

A PROJECT REPORT
ON
CLASSROOM QUIZ

Submitted in partial fulfilment of the requirement For the award of degree of

Of

BACHELOR OF COMPUTER APPLICATIONS Of
BANGALORE UNIVERSITY

2018

By

Dharamraj Bhatt

(16NCSB7012)

Under the guidance of

Prof. Sailaja



DEPARTMENT OF COMPUTER APPLICATIONS
ST. FRANCIS DE SALES COLLEGE

BENGALURU-560100

DECLARATION

I, **DHARAMRAJ BHATT**, hereby declare that the Project Report titled “**CLASSROOM QUIZ**” submitted to Bangalore University, Bengaluru in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Applications is a report of work done by me under the supervision of **Prof. Sailaja** I also declare that this report any part of it has not been submitted to any other University/Institute for the award of any degree.

Signature of student

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and regards to my external guide **Mrs. Rhicha** Tiwari for her constant inspiration, supervision and invaluable guidance during the training. I would also like to thank **Mrs Jayshree** of Digitage Technologies for giving me such an opportunity to continue my training in Digitage Technologies and providing the facility. At last I would also like to extend my sincere gratitude to all my faculty members and specially **Prof. Sailaja** for giving their valuable suggestions.

Signature of Student

Sign of HOD

Sign of Internal Guide

ABSTRACT

Technology in the field of education is constantly evolving, growing and this increase will continually offer new and extraordinary advances in the field of education. People want efficient and less time consuming way of learning. Traditional way of learning and conducting quiz or examination was fully based on paper work which is wastage of time, effort money and so on. The online classroom quiz will simplify this task. classroom quiz will allow the users to participate for the particular quiz in the particular subject. it can also give the details of the quiz name, password for quiz subject and quizzes which are already taken. The use of the Web (World Wide Web) has had many positive effects on education. It overcomes time and space limitations in traditional schools. Teachers and students are now using the Web to access vast amounts of information and resources in the cyberspace. The main aim of classroom Quiz is to effectively estimate the candidate completely via a totally automated system which besides preserving time, offers swifter outcomes. Generally, pupil are provided with paper, pen etc for taking the test but the classroom Quiz doesn't require all these. Using this web based application program Teacher can upload quiz, it will be appear on student's dashboard. student can attend quiz and his score will be appear on teacher's dashboard by student's name.

SL.N O	CONTENTS		PAGE NO.
01.	INTRODUCTION		1-6
	1.1	Basic Introduction of Project	2
	1.2	Objective and Scope	2
	1.3	Tools and Technologies used	3
	1.4	Proposed Systems	6
	1.5	Limitations of Existing System	6
02.	SYSTEM ANALYSIS		7-14
	2.1	Preliminary Analysis & Information Gathering	8
	2.2	Input/Output	10
	2.3	System Design and Development	10
	2.4	System Requirements Specification	12
	2.5	Software Engineering Model Used	13
03.	SYSTM DESIGN		15-30
	3.1	Project Planning	16
	3.2	Modules	17

	3.3	Data Flow Diagram	18
	3.4	E-R Diagram	20
	3.5	Database Design	21
	3.6	Screen Shorts	23
04.	TESTING		30-34
	4.1	Introduction to Testing	31
	4.2	Types of Testing	31
05.	IMPLEMENTATION & MAINTAINANCE		35-88
06.	CONCLUSION		89-90
07.	FUTURE ENHANCEMENT		91-92
08.	REFERENCES		93-94

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1. Basic introduction of Project:

Recent advances in the Web have rapidly changed our life in various ways. These advances provide new ways for people to communicate on a global scale and assess vast amounts of information. The Web provides educators with opportunities to implement a range of new teaching and learning practices, which redefine classroom learning experiences. classroom quiz is a basic quiz app. There are mainly two actors:

1. **student**
2. **Teacher**

These are the functionality of the users-

1. Teacher

- i. Can create quiz after getting logged in!
- ii. Can enter subjects and enter question with its options and answer at the time of creating quiz.
- iii. Question for each quiz can be added according to teacher's desire

2. Student

- i. Can search quiz according to their interest.
- ii. Quiz can start by clicking start button.
- iii. After completing all questions, result will be displayed Automatically in student and teacher's dashboard.

1.2. Objective and Scope:

The classroom quiz to provides a common platform to connect student and teacher online. It is developed for student in schools, colleges and institutes to participate in quiz to enhance their knowledge. thus the main objective of the project is to develop an interactive python web based Django application to conduct quiz sessions in the class for different topics.

Scope:

This project has a wide scope as it is better than the manual tests. Following are some of its advantages

- i. Both educational organizations and corporate companies can employ this Quiz.
- ii. It, being an application based on web may be taken at any place or any time as the location is not considered.
- iii. The presence of the examiner is not required while the candidate is appearing for the test.
- iv. Easy way to test knowledge.
- v. User-friendly app for easy understanding
- vi. Can reduce usage of resources like paper.

1.3. Tools and Techniques

Python: Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly

fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

Django: Django is a free and open source web application framework written in Python. A framework is nothing more than a collection of modules that make development easier. They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch.

This is how websites - even simple ones designed by a single person - can still include advanced functionality like authentication support, management and admin panels, contact forms, comment boxes, file upload support, and more. In other words, if you were creating a website from scratch you would need to develop these components yourself. By using a framework instead, these components are already built, you just need to configure them properly to match your site.

The official project site describes Django as "a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source."

Django offers a big collection of modules which you can use in your own projects. Primarily, frameworks exist to save developers a lot of wasted time and headaches and Django is no different.

You might also be interested in learning that Django was created with front-end developers in mind. "Django's template language is designed to feel comfortable and easy-to-learn to those used to working with HTML, like designers and front-end developers. But it is also flexible and highly extensible, allowing developers to augment the template language as needed."

SQLite-SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource. SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional. SQLite database is integrated with the application that accesses the database. The applications interact with the SQLite database read and write directly from the database files stored on disk. SQLite uses dynamic types for tables. It means you can store any value in any column, regardless of the data type. SQLite allows a single database connection to access multiple database files simultaneously. This brings many nice features like joining tables in different databases or copying data between databases in a single command. SQLite is capable of creating in-memory databases which are very fast to work with.

HTML and CSS-HTML is the language for describing the structure of Web pages. HTML gives authors the means to:

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links, at the click of a button.
- Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

With HTML, authors describe the structure of pages using *markup*. The *elements* of the language label pieces of content such as “paragraph,” “list,” “table,” and so on.

CSS is the language for describing the presentation of Web pages, including colors, layout, and fonts. It allows one to adapt the presentation to different types of devices, such as large screens, small screens, or printers. CSS is independent of HTML and can be used with any XML-based markup language. The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to different environments. This is referred to as the separation of structure (or: content) from presentation.

1.4. Proposed System:

- i. The development of this new system contains the following activities, which try to automate the entire process keeping in the view of database integration approach.
- ii. This system maintains user's personal and contact details.
- iii. User friendliness is provided in the application with various controls provided by system rich user interface.
- iv. This system makes the overall project management much easier and flexible.
- v. Various classes have been used for maintain the details of all the users and catalog.
- vi. Authentication is provided for this application only registered users can access.
- vii. Result features is provided for both teacher and student.
- viii. The system provides facilities to track the all activities of faculties and students.
- ix. System provides facility to prepare online question paper, and has facility for online exam also.
- x. System also provides facility to upload and view information.
- xi. This system is providing accessibility control to data with respect to users.

1.5 . Existing System:

- i. This existing system is not providing secure registration and profile management of all the users properly.
- ii. This manual system gives us very less security for saving data and some data may be lost due to mismanagement.
- iii. The system doesn't provides facility to track all the activities of student and faculties.
- iv. The system doesn't provide any functionalities to prepare online exam questions, online exam for students.

CHAPTER 2

SYSTEM ANALYSIS

2.1. Preliminary Analysis & Information Gathering

The first and foremost strategy for development of a project starts from the thought of designing a simple form that accepts order and product details for a small firm in which it is easy and convenient for taking order and maintaining customer details, there is a search engine, report generation and also editing options for placing orders and adding products. When it is approved by the organization and our project guide the first activity, i.e preliminary investigation begins. The activity has three parts:

I. Request Clarification

II. Feasibility study

III. Request approval

Request Clarification

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires. Here our project is basically meant for management of restaurant orders. In today's busy schedule man need everything should be provided in a ready made manner. So taking into consideration of the vastly use of net in day to day life, the corresponding development of the portal came into existence.

Feasibility Study

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

I. Operational Feasibility

II. Economic Feasibility

III. Technical Feasibility

Operational Feasibility:

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

Economic Feasibility:

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at any time. The Virtual Private Network is to be developed using the existing resources of the organization. So the project is economically feasible.

Technical Feasibility:

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQLite3 server and Web Logic Server are used to develop the system. The technical feasibility has been carried out. The system is technically

feasible for development and can be developed with the existing facility.

Request Approval

Not all request projects are desirable or feasible. Some organization receives so many project requests from client users that only few of them are pursued. However, those projects that are both feasible and desirable should be put into schedule. After a project request is approved, its cost, priority, completion time and personnel requirement is estimated and used to determine where to add it to any project list. Truly speaking, the approval of those above factors, development works can be launched.

2.3. SYSTEM DESIGN AND DEVELOPMENT

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things

- I. What data should be given as input?
- II. How the data should be arranged or coded?
- III. The dialog to guide the operating personnel in providing input.
- IV. Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

- i. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- ii. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry casier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- iii. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in mair of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making,

- i. Designing computer output should proceed in an organized, well thought out manner the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements
- ii. Select methods for presenting information.

- iii. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives

- ❖ Convey information about past activities, current status or projections of the future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.

2.4. SOFTWARE REQUIREMENTS

HARDWARE REQUIREMENTS

Processor : Intel i3 and above

System Bus : 64 bits

RAM : 4GB and above

HDD : 500GB and above

SOFTWARE REQUIREMENTS

Operating System : Microsoft windows 10 and above

Coding Language : Django ,HTML, CSS

Front End : Python

Back End : SQLite3

2.5 SOFTWARE ENGINEERING MODEL USED

SDLC METHODOLOGIES

This document plays a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system it means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article. "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

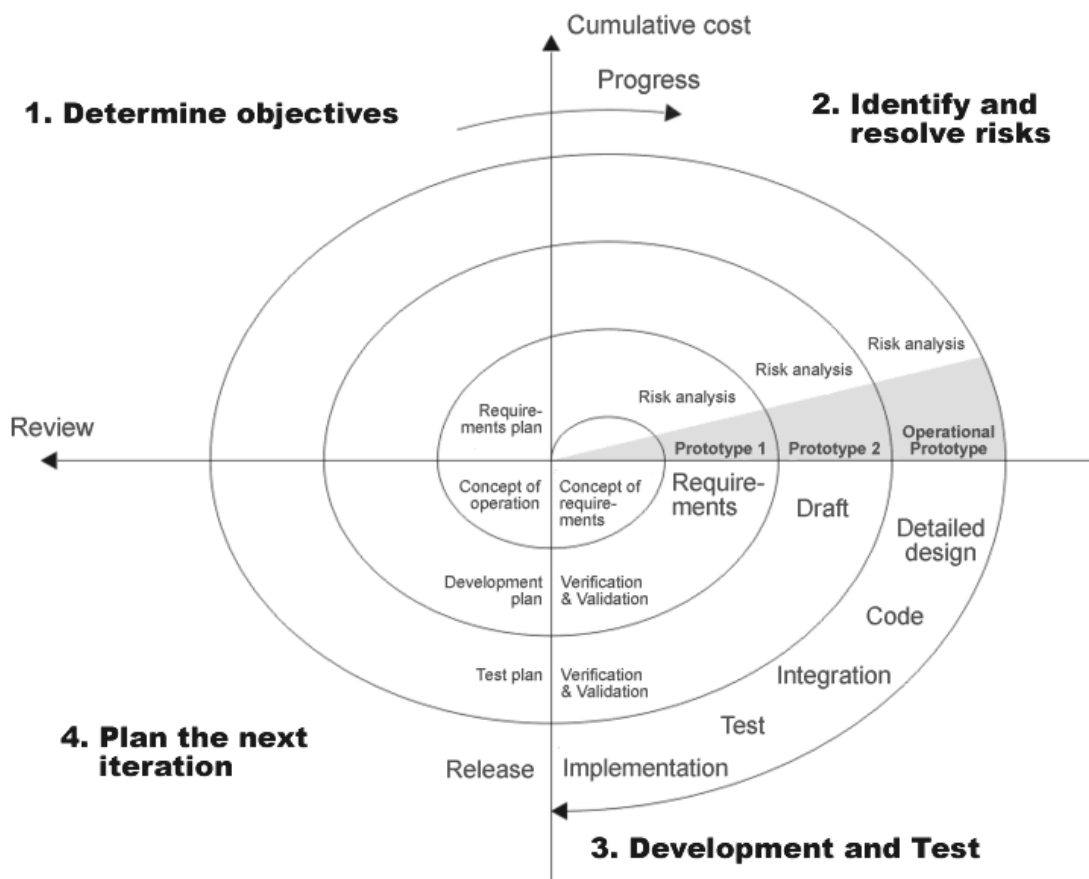
As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- i. The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- ii. A preliminary design is created for the new system.
- iii. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- iv. A second prototype is evolved by a fourfold procedure:
 1. Evaluating the first prototype in terms of its strengths, weakness, and risks
 2. Defining the requirements of the second prototype.
 3. Planning and designing the second prototype.
 4. Constructing and testing the second prototype.

