

ABSTRACT

This project titled as “*Quizzi*” will be developed to overcome the time consuming process of manual quiz system. Apart from that in current system, checking the answer sheets after taking test, wastes the examiner’s time, so this application will check the correct answer and carry the examination process in an effective manner.

The main aim of this project is to computerize the existing manual system and help the examiners to save their valuable time and important data. This project helps in managing the details of every quiz, marks, subjects for a long time and will be easily accessible.

This project will be developed keeping in view that an examiner can conduct as many quizzes as he/she wants to, where each student will not be attempting the same question at a time during the quiz time(questions will be in random sequence for each student). The project will reduce the manual process in managing examinations and issues regarding that.

Functionalities of the project will be as following :

MODULES :

Admin module – 20001-CM-005 – G. Poojitha

Classes Management module – 20001-CM-051 – T. Likitha Reddy

Quizzes Management module – 20001-CM-003 – Y. Ankitha

Grading module – 20001-CM-010 – B. Anjali

SOFTWARE REQUIREMENTS :

- Flask (Version 2.2.2)
- Jinja2 (Version 3.1.2)
- Werkzeug (Version 2.2.2)
- HTML5
- CSS3
- Bootstrap(Version 5.3)
- SQLite3

HARDWARE REQUIREMENTS :

- Any Operating System with a WEB BROWSER
- Minimum required RAM 4GB

INSTALLATION

Installation of VS Code

Step 1: Visit the [official website](#) of the **Visual Studio Code** using any web browser like Google Chrome, Microsoft Edge, etc.

Step 2: Press the “**Download for Windows**” button on the website to start the download of the Visual Studio Code Application.

Step 3: When the download finishes, then the Visual Studio Code icon appears in the downloads folder.

Step 4: Click on the installer icon to start the installation process of the Visual Studio Code.

Step 5: After the Installer opens, it will ask you for accepting the terms and conditions of the Visual Studio Code. Click on **I accept the agreement** and then click the **Next** button.

Step 6: Choose the location data for running the Visual Studio Code. It will then ask you for browsing the location. Then click on **Next** button.

Step 7: Then it will ask for beginning the installing setup. Click on the **Install** button.

Step 8: After clicking on Install, it will take about 1 minute to install the Visual Studio Code on your device.

Step 9: After the Installation setup for Visual Studio Code is finished, it will show a window like this below. Tick the “**Launch Visual Studio Code**” checkbox and then click **Next**.

Step 10: After the previous step, the **Visual Studio Code window** opens successfully. Now you can create a new file in the Visual Studio Code window and choose a language of yours to begin your programming journey!

Installation of Flask

Install Flask

Within the activated environment, use the following command to install Flask:

```
$ pip install Flask
```

3. INTRODUCTION

A Quiz application is a software that allows users to create and take quizzes online. It is typically used by educators to create and distribute assessments, but can also be used by businesses, organizations, and individuals to create and administer quizzes for various purposes.

The Quiz application will work similar to Google Classroom, in that it will provide a platform for teachers to create and manage quizzes, assign them to students, and track their progress. In addition, it will also provide a way for students to take quizzes.

Some key features of the Quiz application will include:

- ➔ Class creation and management: Teachers can create and manage classes, invite students.
- ➔ Quiz creation and distribution: Teachers can create and distribute quizzes.
- ➔ Overall, the Quiz application will provide a comprehensive and user-friendly platform for educators to create and manage quizzes and for students to take quizzes.

If anyone wants to access this application, he/she has to register as a Teacher/Student and has to login to use the application.

When a user login as a Teacher, he/she will be able to create and distribute classes to the Students.

Teacher can create and distribute Quizzes with already created classes.

When a user login as a Student, he/she can join the classes using the class name provided by his/her teacher.

Student can attempt various quizzes added by his/her teacher in the respective classes and get immediate score and solutions to the questions after submitting the quiz.

Immediately after a student submits his/her quiz, respective class teacher can view the results of the students attempted the quiz.

4. ANALYSIS

The main objective of “Quizzi” is to facilitate a user friendly environment for all users and reduces the manual effort. In past days quiz is conducted manually but in further resolution of the technology we are able to generate the score and pose the queries automatically. The functional requirements include to create users that are going to participate in the quiz, automatic score and report generation for admin privilege users. In this application, all the permissions lies with the administrator i.e., specifying the details of the quiz.

4.1 Existing System :

The current quizzes are being conducted through pen and paper in which test-taker i.e., Student is responsible for providing answers manually which are then manually graded by teacher/instructor. They are generally considered to be less sophisticated than online quizzing systems.

Disadvantages :

1. **Time consuming** : Grading a large number of quizzes manually can take a significant amount of time for the teacher.
2. **Limited Scalability** : Manual quizzing systems are not well suited for large classes, as it can be difficult to manage large number of quizzes.
3. **Lack of real-time assessment** : With manual quizzing, students have to wait for the teacher to announce the results.

4.2 Proposed System :

The proposed web application replaces the current manual quizzing system and makes quizzing online. It manages the details of every quiz, marks and subjects which takes long time with manual system. The instructor can conduct as many quizzes as he/she wants to where each student will not be attempting same question at a time during the quiz time (questions will be in random sequence for each student).

Advantages :

1. **Convenience** : These quizzes can be taken anywhere as long as the student has internet access.
2. **Immediate result** : Students get instant results and solutions for the attempted quiz, allowing the students to identify the areas where they need to improve.
3. **Data tracking** : These quizzes allow teachers to track students' performance, making it easier to identify the areas where the student needs extra support.

- 4. Cost effective :** These quizzes are less expensive than traditional methods of testing as they do not require materials such as pen/pencil and papers, etc.
- 5. Accessibility :** These quizzes can be made accessible for students with disabilities.
- 6. Self-paced Learning :** Students can work through the quizzes at their own pace and repeat them as many times as needed.

4.3 Requirements Specification:

4.3.1 Hardware Requirements :

Processor	:	X86-64bit Compatible processor with a minimum 2.40GHz Clock Speed
RAM	:	4GB or more
Hard Disk	:	256GB or more
Monitor	:	VGA/SVGA

4.3.2 Software Requirements :

Operating System	:	Windows 10 and more
Front End	:	HTML5, CSS3, JS ES13, Bootstrap v5.2
Back End	:	SQLITE3,DB BROWSER

5. TECHNOLOGIES USED

5.1 Software Description

5.1.1 What is VS Code 2022 ?

Visual Studio Code (VS Code) is a popular, open-source code editor developed by Microsoft. It has a wide range of features, including syntax highlighting, debugging, and integrated source control. In 2022, there may have been several updates and new features added to VS Code, including new extensions, improved performance, and bug fixes. Additionally, the VS Code team may continue to improve its support for various programming languages and development environments. Overall, VS Code is a widely-used and respected code editor among developers for its flexibility and ease of use.

5.1.2 What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python.

Python has a reputation as a beginner-friendly language, replacing Java as the most widely used introductory language because it handles much of the complexity for the user, allowing beginners to focus on fully grasping programming concepts rather than minute details.

Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is an open source community language, so numerous independent programmers are continually building libraries and functionality for it.

Python Use Cases

- Creating web applications on a server
- Building workflows that can be used in conjunction with software
- Connecting to database systems
- Reading and modifying files
- Performing complex mathematics
- Processing big data
- Fast prototyping
- Developing production-ready software

Features and Benefits of Python

- Compatible with a variety of platforms including Windows, Mac, Linux, Raspberry Pi, and others
- Uses a simple syntax comparable to the English language that lets developers use fewer lines than other programming languages
- Operates on an interpreter system that allows code to be executed immediately, fast-tracking prototyping
- Can be handled in a procedural, object-orientated, or functional way

Python Syntax

- Somewhat similar to the English language, with a mathematical influence, Python is built for readability
- Unlike other languages that use semicolons and/or parentheses to complete a command, Python uses new lines for the same function
- Defines scope (i.e., loops, functions, classes) by relying indentation, using whitespace, rather than braces (aka curly brackets)

5.1.3 What is Flask?

Flask is a web framework that provides libraries to build lightweight web applications in python. It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

When developing a web application, it is important to separate *business logic* from presentation logic. *Business logic* is what handles user requests and talks to the database to build an appropriate response. *Presentation logic* is how the data is presented to the user, typically using HTML files to build the basic structure of the response web page, and CSS styles to style HTML components. For example, in a social media application, you might have a username field and a password field that can be displayed only when the user is not logged in. If the user is logged in, you display a logout button instead. This is the presentation logic. If a user types in their username and password, you can use Flask to perform business logic: You extract the data (the username and password) from the request, log the user in if the credentials are correct or respond with an error message. How the error message is displayed will be handled by the presentation logic.

5.1.4 What is HTML ?

HTML stands for HyperText Markup Language. It is used to design web pages using the markup language. HTML is the combination of Hypertext and Markup

language. Hypertext defines the link between the web pages and markup language defines the text document within the tag that define the structure of web pages.

HTML is used to create the structure of web pages that are displayed on the World Wide Web (www). It contains Tags and Attributes that are used to design the web pages. Also, we can link multiple pages using Hyperlinks. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content.

HTML Tags that are used in this project:

- **<!DOCTYPE html>** : All HTML documents must start with a **<!DOCTYPE>** declaration. The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect.
- **<html lang="en">** : The lang attribute specifies the language of the element's content. Common examples are "en" for English, "es" for Spanish, "fr" for French and so on.
- **<head>** : The **<head>** tag in HTML is used to define the head portion of the document which contains information related to the document.
- **<meta charset="UTF-8">** : tells the browser to use the utf-8 character encoding when translating machine code into human-readable text and vice versa to be displayed in the browser.
- **<meta name="viewport" content="width=device-width, initial-scale=1.0">** : Setting the viewport meta tag lets you control the width and scaling of the viewport so that it's sized correctly on all devices.

- **<title>** : HTML title tag is used to provide a title name for your webpage. The HTML title tag must be used inside the <head> tag.
- **<script>** : HTML script tag is used to specify type of script such as Javascript. It allows you to place a script within your HTML document.
- **<body>** : The <body> HTML element represents the content of an HTML document. There can be only one <body> element in a document.
- **<header>** : The <header> tag defines the header of a page or a section. It usually contains a logo, search, navigational links, etc. This tag doesn't present a new section in the outline
- **<nav>** : The <nav> element in HTML provides a set of navigation links, either within the current document or to other documents. Common navigation sections that use the <nav> element include indexes, table of contents, menus, etc.
- **<div>** : The div tag is generally used by web developers to group HTML elements together and apply CSS styles to many elements at once
- **<a>** : The HTML <a> tag is used for creating a hyperlink to either another document, or somewhere within the current document.
- **** : The image tag allows us to insert images into a web page. It has no closing tag.
- **<h1> to <h6>** : This tag defines the HTML headings. The heading tag makes the text bigger and bold compared to the plain text. There are six heading tags

in HTML: h1, h2, h3, h4, h5, h6. The <h1> tag represents the most important heading while <h6> is for the least important ones.

