code.sprint MT

PREPARATION BOOKLET

Undergraduate 2022

DIRECTORATE FOR LEARNING &
ASSESSMENT PROGRAMMES

Table of Contents

Intro 2

Technical Overview 2

- Platform 2
- Development Environment 2

Preparation Resources 3

- Topics you must know (This is the important stuff) 3
- Topics you should know to get a competitive edge 3

Judgement Criteria 4

Submission Criteria 5

Intro

Welcome to the code.sprint 2022 competition. During this two-day competition, you will be flexing your coding and UI/UX design muscles to create a web app!

This booklet provides some resources to help you prepare. In addition, make sure to read carefully the Rules & Regulations and the Terms & Conditions on the code.sprint website.

We wish you the best of luck in the competition!

From the code.sprint undergrad category judging panel 😊

Technical Overview

During the competition, you will be designing and building a web app.

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.

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.

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.

Preparation Resources

The following resources will help you to prepare for the task. Remember you can use any technology you are familiar with for the server-side component.

Topics you must know (This is the important stuff)

Client-Side

1. Building pages and forms using HTML.
2. Validating user input using JavaScript.
3. Styling pages and forms using CSS.
4. AJAX or similar (i.e., the ability to modify the page DOM without reloading the page).

Server-Side

1. Connecting to a relational or NoSQL database, depending on your preference.
2. Creating a secure login mechanism.
3. User registration flow.
4. Processing form data with multiple input types (text fields, radio buttons, check boxes, text areas and so on).

Topics you should know to get a competitive edge

1. React/Angular/Vue or some other client-side JavaScript framework.
2. Bootstrap or some other UI library.
3. Facebook login mechanism.

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!

Criterion	Notes	Maximum Points
Core Functionality		
Design brief 1**	Will only be awarded if the ability works*	**5**
Design brief 2**	Will only be awarded if the ability works*	**5**
Design brief 3**	Will only be awarded if each ability works*	**8**
Design brief 4**	Will only be awarded if the ability works*	**5**
Design brief 5**	Will only be awarded if the ability works*	**5**
Design brief 6**	Will only be awarded if each ability works*	**10**
Design brief 7**	Will only be awarded if the ability works*	**5**
Design brief 8**	Will only be awarded if each ability works*	**7**
Design brief 9**	Will only be awarded if the ability works*	**5**
Design brief 10**	Will only be awarded if the ability works*	**5**
Design brief 11**	Will only be awarded if the ability works*	**5**
Design brief 12**	Will only be awarded if the ability works*	**5**
Design brief 13**	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**

*** These will be revealed in the task booklet.*

** 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 as instructed during the competition. 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.