Assignment 1 – Treasure Hunt web app

Coursework with deadline: Friday, March 16th, 2021 - end of day

Covers learning outcomes:

1. With tutor assistance, plan and develop an interactive computing related product (e.g. a mobile phone app. with database connectivity via the internet) in a team environment.
2. Design, discuss and evaluate a user interface for a computer application.
3. Discuss the business context in which computer applications are developed and use the internet to launch a campaign to market a product.

Special Theme for 2020/21

As COVID-19 pandemic has forced most of the world to adjust its day to day business, so has this module. Many software companies—including Google[1], Facebook[2], etc.—have asked their employees to shift to 'work from home'. While we do hope that the pandemic—and all negative things it has brought—will soon be history, reality is that some things might change for good, and hopefully for the best (e.g. less time wasted in commute, cleaner environment, less traffic, etc.) In our line of business—i.e. software development—work from home (WFH) is not entirely new, and it seems that it will be even more popular in the future.

An important aspect of this assignment is to promote *teamwork*. You will be assigned into teams of 2-4 persons each. And each team member will have to take up a role (and usually multiple ones): management, development, testing, technical writing, etc. While in the past a lot of this 'teamwork' took place in the classroom, this year it will take place remotely.

You are *required* to use some collaboration tools which were already core elements of the module, such as GitHub. But you are *encouraged* to investigate additional tools which will make such collaboration even better. For example, some tools like Slack are widely popular with start-up companies. You are also *expected* to hold regular team meetings, at least once a week. Try to view this as your own start-up and aim to keep it as realistic as possible. Where possible, keep the meetings online, using your favourite chat/audio/video tools.

The Challenge: Build a Treasure Hunt Web App

The challenge is to build an appropriate Web App to use for playing a treasure hunt. This is an online quiz where the answers depend on you finding information at specific locations. Your Web App will need to use a given API in an appropriate way to access the clues (questions) and compete by submitting answers. The API and the testing API will provide practice treasure hunts with questions and fake scores for you to test various aspects of your app. You need to promote your app with a dedicated Web page and other marketing tools. You will use your own team's app to play the Treasure Hunt, so make sure you do a good job making it user-friendly and robust.

General Requirements

The Treasure Hunt App should be developed as a HTML5/CSS/JavaScript Mobile App utilizing code you have already developed in the worksheets, with a few twists. There are some basic requirements you

[1] Forbes, "Google Employees Will Work From Home Through Next Summer", https://www.forbes.com/sites/jackkelly/2020/07/27/google-employees-will-work-from-home-through-next-summer/

[2] CNN Business, "Facebook will let employees work from home until July 2021", https://edition.cnn.com/2020/08/06/tech/facebook-work-from-home/index.html

will need to meet to be able to play the quiz, and the API will support various advanced features which can provide for extra points. Beyond that, it is up to your imagination!
Minimum Requirements:

- Your app must be well designed, follow sound UI/UX guidelines and be designed targeting primarily mobile devices (specifically mid- to high-end smartphones). It should be compatible with any mobile device featuring a modern, HTML5-compatible web browser.
- Your app needs to ask the server for questions one at a time – answers must be submitted to the server for checking. The web service will only give out the next question once a player has answered the previous question correctly. Questions cannot, therefore, be cached (in case you were thinking of cheating…)
- In addition to answering the question with the correct answer, players will also need to be in roughly the right place (otherwise they could just keep guessing, right?) so you will need to include location information (latitude and longitude) with your answer.
- Periodically (e.g. every two minutes), your app should update the server with your current position using the corresponding API call[3]. At minimum, you should update the server with your location right before submitting a location-aware answer.
- There are multiple *modes* of questions: *numerical*, *Boolean*, *text* and *multiple-choice*. Your app should support text-based answers at the minimum. Adjusting your submission UI to the expected answer type will give you extra points (because of the better UX).
    - The answer you send to the web service will always be a text string:
        - For multiple-choice questions, the single letter "A" or "B" or "C" or "D", corresponding to the correct answer.
        - For numerical questions, a text-based representation e.g. "7", or "2.3".
        - For Boolean questions, the words "True" or "False".
        - For textual questions, a word or phrase.
    - Note that the answer checker is case insensitive (i.e. "a" and "A" are the same).
    - Correct answers earn points and wrong answers cost points. The exact number of points depends on the question.
    - You should build into your app the facility for the player to *skip* a question, but the player will normally lose points. Some questions cannot be skipped.
- The server will keep the score, and the app should enable users to view that or the leader-board.
- The treasure hunt will be time limited (e.g. 30 minutes) after which the response to any server request will return an appropriate error message.