- v. At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- vi. The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- vii. The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- viii. The final system is constructed, based on the refined prototype.
- ix. The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:



CHAPTER 3

SYSTEM DESIGN

3.1. Project Planning

Work Assigned	Due date	Description
Basic Introduction of Project, Objective and Scope, Tools and Technologies used, Proposed Systems, Limitations of Existing System,	16-01-2019	The project was chosen and analyzed to find its scope and limitations.
Preliminary Analysis & Information Gathering, Input/Output, System Design and Development, System Requirements Specification, Software Engineering Model Used	30-01-2019	Important information regarding the project was collected and requirements for the system were specified.
Project Planning, Modules,	11-02-2019	Project schedule table was drawn ,also database was collected and designed.

Data Flow Diagram, E-R Diagram, Database Design, Screen Shorts		
Introduction to Testing, Types of Testing	27-02-2019	Testing was implemented.
Implementation & Maintainance	05-03-2019	Coding of the project got completed
Conclusion	30-03-2019	Project conclusion was decided.
Future Enhancement	04-04-2019	Next level of the project was discussed.
References	11-04-2019	References of the project were note down.

3.2. Modules

In this project there are mainly 4 modules. They are

- i. Registration Module
- ii. Login Module
- iii. Student Module
- iv. Teacher Module

Registration Module:

In the registration module form the user has to register the bio data and a unique code is given to each user. Once the user is registered, they can participate in quiz. During

registration user details will be validated, if a user already exists, system will notify the user.

Login Module:

Through this module, registered users, can login. It is used for authenticating the users. After logging in, users can go quiz based on their choice. When the users clicks login button, system will check for the entered values. If both values matches, users should be able to login, else system should generate invalid user message.

Student Module:

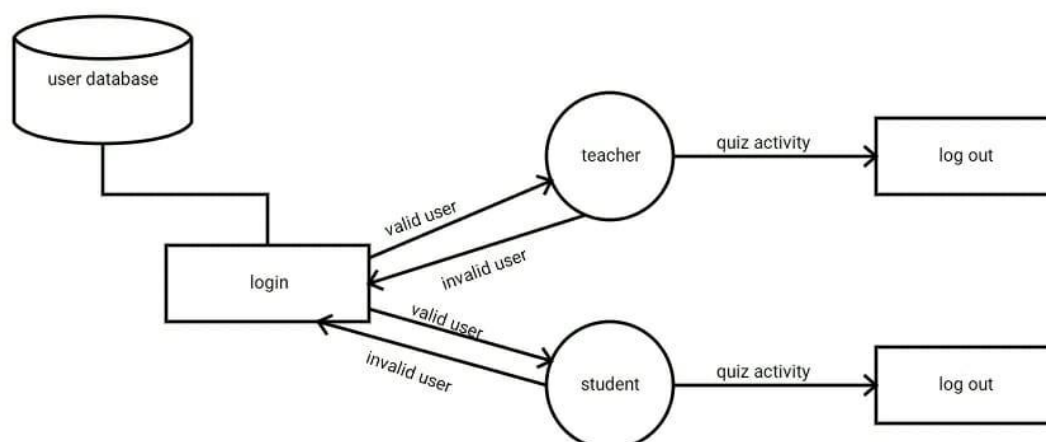
When the uses logs in and gets quiz start part then the users becomes a student. After start the quiz student can give the answer of the given questions and can also view the results

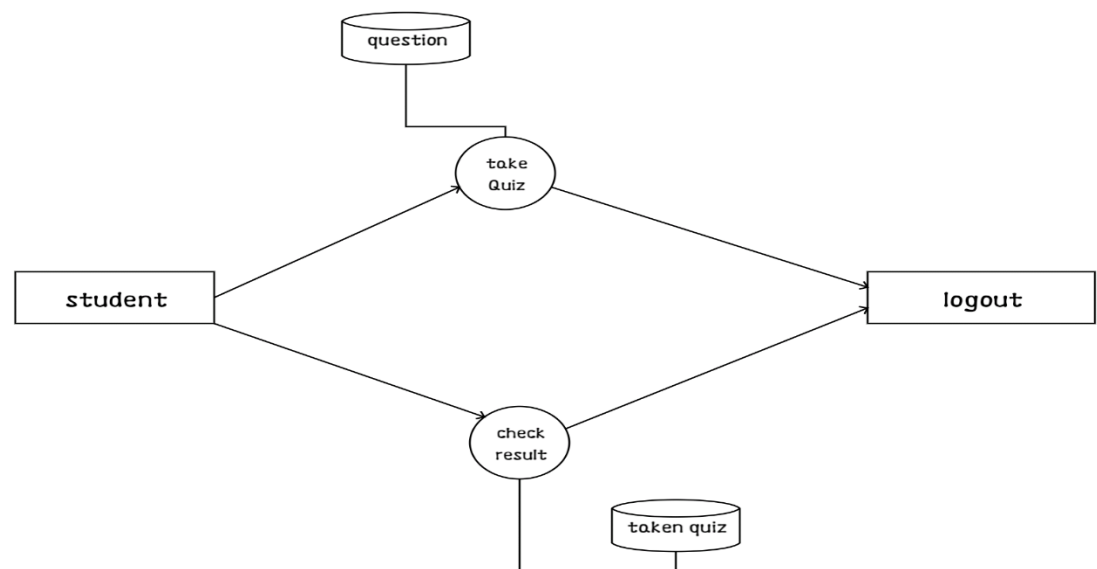
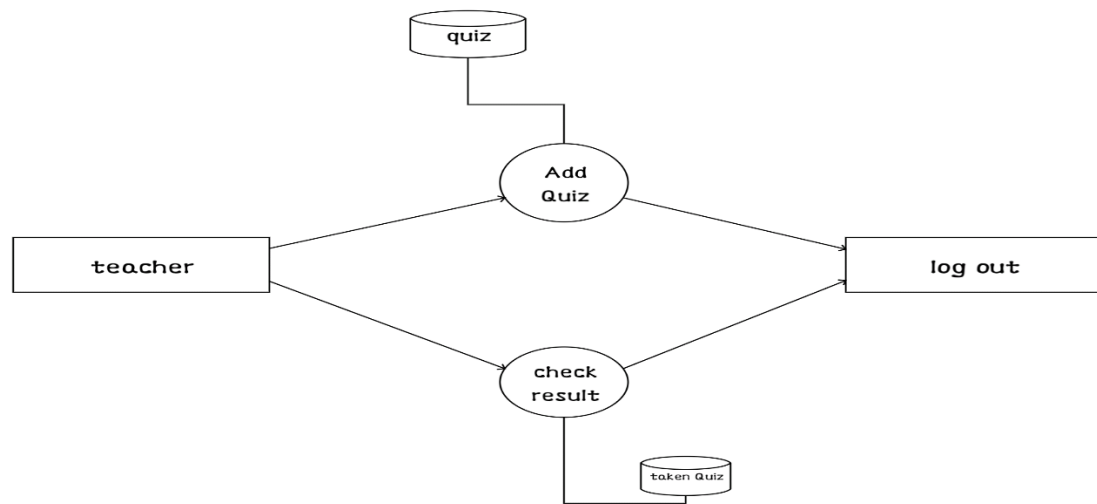
Teacher Module:

When the uses logs in and gets add quiz button the users becomes teacher for the particular quiz. Teacher can creates new quizzes and inside quiz teacher can insert questions and answer. teacher can not participate in a quiz but can see the result of students.

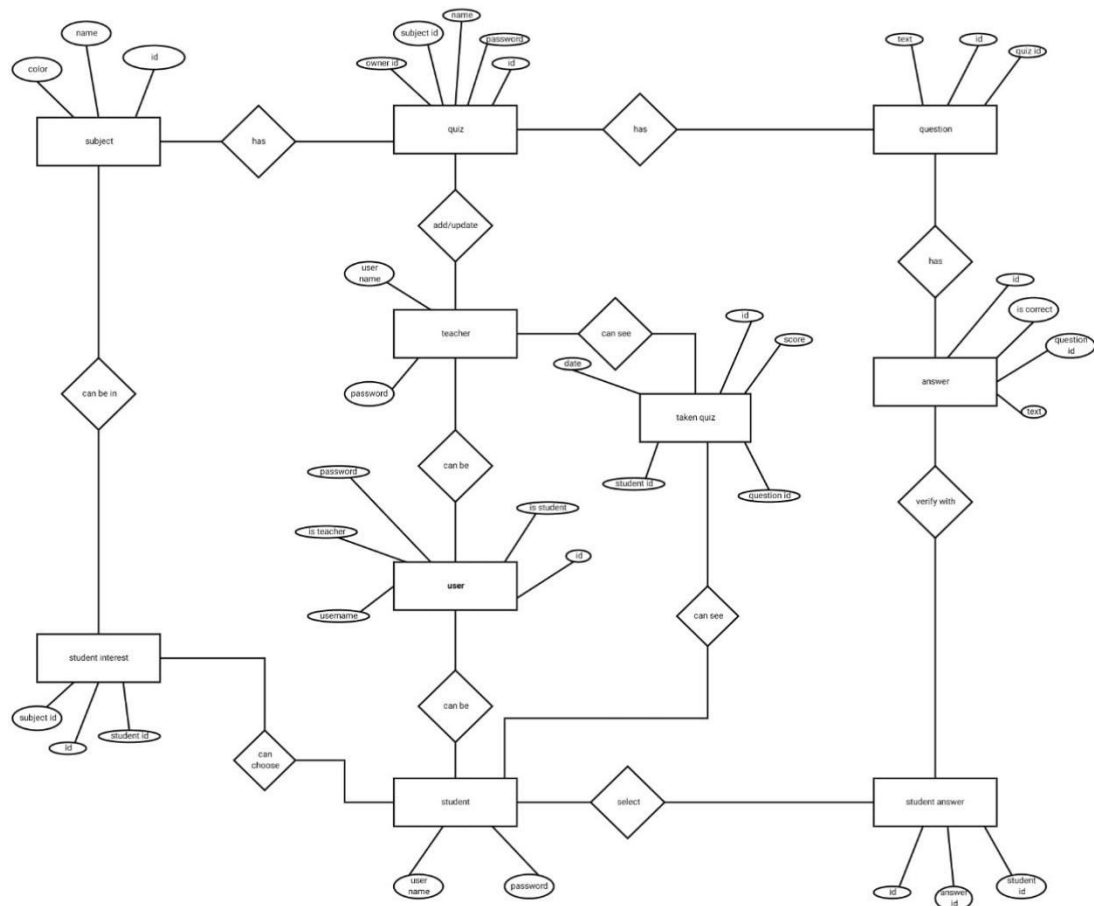
3.3. Data Flow Diagram

0 level DFD



1 level DFD

3.4. E-R Diagram



3.5. Database Design

User table

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
password	varchar(128)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
last_login	datetime	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_superuser	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
username	varchar(150)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		
first_name	varchar(30)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
last_name	varchar(150)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
email	varchar(254)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_staff	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_active	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
date_joined	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_student	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_teacher	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Quiz table

