6COSC022W – Advanced Server-Side Web Programming

QUIZ WEBSITE

Student: Shanmugaratnam Mohanaranjan
(w1870584/18705841)

Lecturer: Dr Simon Courtenage

BEng Software Engineering degree

at the University of Westminster.

School of Computer Science & Engineering

University of Westminster

Date: **1pm Monday 11th March 2024**

Introduction

In this coursework, you will submit a report in which you will design a simple web application and critically discuss the technologies needed to implement it.

1. Choose what application to develop

You should choose which web application you will design from the following list:
1. A simple social networking application based on a shared interest: either photography, cartoons, books or holiday destinations
2. A technical question and answer website, where people can pose questions on technical issues or problems, and other people can help answer them
3. An investment forum where people can post about their favourite stock and shares, and predict whether a stock will go up or down
4. A quiz website that allows users to create quizzes and to take other people's quizzes on various topics

Having decided what application you will design, you should write a short introductory paragraph setting out why you chose this application and your idea about the main aim of the application.

2. A technical question and answer website, where people can pose questions on technical issues or problems, and other people can help answer them

The quiz website described above promotes a sense of community and intellectual engagement among its users by offering a platform for creating and taking quizzes on a variety of themes. Users can pursue their interests, push themselves, and share their knowledge with others, so contributing to a thriving community of learners and enthusiasts.

The quiz creation interface is easy to use, and users may create quizzes using a variety of content forms such as questions, multiple-choice responses, photographs, videos, and audio samples. Quizzes can be categorised by topic, making it simple for users to find quizzes that are relevant to their interests, whether in history, science, literature, movies, music, or sports. The portal also has a search function for finding popular and highly scored quizzes.

Users can interact with quizzes by answering questions, receiving rapid results, and rating them based on difficulty, relevancy, and fun. Users can use social sharing and engagement capabilities to share their quiz scores and participate in discussions with other members of the community. Furthermore, community moderation ensures that quizzes are of high quality and accurate, which improves the entire user experience.

In conclusion, A technical question and answer website, where people can pose questions on technical issues or problems, and other people can help answer them

this quiz website is a wonderful resource for entertainment, learning, and community participation, catering to a wide range of user interests while also encouraging knowledge exchange and friendly competition.

2. Decide requirements

The exact requirements for the application you choose are up to you. However, I suggest that your application should reasonably be expected to include:

a. account management – people should be able to register and create accounts, and login and logout

b. creating content in some way (for example, in the social network application, a user creates a post and others comment, or someone asks a question and another person answers, or someone upvotes an answer)

c. finding (in different ways, such as keywords or tags) and browsing content

(Real-world sites also include management or administrations interfaces - for example to ban users or delete content - but you do not need to do this for this coursework to keep it manageable).

Your requirements can be a short set of bullet points or a table, but should be grouped into those that are essential (those without which the application will not work), desirable (those that, while not essential, still add real value), and luxury (nice to have if there's any time to add them).

If there is anything non-obvious about a requirement, you should add an explanation to the bullet-point. Note that since this is a single coursework, the requirements should not resemble a full project! If you are unsure about whether you are doing too little or too much, then please ask. As a rule of thumb, your requirements should cover the basic points (a) – (c) above, but not go beyond that in any non- trivial way. Remember, you will have to implement your requirements in the next coursework, so don't set yourself too much to do.

Here are the requirements for the quiz website application:

Essential:

1. **Account Management:**
    - Users can register for new accounts.
    - Registered users can log in and log out of their accounts.
2. **Creating Quizzes:**
    - Registered users can create quizzes by providing questions and multiple-choice answers.
    - Quiz creators can assign categories and tags to their quizzes for easy browsing.
3. **Taking Quizzes:**
    - Users can take quizzes created by others.
    - The application should provide immediate feedback on quiz results.

4. **Browsing Quizzes:**
    - Users can browse quizzes based on categories, tags, or keywords.
    - Quizzes should be searchable and filterable for ease of navigation.

Desirable:

1. **User Profiles:**
    - Users can customize their profiles with avatars, bios, and other personal details.
    - Profiles display a user's created quizzes and quiz-taking history.
2. **Social Features:**
    - Users can like and share quizzes with others.
    - Users can follow other users to receive updates on their quiz activity.
3. **Leaderboards:**
    - The application can feature leaderboards to showcase top quiz creators and high scorers.

Luxury:

1. **Quiz Recommendations:**
    - The application suggests quizzes based on a user's quiz-taking history and preferences.
2. **Interactive Elements:**
    - Quizzes may include multimedia elements such as images, videos, or audio clips.
3. **Quiz Statistics:**
    - Users can view detailed statistics on their quiz performance, including average scores and completion rates.