- **<p>** : This tag defines a paragraph. When we use the <p> tag, the web browser automatically inserts a single blank line before and after each <p> element to make the text more readable
- **<!-- Comment -->** : The comment tag helps programmers to understand the HTML source code. The comments are not visible on the web page in a browser.
- **<button>** : The <button> tag is used to create a clickable button within HTML form on your webpage. You can put content like text or image within the <button>.....</button> tag.
- **** : The HTML tag is used for specifying an unordered list, which groups a collection of items having no numerical order.
- **** : Each element of an unordered list is declared inside the tag.
- **<i>** : The <i> tag in HTML is used to display icons in the HTML document.
- **<Section>** : The HTML <section> tag is used to define sections in a document. When you put your content on a web page, it may contains many chapters, headers, footers, or other sections on a web page that is why HTML <section> tag is used.
- **** : The HTML tag is used for grouping and applying styles to inline elements. There is a difference between the span tag and the div tag.

The span tag is used with inline elements whilst the div tag is used with block-level content.

- **<video>** : The <video> tag is used to embed video content in a document, such as a movie clip or other video streams.
- **<center>** : The <center> tag in HTML is used to set the alignment of text into the center.
- **<footer>** : HTML <footer> tag is used to define a footer for a document or a section. It is generally used in the last of the section (bottom of the page).
- **
** : The
 tag in HTML document is used to create a line break in a text.
- **<iframe>** : HTML Iframe is used to display a nested webpage (a webpage within a webpage). The HTML <iframe> tag defines an inline frame, hence it is also called as an Inline frame.
- **<form>** : The <form> tag is used to create an HTML form for user input.
- **<hr>** : The HTML <hr> tag is used for creating a horizontal line.

5.1.5 What is CSS?

Cascading Style Sheets, fondly referred to as CSS, is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page. It describes how a webpage should look: it prescribes colors, fonts, spacing, and much more. In short, you can make your website look however you want. CSS lets developers and

designers define how it behaves, including how elements are positioned in the browser.

While html uses tags, css uses rulesets. CSS is easy to learn and understand, but it provides powerful control over the presentation of an HTML document.

CSS Properties That are Used In this project:

- ***** :The asterisk (*) is known as the **CSS universal selectors**. It can be used to select any and all types of elements in an HTML page.
- **:root** : The :root selector matches the document's root element. In HTML, the root element is always the html element.
- **box-sizing** :The box-sizing property defines how the width and height of an element are calculated: should they include padding and borders, or not.
- **Margin** :The margin property sets the margins for an element
- **Padding** : An element's padding is the space between its content and its border.
- **font-family** : The font-family property specifies the font for an element.
- **background-color** : The background-color property specifies the background color of an element.
- **text-decoration** : the text-decoration property specifies the decoration added to text

- **::-webkit-scrollbar** : CSS pseudo-element is used to style scrollbar of an element.
- **::-webkit-scrollbar-track** : It defines the track of the scrollbar where the scroll-thumb scrolls.
- **Background** : The CSS background properties are used to add background effects for elements.
- **::-webkit-scrollbar-thumb** : This is used to style the moving element in the scrollbar-track.
- **Display** : The display property specifies the display behavior (the type of rendering box) of an element
- **border-radius** : The border-radius property defines the radius of the element's corners.
- **Transition** : CSS transitions allows you to change property values smoothly, over a given duration.
- **Border** : The CSS border properties allow you to specify the style, width, and color of an element's border.
- **Color** : Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.
- **font-weight** : The font-weight property sets how thick or thin characters in text should be displayed.

- **font-size** : The font-size property sets the size of a font.
- **text-transform** :The text-transform property controls the capitalization of text
- **letter-spacing** : The letter-spacing property increases or decreases the space between characters in a text.
- **border-color** : The border-color property sets the color of an element's four borders.
- **: hover** :The :hover selector is used to select elements when you mouse over them.
- **line-height** : The line-height property specifies the height of a line.
- **margin-bottom** : The margin-bottom property sets the bottom margin of an element.
- **text-transform** : The text-transform property controls the capitalization of text.
- **Height** : height property are used to set height of an element.
- **box-shadow** :The CSS box-shadow property is used to apply one or more shadows to an element.
- **Position** :The position property specifies the type of positioning method used for an element.

- **Content** : The content property is used with the ::before and ::after pseudo-elements, to insert generated content.
- **Left** : The left property affects the horizontal position of a positioned element.
- **Bottom** : The bottom property affects the vertical position of a positioned element. This property has no effect on non-positioned elements.
- **::after** : The ::after selector inserts something after the content of each selected element(s).
- **Top** : The top property affects the vertical position of a positioned element. This property has no effect on non-positioned elements.
- **min-height** : The min-height property defines the minimum height of an element.
- **background-repeat** : Sets how a background image will be repeated
- **background-position** : the background-position property sets the starting position of a background image.
- **background-attachment** : The background-attachment property sets whether a background image scrolls with the rest of the page, or is fixed.
- **flex-direction** : The flex-direction property specifies the direction of the flexible items.

- **align-items** : The align-items property specifies the default alignment for items inside the flexible container.
- **justify-content** : The justify-content property aligns the flexible container's items when the items do not use all available space on the main-axis (horizontally).
- **text-align** :The text-align property specifies the horizontal alignment of text in an element.
- **Overflow** :The overflow property specifies what should happen if content overflows an element's box.
- **min-width** : The min-width property defines the minimum width of an element.
- **object-fit** : The CSS object-fit property is used to specify how an or <video> should be resized to fit its container.
- **z-index** :The z-index property specifies the stack order of an element.
- **margin-top** : The margin-top property sets the top margin of an element.
- **padding-bottom** : The padding-bottom property sets the bottom padding (space) of an element.
- **padding-top** : The padding-top property sets the top padding (space) of an element.
- **margin-left** : The margin-left property sets the left margin of an element.

- **margin-right** : The margin-right property sets the right margin of an element.
- **background-image** : The background-image property specifies an image to use as the background of an element.
- **background-repeat** : The background-repeat property sets if/how a background image will be repeated.
- **background-position** : The background-position property sets the starting position of a background image.
- **border-bottom** : The border-bottom-style property in CSS is used to **set the style of the bottom border of an element.**
- **background-position**: The background-position property sets the starting position of a background image.

5.1.6 What is BOOTSTRAP?

Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website. It is absolutely free to download and use. It is a front-end framework used for easier and faster web development. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others. It facilitates you to create responsive designs.

Bootstrap Classes and tags used in this project are :

- **Container** :Containers are the most basic layout element in Bootstrap and are required when using our default grid system.

- **<nav>**:this tag in bootstrap is used to create a navigation bar in a document in which we can add sections
- **navbar-brand** : Added to a link or a header element inside the navbar to represent a logo or a header
- **navbar-fixed-top** : Makes the navbar stay at the top of the screen
- **navbar-toggler** : Styles the button that should open the navbar on small screens.
- **nav-link** : Styles an element to look like a link inside the navbar
- **disabled** : Disables a button
- **row** :In Bootstrap, the "row" class is used mainly to hold columns in it.

The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

We should be careful that the sum should not exceed 12 columns. And also its not compulsory to have 12 columns.
- **Margin** :a margin is the space around an element's border.
- **padding** :padding is the space between an element's border and the element's content.
- **img-fluid** : This class helps an image to adjust its size according to the screen size. Indirectly it makes an image as a responsive image

- **order** : order class rearranges the column inside a row. It helps to control the visual order of your content inside a webpage.
- **flex-row** : This class is used to display the flex items horizontally (side by side). This is default.
- **flex-column** : This class is used to display the flex items vertically (one on another).
- **Justify-content-*** : This class is used to change the alignment of the flex items. In the place of “*” we can give start, end, center, between, around.
- **align-items-*** : This class is used for Controlling the vertical alignment of **single rows** of flex items, and in the place of ‘*’ we can give .align-items-start, .align-items-end, .align-items-center, .align-items-baseline.
- **text-center** : You can align the text to the center by simply adding alignment class *.text-center* to the parent div.
- **card** : With this class we can create a card with a rounded border inside which you can place text, images, links, and etc.

5.1.7 What is SQLITE3?

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a popular choice as an embedded database for local/client storage in application software such as web browsers. It is also used in many other applications that need a lightweight, embedded database.

SQLite is ACID-compliant and implements most of the SQL standards, using a dynamically and weakly typed SQL syntax that does not guarantee domain

integrity

To use SQLite in a C/C++ program, you can use the sqlite3 API, which provides a lightweight, simple, self-contained, high-reliability, full-featured, and SQL database engine. The API is implemented as a library of C functions that can be called from your program. One of the main benefits of using SQLite is that it is very easy to get started with.

5.1.8 What is DB BROWSER ?

DB Browser for SQLite (DB4S) is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.

DB4S is for users and developers who want to create, search, and edit databases. DB4S uses a familiar spreadsheet-like interface, and complicated SQL commands do not have to be learned.

Controls and wizards are available for users to:

- Create and compact database files
- Create, define, modify and delete tables
- Create, define, and delete indexes
- Browse, edit, add, and delete records
- Search records
- Import and export records as text
- Import and export tables from/to CSV files
- Import and export databases from/to SQL dump files
- Issue SQL queries and inspect the results
- Examine a log of all SQL commands issued by the application
- Plot simple graphs based on table or query data

5.1.9 What is Jinja2 ?

Jinja2 is a modern day templating language for Python developers. It was made after Django's template. It is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Why do we need Jinja 2?

1. **Sandboxed Execution:** It provides a protected framework for automation of testing programs, whose behaviour is unknown and must be investigated.
2. **HTML Escaping:** Jinja 2 has a powerful automatic HTML Escaping, which helps preventing Cross-site Scripting ([XSS Attack](#)). There are special characters like >,<,&, etc. which carry special meanings in the templates. So, if you want to use them as regular text in your documents then, replace them with entities. Not doing so might lead to XSS-Attack.
3. **Template Inheritance:** This is the most important feature, which I will expand on to later in the post.

5.1.10 What is Werkzeug?

Werkzeug is a comprehensive [WSGI](#) web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.

It includes:

- An interactive debugger that allows inspecting stack traces and source code in the browser with an interactive interpreter for any frame in the stack.
- A full-featured request object with objects to interact with headers, query args, form data, files, and cookies.
- A response object that can wrap other WSGI applications and handle streaming data.
- A routing system for matching URLs to endpoints and generating URLs for endpoints, with an extensible system for capturing variables from URLs.
- HTTP utilities to handle entity tags, cache control, dates, user agents, cookies, files, and more.
- A threaded WSGI server for use while developing applications locally.
- A test client for simulating HTTP requests during testing without requiring running a server.

