code.sprint MT

TASK BOOKLET

Undergraduate Category

2022

ICE Malta

GOVERNMENT OF MALTA
RESEARCH AND INNOVATION
MINISTRY FOR EDUCATION, SPORT, YOUTH
DIRECTORATE FOR LEARNING AND ASSESSMENT PROGRAMMES

QuizzerMT

Hello!

Welcome to the CODESPRINT 2022 competition. During these two days, you will be flexing your coding and UI/UX design muscles to create a web app for the educational sector.

1. Design Brief

The Ministry for Education and Employment wishes to develop a web app to help teachers and students following any subject within the local educational system. The app will benefit any school (state, church, private), but will be focused on students from secondary level upwards.

The app, QuizzerMT, will allow teachers to create multiple-choice quizzes, which can then be attempted by students. The idea is to allow teachers to assess the level of understanding of their students, and for students to see where they may need further study to reach the required level.

The Ministry has developed a table of the core functionality required in the app, which is shown below.

	Functionality	Notes
Teachers/Admins		
1	**Ability to login**	You can have preset accounts for teachers or administrators which are already marked as teacher/administrators.
2	**Ability to add a quiz**	
3	**Ability to define and change quiz attributes**	• Quiz Name • Quiz Slug (a short name for the quiz without spaces) • Description • Passing score • Whether or not to show correct answers • If correct answers are shown, whether the correct answer is shown after each question, or at the end of the quiz. • A message to be shown when the student successfully completes the quiz. • A message to be shown when the student fails the quiz.
4	**The ability to set the quiz to randomize question order**	
5	**The ability to insert fields in the end-of-quiz messages**	For example, a teacher can set the end of quiz message to: "Congratulations! You passed the quiz with a score of [[score]] out of [[max_score]]". The items in brackets would then be filled in by the system.
6	**The ability to add questions to the quiz**	Each question must consist of: • The question. • Question type: single choice or multiple choice. • The answers (any number should be supported). • The ability to define which answer (single-choice) or answers (multiple-choice) is/are correct. • The score (number of points) for each correct answer.
7	**The ability to categorise questions in a quiz, so the student can see their score by category.**	
Students		
8	**Ability to login**	You can have pre-set accounts for students. However, ideally, a student should be able to register without a pre-set account.
9	**Ability to view the list of quizzes & start a quiz**	
10	**Quiz navigation**	• Questions should be displayed one at a time, with next and previous buttons. • A **flag** button allows a student to mark questions for review.
11	**Review**	A review, displayed at the end of the quiz before the student submits, shows which questions are answered, unanswered or flagged.
12	**Optionally, a progress bar should be displayed as the student answers quiz questions.**	
13	**Results screen**	The results screen should show: • Number of questions attempted. • Score. • Score by category. • If enabled by the teacher, the correct answer for each question. • The end-of-quiz message as defined by the teacher.

2. Technical Guidelines

The method you choose to implement the app is up to you. However, the following technical guidelines are intended to help ensure you stay on the right track.

2.1 Reference Implementation

The judging panel has created a reference implementation of this app in 16 hours. This is to ensure that the task given is possible within the timeframe allocated. A video of this app in operation is available below. We highly recommend that you watch this video carefully, to get an idea of the functionality and level of polish the judging panel is expecting.

https://codesprint2022.s3.eu-central-1.amazonaws.com/QuizzerDemo.mp4

2.2 Platform

Your app must run on Google Chrome, Apple Safari, and Mozilla Firefox. For the purposes of this competition, having your app run on one of these platforms is acceptable – i.e., **your app does not need to be cross-platform.**

2.3 Development Environment

You are free to use any development environment you wish. Of course, the client must be written in HTML, CSS and JavaScript, however, you can use any server-side technology you wish (Node, PHP, .NET, etc…). In any case, instructions on how to setup the system should be provided so the judges can run your solution.

2.5 Name

QuizzerMT is a sample name – you are free to call your app whatever you want ☺

3. Judgement Criteria

Your submission will be given a maximum of 155 points. The criteria by which points are awarded are detailed below. **Note that you do not need to achieve all the criteria**, however, the more criteria you achieve, the greater your chances of winning! The numbers in **[brackets]** refer to the functionality in the design brief.

Criterion	Notes	Max Points
Core Functionality		
[1] Ability to login	Will only be awarded if the ability works*	**5**
[2] Ability to add a quiz	Will only be awarded if the ability works*	**5**
[3] Ability to define & change quiz attributes	Will only be awarded if each ability works*	**8**
[4] Ability to randomize question order	Will only be awarded if the ability works*	**5**
[5] Ability to insert fields in end-of-quiz messages	Will only be awarded if the ability works*	**5**
[6] Ability to add quiz questions	Will only be awarded if each ability works*	**10**
[7] Ability to categorise quiz questions	Will only be awarded if the ability works*	**5**
[8] Ability to register new student accounts	Will only be awarded if each ability works*	**7**
[9] Ability to view quizzes & start a quiz	Will only be awarded if the ability works*	**5**
[10] Quiz navigation & flagging	Will only be awarded if the ability works*	**5**
[11] Review screen	Will only be awarded if the ability works*	**5**
[12] Progress bar	Will only be awarded if the ability works*	**5**
[13] Results screen	Will only be awarded if each ability works*	**10**
UI/UX		
Neat/Aesthetically pleasant user interface	Rather than 'flair', we are looking for a neat, organized, and functional UI	**10**
App is easy to use	The user should not need a manual to use the app	**5**
Validation	All input fields should be validated, ideally before form submission, especially when the teacher is inputting quiz data.	**10**
Code Quality		
Code is organized into packages/modules/units etc.		**5**
Separation between presentation and logic layers	For example, using a REST API model	**10**
Consistent and correct use of a programming paradigm	Such as OOP, AOP, functional etc.	**5**
Function cohesion	Functions should be kept small, and do one thing, without being too dependent on other functions	**5**
Inline documentation	i.e., comments	**5**
Maintainable code	Ex: use of abstract classes, interfaces, function prototypes etc. Depending on the programming paradigm chosen	**5**

Additional Functionality/Features		
Sample data is pre-loaded into the application	The app should come with sample quizzes	**10**
Sample data is persistently stored	The sample data should be used to 'seed' the permanent storage mechanism of the app, such as a database or structured file, rather than hard-coded sample data being read every time the app is launched	**5**

** This means that abilities that are partially working, or not working, will be graded zero, regardless of code/design created to support this ability.*

Submission Criteria

At the end of the time allocated to this competition, you must submit your code to the judging panel. The code, including all assets and other resources, must be submitted as a folder or compressed archive.

You will also be required to demonstrate your application running.

NOTES


eSkills
eSkills Malta Foundation


LeoVegas


FreeHour