Requirement	Description
User Registration and Authentication	Users should be able to register accounts and log in securely.
Quiz Creation Interface	Provide a user-friendly interface for creating quizzes with various content formats such as questions and media.
Categorization	Allow users to categorize quizzes by topic (e.g., history, science, literature) for easy navigation.
Search Functionality	Implement a search feature for users to discover trending and top-rated quizzes.
Quiz Taking and Evaluation	Users should be able to take quizzes, receive quick results, and evaluate them based on difficulty and enjoyment.
Social Sharing and Interactions	Enable users to share quiz results and engage in discussions through social media and within the platform.
Community Moderation	Implement moderation mechanisms to ensure the quality and correctness of quizzes submitted by users.
User Profiles and Activity Tracking	Provide user profiles where users can track their quiz activity, achievements, and contributions.
Responsive Design	Ensure the website is responsive and accessible across different devices and screen sizes.
Security Measures	Implement robust security measures to protect user data, prevent unauthorized access, and mitigate risks.
Feedback and Support	Offer channels for users to provide feedback, report issues, and seek assistance from support staff.

These requirements cover the basic functionalities of account management, creating, taking, and browsing quizzes while also including some additional features to enhance user experience and engagement.

3. Create an outline design

You should create an outline design for your application using (i) class diagrams, (ii) a database schema, and (iii) screen mockups. This design should be sufficient to allow a reader to know

a. What classes will need to be written
b. What database tables will be needed and what information they will contain (this includes field names and types)
c. What, roughly, the pages will look like and what information they will contain, as well as how, roughly, how one page is linked to another

The class diagrams should show controllers, models and views as classes, together with (a) the relationships between them, and (b) the main methods or functions in the controller and model classes.

The database schema can be represented diagrammatically (as shown on this page: https://database.guide/what-is-a-database-schema/), with relationships between tables where you think they need to exist, or you can use UML notation in the form of class diagrams (as you may have learnt to do on 5COSC020W Database Systems).

Screen mockups should resemble the wireframes on this page: https://www.smartdraw.com/website-wireframe/. This page also discusses sitemaps, which you will also find useful. Your wireframes should include some example data, rather than be entirely generic.

Again, if anything is non-obvious to a reader, you should briefly explain it

Outline Design for Quiz Application:

(i) Class Diagrams:

Admin	User	Quiz
- registerUser()	- userId	- createQuiz()
- loginUser()	- username	- takeQuiz()
- logoutUser()	- password	

Explanation:

- Admin: Manages user-related actions such as registration, login, and logout.
- User: Represents user data including user ID, username, and password.
- QuizController: Handles quiz-related actions like creating and taking quizzes.

(ii) Database Schema:

User Table
userId (PK)
username
password

Quiz Table
quizId (PK)
userId (FK)
title
category
tags

Question Table
questionId (PK)
quizId (FK)
question_text

Answer Table
answerId (PK)
questionId (FK)
answer_text
isCorrect

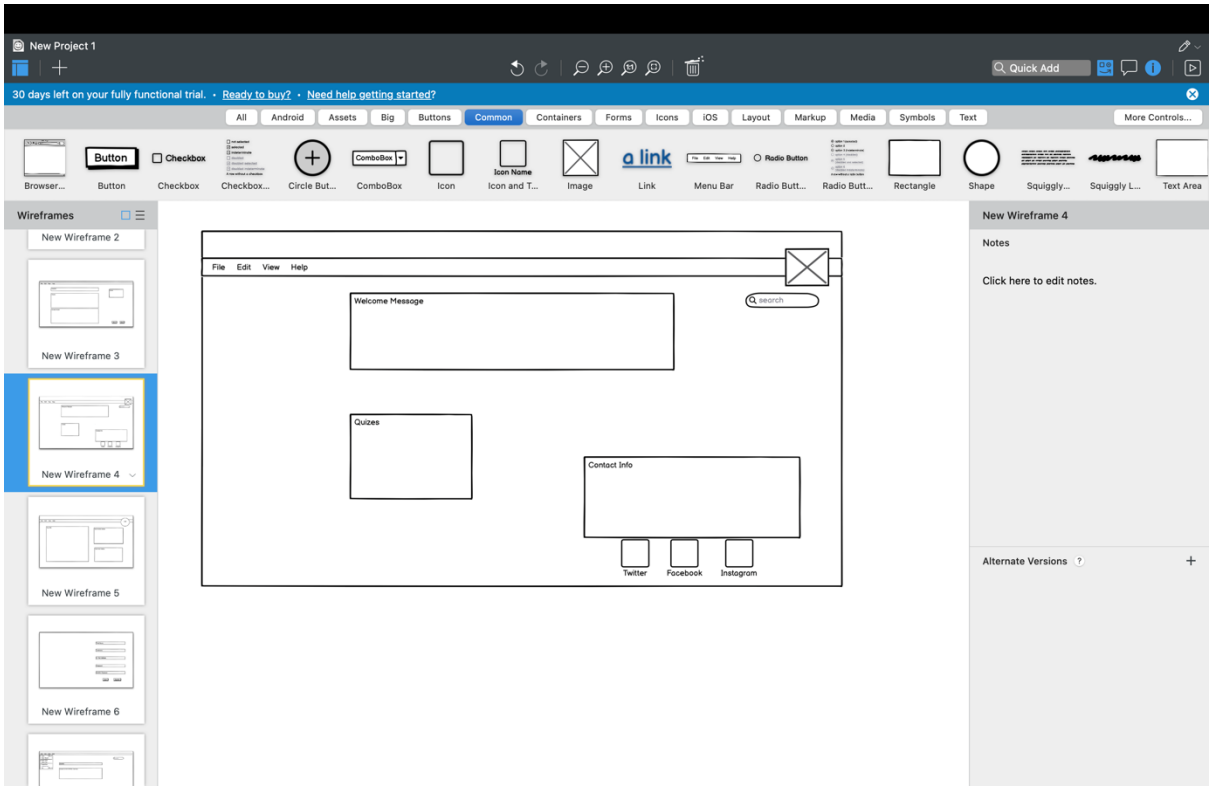
Explanation:

- User Table: Stores user information including their unique ID, username, and password.
- Quiz Table: Contains quiz details such as title, category, and tags, linked to the user who created it.
- Question Table: Holds questions associated with quizzes.
- Answer Table: Stores possible answers for each question along with correctness indicators.

(iii) Screen Mockups:

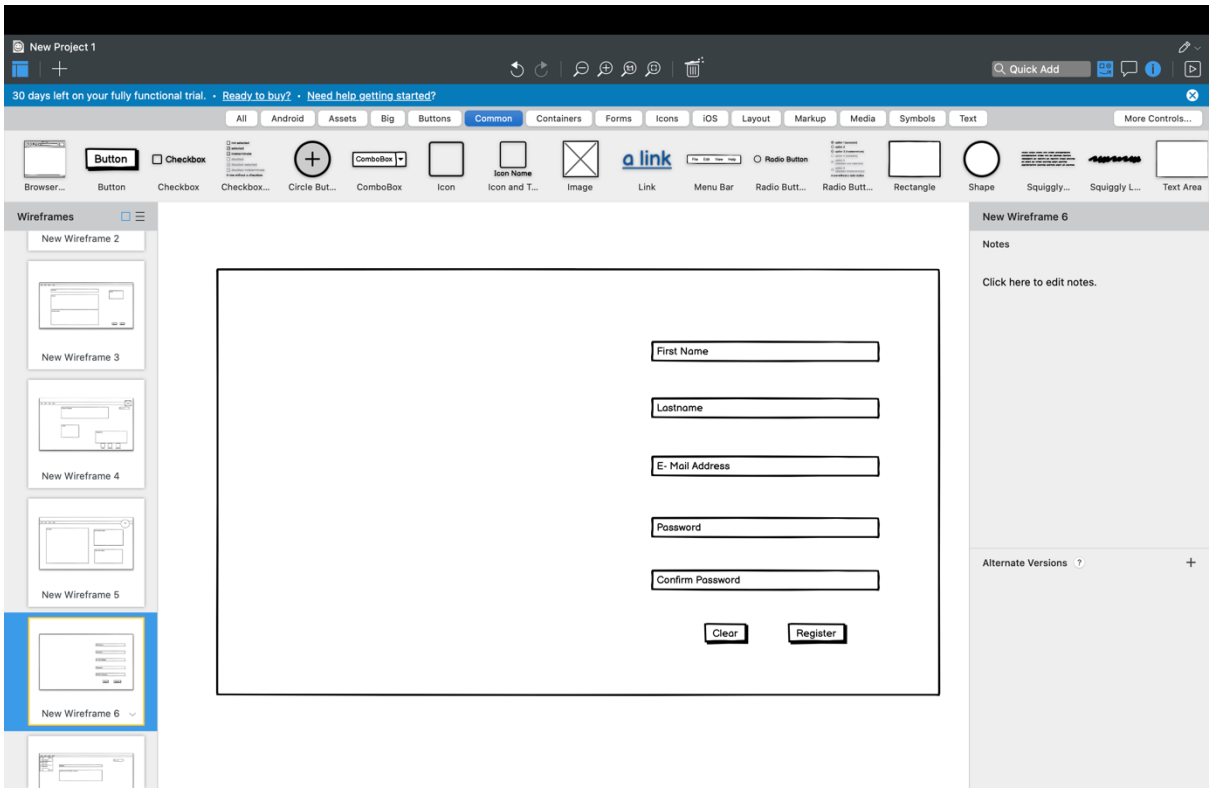
1. **Homepage:**

   - Features navigation links to login, register, and browse quizzes.
   - Displays trending quizzes and categories.