Minimum Requirements

The minimum requirements for a pass mark are as follows:

1. A landing page which includes information about the project and the team. This page should be designed in a way that *promotes* your app to be used by the wider public. This is the place to *prove* your marketing efforts (videos, Twitter/Facebook integration, etc.)
2. A web app that allows anyone to *play* a full session of the treasure hunt making a reasonably proper use of the Treasure Hunt API[4]. See detailed marking scheme next page.
3. A testing page which includes a description of your approach for verifying and validating your Web App. The minimum is some unit test (see the Testing API[5]) and a User Acceptance test plan.
4. All work must be continuously developed using the recommended collaboration platform (GitHub)[6]. Your repo should have evidence of individual contributions and teamwork.

[3] You can easily do this using the `setInterval()` function: https://www.w3schools.com/js/js_timing.asp
[4] The Treasure Hunt API is described at: https://codecyprus.org/th/guide
[5] The Treasure Hunt *testing* API is described at: https://codecyprusorg.appspot.com/th/testing
[6] You can (optionally) start your project by cloning: https://github.com/NPaspallis/co1111-assignment1

Further Requirements

To make your app more interesting and entertaining, there are many "extra" features you can include (for more marks, obviously) that the server will support via its API.

- You could include a QR Code reader so that if a question includes a QR it could scan it directly and if it is a URL allow to open it, or if it is text, allow to insert it.
- You could show a map of the locations you have visited so-far, and your path between them (i.e. draw your path since the beginning of the treasure hunt on a map)
- You can ask the server how many points the other players of the game have got so far:
    - These could be shown as a "leader-board" if you like...
    - ...or combined with the map from the previous suggestion

Other features may be added according to your skill and imagination! For example, you could add a Web App Manifest to allow Chrome to show the Add to Home Screen prompt[7]. Or you could integrate with social media, so you could easily post your Treasure Hunt score on Twitter/Facebook.

What to submit

You need to submit the following:

- The URL of your publicly accessible project's landing page (e.g. via GitHub Pages) as a comment,
- Also, the URL of your main GitHub repository, again as a comment in the submission page,
- A ZIP file[8] containing the latest version of your GitHub project's repository. First, make sure your GitHub project is in sync with your individual local repositories. Then ZIP your project folder (the one containing the ".git" folder). It is important to submit your full Git project (not the latest version only), so the markers can have access to all your commits.

Marking scheme (to determine Fail or Pass)

You get a grade of **0-35% Fail** if you do *not* meet the minimum requirements.

The minimum requirements to achieve a **40% Pass** grade are:

- You develop a reasonable **landing page** which has at least these features:
    - ☐ It clearly explains the scope of your project,
    - ☐ It clearly lists the team members and their role in the project,
    - ☐ It clearly explains your digital marketing efforts to promote your project.
- You develop an implementation of a **JavaScript-based Web App** that allows you to play a full session of treasure hunt. It should have at minimum these features:
    - ☐ No major bugs/crashes/freezes,
    - ☐ It can be used on mobile devices,
    - ☐ Its UI is functional and features at least a basic level of usability,
    - ☐ It uses the provided Treasure Hunt API,
    - ☐ It allows to view the list of available treasure hunts,
    - ☐ It allows to start a new session and answer questions,
    - ☐ It informs the user when they finish the session and shows the score.
- You develop a **testing page** which includes notes of your testing efforts. At the very minimum you need to have:
    - ☐ A Unit Test for some aspect of your Web App,
    - ☐ A User Acceptance test plan detailing the scenarios in which you test the app, along with the results,

[7] See Chrome's page at: https://developers.google.com/web/fundamentals/web-app-manifest

[8] A good zipping tool is 7zip, available for free at: https://www.7-zip.org

- A basic usability assessment of the final app (e.g. using Nielsen's 10 Heuristics).