Werkzeug doesn't enforce any dependencies. It is up to the developer to choose a template engine, database adapter, and even how to handle requests. It can be used to build all sorts of end user applications such as blogs, wikis, or bulletin boards.

[Flask](#) wraps Werkzeug, using it to handle the details of WSGI while providing more structure and patterns for defining powerful applications.

5.2 Coding

Myproject.py

```
from flask import Flask, render_template, request, redirect, flash, url_for,
session

import sqlite3

from datetime import *

from flask_session import Session

curdir = r"E:\ANKITHA\Quizzi"
app = Flask(__name__)
app.config["SECRET_KEY"] = "uytraw35467y8uoikkijh"
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

@app.route("/")
def home():
    return render_template('home.html')

@app.route("/register")
def register():
    return render_template("register.html")

@app.route("/register", methods=["POST", "GET"])
def registration():
    fname = request.form['namef']
    username = request.form['emaile']
```

```

    prof = request.form['prof']
    password = request.form['passwordp']
    conpassword = request.form['passwordcon']
    con = sqlite3.connect(curdir + "\\Project.db")
    cur = con.cursor()

    if cur.execute("SELECT username from Register WHERE
username=:username",{ "username":username }).fetchone():
        flash("username already exists, try another","danger")
        return redirect("register")
    else:
        if password == conpassword:
            cur.execute("INSERT INTO Register VALUES
(?,?,?,?)",(fname,username,password,prof,))
            con.commit()
            con.close()
            flash("Registered Successfully!!","success")
            return redirect(url_for('login'))
        else:
            flash("Password doesn't match","danger")
            return render_template('register.html')

@app.route("/login")
def login():
    return render_template('login.html')

@app.route("/login", methods=["POST","GET"])
def loginpage():
    username = request.form["user"]
    session["user"] = username
    password = request.form["password"]

```

```

con = sqlite3.connect(curdir + "\\Project.db")

cur = con.cursor()

user = cur.execute("SELECT username from Register WHERE
username=:username",{ "username":username}).fetchone()

passw = cur.execute("SELECT password from Register WHERE
username=:username",{ "username":username}).fetchone()

prof = cur.execute("SELECT prof from Register WHERE
username=:username",{ "username":username}).fetchone()

con.commit()
con.close()

prof_data = ""
pass_data = ""
user_data = ""

if user is None:
    flash("No such user","danger")
    return render_template("login.html")
else:
    for a in prof:
        prof_data = prof_data + a
    for a in passw:
        pass_data = pass_data + a
    for a in user:
        user_data = user_data + a
    if password == pass_data:
        flash("Login Succesfull!!","success")
        if prof_data == "Teacher":
            return redirect(url_for('teacher',user = user_data))
        else:
            return redirect(url_for('student'))
    else:

```



```

        flash("Incorrect password","danger")
        return render_template("login.html")

@app.route("/logout")
def logout():
    session.pop("user",None)
    return redirect('/')

@app.route("/teacher")
def teacher():
    username = session.get("user")
    con = sqlite3.connect(curdir + "\Project.db")
    cur = con.cursor()
    if cur.execute("SELECT name FROM classes WHERE
username=:username",{ "username":username }).fetchone():
        classes = cur.execute("SELECT name FROM classes WHERE
username=:username",{ "username":username }).fetchall()
        con.commit()
        con.close()
        return render_template("teacher.html",classes=classes)
    else:
        return render_template("teacher.html")

@app.route("/student")
def student():
    username = session.get("user")
    con = sqlite3.connect(curdir + "\Project.db")
    cur = con.cursor()
    classes = cur.execute("SELECT classname FROM studentinfo WHERE
username=:username",{ "username":username }).fetchall()

```

```

        con.commit()

        con.close()

        return render_template("student.html",classes=classes)

@app.route("/studjoin")
def studjoin():
    return render_template("studjoin.html")

@app.route("/classjoin", methods = ["POST","GET"])
def classjoin():
    classname = request.form["name"]
    #session["myclass"] = classname
    username = session.get("user")
    con = sqlite3.connect(curdir + "\\Project.db")
    cur = con.cursor()

    classes = cur.execute("SELECT classname FROM studentinfo WHERE
username=:username",{ "username":username}).fetchall()

    tclasses = cur.execute("SELECT name FROM classes").fetchall()
    con.commit()
    con.close()
    n=len(classes)
    tn = len(tclasses)
    suc1 = False
    suc2 = False
    count = 0
    for j in range(n):
        if classname == classes[j][0]:
            suc1 = True
            flash("you already joined the class","danger")

```

```

        return render_template("student.html",classes=classes)
    else:
        pass
    for j in range(tn):
        if classname != tclasses[j][0]:
            count = count + 1
            if count == tn:
                suc2 = True
                flash("No Such Class Found!", "danger")
                return render_template("student.html",classes=classes)
        else:
            if suc1== False and suc2== False:
                con = sqlite3.connect(curdir + "\Project.db")
                cur = con.cursor()
                cur.execute("INSERT INTO studentinfo VALUES
(?,?)",(classname,username,))
                con.commit()

                classes = cur.execute("SELECT classname FROM studentinfo
WHERE username=:username",{ "username":username }).fetchall()
                con.close()
                flash("joined class successfully","success")
                return render_template("student.html",classes=classes)

@app.route("/classopen",methods = ["POST","GET"])
def classopen():
    session["myclass"]= request.form["sclass"]
    return render_template("classopen.html")

@app.route("/quizdisplay")
def quizdisplay():

```

```

        classname=session.get("myclass")

        con = sqlite3.connect(curdir + "\\Project.db")

        cur = con.cursor()

        quizzes=cur.execute("SELECT Quiz from Quizzes WHERE
classname=:classname",{ "classname":classname}).fetchall()

        con.commit()

        con.close()

        return render_template("classopen.html",quizzes=quizzes,name=classname)

@app.route("/studjoined")
def studjoined():

    return render_template("studjoined.html")

@app.route("/quizjoined", methods = ["POST","GET"])
def quizjoined():

    return render_template("quizjoined.html")

@app.route("/quizzattempt",methods =["POST","GET"])
def quizzattempt():

    quizname = request.form["name"]

    session["quizname"] = quizname

    username = session.get("user")

    classname = session.get("myclass")

    con = sqlite3.connect(curdir + "\\Project.db")

    cur = con.cursor()

    qname = cur.execute("SELECT Quiz FROM Quizzes WHERE
classname=:classname",{ "classname":classname}).fetchall()

    aqname = cur.execute("SELECT quizname FROM Result WHERE
username=:username AND
quizname=:quizname",{ "username":username,"quizname":quizname}).fetchall()

```

```

l = len(qname)
l1 = len(aqname)
suc = False
suc1 = False
count = 0
for j in range(l):
    if quizname != qname[j][0]:
        count = count + 1
        if count == l:
            suc = True
            flash("No Such Quiz Found!", "danger")
            return render_template("classopen.html")
for j in range(l1):
    if quizname == aqname[j][0]:
        suc1 = True
        flash("Quiz already attempted!", "danger")
        return render_template("classopen.html")

if suc == False and suc1 == False:
    table = tuple(cur.execute("SELECT * FROM "+quizname))
    con.close()

    global
    q1,q1o1,q1o2,q1o3,q1o4,q2,q2o1,q2o2,q2o3,q2o4,q3,q3o1,q3o2,q3o3,q3o4,q4,q4o1
    ,q4o2,q4o3,q4o4,q5,q5o1,q5o2,q5o3,q5o4

    global q1c,q2c,q3c,q4c,q5c
    q1,q1o1,q1o2,q1o3,q1o4,q1c =
table[0][0],table[0][1],table[0][2],table[0][3],table[0][4],table[0][5]
    q2,q2o1,q2o2,q2o3,q2o4,q2c =
table[1][0],table[1][1],table[1][2],table[1][3],table[1][4],table[1][5]

```

```

        q3,q3o1,q3o2,q3o3,q3o4,q3c =
table[2][0],table[2][1],table[2][2],table[2][3],table[2][4],table[2][5]

        q4,q4o1,q4o2,q4o3,q4o4,q4c =
table[3][0],table[3][1],table[3][2],table[3][3],table[3][4],table[3][5]

        q5,q5o1,q5o2,q5o3,q5o4,q5c =
table[4][0],table[4][1],table[4][2],table[4][3],table[4][4],table[4][5]

    return render_template("studquiz.html",

        q1=q1,q1o1=q1o1,q1o2=q1o2,q1o3=q1o3,q1o4=q1o4,q1c=q1c,
        q2=q2,q2o1=q2o1,q2o2=q2o2,q2o3=q2o3,q2o4=q2o4,q2c=q2c,
        q3=q3,q3o1=q3o1,q3o2=q3o2,q3o3=q3o3,q3o4=q3o4,q3c=q3c,
        q4=q4,q4o1=q4o1,q4o2=q4o2,q4o3=q4o3,q4o4=q4o4,q4c=q4c,

        q5=q5,q5o1=q5o1,q5o2=q5o2,q5o3=q5o3,q5o4=q5o4,q5c=q5c,quizname=quizname
    )

@app.route("/attemptedquiz",methods=["POST","GET"])
def attemptedquiz():
    username=session.get("user")
    quizname=session.get("quizname")
    classname=session.get("myclass")
    q1s = request.form["q1"]
    session["q1"]=q1s
    q2s = request.form["q2"]
    session["q2"]=q2s
    q3s = request.form["q3"]
    session["q3"]=q3s
    q4s = request.form["q4"]
    session["q4"]=q4s
    q5s = request.form["q5"]
    session["q5"]=q5s
    con = sqlite3.connect(curdir + "\\Project.db")

```

```

        cur = con.cursor()

        cur.execute("INSERT INTO attempted VALUES
(?,?,?)",(username,quizname,classname,))

        con.commit()

        con.close()

        return redirect(url_for('submitquiz'))

@app.route("/quizadd", methods = ["POST","GET"])
def quizadd():

    ques1 =
(request.form["q1"],request.form["q1o1"],request.form["q1o2"],request.form["q1o3"
],request.form["q1o4"],request.form["q1c"])

    ques2 =
(request.form["q2"],request.form["q2o1"],request.form["q2o2"],request.form["q2o3"
],request.form["q2o4"],request.form["q2c"])

    ques3 =
(request.form["q3"],request.form["q3o1"],request.form["q3o2"],request.form["q3o3"
],request.form["q3o4"],request.form["q3c"])

    ques4 =
(request.form["q4"],request.form["q4o1"],request.form["q4o2"],request.form["q4o3"
],request.form["q4o4"],request.form["q4c"])

    ques5 =
(request.form["q5"],request.form["q5o1"],request.form["q5o2"],request.form["q5o3"
],request.form["q5o4"],request.form["q5c"])

    clist = [ques1,ques2,ques3,ques4,ques5]

    now = datetime.now()

    quiz = request.form["quizname"] + now.strftime("%d%m%Y%H%M%S")

    #cname = session.get("class")

    cname = request.form["classname"]

    con= sqlite3.connect(curdir + "\\Project.db")

    cur = con.cursor()

    quizzes = cur.execute("SELECT Quiz FROM Quizzes").fetchall()

    con.commit()