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
name	varchar(255)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
owner_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
subject_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
password	varchar(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Taken quiz

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
score	real	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
date	datetime	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
quiz_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
student_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Subject

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
name	varchar(30)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
color	varchar(7)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Answer

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
text	varchar(255)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
is_correct	bool	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
question_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Question

Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
text	varchar(255)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
quiz_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

Student interest

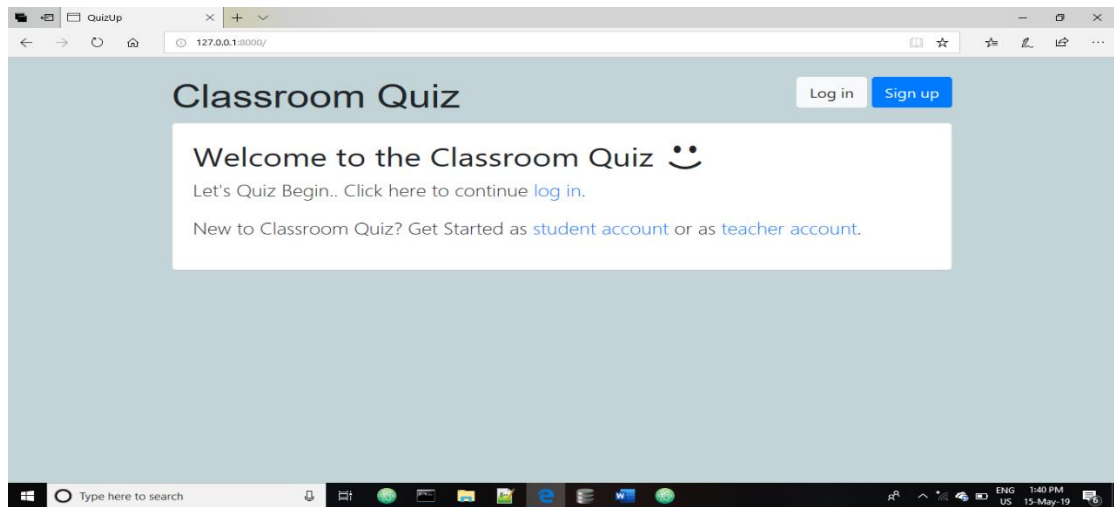
Name	Type	NN	PK	AI	U	Default
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
student_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
subject_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Student answer

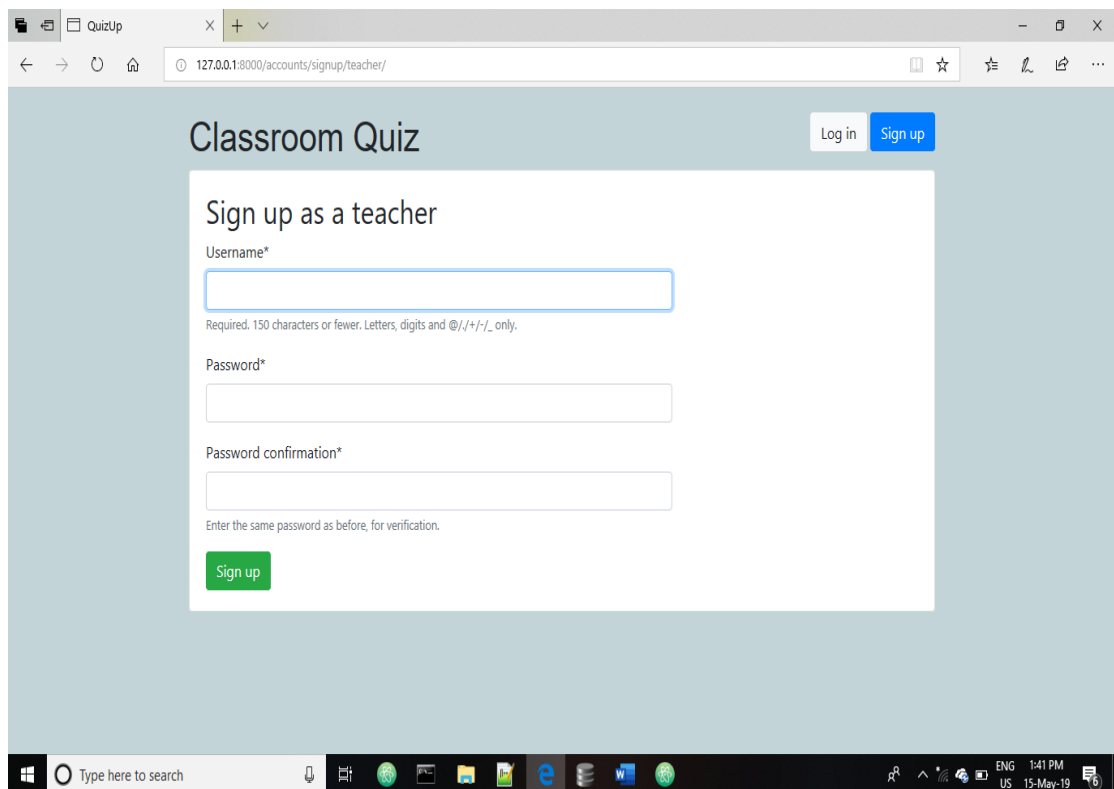
Name	Type	NN	PK	AI	U	Default	Check
id	integer	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
answer_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		
student_id	integer	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		

3.6. Screen Shorts

Home Page



Teacher registration form



Student registration form

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/accounts/signup/student/`. The page title is "Classroom Quiz". In the top right corner, there are two buttons: "Log in" and "Sign up". The main content area is a white box titled "Sign up as a student". It contains the following fields and options:

- Username***: A text input field. Below it, a note reads: "Required. 150 characters or fewer. Letters, digits and @/+/./- only."
- Password***: A text input field.
- Password confirmation***: A text input field. Below it, a note reads: "Enter the same password as before, for verification."
- Interests***: A list of checkboxes with labels: Arts, Computing, Math, Biology, and History.
- Sign up**: A green button at the bottom of the form.

The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates the language is ENG US and the time is 1:40 PM on 15-May-19.

Log in page

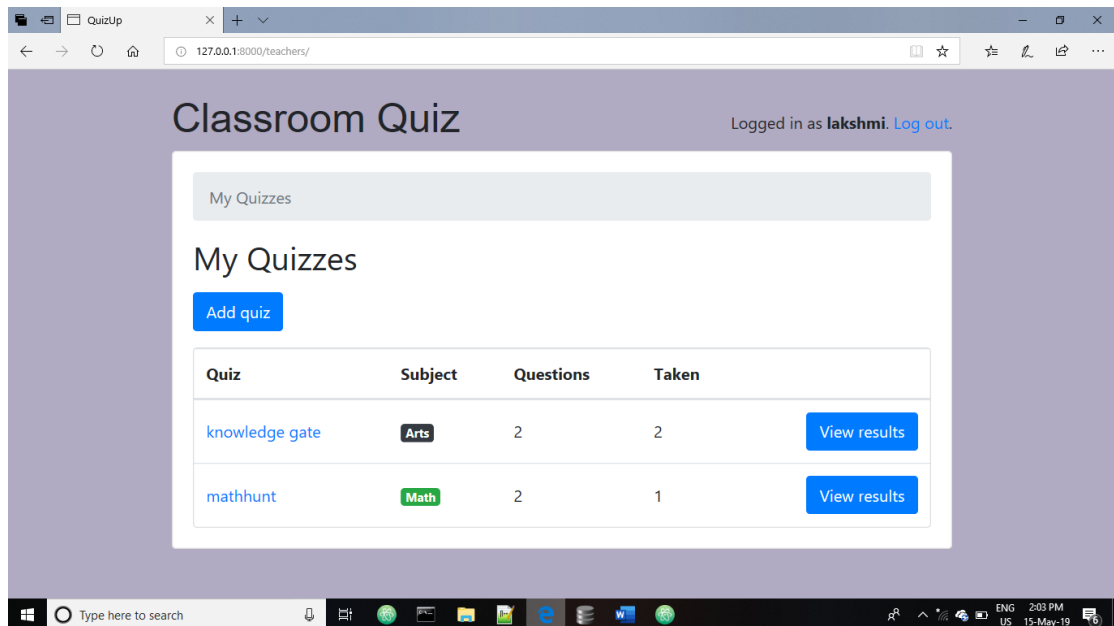
The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000/accounts/login/`. The page title is "Classroom Quiz". In the top right corner, there are two buttons: "Log in" and "Sign up". The main content area is a white box titled "Log in". It contains the following fields and a button:

- Username***: A text input field.
- Password***: A text input field.
- Log in**: A blue button at the bottom of the form.

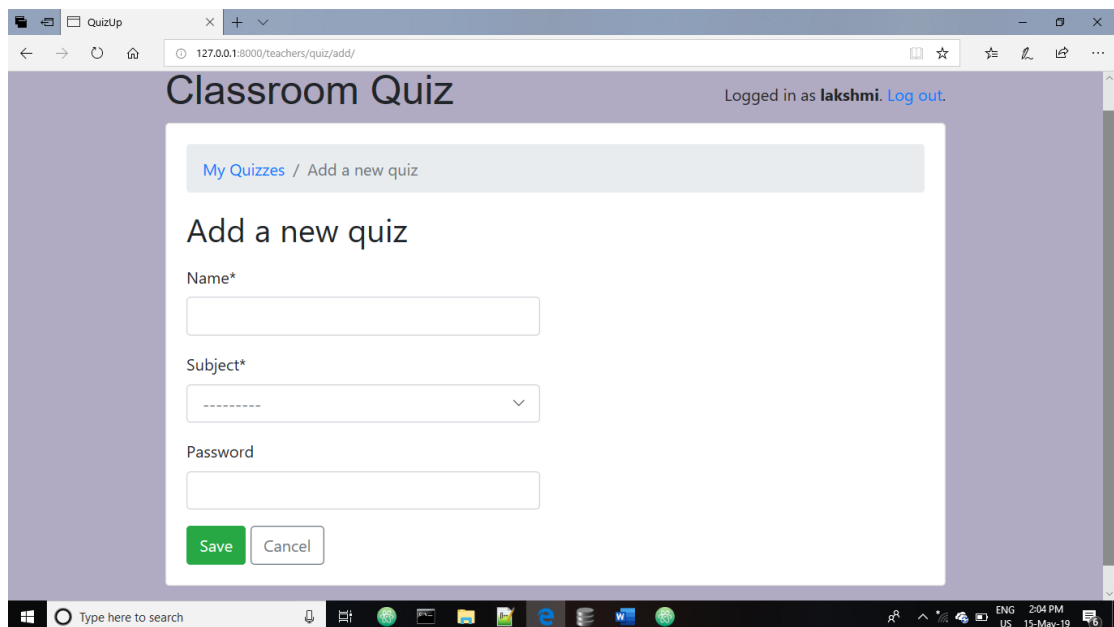
The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates the language is ENG US and the time is 1:59 PM on 15-May-19.

For teacher:

Created Quizzes



Add new quiz



Add new Quiz

The screenshot shows a web browser window with the URL `127.0.0.1:8000/teachers/quiz/8/question/add/`. The page title is "Classroom Quiz" and it shows the user is logged in as "lakshmi". The breadcrumb trail is "My Quizzes / class test / Add a new question". The main heading is "Add a new question". Below this, a text prompt says: "Add first the text of the question. In the next step you will be able to add the possible answers." There is a text input field labeled "Question*" and two buttons: "Save" (green) and "Cancel" (white with a grey border). The Windows taskbar at the bottom shows the search bar and various application icons.

