login", ex);

}

**else**

res.sendRedirect("/Web/index.jsp");

} }

// THE CLASS WHICH SHOWS OLD QUESTIONS

@WebServlet("/GetOldQuestions")

**public** **class** GetOldQuestions **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

List<String> list = **new** ArrayList<>();

**if**(session.getAttribute("login") != **null**)

{

StudentContainer ex = (StudentContainer) session.getAttribute("login");

String ID = ex.GetStudentID();

list = **new** Data().GetOldExamCourse(ID);

String json = **new** Gson().toJson(list);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

String class\_name = req.getParameter("name");

HttpSession session = req.getSession();

List<String> list = **new** ArrayList<>();

**if**(session.getAttribute("login") != **null**)

{

list = **new** Data().GetOldExamQuestions(class\_name);

String json = **new** Gson().toJson(list);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

**else**

res.sendRedirect("/Web/index.jsp");

}

}

// THE CLASS WHICH GETS PREVIOUS QUSTION ANSWERS IN SESSION

@WebServlet("/get\_prev\_quest")

**public** **class** GetPrevQuestAnswer **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

HttpSession session = request.getSession(**true**);

HashMap<String,String> hash = **new** HashMap<>();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Student")

{

StudentContainer ex = (StudentContainer)session.getAttribute("login");

**int** last = ex.GetLastQuestionNum() - 1;

**if**(last >= 0)

{

hash.put("status", "valid");

hash.put("question", ex.GetQuestions(last));

hash.put("ans\_A",ex.GetAnswer\_A(last));

hash.put("ans\_B",ex.GetAnswer\_B(last));

hash.put("ans\_C",ex.GetAnswer\_C(last));

hash.put("ans\_D",ex.GetAnswer\_D(last));

hash.put("type",ex.GetQuestionType(last));

hash.put("number",String.*valueOf*(last+1));

**if**(ex.IsStudentAnswerContains(last+1))

hash.put("checked",ex.GetStudentAnswers(last+1));

**else**

hash.put("checked","false");

/\*list.add(ex.GetQuestions(last));

list.add(ex.GetAnswer\_A(last));

list.add(ex.GetAnswer\_B(last));

list.add(ex.GetAnswer\_C(last));

list.add(ex.GetAnswer\_D(last));

list.add(Integer.toString(last+1));

list.add(ex.GetQuestionType(last));

if(ex.IsStudentAnswerContains(last+1))

list.add(ex.GetStudentAnswers(last+1));

\*/

String json = **new** Gson().toJson(hash);

ex.SetLastQuestionNum(last);

session.setAttribute("login", ex);

response.setContentType("application/json");

response.setCharacterEncoding("UTF-8");

response.getWriter().write(json);

}

**else**

{

hash.put("status", "invalid");

String json = **new** Gson().toJson(hash);

response.setContentType("application/json");

response.setCharacterEncoding("UTF-8");

response.getWriter().write(json);

}

}

**else**

response.sendRedirect("Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

HttpSession session = request.getSession(**true**);

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Student")

{

String myJsonData = request.getParameter("checked");

StudentContainer ex = (StudentContainer)session.getAttribute("login");

**int** last = ex.GetLastQuestionNum()+1;

System.***out***.println("Answer is : "+myJsonData+" QuestNumber : "+last);

**if**(!ex.IsStudentAnswerContains(last) && !myJsonData.contains("false") && myJsonData != **null**)

ex.PutStudentAnswers(last, myJsonData);

**else** **if**(ex.IsStudentAnswerContains(last) && !ex.GetStudentAnswers(last).contains(myJsonData))

ex.ReplaceStudentAnswers(last, myJsonData);

session.setAttribute("login", ex);

}

**else**

response.sendRedirect("Web/index.jsp");

}

}

// THE CLASS WHICH GETS RESULTS OF AN EXAM IN SESSION

WebServlet("/GetResults")

**public** **class** GetResults **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException

{

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Student")

{

StudentContainer ex = (StudentContainer)session.getAttribute("login");

List<String> list = **new** Data().GetExamResult(ex.GetStudentID());

String json = **new** Gson().toJson(list);

res.setContentType("application/json");

res.setCharacterEncoding("UTF-8");

res.getWriter().write(json);

}

**else**

res.sendRedirect("Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

doGet(req, res);

}

}

// THE CLASS WHICH IS THE LOGS IN THE SESSION

**public** **class** LoginPage **extends** HttpServlet

{

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** **boolean** IsAdmin(String name)

{

**boolean** check = **false**;

Pattern VALID\_EMAIL\_ADDRESS\_REGEX =

Pattern.*compile*("^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$", Pattern.***CASE\_INSENSITIVE***);

Matcher m = VALID\_EMAIL\_ADDRESS\_REGEX.matcher(name);

**if**(m.find())

{

System.***out***.println("Admin : "+name);

check = **true**;

}

**return** check;

}

@Override

**public** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** IOException, ServletException

{

String name = req.getParameter("email");

HttpSession session = req.getSession(**true**);

//HttpSession session = request.getSession(); Gets existing sessions.

//session.setAttribute("Session\_name",object);Object(class,String,Integer,Float...)

//session.getAttribute("Session\_name")Object called.