```

```

    if quiz in quizzes:
        flash("Quiz name already exists","danger")
        return render_template("addquiz.html")
    else:
        cur.execute("INSERT INTO Quizzes VALUES (?,?)",(quiz,cname,))
        con.commit()

        cur.execute("CREATE TABLE " + quiz + "(ques text,op1 text,op2 text,
op3 text,op4 text,cop text)")

        query1 = "INSERT INTO " + quiz + " (ques,op1,op2,op3,op4,cop)
VALUES (?, ?, ?, ?, ?, ?);"

        cur.executemany(query1,clist,)
        con.commit()
        con.close()

        flash("quiz added succesfully","success")
        return redirect("teacher")

@app.route("/previous1", methods = ["POST","GET"])
def previous1():
    username=session.get("user")
    classname=session.get("myclass")
    con= sqlite3.connect(curdir + "\Project.db")
    cur = con.cursor()

    quizzes = cur.execute("SELECT quizname FROM attempted WHERE
classname=:classname AND
username=:username",{ "classname":classname,"username":username}).fetchall()

    con.commit()
    con.close()

    return
    render_template("previousquizes.html",quizzes=quizzes,username=username)

```



```

@app.route("/submitquiz", methods = ["POST","GET"])
def submitquiz():
    count = 0
    username = session.get("user")
    quizname = session.get("quizname")
    q1s=session.get("q1")
    q2s=session.get("q2")
    q3s=session.get("q3")
    q4s=session.get("q4")
    q5s=session.get("q5")
    con = sqlite3.connect(curdir + "\\Project.db")
    cur = con.cursor()
    clist = cur.execute("SELECT cop FROM " + quizname).fetchall()
    q1c = clist[0][0]
    q2c = clist[1][0]
    q3c = clist[2][0]
    q4c = clist[3][0]
    q5c = clist[4][0]
    if q1s == q1c:
        count = count + 1
    if q2s == q2c:
        count = count + 1
    if q3s == q3c:
        count = count + 1
    if q4s == q4c:
        count = count + 1
    if q5s == q5c:
        count = count + 1
    con = sqlite3.connect(curdir + "\\Project.db")

```

```

        cur = con.cursor()

        cur.execute("INSERT INTO Result VALUES
        (?, ?, ?)", (username, quizname, count,))

        con.commit()

        con.close()

        return render_template("submitquiz.html",
        q1s=q1s, q2s=q2s, q3s=q3s, q4s=q4s, q5s=q5s,
        q1=q1, q1o1=q1o1, q1o2=q1o2, q1o3=q1o3, q1o4=q1o4, q1c=q1c,
        q2=q2, q2o1=q2o1, q2o2=q2o2, q2o3=q2o3, q2o4=q2o4, q2c=q2c,
        q3=q3, q3o1=q3o1, q3o2=q3o2, q3o3=q3o3, q3o4=q3o4, q3c=q3c,
        q4=q4, q4o1=q4o1, q4o2=q4o2, q4o3=q4o3, q4o4=q4o4, q4c=q4c,
        q5=q5, q5o1=q5o1, q5o2=q5o2, q5o3=q5o3, q5o4=q5o4, q5c=q5c,
        score=count)

```

```

@app.route("/result", methods=["POST", "GET"])
def studentresult():

    quizname=request.form["qname"]

    classname =session.get("myclass")

    con = sqlite3.connect(curdir + "\Project.db")

    cur = con.cursor()

    username = session.get("user")

    count = cur.execute("SELECT score FROM Result WHERE
username=:username AND
quizname=:quizname", {"username":username, "quizname":quizname}).fetchone()

    score = count[0][0]

    ques = cur.execute("SELECT ques FROM " + quizname).fetchall()

    cop = cur.execute("SELECT cop FROM " + quizname).fetchall()

    con.commit()

    con.close()

```

```
        return  
render_template("quizdisplay.html",quizname=quizname,username=username,score  
=score,ques=ques,cop=cop)
```

```
@app.route("/classs", methods=["POST","GET"])
```

```
def classs():
```

```
    #global my_class
```

```
    #my_class = request.form["my_class"]
```

```
    session["class_my"] = request.form["my_class"]
```

```
    return render_template("quizadd.html")
```

```
@app.route("/addclass")
```

```
def addclass():
```

```
    return render_template("addclass.html")
```

```
@app.route("/classadd", methods = ["POST","GET"])
```

```
def classadded():
```

```
    name = request.form["name"]
```

```
    #session["class"] = name
```

```
    username = session.get("user")
```

```
    con = sqlite3.connect(curdir + "\Project.db")
```

```
    cur = con.cursor()
```

```
    classes = cur.execute("SELECT name FROM classes").fetchall()
```

```
    n=len(classes)
```

```
    suc = False
```

```
    for j in range(n):
```

```
        if name == classes[j][0]:
```

```
            flash("Class name already exists","danger")
```

```
            suc=True
```

```
            return render_template("teacher.html")
```

```

        else:
            pass
    if suc == False:
        cur.execute("INSERT INTO classes VALUES (?,?)",(name,username,))
        con.commit()
        flash("class added successfully","success")
        return render_template("addquiz.html")

@app.route("/addquiz", methods=["POST","GET"])
def addquiz():
    return render_template("addquiz.html")

@app.route("/prac")
def prac():
    return render_template("prac.html")

@app.route("/previous",methods = ["POST","GET"])
def previousQuiz():
    con = sqlite3.connect(curdir + "\\Project.db")
    cur = con.cursor()
    classname = session.get("class_my")
    #classname = request.form["classname"]
    prevquiz = cur.execute("SELECT Quiz FROM Quizzes WHERE
classname=:classname",{ "classname":classname}).fetchall()
    return render_template("prevquiz.html",prevquiz = prevquiz)

@app.route("/tresult",methods=["POST","GET"])
def tresult():
    quizname = request.form["qname"]

```

```
con = sqlite3.connect(curdir + "\\Project.db")

cur = con.cursor()

username = cur.execute("SELECT username FROM Result WHERE
quizname=:quizname",{ "quizname":quizname}).fetchall()

result = cur.execute("SELECT score FROM Result WHERE
quizname=:quizname",{ "quizname":quizname}).fetchall()

l = len(username)

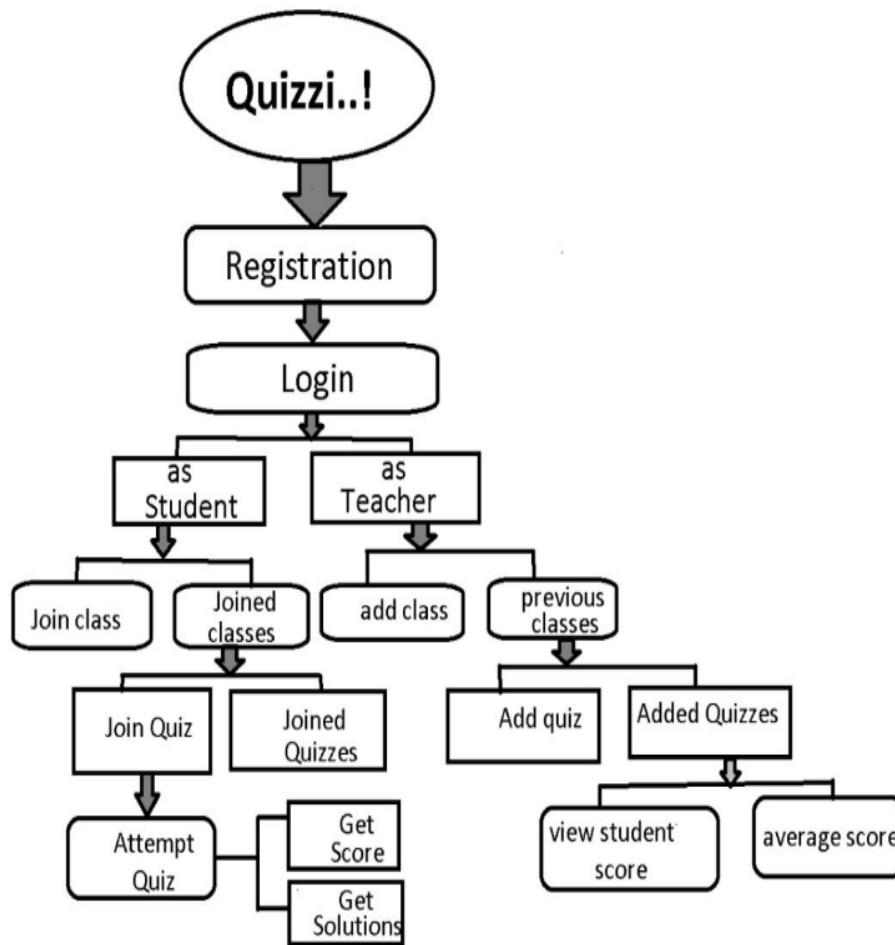
return
render_template("resultdisplay.html",username=username,result=result,l=l)

if __name__ == "__main__":

    app.run(debug=True)
```

6. SYSTEM DESIGN

6.1 User's Flow in the Website



Home page

In-home there will be a navigation bar that consists of a logo and various navigation bar sections like . Home, Registration, Login, and About.

Registration page

It contains some fields like username,password,register as.

User can select any one option(Teacher/Student) in register as field.After registration it shows message Like Register is successfully done.

Login page

After registration the user can able to login into the application by entering username,password.

Addclass page

After login into the application the teacher can able to create classes with a Specific classname,

Addquiz page

After creating classes the teacher can able to create quiz in already created class by Classname and quizname.

Previous quizzes page

Which are the previous added quizzes are displayed in previous added quizzes page.

Joinclass page

If the user can select student ,then he can join the classes with classname which is Created by teacher.