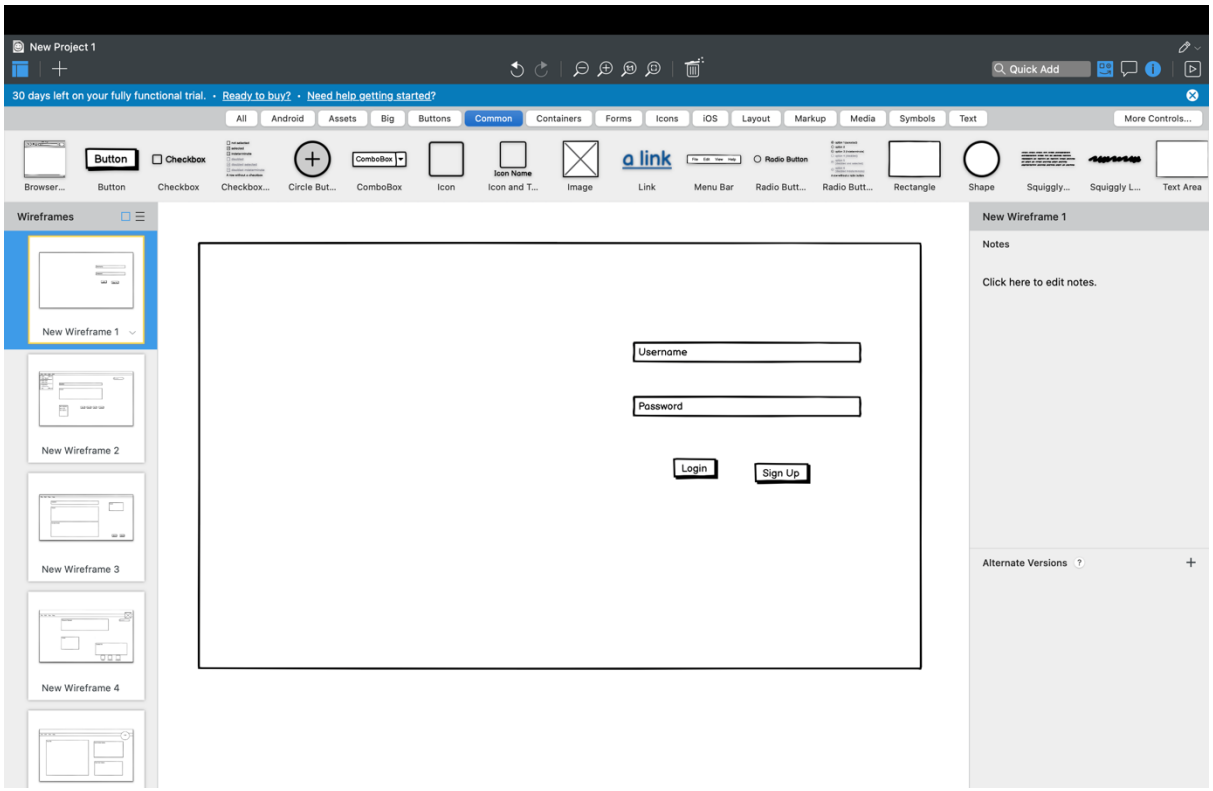
2. **Registration Page:**

- Includes fields for username, email, and password.
- Submit button for user registration.