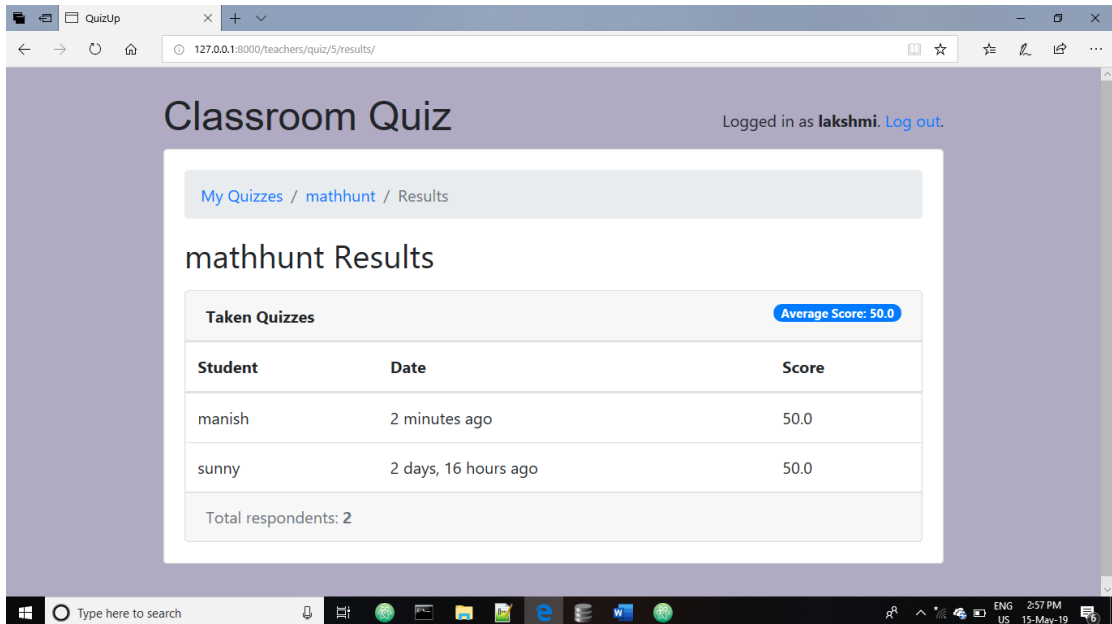
Add Answers

The screenshot shows the same web browser window, but the URL is `127.0.0.1:8000/teachers/quiz/8/question/13/`. The page title is "Classroom Quiz" and the user is logged in as "lakshmi". The breadcrumb trail is "My Quizzes / class test / what is bit?". The main heading is "Add Answers". A green notification box at the top says: "You may now add answers/options to the question." Below this, the question text "what is bit?" is shown in a text input field. There is a table with the following structure:

Answers	Correct?	Delete?
central processing unit	<input type="checkbox"/>	<input type="checkbox"/>
package	<input type="checkbox"/>	<input type="checkbox"/>
8 bit data	<input type="checkbox"/>	<input type="checkbox"/>
small unit of data	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table, a note states: "Your question may have at least 2 answers and maximum 10 answers. Select at least one correct answer." At the bottom, there are three buttons: "Save changes" (green), "Nevermind" (white with a grey border), and "Delete" (red).

Result of all student in particular quiz



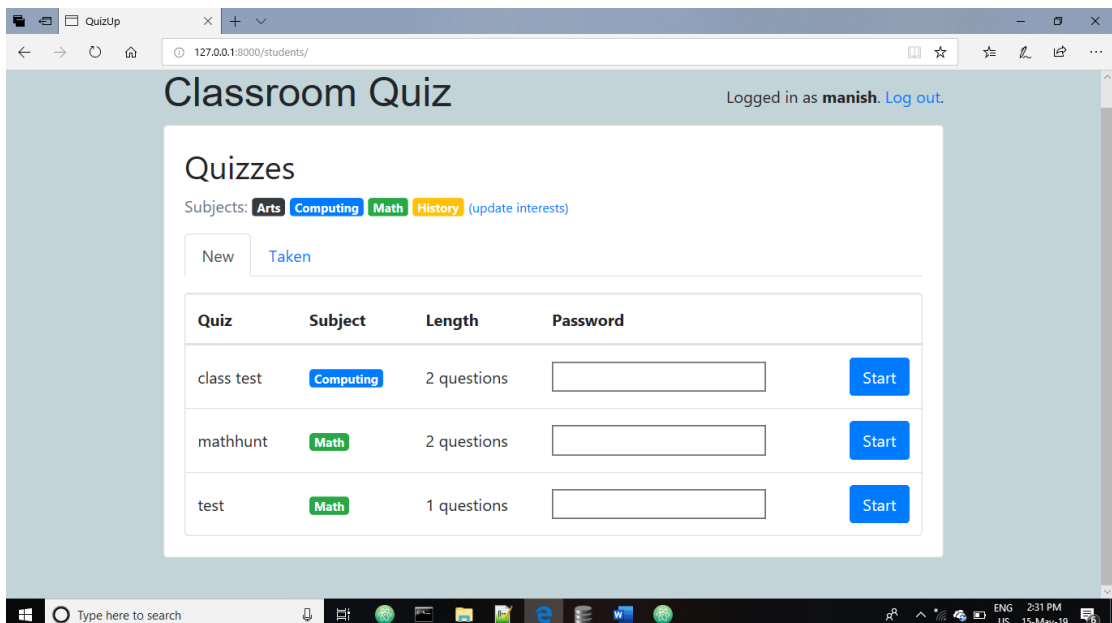
The screenshot shows the 'Classroom Quiz' interface for a user logged in as 'lakshmi'. The page displays the results for a quiz named 'mathhunt'. The breadcrumb trail is 'My Quizzes / mathhunt / Results'. The main heading is 'mathhunt Results'. Below this, there is a table titled 'Taken Quizzes' with an 'Average Score: 50.0' badge. The table has three columns: 'Student', 'Date', and 'Score'. It lists two students: 'manish' who took the quiz '2 minutes ago' with a score of '50.0', and 'sunny' who took it '2 days, 16 hours ago' with a score of '50.0'. At the bottom of the table, it states 'Total respondents: 2'.

Student	Date	Score
manish	2 minutes ago	50.0
sunny	2 days, 16 hours ago	50.0

Total respondents: 2

For student:

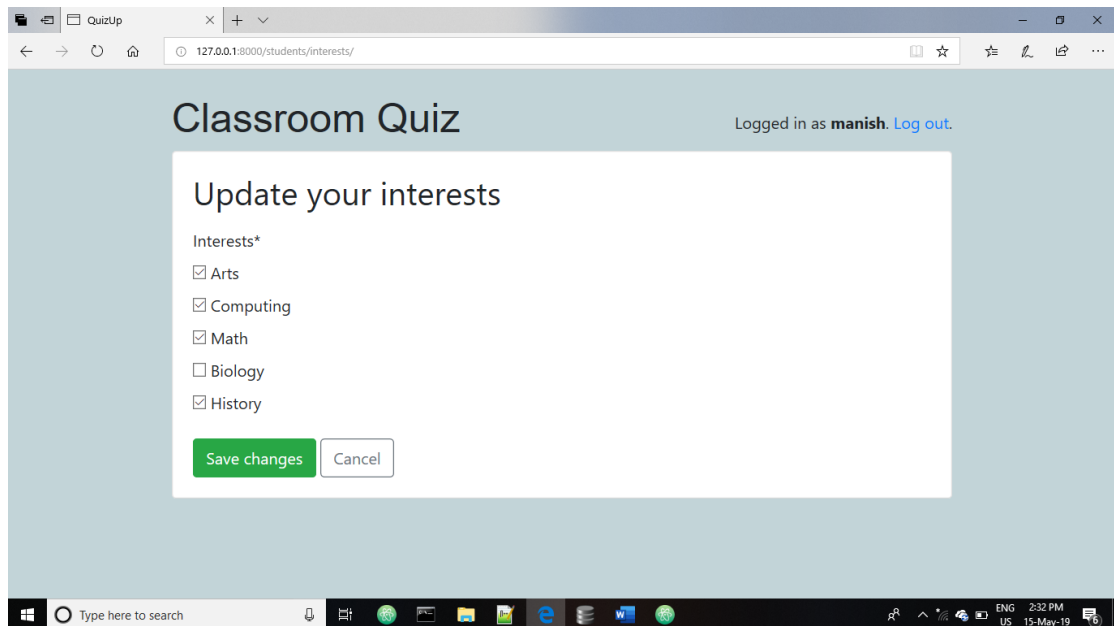
Total Quizes



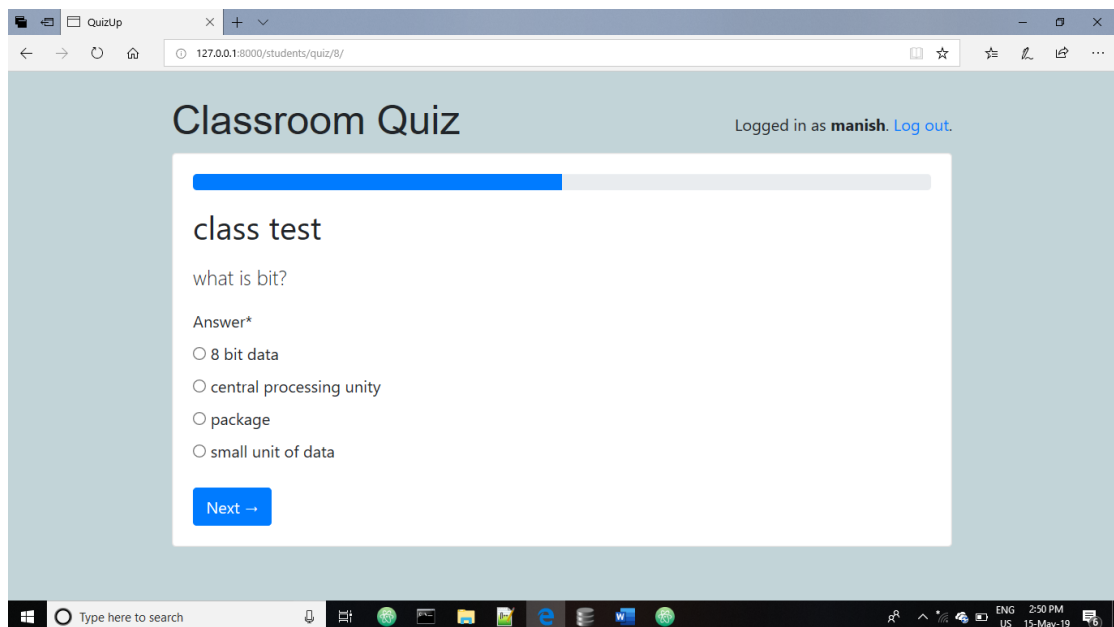
The screenshot shows the 'Classroom Quiz' interface for a user logged in as 'manish'. The page displays a list of quizzes under the heading 'Quizzes'. Below the heading, there are subject filters: 'Arts', 'Computing', 'Math', and 'History', with a link to '(update interests)'. There are two tabs: 'New' and 'Taken'. The 'Taken' tab is selected. Below the tabs is a table with columns: 'Quiz', 'Subject', 'Length', and 'Password'. The table lists three quizzes: 'class test' (Computing, 2 questions), 'mathhunt' (Math, 2 questions), and 'test' (Math, 1 question). Each row has a 'Start' button next to a password input field.