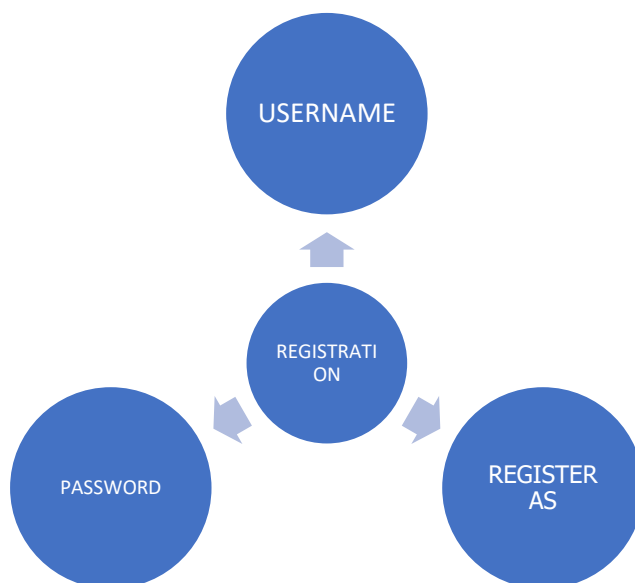
Joinquiz page

After joining the classes the student can able to attempt the quizzes by classname \$ quizname

Score page

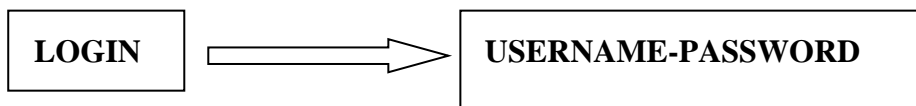
After attempting the quizzes the score is visible to the students,how many right answers and wrong answers he put .

6.2 REGISTRATION

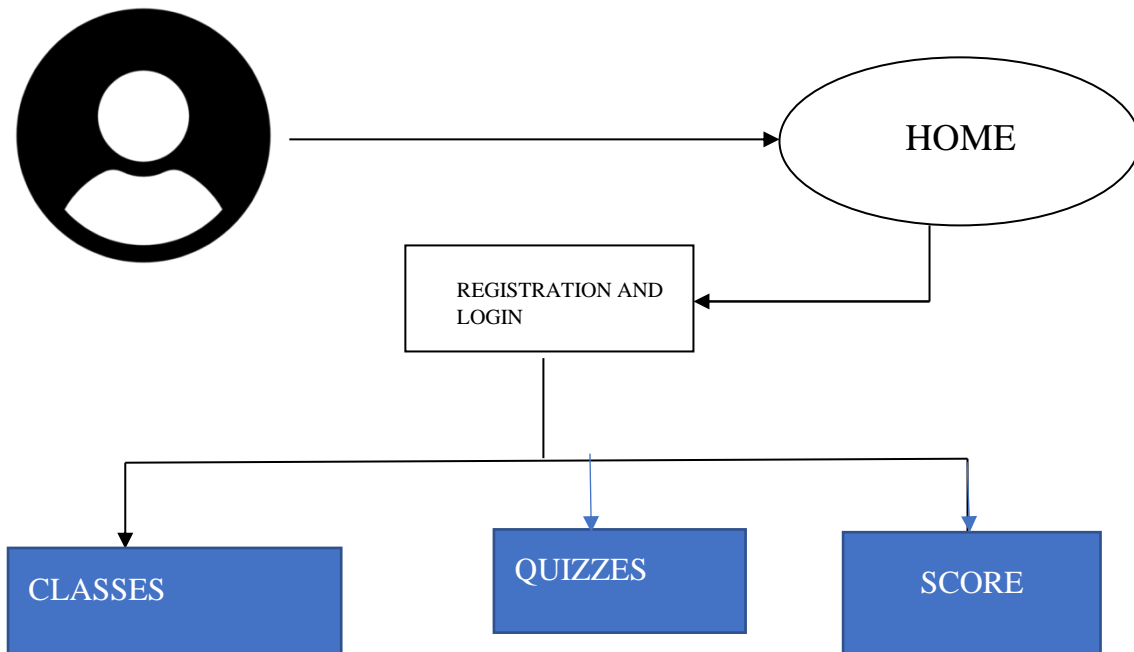


:

6.3 LOGIN



6.4 USER'S SEQUENCE



7. TESTING

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to variety of tests-on- line response, Volume Street, recovery and security and usability test. A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “al gears mesh”, that is the internal operation of the product

performs according to the specification and all internal components have been adequately exercised.

7.1 UNIT TESTING

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’. The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

7.2 INTEGRATION TESTING

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

- i) Top-down integration testing.
- ii) Bottom-up integration testing.

7.3 The Box Approach:

Software testing methods are traditionally divided into white Box Testing & Black Box Testing these two approaches are used to describe the point of view of the test engineer, when he is assigned with an application to test and design the test cases. WHITE BOX TESTING APPROACH:

7.3.1 White Box Testing:

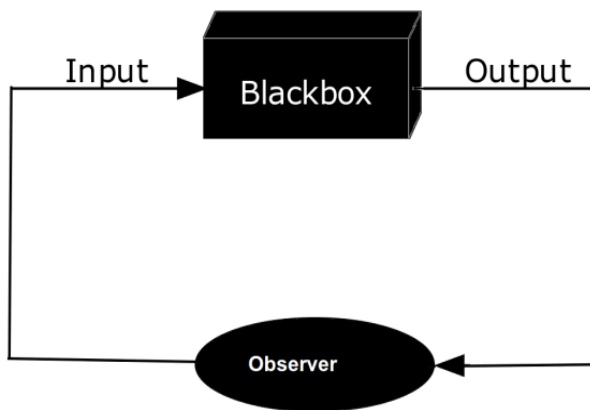
Transparent Box Testing, and Structure Testing tests internal structures of the application or working of a program. In White Box Testing, an internal perspective of the system as well as programming skills is used to design test cases. While White box testing can be applied at unit, integrated levels of the system, it is generally preferred to be done at unit level.



7.3.2 Black Box testing:

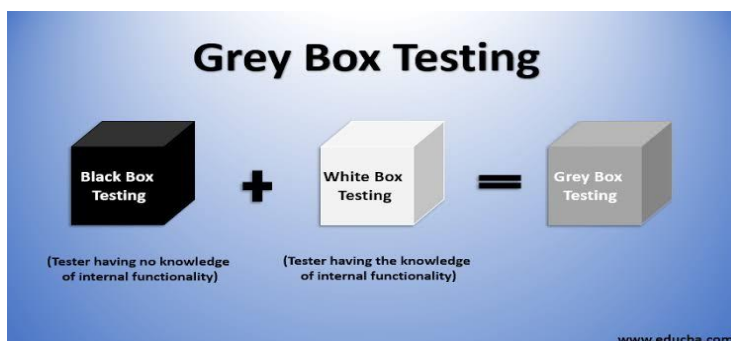
- ✓ Black box testing is done to find incorrect or missing function
- ✓ Interface error
- ✓ Errors in external database access
- ✓ Performance errors
- ✓ Initialization and termination errors

This type of testing approach is mostly preferred, as it is the most convenient type of software Testing. Here the entire Software product is taken as a single component, i.e. a Black Box, and then its functionality is tested without any knowledge of its internal implementation. The person testing has to be aware what the application is supposed to do, but not how it does what is supposed to do.



7.3.3 Gray Box Testing:

The most sophisticated and complicated way of Software Testing is the Grey Box Testing. This testing method involves checking, which needs knowledge of the entire internal data structure and algorithms hired to develop that particular application. Once known, those components are taken either at Black or White Level again, and are tested by the assigned tester. The advantages of this level of testing is that the tester is not required to have access to full source of the software he is testing Only handling him the algorithms and implementation details will do the work. In general it is the combination of Black box and white box testing



7.4 VALIDATION TESTING

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

7.5 USER ACCEPTANCE TESTING

User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by

constantly keeping in touch with prospective system at the time of developing changes whenever required.

7.6 OUTPUT TESTING

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs.

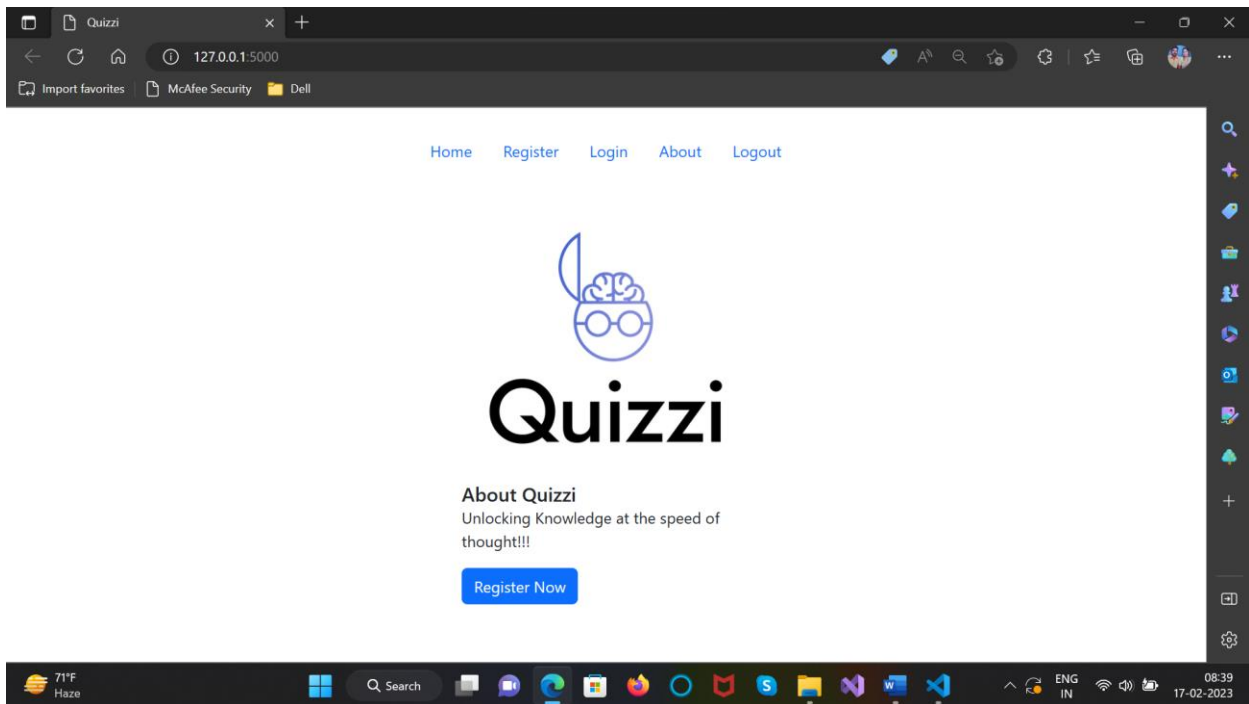
7.7 IMPLEMENTATION OF TEST CASES

Test Case No.	Description	Test Data	Expected Output	Actual Output	Status (pass/fail)
#1	Verify response When a valid email Address and password is used during login.	Username: yankithareddy@gmail.com Password: **34	Login Successful!!	Login Successful!!	Pass
#2	Verify response when an invalid email address or password is used during login.	Username: ankithareddy@gmail.com Password: **34	No Such user!!	No Such user!!	Pass
#3	Verify response when a user log in as teacher and create a class which doesn't exist already.	Class name: Java	Class added Successfully!	Class added Successfully!	Pass
#4	Verify response when a User log in as teacher and create a class which already exists.	Class name: Java	Class name already exists!	Class name already exists!	Pass
#5	Verify response when a User log in as student and join a class which was not joined earlier.	Class name: Java	Class joined Successfully!	Class joined Successfully!	Pass
#6	Verify response when a User log in as student and join a class which was joined already.	Class name: Java	Class already joined!	Class already joined!	Pass
#7	Verify response when a User log in as teacher And create quizzes in the Class.	Quiz name: Features of Java	Quiz added Successfully!	Quiz added Successfully!	Pass