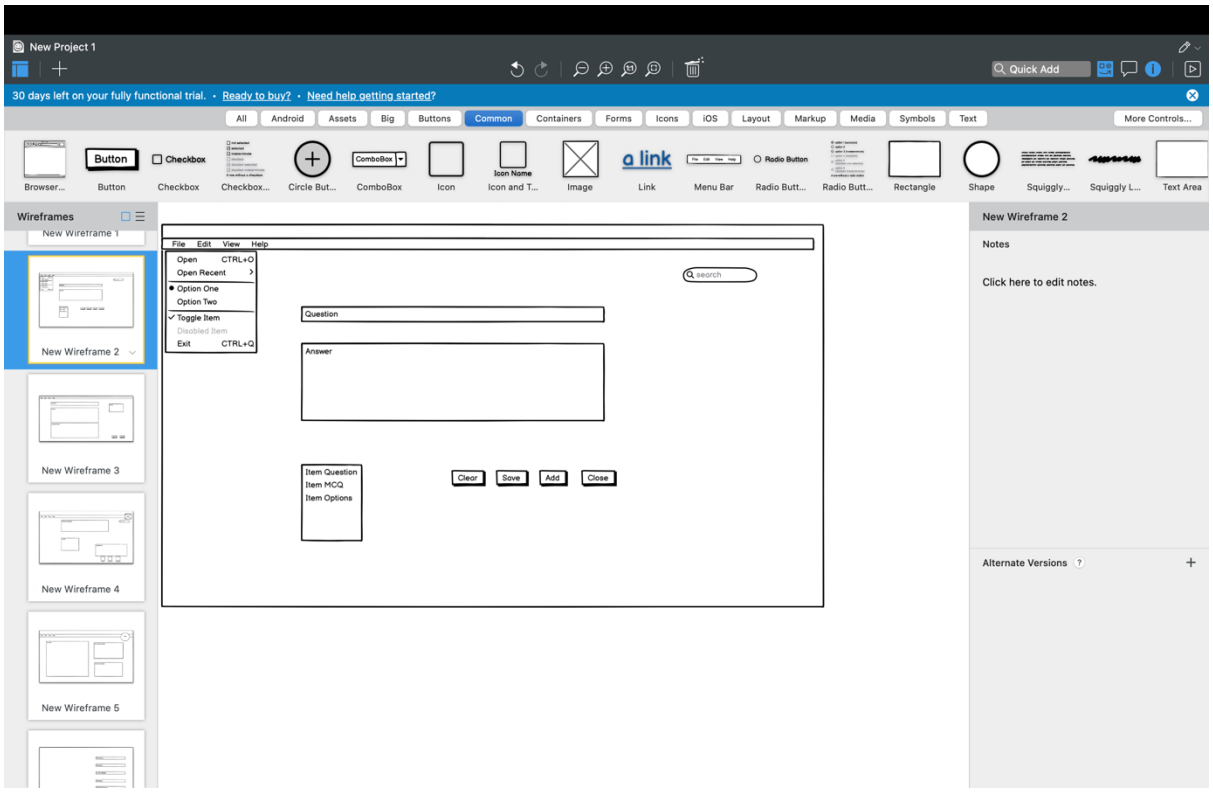
3. **Login Page:**

- Fields for username and password.
- Option to remember login credentials.