Quiz	Subject	Length	Password
class test	Computing	2 questions	<input type="password"/>
mathhunt	Math	2 questions	<input type="password"/>
test	Math	1 questions	<input type="password"/>

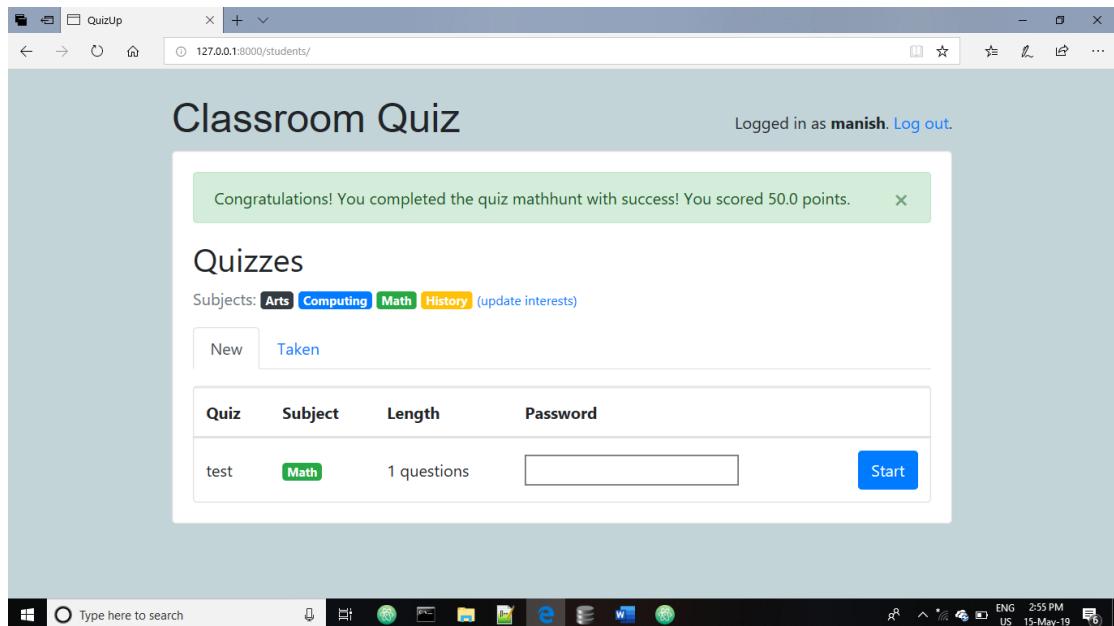
Update interest



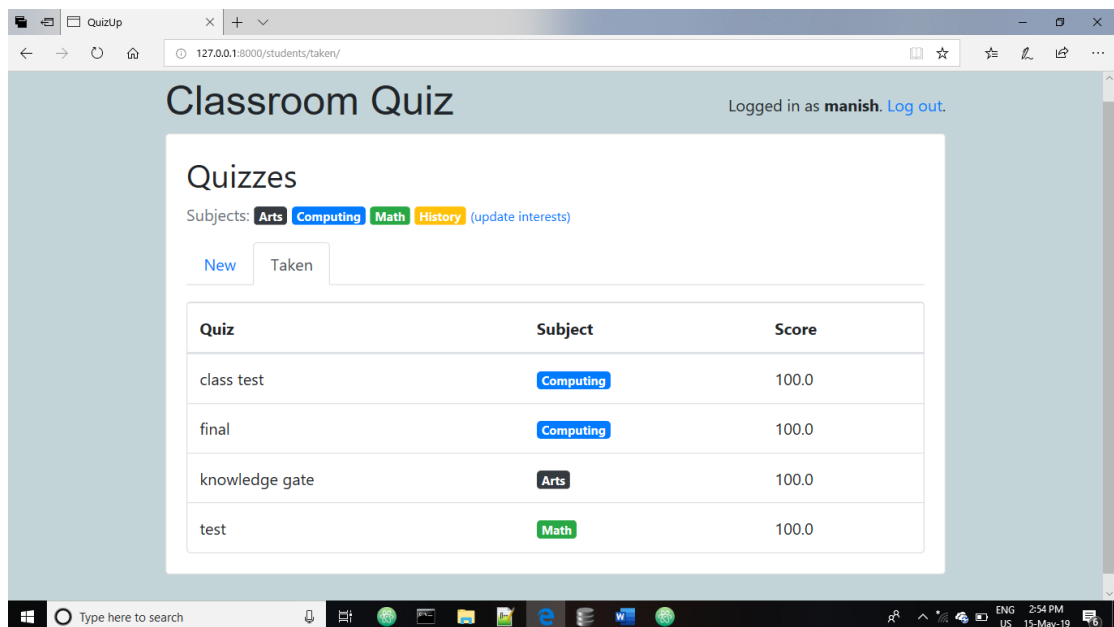
Quiz



Result of Quiz



Quizzes taken by student



CHAPTER 4

TESTING

4.1. INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

4.2. TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. This is the testing of individual software units of the application. It is done after the completion of an individual with integration. This structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements. System documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

invalid Input : identified classes of invalid input must be rejected

Functions : identified functions must be exercised

Output : identified classes of applications must be exercised.

Systems Procedures :interfacing system or procedures must be invoked

Organization and preparation of functional test is used on requirements key functions, or special test cases. In addition systematic coverage pertaining to identify Business process flows data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests must be written from a definitive source document, such as specification or requirements document such specification or requirements document is a test in which the software under test is treated as a black box you cannot “see” into it. The test provides inputs and responds to cut without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- i. All field entries must work properly.
- ii. Pages must be activated from the identified link.
- iii. The entry screen, messages and responses must not be delayed.

Features to be tested

- i. Verify that the entries are of the correct format.
- ii. No duplicate entries should be allowed.
- iii. All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental Integration testing of two or more integrated software components in a single platform to produce failures caused by

interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or one step up software applications at the company level interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 5

IMPLEMENTATION

&

MAINTAINANCE

CODING:

CLASSROOM/views/classroom.py

```
from django.shortcuts import redirect, render

from django.views.generic import TemplateView

class SignUpView(TemplateView):

    template_name = 'registration/signup.html'

    #homePage

    def home(request):

        if request.user.is_authenticated:

            if request.user.is_teacher:

                return redirect('teachers:quiz_change_list')

            else:

                return redirect('students:quiz_list')

        return render(request, 'classroom/home.html')
```

CLASSROOM/views/student.py

```
from django.contrib import messages

from django.contrib.auth import login

from django.contrib.auth.decorators import login_required

from django.db import transaction

from django.db.models import Count

from django.http import JsonResponse

from django.shortcuts import get_object_or_404, redirect, render
```

```
from django.urls import reverse_lazy

from django.utils.decorators import method_decorator

from django.views.generic import CreateView, ListView, UpdateView


from ..decorators import student_required

from ..forms import StudentInterestsForm, StudentSignUpForm, TakeQuizForm

from ..models import Quiz, Student, TakenQuiz, User


class StudentSignUpView(CreateView):

    model = User

    form_class = StudentSignUpForm

    template_name = 'registration/signup_form.html'

    def get_context_data(self, **kwargs):

        kwargs['user_type'] = 'student'

        return super().get_context_data(**kwargs)

    def form_valid(self, form):

        user = form.save()

        login(self.request, user)

        return redirect('students:quiz_list')

    @method_decorator([login_required, student_required], name='dispatch')

class StudentInterestsView(UpdateView):

    model = Student

    form_class = StudentInterestsForm

    template_name = 'classroom/students/interests_form.html'
```

```
success_url = reverse_lazy('students:quiz_list')

def get_object(self):

return self.request.user.student

def form_valid(self, form):

messages.success(self.request, 'Interests updated with success!')

return super().form_valid(form)


@method_decorator([login_required, student_required], name='dispatch')

class QuizListView(ListView):

model = Quiz

ordering = ('name',)

context_object_name = 'quizzes'

template_name = 'classroom/students/quiz_list.html'

def get_queryset(self):

student = self.request.user.student

student_interests = student.interests.values_list('pk', flat=True)

taken_quizzes = student.quizzes.values_list('pk', flat=True)

queryset = Quiz.objects.filter(subject__in=student_interests) \

.exclude(pk__in=taken_quizzes) \

.annotate(questions_count=Count('questions')) \

.filter(questions_count__gt=0)

return queryset


@method_decorator([login_required, student_required], name='dispatch')
```

```
class TakenQuizListView(ListView):

    model = TakenQuiz

    context_object_name = 'taken_quizzes'

    template_name = 'classroom/students/taken_quiz_list.html'

    def get_queryset(self):

        queryset = self.request.user.student.taken_quizzes \

        .select_related('quiz', 'quiz__subject') \

        .order_by('quiz__name')

        return queryset


@login_required
@student_required

def take_quiz(request, pk):

    quiz = get_object_or_404(Quiz, pk=pk)

    student = request.user.student

    print(request.POST)

    if 'password' in request.POST:

        if request.POST['password'] == quiz.password or quiz.password is None:

            if student.quizzes.filter(pk=pk).exists():

                return render(request, 'students/taken_quiz_list.html')

            else:

                return JsonResponse({'error': 'Password did not match'})
```

```
total_questions = quiz.questions.count()

unanswered_questions = student.get_unanswered_questions(quiz)

total_unanswered_questions = unanswered_questions.count()

progress = 100 - round((((total_unanswered_questions - 1) / total_questions) * 100))

question = unanswered_questions.first()

if request.method == 'POST':

    form = TakeQuizForm(question=question, data=request.POST)

    if form.is_valid():

        with transaction.atomic():

            student_answer = form.save(commit=False)

            student_answer.student = student

            student_answer.save()

            if student.get_unanswered_questions(quiz).exists():

                return redirect('students:take_quiz', pk)

            else:

                correct_answers = student.quiz_answers.filter(answer__question__quiz=quiz,
                    answer__is_correct=True).count()

                score = round((correct_answers / total_questions) * 100.0, 2)

                TakenQuiz.objects.create(student=student, quiz=quiz, score=score)

                if score < 50.0:

                    messages.warning(request, 'Better luck next time! Your score for the quiz %s was
                    %s.' % (quiz.name, score))

            else:
```

```
messages.success(request, 'Congratulations! You completed the quiz %s with success!
You scored %s points.' % (quiz.name, score))
```

```
return redirect('students:quiz_list')
```

```
else:
```

```
form = TakeQuizForm(question=question)
```

```
return render(request, 'classroom/students/take_quiz_form.html', {
```

```
'quiz': quiz,
```

```
'question': question,
```

```
'form': form,
```

```
'progress': progress
```

```
}})
```

```
else:
```

```
form = TakeQuizForm(question=question)
```

```
return render(request, 'classroom/students/take_quiz_form.html', {
```

```
'quiz': quiz,
```

```
'question': question,
```

```
'form': form,
```

```
'progress': progress
```

```
}})
```

CLASSROOM/views/teacher.py

```
from django.contrib import messages
```

```
from django.contrib.auth import login
```

```
from django.contrib.auth.decorators import login_required

from django.db import transaction

from django.db.models import Avg, Count

from django.forms import inlineformset_factory

from django.shortcuts import get_object_or_404, redirect, render

from django.urls import reverse, reverse_lazy

from django.utils.decorators import method_decorator

from django.views.generic import (CreateView, DeleteView, DetailView, ListView,
UpdateView)

from ..decorators import teacher_required

from ..forms import BaseAnswerInlineFormSet, QuestionForm, TeacherSignUpForm

from ..models import Answer, Question, Quiz, User

class TeacherSignUpView(CreateView):

    model = User

    form_class = TeacherSignUpForm

    template_name = 'registration/signup_form.html'

    def get_context_data(self, **kwargs):

        kwargs['user_type'] = 'teacher'

        return super().get_context_data(**kwargs)

    def form_valid(self, form):

        user = form.save()

        login(self.request, user)

        return redirect('teachers:quiz_change_list')
```



```
@method_decorator([login_required, teacher_required], name='dispatch')

class QuizListView(ListView):

    model = Quiz

    ordering = ('name', )

    context_object_name = 'quizzes'

    template_name = 'classroom/teachers/quiz_change_list.html'

    def get_queryset(self):

        queryset = self.request.user.quizzes \

            .select_related('subject') \

            .annotate(questions_count=Count('questions', distinct=True)) \

            .annotate(taken_count=Count('taken_quizzes', distinct=True))

        return queryset

@method_decorator([login_required, teacher_required], name='dispatch')

class QuizCreateView(CreateView):

    model = Quiz

    fields = ('name', 'subject', 'password')

    template_name = 'classroom/teachers/quiz_add_form.html'

    def form_valid(self, form):

        quiz = form.save(commit=False)

        quiz.owner = self.request.user

        quiz.save()

        messages.success(self.request, 'The quiz was created with success! Go ahead and add some questions now.')
```

```

return redirect('teachers:quiz_change', quiz.pk)

@method_decorator([login_required, teacher_required], name='dispatch')

class QuizUpdateView(UpdateView):

    model = Quiz

    fields = ('name', 'subject', 'password')

    context_object_name = 'quiz'

    template_name = 'classroom/teachers/quiz_change_form.html'

    def get_context_data(self, **kwargs):

        kwargs['questions'] =
        self.get_object().questions.annotate(answers_count=Count('answers'))

    return super().get_context_data(**kwargs)

    def get_queryset(self):

        """

        This method is an implicit object-level permission management

        This view will only match the ids of existing quizzes that belongs

        to the logged in user.

        """

        return self.request.user.quizzes.all()

    def get_success_url(self):

        return reverse('teachers:quiz_change', kwargs={'pk': self.object.pk})

    @method_decorator([login_required, teacher_required], name='dispatch')

    class QuizDeleteView(DeleteView):

        model = Quiz