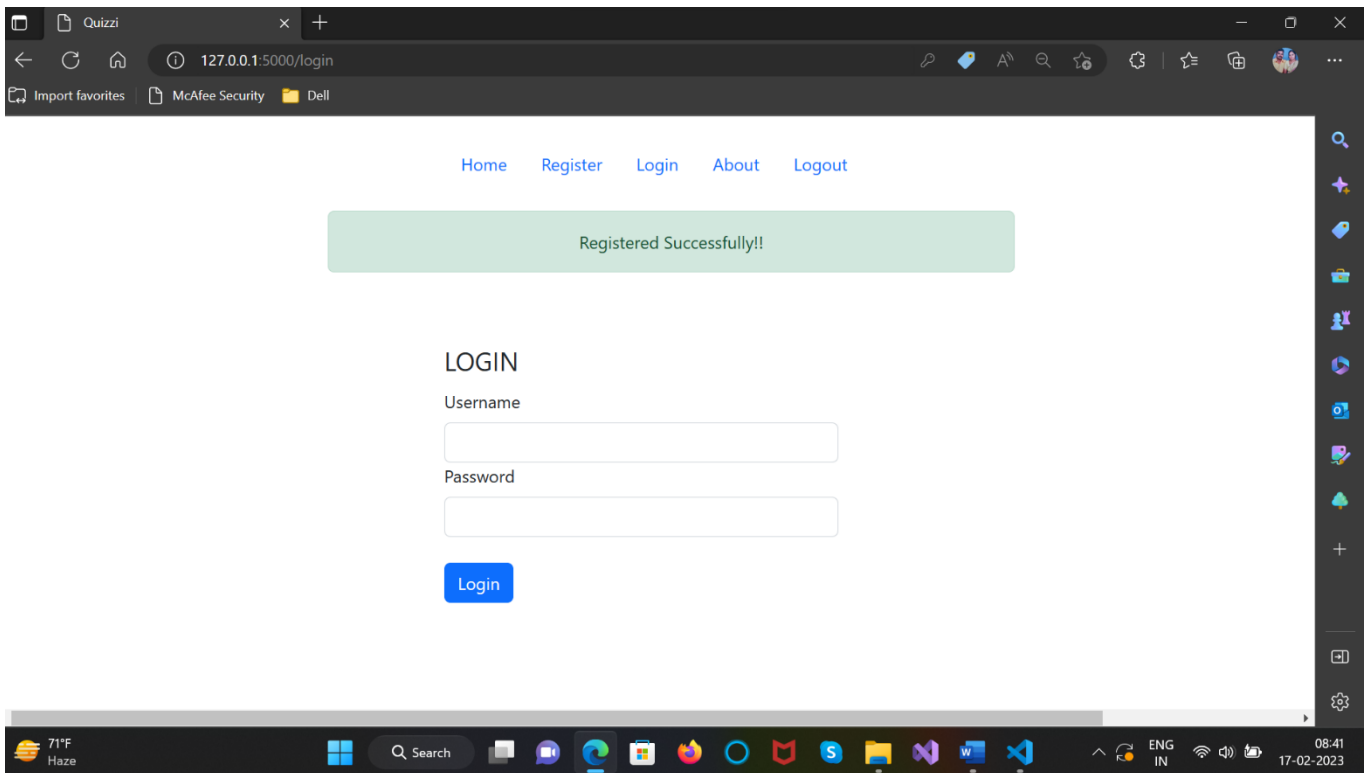
#8	Verify response when a User log in as student and attempt a quiz.	Quiz name: Features of Java	Your Score: **	Your Score: **	Pass
#9	Verify the functionality Of if-else if-else Statements when a user Login as teacher/student.	Username: yankithareddy@gmail.com Or Username: reddytlikky@gmail.com	Visibility of Joined classes Or Visibility of Created Classes	Visibility of Joined classes Or Visibility of Created Classes	Pass
#10	Verify the functionality Of score calculation of Score when a student attempted quiz.	Attempting 3 correct answers out Of 5 questions.	Your score:3 (with summary Of the Quiz)	Your score:3 (with summary Of the Quiz)	Pass