- A **teamwork minutes** (notes) page which shows evidence of your teamwork. At the very minimum you need to have:
  - A brief description of the team members and their roles,
  - Minutes (i.e. notes) for at least 5 meetings or progress reports.
- In terms of deployment, your minimum requirements are:
  - Submit the URL of your project's public landing page as a comment,
  - Submit the URL of your repository as a comment,
  - Inclusion of the minimum four files: index.html, app.html, test.html, notes.html (or notes.md) – see details below.

Important note: Since you will be submitting only the Git project, you need to include appropriate Web pages realizing the required features:

1. An "index.html" which must be the landing page and provides the information regarding your app in a professional manner (similar to what you would expect in a commercial site).
2. An "app.html" which realizes the Web App for the treasure hunt. You can use a single page for the whole app or multiple pages (e.g. individual pages for "start", "question", "leaderboard" etc.) but the starting page must be named "app.html".
3. A "test.html" page which provides a description of your testing efforts. This page can be linked from "index.html" or not (your choice). Even though this page does not need to be particularly nice, it should still be usable and present your work in a meaningful and academically appropriate way.
4. A "notes.html" or "notes.md" page where you log your progress in the project. Ideally you will have weekly (or more frequent) entries, explaining what you worked on since the previous meeting, and short descriptions of what individual members worked on.

Further marking scheme (to determine classification)

Assuming you have completed the *minimum requirements* outlined earlier, your grade is decided as follows (higher grade bands assume you also satisfy all previous grading bands' requirements)[9]:

Section	Weight	50+% (2:2)	60+% (2:1)	70+% (first)	Grade
Landing page and Marketing	20%	Some evidence of engagement with social media. Clearly demonstrating the advantages of the team/project.	A well-thought design which engages visitors. Also use and mention of methods to achieve higher ranking in search engines[10].	Evidence of commercial quality page with strong engagement with multiple social media (e.g. Twitter, YouTube, Facebook, etc.)	
Web App	50%	Implements most features of the API, including location updates. At least a basic level of usability that allows you to efficiently participate to the treasure hunt game. Elegance of code (i.e. sound use of HTML/CSS, proper JavaScript code with functions, proper names, comments, etc.) Also, properly handling error messages, informing the player accordingly.	An excellent implementation with no crashes and a very high level of usability (e.g. input UI depends on the answer type, etc.) Allows the user to resume unfinished games if they navigate out of the game. Implements all API features plus a QR code scanner. Allows navigating to a URL (if included in the questions). Good design which works well on multiple screen sizes.	Above and beyond project with extensive use of all data and functionality of the provided API, and an implementation of all requirements plus extra features such as a record of past questions and answers, a map view of the questions, and generally anything towards a commercial-grade Web App. Also, evidence of proper use of Object Orientation in your JavaScript code.	
Testing	15%	An elaborate User Acceptance test plan with evidence of proper testing.	An elaborate usability assessment with consideration of Nielsen's heuristics.	Extensive and proper use of unit tests to verify your JavaScript code. An excellent User Acceptance plan.	
Teamwork [11]	15%	Evidence of good collaborative work with coherent records in the *notes* page as well as in the Git log.	Extensive and detailed evidence of great teamwork, with notes of multiple team meetings (e.g. 8+).	Excellent teamwork as evidenced by the *notes* page and Git log as well as by the final product.	

[9] Also note that your individual grade will be rounded to the nearest band mark (e.g. 52, 55, 58, 62, etc.)

[10] If you include any features which are not easily perceived by the marker, make sure to mention it (e.g. in the *notes* page) so that the marker can check the feature out and grade you for it.

[11] In teamwork projects, not all members contribute the same. This is understood, and we will accept submissions where the team members did not contribute *exactly* equal effort but nevertheless managed to positively contribute to their team's success. Note though that the markers reserve the right to adjust a team member's mark, positively or negatively relative to their team-assigned mark, based on the evidence and their academic judgement. In any case, a student is expected to make a meaningful contribution to the team effort to be eligible for a pass grade (i.e. >=40).

Academic year: 2020-21