```

```
context_object_name = 'quiz'

template_name = 'classroom/teachers/quiz_delete_confirm.html'

success_url = reverse_lazy('teachers:quiz_change_list')

def delete(self, request, *args, **kwargs):

    quiz = self.get_object()

    messages.success(request, 'The quiz %s was deleted with success!' % quiz.name)

    return super().delete(request, *args, **kwargs)

def get_queryset(self):

    return self.request.user.quizzes.all()

@method_decorator([login_required, teacher_required], name='dispatch')

class QuizResultsView(DetailView):

    model = Quiz

    context_object_name = 'quiz'

    template_name = 'classroom/teachers/quiz_results.html'

    def get_context_data(self, **kwargs):

        quiz = self.get_object()

        taken_quizzes = quiz.taken_quizzes.select_related('student__user').order_by('-date')

        total_taken_quizzes = taken_quizzes.count()

        quiz_score = quiz.taken_quizzes.aggregate(average_score=Avg('score'))

        extra_context = {

            'taken_quizzes': taken_quizzes,

            'total_taken_quizzes': total_taken_quizzes,

            'quiz_score': quiz_score
```

```
}

kwargs.update(extra_context)

return super().get_context_data(**kwargs)

def get_queryset(self):

    return self.request.user.quizzes.all()

@login_required

@teacher_required

def question_add(request, pk):

    quiz = get_object_or_404(Quiz, pk=pk, owner=request.user)

    if request.method == 'POST':

        form = QuestionForm(request.POST)

        if form.is_valid():

            question = form.save(commit=False)

            question.quiz = quiz

            question.save()

            messages.success(request, 'You may now add answers/options to the question.')

            return redirect('teachers:question_change', quiz.pk, question.pk)

        else:

            form = QuestionForm()

    return render(request, 'classroom/teachers/question_add_form.html', {'quiz': quiz,
    'form': form})

@login_required

@teacher_required
```

```
def question_change(request, quiz_pk, question_pk):

    quiz = get_object_or_404(Quiz, pk=quiz_pk, owner=request.user)

    question = get_object_or_404(Question, pk=question_pk, quiz=quiz)

    AnswerFormSet = inlineformset_factory(

        Question, # parent model

        Answer, # base model

        formset=BaseAnswerInlineFormSet,

        fields=('text', 'is_correct'),

        min_num=2,

        validate_min=True,

        max_num=10,

        validate_max=True

    )

    if request.method == 'POST':

        form = QuestionForm(request.POST, instance=question)

        formset = AnswerFormSet(request.POST, instance=question)

        if form.is_valid() and formset.is_valid():

            with transaction.atomic():

                form.save()

                formset.save()

            messages.success(request, 'Question and answers saved with success!')

    return redirect('teachers:quiz_change', quiz.pk)
```

```
else:

form = QuestionForm(instance=question)

formset = AnswerFormSet(instance=question)

return render(request, 'classroom/teachers/question_change_form.html', {

'quiz': quiz,

'question': question,

'form': form,

'formset': formset

}))

@method_decorator([login_required, teacher_required], name='dispatch')

class QuestionDeleteView(DeleteView):

model = Question

context_object_name = 'question'

template_name = 'classroom/teachers/question_delete_confirm.html'

pk_url_kwarg = 'question_pk'

def get_context_data(self, **kwargs):

question = self.get_object()

kwargs['quiz'] = question.quiz

return super().get_context_data(**kwargs)

def delete(self, request, *args, **kwargs):

question = self.get_object()

messages.success(request, 'The question %s was deleted with success!' %

question.text)
```

```
return super().delete(request, *args, **kwargs)

def get_queryset(self):

return Question.objects.filter(quiz__owner=self.request.user)

def get_success_url(self):

question = self.get_object()

return reverse('teachers:quiz_change', kwargs={'pk': question.quiz_id})
```

CLASSROOM/admin.py

```
from .models import User, Student,
StudentAnswer, Subject, Answer, Question, Quiz, TakenQuiz

from django.contrib.auth.models import AbstractUser

from django.contrib import admin

admin.site.register(User)

admin.site.register(Student)

admin.site.register(StudentAnswer)

admin.site.register(Subject)

admin.site.register(Answer)

admin.site.register(Question)

admin.site.register(Quiz)

admin.site.register(TakenQuiz)
```

CLASSROOM/apps.py

```
from django.apps import AppConfig

class ClassroomConfig(AppConfig):

    name = 'classroom'
```

CLASSROOM/decorators.py

```
from django.contrib.auth import REDIRECT_FIELD_NAME

from django.contrib.auth.decorators import user_passes_test

def student_required(function=None,
    redirect_field_name=REDIRECT_FIELD_NAME, login_url='login'):
    """
    Decorator for views that checks that the logged in user is a student,
    redirects to the log-in page if necessary.
    """
    actual_decorator = user_passes_test(
        lambda u: u.is_active and u.is_student,
        login_url=login_url,
        redirect_field_name=redirect_field_name
    )
    if function:
        return actual_decorator(function)
    return actual_decorator

def teacher_required(function=None,
    redirect_field_name=REDIRECT_FIELD_NAME, login_url='login'):
```


'''

Decorator for views that checks that the logged in user is a teacher,
redirects to the log-in page if necessary.

'''

```
actual_decorator = user_passes_test(
    lambda u: u.is_active and u.is_teacher,
    login_url=login_url,
    redirect_field_name=redirect_field_name
)

if function:

    return actual_decorator(function)

return actual_decorator
```

CLASSROOM/form.py

```
from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.db import transaction

from django.forms.utils import ValidationError

from classroom.models import (Answer, Question, Student, StudentAnswer,
    Subject, RecruiterDetails, Job, User)

class TeacherSignUpForm(UserCreationForm):

    class Meta(UserCreationForm.Meta):

        model = User
```

```
def save(self, commit=True):  
  
    user = super().save(commit=False)  
  
    user.is_teacher = True  
  
    if commit:  
  
        user.save()  
  
    return user
```

```
class StudentSignUpForm(UserCreationForm):  
  
    interests = forms.ModelMultipleChoiceField(  
  
        queryset=Subject.objects.all(),  
  
        widget=forms.CheckboxSelectMultiple,  
  
        required=True  
    )  
  
    class Meta(UserCreationForm.Meta):  
  
        model = User  
  
        @transaction.atomic  
  
        def save(self):  
  
            user = super().save(commit=False)  
  
            user.is_student = True  
  
            user.save()  
  
            student = Student.objects.create(user=user)  
  
            student.interests.add(*self.cleaned_data.get('interests'))
```

```
return user

class StudentInterestsForm(forms.ModelForm):

    class Meta:

        model = Student

        fields = ('interests',)

        widgets = {

            'interests': forms.CheckboxSelectMultiple

        }

class QuestionForm(forms.ModelForm):

    class Meta:

        model = Question

        fields = ('text',)

class BaseAnswerInlineFormSet(forms.BaseInlineFormSet):

    def clean(self):

        super().clean()

        has_one_correct_answer = False

        for form in self.forms:

            if not form.cleaned_data.get('DELETE', False):

                if form.cleaned_data.get('is_correct', False):

                    has_one_correct_answer = True

                    break

        if not has_one_correct_answer:
```

```
raise ValidationError('Mark at least one answer as correct.',
code='no_correct_answer')

class TakeQuizForm(forms.ModelForm):

    answer = forms.ModelChoiceField(

    queryset=Answer.objects.none(),

    widget=forms.RadioSelect(),

    required=True,

    empty_label=None)

    class Meta:

        model = StudentAnswer

        fields = ('answer', )

    def __init__(self, *args, **kwargs):

        question = kwargs.pop('question')

        super().__init__(*args, **kwargs)

        self.fields['answer'].queryset = question.answers.order_by('text')
```

CLASSROOM/models.py

```
from django.contrib.auth.models import AbstractUser

from django.db import models

from django.utils.html import escape, mark_safe

from django.contrib.auth.models import User

class User(AbstractUser):
```

```
is_student = models.BooleanField(default=False)

is_teacher = models.BooleanField(default=False)

class Subject(models.Model):

    name = models.CharField(max_length=30)

    color = models.CharField(max_length=7, default='#007bff')

    def __str__(self):

        return self.name

    def get_html_badge(self):

        name = escape(self.name)

        color = escape(self.color)

        html = '<span class="badge badge-primary" style="background-color: %s">%s</span>' % (color, name)

        return mark_safe(html)

class Quiz(models.Model):

    owner = models.ForeignKey(User, on_delete=models.CASCADE,
                               related_name='quizzes')

    name = models.CharField(max_length=255)

    subject = models.ForeignKey(Subject, on_delete=models.CASCADE,
                                related_name='quizzes')

    password = models.CharField(blank=True, null=True, max_length=10)

    def __str__(self):

        return self.name

class Question(models.Model):
```

```
quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE,
related_name='questions')

text = models.CharField('Question', max_length=255)

def __str__(self):

return self.text

class Answer(models.Model):

question = models.ForeignKey(Question, on_delete=models.CASCADE,
related_name='answers')

text = models.CharField('Answer', max_length=255)

is_correct = models.BooleanField('Correct answer', default=False)

def __str__(self):

return self.text

class Student(models.Model):

user = models.OneToOneField(User, on_delete=models.CASCADE,
primary_key=True)

quizzes = models.ManyToManyField(Quiz, through='TakenQuiz')

interests = models.ManyToManyField(Subject, related_name='interested_students')

def get_unanswered_questions(self, quiz):

answered_questions = self.quiz_answers \

.filter(answer__question__quiz=quiz) \

.values_list('answer__question__pk', flat=True)

questions = quiz.questions.exclude(pk__in=answered_questions).order_by('text')

return questions

def __str__(self):
```

```
return self.user.username

class TakenQuiz(models.Model):

    student = models.ForeignKey(Student, on_delete=models.CASCADE,
                                related_name='taken_quizzes')

    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE,
                             related_name='taken_quizzes')

    score = models.FloatField()

    date = models.DateTimeField(auto_now_add=True)

class StudentAnswer(models.Model):

    student = models.ForeignKey(Student, on_delete=models.CASCADE,
                                related_name='quiz_answers')

    answer = models.ForeignKey(Answer, on_delete=models.CASCADE,
                              related_name='+')

class RecruiterDetails(models.Model):

    first_name = models.CharField(max_length= 255)

    last_name = models.CharField(max_length= 255)

    email = models.EmailField(max_length= 70,blank= True, null=True, unique= True)

    mobile = models.CharField(max_length= 10)

    organization_name = models.CharField(max_length= 255, blank= True, unique=
    True)

    organization_email = models.EmailField(max_length= 70, blank= True, null=True,
    unique= True)

    organization_description = models.CharField(max_length= 255)

    organization_logo = models.ImageField(upload_to='organization_logo', blank=True)

    def __str__(self):
```

```
return self.first_name + " " + self.last_name
```

CLASSROOM/urls.py

```
from django.urls import include, path

from .views import classroom, students, teachers

urlpatterns = [

    path("", classroom.home, name='home'),

    path('students/', include([

        path("", students.QuizListView.as_view(), name='quiz_list'),

        path('interests/', students.StudentInterestsView.as_view(), name='student_interests'),

        path('taken/', students.TakenQuizListView.as_view(), name='taken_quiz_list'),

        path('quiz/<int:pk>', students.take_quiz, name='take_quiz'),

    ], 'classroom')), namespace='students')),

    path('teachers/', include([

        path("", teachers.QuizListView.as_view(), name='quiz_change_list'),

        path('quiz/add/', teachers.QuizCreateView.as_view(), name='quiz_add'),

        path('quiz/<int:pk>', teachers.QuizUpdateView.as_view(), name='quiz_change'),

        path('quiz/<int:pk>/delete/', teachers.QuizDeleteView.as_view(), name='quiz_delete'),

        path('quiz/<int:pk>/results/', teachers.QuizResultsView.as_view(),

            name='quiz_results'),

        path('quiz/<int:pk>/question/add/', teachers.question_add, name='question_add'),

        path('quiz/<int:quiz_pk>/question/<int:question_pk>', teachers.question_change,

            name='question_change'),
```



```

path('quiz/<int:quiz_pk>/question/<int:question_pk>/delete/',
teachers.QuestionDeleteView.as_view(), name='question_delete'),

], 'classroom'), namespace='teachers')),

]