//session.invalidate();Removes all existing sessions.

//HttpSession session;

**if**(session.getAttribute("login") == **null**)

{

**if**(IsAdmin(name))

{

session.setAttribute("user\_type", "Admin"); //Creating the session.

TeacherContainer ad = **new** TeacherContainer();

ad.SetAdminEmail(name);

//Taking information from database.

//I take the information and put them into TeacherContainer class then create object.

**int** ID = **new** Data().GetTeacherID(name);

ad.SetAdminID(ID);

ad.SetAdminName(**new** Data().GetName(name));

ad.SetAdminSurname(**new** Data().GetSurName(name));

ad.SetCourses(**new** Data().GetTeacherCourses(ID));

ad.SetCourseDate(**new** Data().GetTeacherCourseDate(ID));

ad.SetCourseStart(**new** Data().GetTeacherCourseStart(ID));

ad.SetCourseFinish(**new** Data().GetTeacherCourseEnd(ID));

//Saves the created session into class.

session.setAttribute("login", ad);

}

**else**

{

session.setAttribute("user\_type", "Student");

StudentContainer stu = **new** StudentContainer();

ExamContainer ex = **new** ExamContainer();

CourseContainer co = **new** CourseContainer();

stu.SetStudentName(**new** Data().GetName(name));

stu.SetStudentSurname(**new** Data().GetSurName(name));

stu.SetStudentID(name);

**if**(**new** Data().CheckExam(name))

{

co = **new** Data().GetExamCourse(name);

ex = **new** Data().GetQuestionInformations(name,co.getCourseName(),co.getMaxQuestion());

stu.SetHasExam(**true**);

stu.setExamContainer(ex);

stu.setCourseContainer(co);

stu.SetLastQuestionNum(0);

}

**else**

stu.SetHasExam(**false**);

session.setAttribute("login", stu);

}

}

res.sendRedirect("/Web/index.jsp");

}

@Override

**public** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** IOException, ServletException

{

String name = req.getParameter("email");

String password = req.getParameter("pass");

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login") == **null**)

{

System.***out***.println(name+":"+password);

**try** {

**if**(**new** Data().CheckUserNamePassword(name,password))

doGet(req,res);

**else**

{

**if**(IsAdmin(name))

res.sendRedirect("/Web/admin\_login.jsp");

**else**

res.sendRedirect("/Web/student\_login.jsp");

}

}

**catch** (SQLException e) {e.printStackTrace();}

}

**else**

res.sendRedirect("/Web/index.jsp");

}

}

// THE CLASS WHICH LOGS OUT FROM THE SESSION

**public** **class** LogOut **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Override

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession(**true**);

**if**(session.getAttribute("login") != **null** )

{

**if**(session.getAttribute("user\_type").equals("Admin"))

{

}

**else**

{

StudentContainer stu = (StudentContainer)session.getAttribute("login");

**if**(stu.HasExam())

{

String class\_name = stu.GetExamLesson();

String ID = stu.GetStudentID();

**new** Data().SetExamFinish(ID, class\_name);

**float** result = stu.CalculateResult();

**new** Data().SetExamResult(ID, class\_name, result);

}

}

session.invalidate();

}

res.sendRedirect("/Web/index.jsp");

}

@Override

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

doGet(request, response);

}}

// THE CLASS WHICH REFRESHES THE EXAM PAGE IN A SESSION

@WebServlet("/refresh")

**public** **class** RefreshExam **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**public** RefreshExam() {

**super**();

}

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login") != **null** && session.getAttribute("user\_type") == "Student")

doPost(req, res);

**else**

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

StudentContainer stu = (StudentContainer)session.getAttribute("login");

ExamContainer ex = **new** ExamContainer();

CourseContainer co = **new** CourseContainer();

String ID = stu.GetStudentID();

**if**(stu.HasAnotherExam())

{

co = **new** Data().GetExamCourse(ID);

ex = **new** Data().GetQuestionInformations(ID,co.getCourseName(),co.getMaxQuestion());

stu.SetHasExam(**true**);

stu.SetStudentAnswers(**new** Data().GetStudentAnswers(ID, co.getCourseName()));

stu.setExamContainer(ex);

stu.setCourseContainer(co);

stu.SetLastQuestionNum(0);

}

**else**

stu.SetHasExam(**false**);

session.setAttribute("login", stu);

res.sendRedirect("/Web/index.jsp");}}

// THE CLASS WHICH SIGNS YOU UP TO THE SYSTEM

@WebServlet("/sign\_up")

**public** **class** SignUp **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login")== **null**)

{

String email = req.getParameter("email");

String name = req.getParameter("name");

String surname = req.getParameter("surname");

String password = req.getParameter("pass");

**new** Data().SetNewUser(email,password,name,surname);

}

res.sendRedirect("/Web/index.jsp");

}

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse res) **throws** ServletException, IOException {

HttpSession session = req.getSession();

**if**(session.getAttribute("login")== **null**)

{

String email = req.getParameter("email");

**if**(**new** Data().CheckEmail(email))

doGet(req, res);

**else**

res.sendRedirect("/Web/sign\_up.jsp");

}

**else**

res.sendRedirect("/Web/index.jsp");

}

}