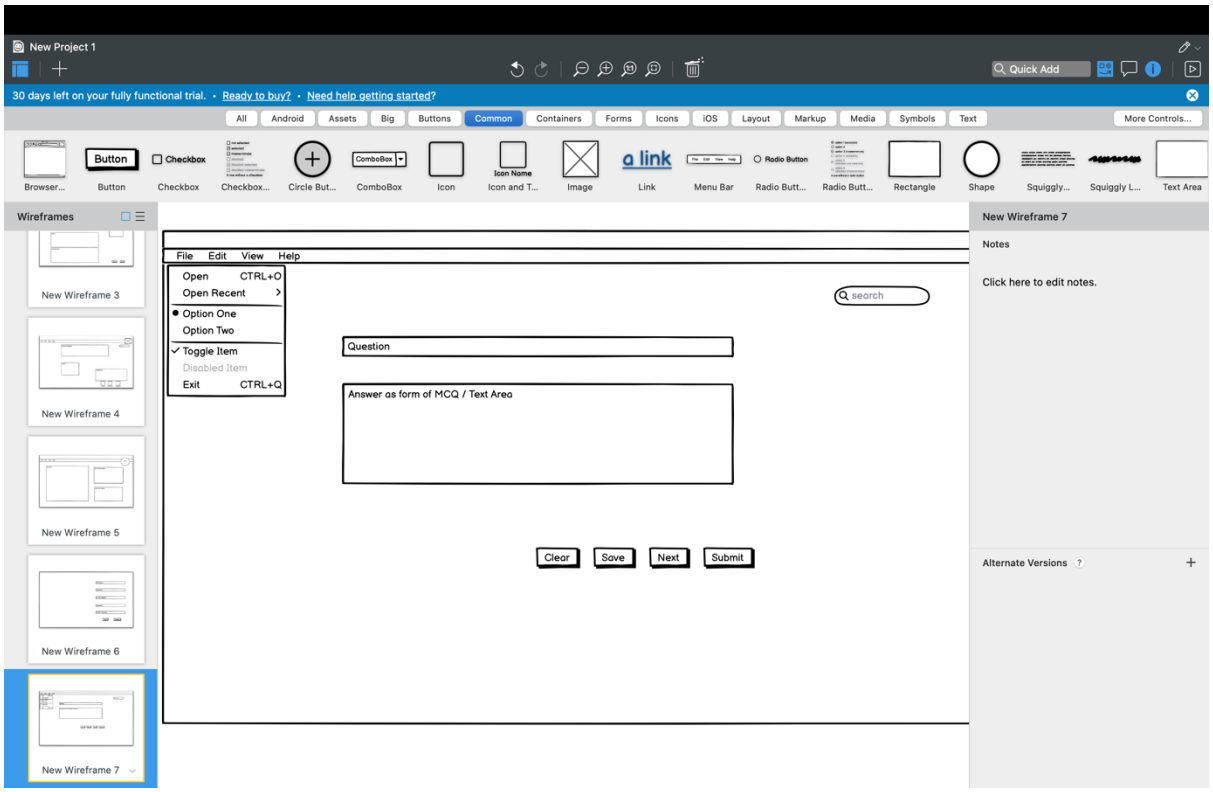
4. **Quiz Creation Page:**

- Form to input quiz title, category, and tags.
- Option to add questions and multiple-choice answers.