```

CLASSROOM/template/classroom/home.html

```

{% extends 'base.html' %}

{% block content %}

<h2>Welcome to the Classroom Quiz <span class="icon-emo-happy"></span></h2>

<p class="lead">

Let's Quiz Begin.. Click here to continue <a href="{% url 'login' %}">log
in</a>.</p>

<p class="lead">New to Classroom Quiz? Get Started as <a href="{% url
'student_signup' %}">student account</a>

or as <a href="{% url 'teacher_signup' %}">teacher account</a>.</p>

{% endblock %}

```

CLASSROOM/template/classroom/student/_header.html

```

<h2>Quizzes</h2>

<p class="text-muted">

Subjects:{% for subject in user.student.interests.all %} {{ subject.get_html_badge
}}{% endfor %}

<a href="{% url 'students:student_interests' %}"><small>(update
interests)</small></a>

```

```
</p>
```

```
<ul class="nav nav-tabs mb-3">
```

```
<li class="nav-item">
```

```
<a class="nav-link{% if active == 'new' %} active{% endif %}" href="{% url
'students:quiz_list' %}">New</a>
```

```
</li>
```

```
<li class="nav-item">
```

```
<a class="nav-link{% if active == 'taken' %} active{% endif %}" href="{% url
'students:taken_quiz_list' %}">Taken</a>
```

```
</li>
```

```
</ul>
```

CLASSROOM/template/classroom/student/interest_form.html

```
{% extends 'base.html' %}
```

```
{% load crispy_forms_tags %}
```

```
{% block content %}
```

```
<h2 class="mb-3">Update your interests</h2>
```

```
<form method="post" novalidate>
```

```
{% csrf_token %}
```

```
{{ form|crispy }}
```

```
<button type="submit" class="btn btn-success">Save changes</button>
```

```
<a href="{% url 'students:quiz_list' %}" class="btn btn-outline-
secondary">Cancel</a>
```

```
</form>
```

```
{% endblock % }
```

CLASSROOM/template/classroom/student/quiz_list.html

```
{% extends 'base.html' % }
```

```
{% block content % }
```

```
{% include 'classroom/students/_header.html' with active='new' % }
```

```
<div class="card">
```

```
<table class="table mb-0">
```

```
<thead>
```

```
<tr>
```

```
<th>Quiz</th>
```

```
<th>Subject</th>
```

```
<th>Length</th>
```

```
<th>Password</th>
```

```
<th></th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
{% for quiz in quizzes % }
```

```
<tr>
```

```
<td class="align-middle">{{ quiz.name }}</td>
```

```
<td class="align-middle">{{ quiz.subject.get_html_badge }}</td>
```

```
<td class="align-middle">{{ quiz.questions_count }} questions</td>
```

```
<form method="POST" action="{ % url 'students:take_quiz' quiz.pk % }">

{ % csrf_token % }

<td class="align-middle">

<input type="text" name="password"/>

</td>

<td class="text-right">

<button type="submit" class="save btn btn-primary">Start</button>

<!--<a href="{ % url 'students:take_quiz' quiz.pk % }" class="btn btn-primary">Start
quiz</a>-->

</td>

</form>

</tr>

{ % empty % }

<tr>

<td class="bg-light text-center font-italic" colspan="4">No quiz matching your
interests right now.</td>

</tr>

{ % endfor % }

</tbody>

</table>

</div>

{ % endblock % }
```

CLASSROOM/template/classroom/student/take_quiz_form.html

```
{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block content %}

<div class="progress mb-3">

<div class="progress-bar" role="progressbar" aria-valuenow="{{ progress }}" aria-
valuemin="0" aria-valuemax="100" style="width:{{ progress }}% "></div>

</div>

<h2 class="mb-3">{{ quiz.name }}</h2>

<p class="lead">{{ question.text }}</p>

<form method="post" novalidate>

{% csrf_token %}

{{ form|crispy }}

<button type="submit" class="btn btn-primary">Next →</button>

</form>

{% endblock %}
```

CLASSROOM/template/classroom/student/taken_quiz_list.html

```
{% extends 'base.html' %}

{% block content %}

{% include 'classroom/students/_header.html' with active='taken' %}

<div class="card">

<table class="table mb-0">
```

```
<thead>

<tr>

<th>Quiz</th>

<th>Subject</th>

<th>Score</th>

</tr>

</thead>

<tbody>

{ % for taken_quiz in taken_quizzes % }

<tr>

<td>{{ taken_quiz.quiz.name }}</td>

<td>{{ taken_quiz.quiz.subject.get_html_badge }}</td>

<td>{{ taken_quiz.score }}</td>

</tr>

{ % empty % }

<tr>

<td class="bg-light text-center font-italic" colspan="3">You haven't completed any
quiz yet.</td>

</tr>

{ % endfor % }

</tbody>

</table>

</div>
```

```
{% endblock % }
```

CLASSROOM/template/classroom/teacher/question_add_form.html

```
{% extends 'base.html' % }
```

```
{% load crispy_forms_tags % }
```

```
{% block content % }
```

```
<nav aria-label="breadcrumb">
```

```
<ol class="breadcrumb">
```

```
<li class="breadcrumb-item"><a href="{ % url 'teachers:quiz_change_list' % }">My  
Quizzes</a></li>
```

```
<li class="breadcrumb-item"><a href="{ % url 'teachers:quiz_change' quiz.pk % }">{{  
quiz.name }}</a></li>
```

```
<li class="breadcrumb-item active" aria-current="page">Add a new question</li>
```

```
</ol>
```

```
</nav>
```

```
<h2 class="mb-3">Add a new question</h2>
```

```
<p class="lead">Add first the text of the question. In the next step you will be able to  
add the possible answers.</p>
```

```
<form method="post" novalidate>
```

```
{% csrf_token % }
```

```
{{ form|crispy }}
```

```
<button type="submit" class="btn btn-success">Save</button>
```

```
<a href="{ % url 'teachers:quiz_change' quiz.pk % }" class="btn btn-outline-  
secondary" role="button">Cancel</a>
```

```
</form>
```

```
{% endblock % }
```

CLASSROOM/template/classroom/teacher/question_change_form.html

```
{% extends 'base.html' % }
```

```
{% load crispy_forms_tags crispy_forms_filters % }
```

```
{% block content % }
```

```
<nav aria-label="breadcrumb">
```

```
<ol class="breadcrumb">
```

```
<li class="breadcrumb-item"><a href="{% url 'teachers:quiz_change_list' %}">My  
Quizzes</a></li>
```

```
<li class="breadcrumb-item"><a href="{% url 'teachers:quiz_change' quiz.pk %}">{{  
quiz.name }}</a></li>
```

```
<li class="breadcrumb-item active" aria-current="page">{{ question.text }}</li>
```

```
</ol>
```

```
</nav>
```

```
<h2 class="mb-3">{{ question.txt }}</h2>
```

```
<form method="post" novalidate>
```

```
{% csrf_token % }
```

```
{{ formset.management_form }}
```

```
{{ form|crispy }}
```

```
<div class="card mb-3{% if formset.errors % } border-danger{% endif %}">
```



```

<div class="card-header">

<div class="row">

<div class="col-8">

<strong>Answers</strong>

</div>

<div class="col-2">

<strong>Correct?</strong>

</div>

<div class="col-2">

<strong>Delete?</strong>

</div>

</div>

</div>

</div>

{ % for error in formset.non_form_errors % }

<div class="card-body bg-danger border-danger text-white py-2">{{ error }}</div>

{ % endfor % }

<div class="list-group list-group-flush list-group-formset">

{ % for form in formset % }

<div class="list-group-item">

<div class="row">

<div class="col-8">

{ % for hidden in form.hidden_fields % }{{ hidden }}{ % endfor % }

{{ form.text|as_crispy_field }}

```

```
{% if form.instance.pk and form.text.value != form.instance.text %}<p class="mb-0
mt-1"><small class="text-muted font-italic"><strong>Old answer:</strong> {{
form.instance.text }}</small></p>{% endif %}
```

```
</div>
```

```
<div class="col-2">
```

```
{{ form.is_correct }}
```

```
</div>
```

```
<div class="col-2">
```

```
{% if form.instance.pk %}
```

```
{{ form.DELETE }}
```

```
{% endif %}
```

```
</div>
```

```
</div>
```

```
</div>
```

```
{% endfor %}
```

```
</div>
```

```
</div>
```

```
<p>
```

```
<small class="form-text text-muted">Your question may have at least
<strong>2</strong> answers and maximum <strong>10</strong> answers. Select at
least one correct answer.</small>
```

```
</p>
```

```
<button type="submit" class="btn btn-success">Save changes</button>
```

```

<a href="{ % url 'teachers:quiz_change' quiz.pk % }" class="btn btn-outline-
secondary" role="button">Nevermind</a>

<a href="{ % url 'teachers:question_delete' quiz.pk question.pk % }" class="btn btn-
danger float-right">Delete</a>

</form>

{ % endblock % }

```

CLASSROOM/template/classroom/teacher/question_delete_confirm.html

```

{ % extends 'base.html' % }

{ % load crispy_forms_tags % }

{ % block content % }

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="{ % url 'teachers:quiz_change_list' % }">My
Quizzes</a></li>

<li class="breadcrumb-item"><a href="{ % url 'teachers:quiz_change' quiz.pk % }">{ {
quiz.name } }</a></li>

<li class="breadcrumb-item"><a href="{ % url 'teachers:question_change' quiz.pk
question.pk % }">{ { question.text } }</a></li>

<li class="breadcrumb-item active" aria-current="page">Confirm deletion</li>

</ol>

</nav>

<h2 class="mb-3">Confirm deletion</h2>

```

```

<p class="lead">Are you sure you want to delete the question <strong>"{{
question.text }}"</strong>? There is no going back.</p>

<form method="post">

{% csrf_token %}

<button type="submit" class="btn btn-danger btn-lg">Yes, I'm sure</button>

<a href="{% url 'teachers:question_change' quiz.pk question.pk %}" class="btn btn-
outline-secondary btn-lg" role="button">Cancel</a>

</form>

{% endblock %}

```

CLASSROOM/template/classroom/teacher/quiz_add_form.html

```

{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block content %}

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="{% url 'teachers:quiz_change_list' %}">My
Quizzes</a></li>

<li class="breadcrumb-item active" aria-current="page">Add a new quiz</li>

</ol>

</nav>

<h2 class="mb-3">Add a new quiz</h2>

<div class="row">

<div class="col-md-6 col-sm-8 col-12">

```

```

<form method="post" novalidate>

{% csrf_token %}

{{ form|crispy }}

<button type="submit" class="btn btn-success">Save</button>

<a href="{% url 'teachers:quiz_change_list' %}" class="btn btn-outline-secondary"
role="button">Cancel</a>

</form>

</div>

</div>

{% endblock %}

```

CLASSROOM/template/classroom/teacher/quiz_change_form.html

```

{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block content %}

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="{% url 'teachers:quiz_change_list' %}">My
Quizzes</a></li>

<li class="breadcrumb-item active" aria-current="page">{{ quiz.name }}</li>

</ol>

```

```
</nav>

<h2 class="mb-3">

{{ quiz.name }}

<a href="{% url 'teachers:quiz_results' quiz.pk %}" class="btn btn-primary float-
right">View results</a>

</h2>

<div class="row mb-3">

<div class="col-md-6 col-sm-8 col-12">

<form method="post" novalidate>

{% csrf_token %}

{{ form|crispy }}

<button type="submit" class="btn btn-success">Save changes</button>

<a href="{% url 'teachers:quiz_change_list' %}" class="btn btn-outline-secondary"
role="button">Cancel</a>

<a href="{% url 'teachers:quiz_delete' quiz.pk %}" class="btn btn-danger float-
right">Delete</a>

</form>

</div>

</div>

<div class="card">

<div class="card-header">

<div class="row">

<div class="col-10">

<strong>Questions</strong>
```

```

</div>

<div class="col-2">

<strong>Answers</strong>

</div>

</div>

</div>

<div class="list-group list-group-flush list-group-formset">

{% for question in questions %}

<div class="list-group-item">

<div class="row">

<div class="col-10">

<a href="{% url 'teachers:question_change' quiz.pk question.pk %}">{{ question.text
}}</a>

</div>

<div class="col-2">

{{ question.answers_count }}

</div>

</div>

</div>

</div>

{% empty %}

<div class="list-group-item text-center">

<p class="text-muted font-italic mb-0">You haven't created any questions yet. Go
ahead and <a href="{% url 'teachers:question_add' quiz.pk %}">add the first
question</a>.</p>