8. OUTPUT SCREENS

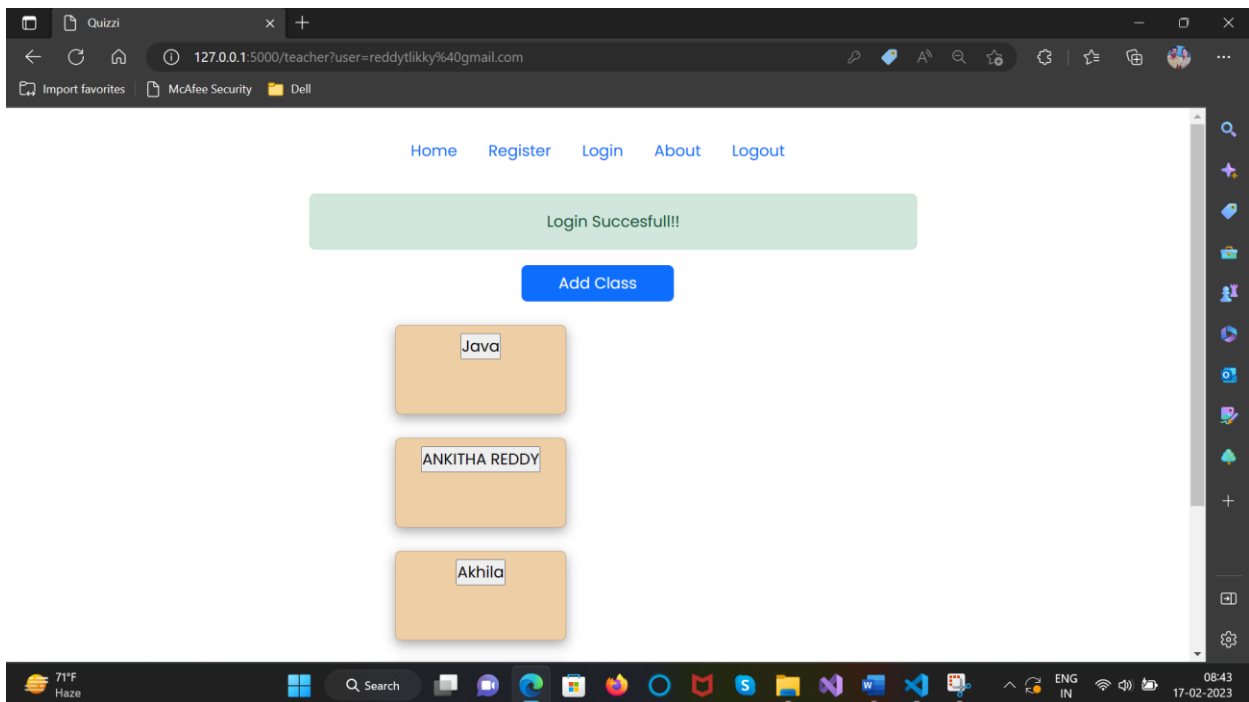
8.1 HOME PAGE



8.1.2 REGISTRATION

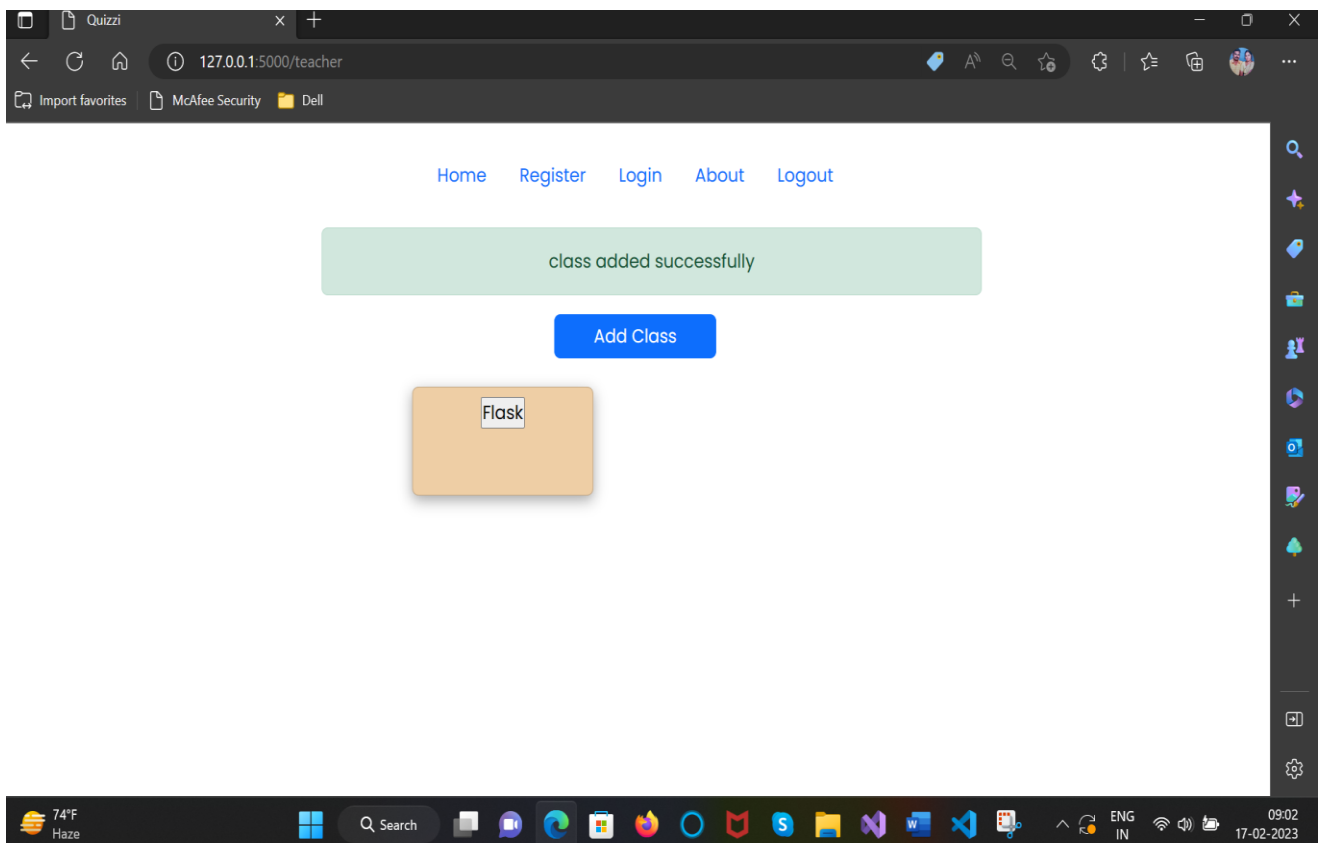


8.1.3 LOGIN

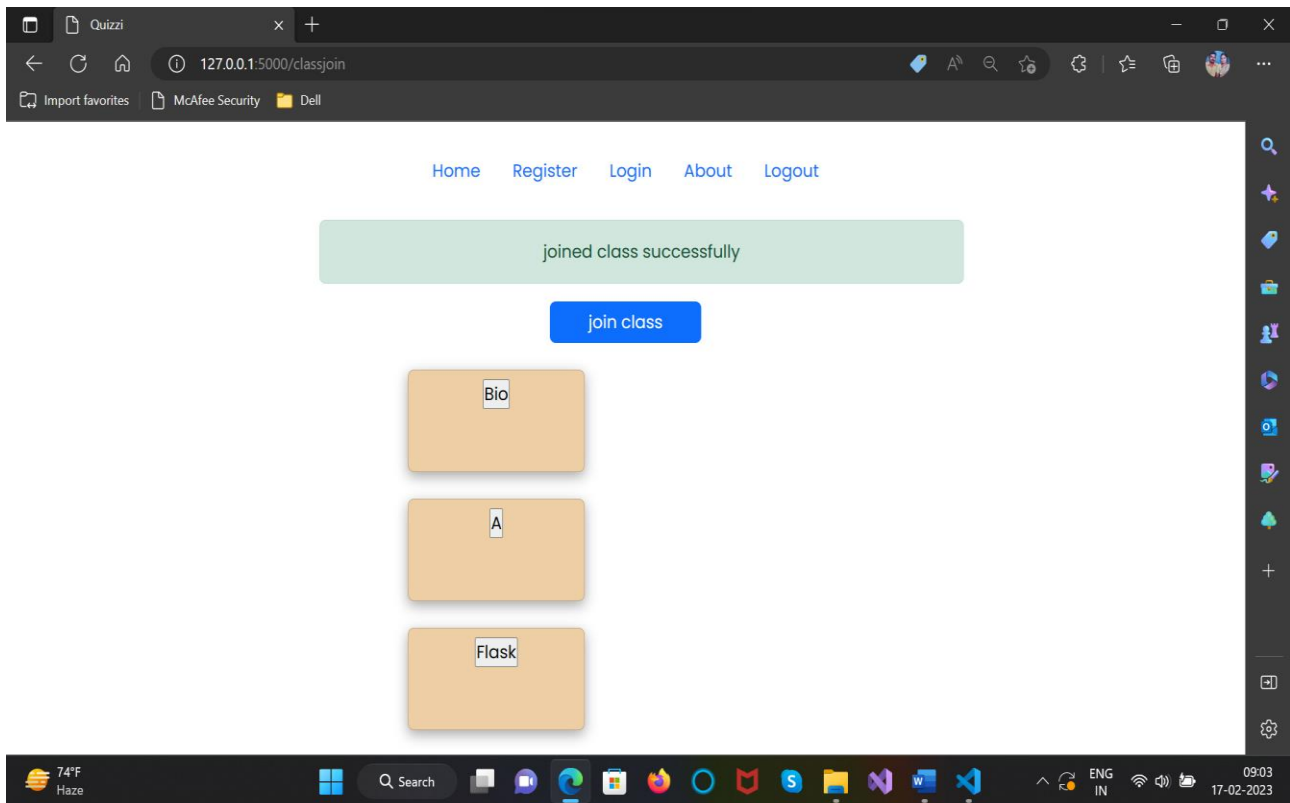


8.2 CLASSES MANAGEMENT

8.2.1 ADD CLASS



8.2.2 JOIN CLASS



8.3 QUIZZES MANAGEMENT

8.3.1 ADD QUIZ(by teacher)

[Not Now](#)

Quiz name:

Class name:

Question 1:

Options:

Question 2:

Options:

Question 3:

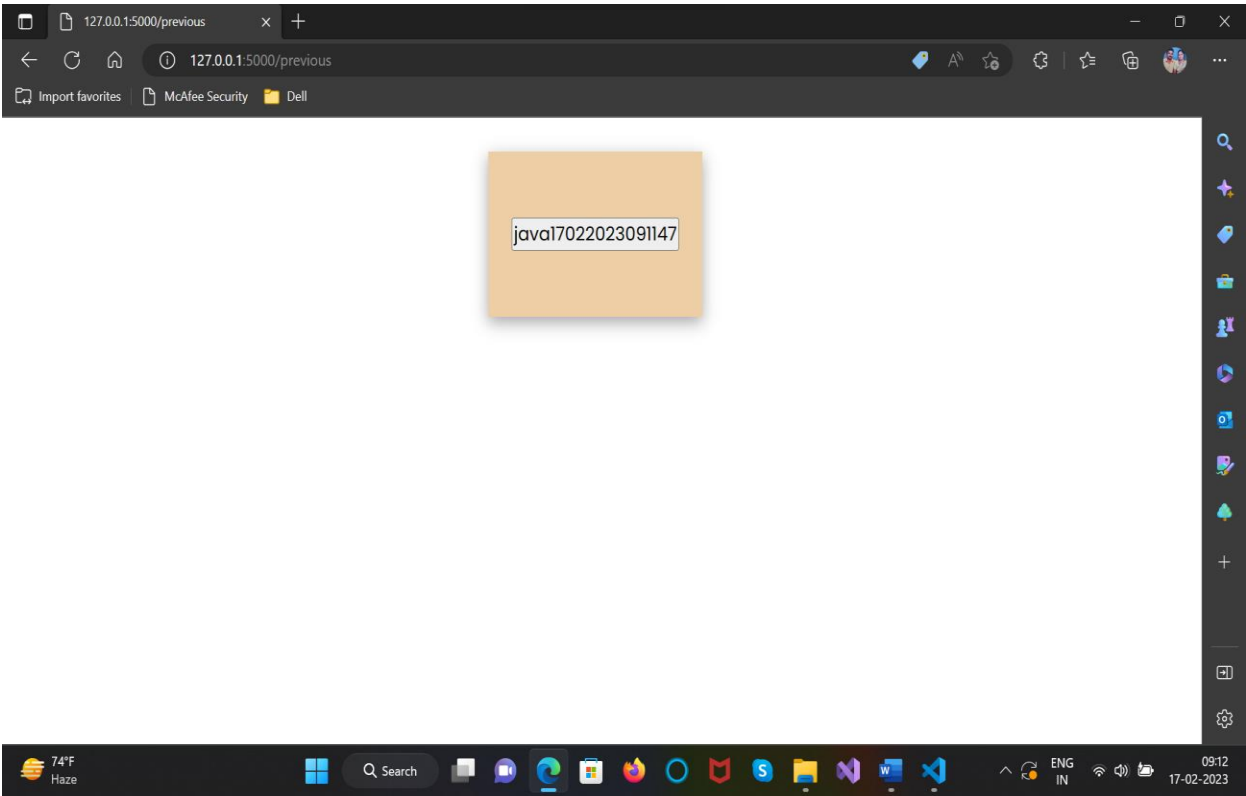
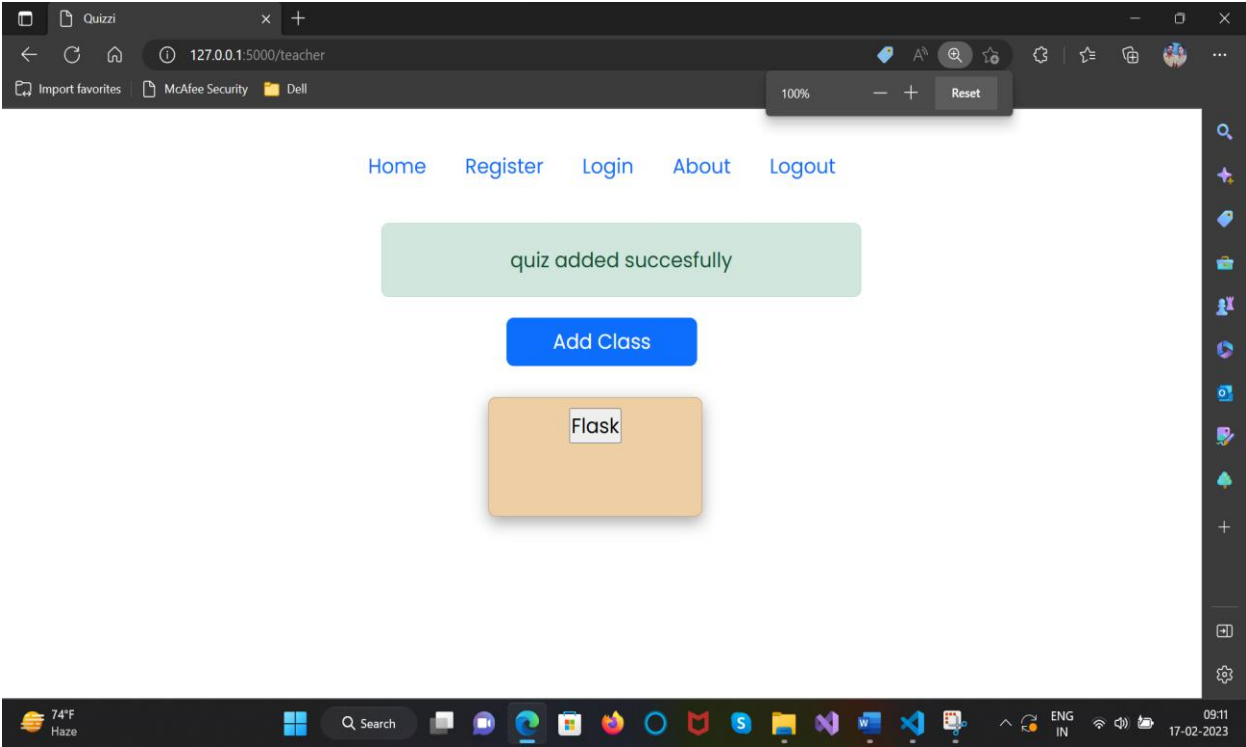
Options:

Question 4:

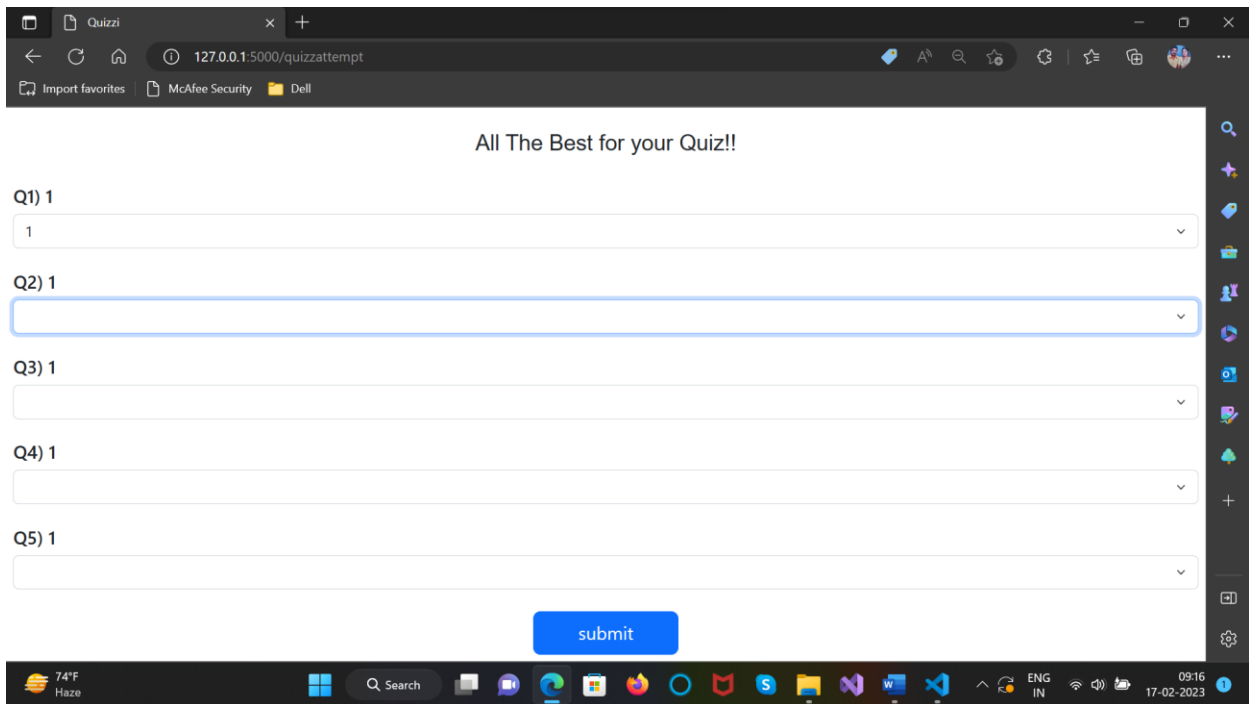
Options:

Question 5:

8.3.2 MANAGE QUIZZES(by teacher)



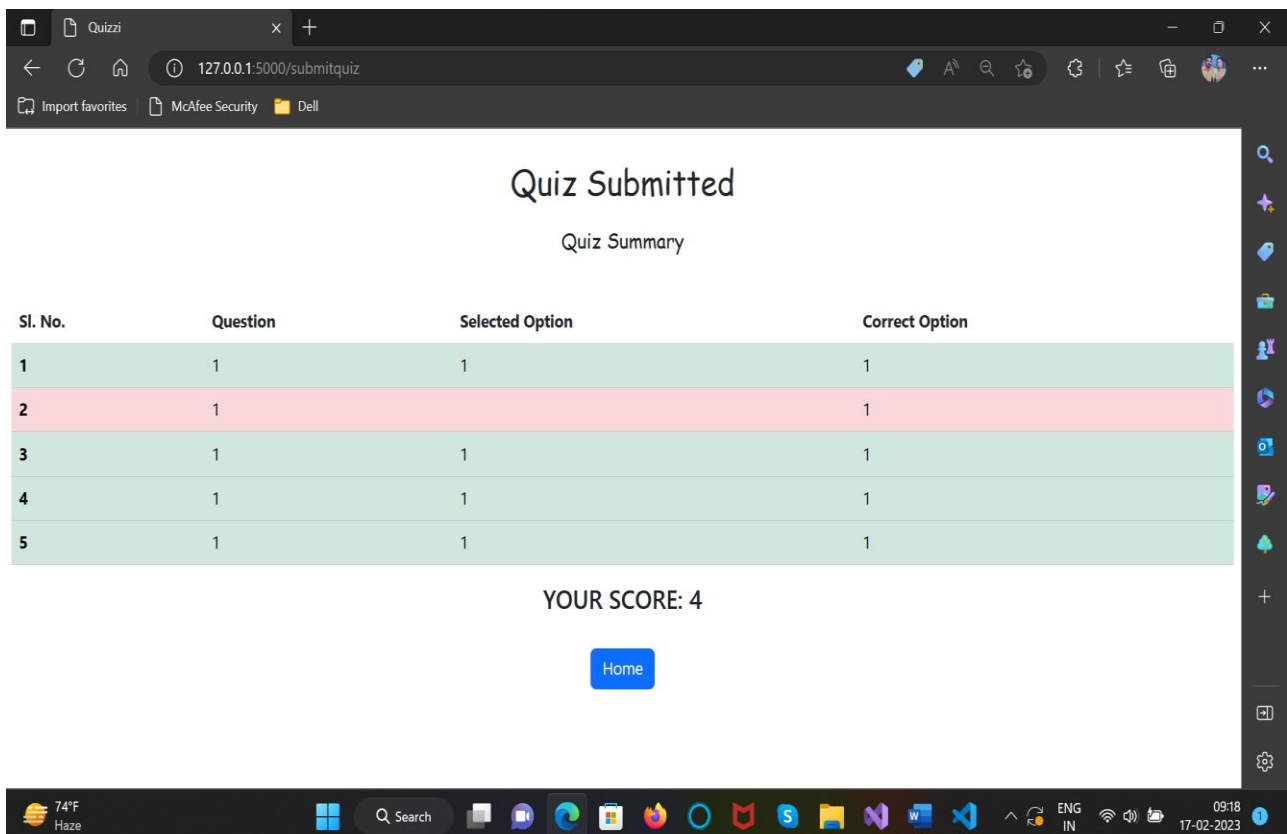
8.3.3 ATTEMPT QUIZ (by Student)



The screenshot shows a web browser window with the URL `127.0.0.1:5000/quizzattempt`. The page displays a message "All The Best for your Quiz!!" and five question input fields labeled Q1) 1, Q2) 1, Q3) 1, Q4) 1, and Q5) 1. Each field contains the number "1". A blue "submit" button is located at the bottom center of the form area. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates a temperature of 74°F, weather of Haze, and the date/time as 09:16 on 17-02-2023.

8.4 GRADING

8.4.1 SCORE CALCULATION AND QUIZ SUMMARY



The screenshot shows a web browser window with the URL `127.0.0.1:5000/submitquiz`. The page displays a message "Quiz Submitted" and a "Quiz Summary" table. The table has four columns: "Sl. No.", "Question", "Selected Option", and "Correct Option". It contains five rows of data, all showing "1" for both the selected and correct options. Below the table, the text "YOUR SCORE: 4" is displayed, followed by a blue "Home" button. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons. The system tray on the right indicates a temperature of 74°F, weather of Haze, and the date/time as 09:18 on 17-02-2023.

Sl. No.	Question	Selected Option	Correct Option
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1

YOUR SCORE: 4

Home

8.4.2 RETRIEVAL OF SCORE (by teacher)

Quizzi

127.0.0.1:5000/tresult

Import favoritesMcAfee SecurityDell

Student Results

Home

SL. No	USERNAME	SCORE
1	studankitha@gmail.com	4

74°F
Haze

Search

ENG
IN09:20
17-02-2023

9. CONCLUSION

9.1 FUTURE SCOPE

Integration with other learning management systems (LMS): Online quiz applications can integrate with other LMS platforms, such as Canvas or Blackboard, to provide a seamless learning experience for students and teachers. This will allow for easier sharing of resources and assignments.

Enhanced features for personalized learning: Online quiz applications can use machine learning algorithms to analyze student data and provide personalized recommendations for further learning. This could include recommending specific study materials, quizzes, or practice exercises based on the student's performance.

Gamification: Gamification can be a powerful motivator for students, and online quiz applications can incorporate elements of gaming to make learning more engaging and fun. This could include badges, rewards, leaderboards, or other game-like features.

Virtual reality and augmented reality: As technology continues to improve, online quiz applications could incorporate virtual reality (VR) or augmented reality (AR) to provide immersive learning experiences. For example, a history quiz could take place in a virtual historical setting, or a science quiz could allow students to explore 3D models of molecules or organisms.

Collaboration and communication: Online quiz applications can facilitate collaboration and communication between students and teachers. This could include features such as discussion forums, group projects, and peer review.

Integration with AI-powered chatbots: Online quiz applications can integrate with AI-powered chatbots to provide personalized feedback and guidance to students. This can be especially useful for providing support outside of regular class hours.

9.2 IMPROVEMENTS TO BE DONE IN FUTURE

User interface and design: Improving the design and user interface can make the quiz project more user-friendly and engaging for students. This can include things like using clear and concise language, creating a visually appealing layout, and making it easy to navigate.

Mobile responsiveness: More and more people are accessing the internet through their mobile devices, so making the quiz project mobile-friendly is essential. This can include responsive design that adjusts to different screen sizes, mobile-specific features like touch-friendly buttons, and optimized loading times.

Accessibility: Ensuring that the quiz project is accessible to all students is crucial. This can include features like high-contrast modes for students with visual impairments, text-to-speech options, and keyboard navigation for students with mobility impairments.

Personalization: Personalization can enhance the student experience by providing customized content and feedback. This can include things like customized quizzes based on a student's interests or learning style, or personalized feedback based on their performance.

Integration with other educational tools: Integration with other educational tools, such as learning management systems (LMS) or student information systems (SIS), can make it easier for teachers to manage their classes and for students to access resources.

Security: Ensuring that the quiz project is secure is critical to protecting student data and ensuring the integrity of the assessment. This can include features like encryption, two-factor authentication, and access controls.

8. BIBLIOGRAPHY

Information for this project is gathered from google from the following websites:

<https://getbootstrap.com/docs/4.0/components/alerts/>

<https://w3cschoool.com/tutorial/flask-tutorial>

<https://www.w3schools.com/python/>

<https://www.geeksforgeeks.org/python-sqlite/>

<https://stackoverflow.com/questions/36439032/how-do-you-pass-through-a-python-variable-into-sqlite3-query>

<https://stackoverflow.com/questions/6986547/python-how-do-you-concatenate-time-to-a-string>

https://www.w3schools.com/TAgs/tag_html.asp