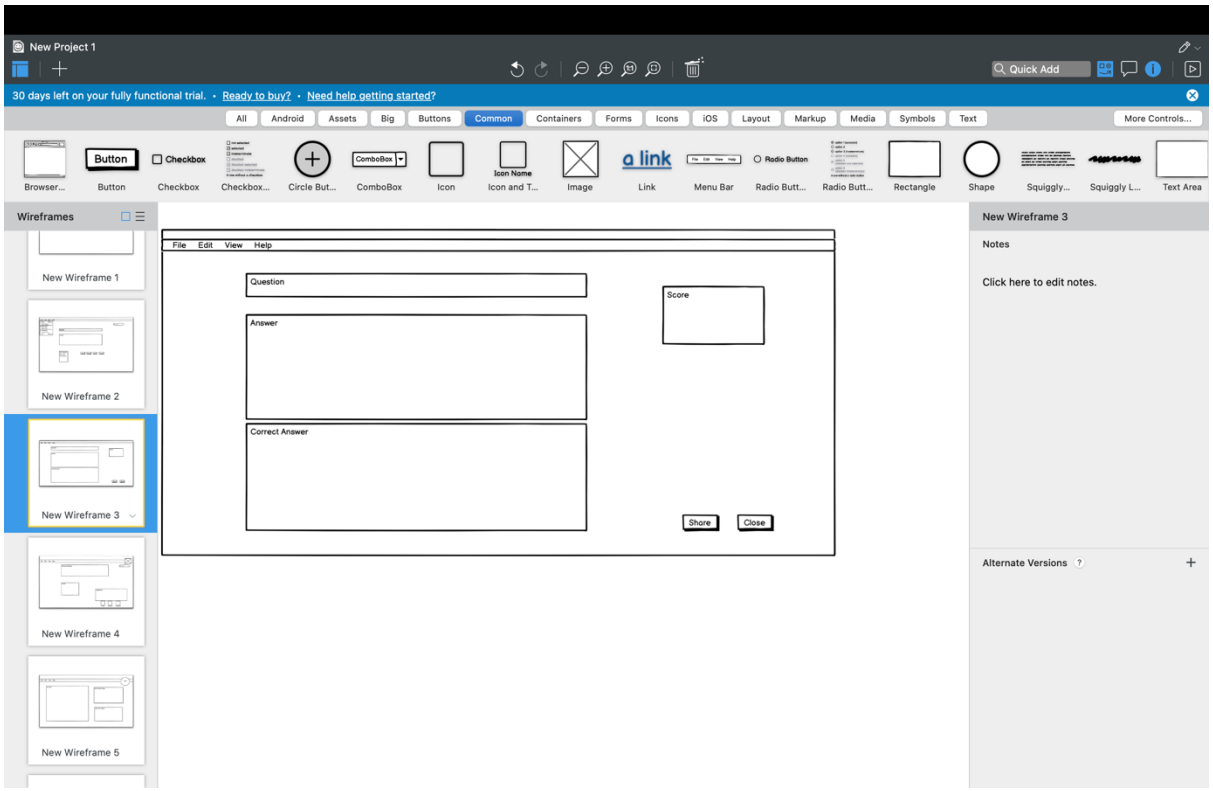
5. **Quiz Taking Page:**

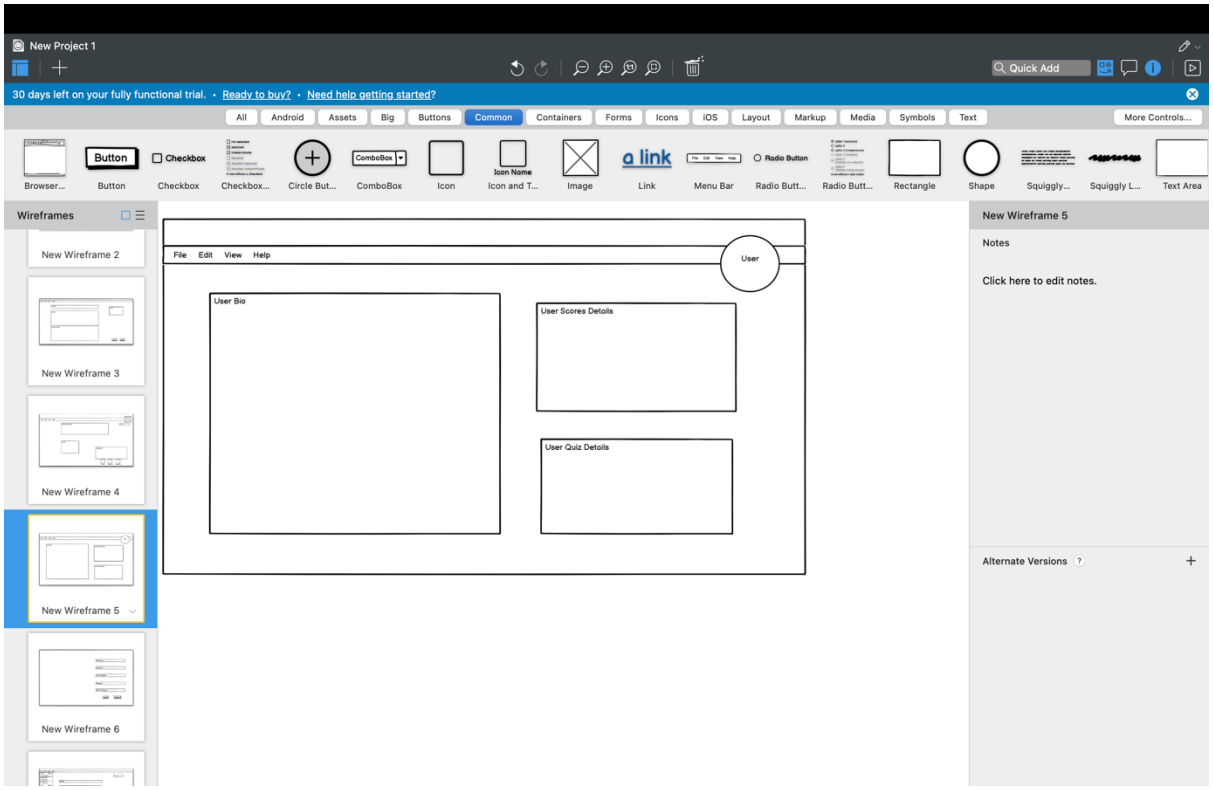
- Displays quiz title and questions.
- Allows users to select answers and submit their quiz.



6. **Quiz Results Page:**

- Shows user's score and correct answers.
- Option to share quiz results on social media.





These mockups provide a visual representation of how users interact with the application, from registration and quiz creation to taking quizzes and viewing results.

4. Discuss technologies

First, read the following articles.

Note (i) these sources are all trying to convince you of their argument – ask yourself if you agree, (ii) some of the information might be a little old but might still be useful.

Second, write a short essay of around 800-1000 words with the title “What Makes a Programming Language Suitable For **Server-Side** Web Development? Is PHP a good language for server-side web programming?”

Your essay should answer the questions in the title, referencing the articles above, in an original way. By “original”, I mean in your own words, using the references to back up or support the points you make yourself. If the authors of the articles make points you disagree with, you should make your disagreement clear and support it with an argument. Note that the first question is not about any particular language such as PHP, but what makes ANY language good for web development. You can then apply this to PHP to argue whether or not it is good for this purpose. Note that just because we use PHP on this module, this does not mean you should decide it is good for server-side web development.

“What Makes a Programming Language Suitable For Server-Side Web Development?

Server-side web development is a crucial process that involves managing backend processes that power dynamic online applications. It involves handling client requests, processing data, communicating with databases, and providing appropriate responses. The choice of the right programming language directly impacts application performance, scalability, security, and maintainability. Therefore, selecting the right language is essential for effective server-side web development.

Factors Making a Programming Language Suitable for Server-Side Web Development:

PHP (Hypertext Pre-processor) is a popular open-source programming language primarily intended for server-side web development. It powers a large section of the web, including popular content management systems such as WordPress and Drupal. However, the usefulness of PHP for server-side web programming is based on various factors:

Performance and Efficiency:

A ideal language should be fast and efficient enough to manage massive amounts of online traffic and complicated server-side processes. Languages with optimised runtime environments, efficient memory management, and concurrency support do well in server-side programming. PHP's speed has increased dramatically over the years as a result of opcode caching, just-in-time compilation, and other optimisations. While PHP may not be as quick as certain compiled languages, it performs well for many online applications, especially when combined with effective caching methods and optimised code. Languages recognised for their fast runtime environments and robust concurrency support, such as Go and Node.js, are popular for developing scalable web applications. These languages excel in effectively managing enormous volumes of requests, ensuring that processes run smoothly even during peak traffic periods.