```

```

</div>

{% endfor %}

</div>

<div class="card-footer">

<a href="{% url 'teachers:question_add' quiz.pk %}" class="btn btn-primary btn-
sm">Add question</a>

</div>

</div>

{% endblock %}

```

CLASSROOM/template/classroom/teacher/quiz_change_list.html

```

{% extends 'base.html' %}

{% block content %}

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item active" aria-current="page">My Quizzes</li>

</ol>

</nav>

<h2 class="mb-3">My Quizzes</h2>

<a href="{% url 'teachers:quiz_add' %}" class="btn btn-primary mb-3"
role="button">Add quiz</a>

<div class="card">

<table class="table mb-0">

```



```

<thead>

<tr>

<th>Quiz</th>

<th>Subject</th>

<th>Questions</th>

<th>Taken</th>

<th></th>

</tr>

</thead>

<tbody>

{% for quiz in quizzes %}

<tr>

<td class="align-middle"><a href="{% url 'teachers:quiz_change' quiz.pk %}">{{
quiz.name }}</a></td>

<td class="align-middle">{{ quiz.subject.get_html_badge }}</td>

<td class="align-middle">{{ quiz.questions_count }}</td>

<td class="align-middle">{{ quiz.taken_count }}</td>

<td class="text-right">

<a href="{% url 'teachers:quiz_results' quiz.pk %}" class="btn btn-primary">View
results</a>

</td>

</tr>

{% empty %}

```

```

<tr>

<td class="bg-light text-center font-italic" colspan="5">You haven't created any quiz
yet.</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

{% endblock %}

```

CLASSROOM/template/classroom/teacher/quiz_delete_confirm.html

```

{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block content %}

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="{ % url 'teachers:quiz_change_list' % }">My
Quizzes</a></li>

<li class="breadcrumb-item"><a href="{ % url 'teachers:quiz_change' quiz.pk % }">{ {
quiz.name } }</a></li>

<li class="breadcrumb-item active" aria-current="page">Confirm deletion</li>

</ol>

</nav>

```

```

<h2 class="mb-3">Confirm deletion</h2>

<p class="lead">Are you sure you want to delete the quiz <strong>"{{ quiz.name
}}"</strong>? There is no going back.</p>

<form method="post">

{% csrf_token %}

<button type="submit" class="btn btn-danger btn-lg">Yes, I'm sure</button>

<a href="{% url 'teachers:quiz_change' quiz.pk %}" class="btn btn-outline-secondary
btn-lg" role="button">Cancel</a>

</form>

{% endblock %}

```

CLASSROOM/template/classroom/teacher/quiz_result.html

```

{% extends 'base.html' %}

{% load crispy_forms_tags humanize %}

{% block content %}

<nav aria-label="breadcrumb">

<ol class="breadcrumb">

<li class="breadcrumb-item"><a href="{% url 'teachers:quiz_change_list' %}">My
Quizzes</a></li>

<li class="breadcrumb-item"><a href="{% url 'teachers:quiz_change' quiz.pk %}">{{
quiz.name }}</a></li>

<li class="breadcrumb-item active" aria-current="page">Results</li>

</ol>

```

```
</nav>

<h2 class="mb-3">{{ quiz.name }} Results</h2>

<div class="card">

<div class="card-header">

<strong>Taken Quizzes</strong>

<span class="badge badge-pill badge-primary float-right">Average Score: {{
quiz_score.average_score|default_if_none:0.0 }}</span>

</div>

<table class="table mb-0">

<thead>

<tr>

<th>Student</th>

<th>Date</th>

<th>Score</th>

</tr>

</thead>

<tbody>

{% for taken_quiz in taken_quizzes %}

<tr>

<td>{{ taken_quiz.student.user.username }}</td>

<td>{{ taken_quiz.date|naturaltime }}</td>

<td>{{ taken_quiz.score }}</td>

</tr>
```

```
{% endfor %}

</tbody>

</table>

<div class="card-footer text-muted">

Total respondents: <strong>{{ total_taken_quizzes }}</strong>

</div>

</div>

{% endblock %}
```

TEMPLATE/base.html

```
{% load static %} <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">

<title>{% block title %}QuizUp{% endblock %}</title>

<link rel="icon" href="{% static 'img/favicon.png' %}">

<link href="https://fonts.googleapis.com/css?family=Clicker+Script|Raleway"
rel="stylesheet">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6J
Xm" crossorigin="anonymous">
```

```

<link rel="stylesheet" type="text/css" href="{ % static 'vendor/fontello-
2f186091/css/fontello.css' % }">

<link rel="stylesheet" type="text/css" href="{ % static 'css/app.css' % }">

{ % if user.is_authenticated and user.is_teacher % }

<link rel="stylesheet" type="text/css" href="{ % static 'css/teachers.css' % }">

{ % else % }

<link rel="stylesheet" type="text/css" href="{ % static 'css/students.css' % }">

{ % endif % }

</head>

<body>

<div class="container my-4">

<div class="row justify-content-center">

<div class="col-md-10 col-sm-12">

<div class="row">

<div class="col-6">

<h1 class="logo">

<a href="{ % url 'home' % }">

Classroom Quiz

{ % if user.is_authenticated % }

{ % if user.is_teacher % }

<span class="icon" data-toggle="tooltip" data-placement="right" title="Teacher
profile"></span>

{ % else % }

```

```

<span class="icon" data-toggle="tooltip" data-placement="right" title="Student
profile"></span>

{ % endif % }

{ % endif % }

</a>

</h1>

</div>

<div class="col-6 text-right">

{ % if user.is_authenticated % }

<p class="pt-3">Logged in as <strong>{{ user.username }}</strong>. <a href="{ %
url 'logout' % }">Log out</a>.</p>

{ % else % }

<a href="{ % url 'login' % }" class="btn btn-light" role="button">Log in</a>

<a href="{ % url 'signup' % }" class="btn btn-primary" role="button">Sign up</a>

{ % endif % }

</div>

</div>

<div class="card mb-3">

<div class="card-body">

{ % for message in messages % }

<div class="alert {{ message.tags }} alert-dismissible fade show" role="alert">

{{ message }}

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

```

```
<span aria-hidden="true">&times;</span>
```

```
</button>
```

```
</div>
```

```
{ % endfor % }
```

```
{ % block content % }
```

```
{ % endblock % }
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG
5KkN" crossorigin="anonymous"></script>
```

```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q
" crossorigin="anonymous"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl
" crossorigin="anonymous"></script>
```

```
<script type="text/javascript">
```

```
$(function () {

$('[data-toggle="tooltip"]').tooltip();
```



```

}))

</script>

</body>

</html>

```

TEMPLATE/registration/login.html

```

{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block content %}

{% if form.non_field_errors %}

<div class="alert alert-danger alert-dismissible fade show" role="alert">

{% for error in form.non_field_errors %}

<p>{% if forloop.last %} class="mb-0"{% endif %}>{{ error }}</p>

{% endfor %}

<button type="button" class="close" data-dismiss="alert" aria-label="Close">

<span aria-hidden="true">&times;</span>

</button>

</div>

{% endif %}

<div class="row">

<div class="col-lg-4 col-md-6 col-sm-8 col-12">

<h2>Log in</h2>

<form method="post" novalidate>

{% csrf_token %}

```

```

<input type="hidden" name="next" value="{{ next }}">

{{ form.username|as_crispy_field }}

{{ form.password|as_crispy_field }}

<button type="submit" class="btn btn-primary">Log in</button>

</form>

</div>

</div>

{% endblock %}

```

TEMPLATE/registration/signup_form.html

```

{% extends 'base.html' %}

{% load crispy_forms_tags %}

{% block content %}

<div class="row">

<div class="col-md-8 col-sm-10 col-12">

<h2>Sign up as a {{ user_type }}</h2>

<form method="post" novalidate>

{% csrf_token %}

<input type="hidden" name="next" value="{{ next }}">

{{ form|crispy }}

<button type="submit" class="btn btn-success">Sign up</button>

</form>

</div>

</div>

```

```
{% endblock % }
```

TEMPLATE/registration/signup.html

```
{% extends 'base.html' % }

{% block content % }

<h2>Sign up for a free account</h2>

<p class="lead">Select below the type of account you want to create</p>

<a href="{% url 'student_signup' %}" class="btn btn-student btn-lg"
role="button">I'm a student</a>

<a href="{% url 'teacher_signup' %}" class="btn btn-teacher btn-lg"
role="button">I'm a teacher</a>

{% endblock % }
```

STATIC/css/apps.css

```
.logo {

font-family: 'Raleway', sans-serif;

}

.logo a {

color: #212529;

text-decoration: none;

}
```

```
footer {  
  
font-size: .9rem;  
  
}
```

```
footer a {  
  
color: #212529;  
  
}
```

```
.list-group-formset label {  
  
display: none;  
  
}
```

```
.list-group-formset .form-group,  
  
.list-group-formset .invalid-feedback {  
  
margin-bottom: 0;  
  
}
```

```
.list-group-formset .form-control {  
  
padding: .25rem .5rem;  
  
font-size: .875rem;  
  
line-height: 1.5;  
  
border-radius: .2rem;
```

```
}
```

```
.btn-student {  
  
color: #fff;  
  
background-color: #91afb6;  
  
border-color: #91afb6;  
  
}
```

```
.btn-student:hover,  
  
.btn-student:active {  
  
color: #fff;  
  
background-color: #608993;  
  
border-color: #608993;  
  
}
```

```
.btn-teacher {  
  
color: #fff;  
  
background-color: #8980a5;  
  
border-color: #8980a5;  
  
}
```

```
.btn-teacher:hover,  
  
.btn-teacher:active {
```

```
color: #fff;

background-color: #66598B;

border-color: #66598B;

}
```

```
.has-danger .radio,

.has-danger .checkbox {

color: #dc3545;

}
```

```
.has-danger .invalid-feedback {

display: block;

}
```

STATIC/css/students.css

```
body {

background-color: #c2d4d8;

}
```

STATIC/css/teachers.css

```
body {

background-color: #b0aac2;

}
```

CHAPTER 6

CONCLUSION

CONCLUSION

The project entitled “CLASSROOM QUIZ “has been proposed to be implementing to replace the manual system paper base quiz. The classroom quiz System is developed using Django-python and SQLite3 fully meets the objectives of the system for which it has been developed. The system has reached a steady state where all bugs have been eliminated. The system is operated at a high level of efficiency and all the teachers and user associated with the system understands its advantage. The system solves the problem. It was intended to solve as requirement specification

CHAPTER 7

FUTURE ENHANCEMENT

Future Enhancement

This project has a wide scope and some future enhancement are needed to make it more dynamic web app. Following are some of its future enhancement-

- i. There can be added the feature of time limitation for each quiz.so that the quiz can be complete in given time.
- ii. this app can be used for the recruitment process of software companies which will be able to save time and efforts to eliminate unwanted candidates to appear for personal interview by travelling a long distance.
- iii. In this application, answer will not display only result will be displayed at last. It is our future enhancement to display answer while answering the question

CHAPTER 8

REFERENCES

References

- i. <http://www.softwaresuggest.com>
- ii. <http://financesonline.com>
- iii. <http://www.slideshare.net>
- iv. <https://www.python.org>
- v. <https://stackoverflow.com>
- vi. <https://www.quora.com>
- vii. “Web Programming”-by Srikanth .S ,Skyward Publishers
- viii. DjangoDocumentationRelease1.10.6.dev20170211184251DjangoSoftware
FoundationFebruary11,2017
- ix. Using SQLite- by Jay A. Kreibich