Web Frameworks and Libraries:

The availability of powerful web frameworks and libraries makes server-side programming much easier by offering ready-made solutions for common tasks like routing, database access, and user authentication. A language with a diverse ecosystem of frameworks and libraries enables developers to create and maintain web applications more quickly. PHP includes a diverse ecosystem of frameworks and libraries, like Laravel, Symfony, and CodeIgniter, that speed web development and give strong solutions to common tasks. These frameworks abstract away most of the complexity of server-side programming while promoting best practices like MVC (Model-View-Controller) architecture and RESTful API design. Consider Python and Ruby, both of which offer substantial tool sets and libraries that can speed up development. Frameworks, such as Django and Ruby on Rails prioritise security best practices to make development workflows more efficient and safe. Django, for example, has built-in protection against common security issues like SQL injection and cross-site scripting (XSS), letting developers to focus on building powerful applications.

Scalability:

Scalability is crucial for server-side applications to handle increasing workloads and accommodate growing user bases. Languages that support horizontal scaling through distributed computing and asynchronous processing are preferred for building scalable web applications. PHP applications can be scaled horizontally by deploying them across multiple servers and using load balancers to distribute incoming traffic. However, PHP's shared-nothing architecture may present challenges in handling concurrent requests and managing session data in distributed environments compared to languages like Node.js or Go.

Security Features:

Server-side languages should have security features and procedures to prevent typical online vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). Strong typing, input validation, and safe coding techniques are required to ensure the security of server-side applications. PHP provides advanced security features and functions, including input validation, data sanitization, and cryptographic operations. However, PHP's history of security flaws, such as those found in outdated versions or badly written code, emphasises the significance of using safe development techniques and maintaining up to current with security updates.

Community Support and Documentation:

A good developer community contributes to the ongoing enhancement and evolution of server-side languages and frameworks. Comprehensive documentation, tutorials, and online tools help developers understand and debug server-side technology. PHP has a big and active developer community that contributes to its development, maintains frameworks and libraries, and provides help via forums, blogs, and documentation. Because of the number of resources, developers may easily learn PHP and handle difficulties that arise during server-side development.

The selection of a programming language for server-side web development is influenced by considerations such as performance, scalability, security, community support, and framework/library availability. PHP is a popular choice for developing dynamic and interactive web applications because of its simplicity and ease of use. With adequate optimisation, safe coding techniques, and contemporary PHP frameworks, developers may efficiently use PHP for server-side web development.

Maintainability and extensibility are crucial for the long-term viability of online applications. Frameworks like Laravel and Flask promote clean code architecture and allow for extensibility via plugins and extensions. Community support is essential for knowledge exchange, cooperation, and innovation within developer groups. Languages with strong communities create an atmosphere conducive to continual progress by providing extensive documentation, courses, and online tools.

Despite its restrictions, PHP remains a viable option for projects requiring quick development and deployment. Developers must exercise caution in reducing possible dangers by adhering to security best practices and utilising modern PHP frameworks.

Selecting a programming language for server-side web development should be an intentional process that considers issues such as performance, security, maintainability, and community support. Each language has its own set of advantages and disadvantages, and developers must measure these against their specific project needs. The programming language used to create server-side web applications significantly impacts their lifespan and profitability.

References

1. Crawford, T. & Hussain, T. 2017, *A Comparison of Server Side Scripting Technologies*, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), Athens. (https://www.proquest.com/docview/2139472343?pq-origsite=gscholar&fromopenview=true)
2. K. Lei, Y. Ma and Z. Tan, "Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js," *2014 IEEE 17th International Conference on Computational Science and Engineering*, 2014, pp. 661-668, doi: 10.1109/CSE.2014.142. (https://ieeexplore.ieee.org/abstract/document/7023652)
3. Purer, Klaus. "PHP vs. Python vs. Ruby–The web scripting language shootout." *Vienna University of Technology* (2009). (http://online.verypdf.com/u/8925/api/20140616-010712-3660462726.pdf)
4. 8 Reasons Why PHP Is Still So Important for Web Development (https://www.jobsity.com/blog/8-reasons-why-php-is-still-so-important-for-web-development)
5. Is PHP Worthy of Developers' Hate (https://www.altamira.ai/blog/is-php-worthy-of-developers-hate)