Aufgabenstellung

Zur Festigung des erworbenen Wissens durch die Angular App 'Tour of Heroes' und der Angular App 'BookMonkey' aus dem Angular Buch, soll eine eigene Angular SPA programmiert werden.

Mit dieser soll es möglich sein sich auf die Linux LPIC-1 Zertifizierungsprüfungen vorzubereiten. Die App soll ähnlich dem "lpisim" sein, der zur Zeit für diese Vorbereitung bei der BITLC GmbH zur Prüfungssimulation eingesetzt wird, allerdings mit 2 zusätzlichem wichtigen Ergänzungen ausgestattet werden.

Diese Ergänzungen sind einmal ein **Learn-Modus,** der es ermöglicht sich genauer mit den Fragen und Antworten auseinander zu setzten und ein **Check-Modus**, bei dem das Gelernte direkt bei jeder beantworten Frage geprüft wird. Und natürlich soll die App einen **Exam-Modus** enthalten, mit dem verschiedene Prüfungsszenarien durchgespielt werden können, wie das bei den meisten PrüfungsSimulatoren so üblich ist, wie Anzahl der Frage festlegen, sowie Fragenreihenfolge und/oder Antwortreihenfolge zu mischen.

Der Name der Angular App soll sein:

LLCE App - Linux Learn/Check/Exam App

Info zur Angular Modul Bewertung

Zur Bewertung des Angular Modul wird die Erstellung und Dokumentation, des durch diese Aufgabenstellung vorgebenen Angular Projekts herangezogen.

Bewertet wird:

50% Programmierung (Design/Funktion/Code) +

50% Dokumentation (nach IHK Vorgabe für ein IHK Projekt)

Empfohlene Bearbeitungsreihenfolge

Für das vollständige Erstellen des LLCE App ähnlich dem „lpisim" mit Learn-/Check-/ExamModus ist die zur Verfügung stehende Zeit ausreichend. Trotzdem wird empfohlen, die Entwicklung in der unten angegebenen Reihenfolge durchzuführen.

Diese schrittweise Entwicklung einzelner Komponenten ermöglicht es, sich nicht zu überfordern und alles auf einmal machen zu wollen. Sind die ersten Teilabschnitte entwickelt, hat man eher die Möglichkeit abzuschätzen, welche weiteren Teilabschnitte in der gegebenen Zeit noch entwickelt werden können oder auch nicht.

A. Fragen verarbeiten:

120 Fragen aus der dem lpisim beiliegenden LPI-2019-1-101d-QA(LPIC-1 101) oder LPI-2019-1-102d-QA (LPIC-1 102) Datei. Mit einem, vom Dozenten bereitgestelltem Programm, kann aus den genannten Dateien eine JSON Datei, mit einer gegebenen Datenstruktur, erstellt werden. Dabei kann man die Datenstruktur übernehmen (was empfohlen wird).

1. **Fragentypen** verarbeiten: Vorgehensreihenfolge:
   a. Die App sollte in der Lage sein ***zunächst*** nur **Multiple-Choice-Fragen** zu verarbeiten.

b. Eine Erweiterung soll es später ermöglichen nur **Single-Choice-Fragen** zu verarbeiten.
c. Eine weitere Erweiterung soll dann das Verarbeiten von nur **Fill-In-Fragen** ermöglichen.

2. Alle Fragentypen (A.1: a,b,c) in einem Fragenmodul zusammen verarbeiten.

Es wird empfohlen zunächst nur **A.1.a.**, also die App zunächst nur mit **Multiple-Choice-Fragen** für den Learn-Modus, den Check-Modus und den Exam-Modus zu erstellen. Später die anderen Fragentypen hinzuzufügen, was einfacher ist und anschließend gemischt verarbeiten.

B. Infos zu den **Learn- / Check- / Exam-Modi:**

Learn-Modus:

Die App soll die Möglichkeit zum reinen Lernen des Stoffs bieten. Die Fragen sollen in Listenform aber auch als jeweils einzelne angezeigte Frage durchlaufen werden können. Mit der Möglichkeit zu jeweils einer Frage die richtige(n) Lösungen, aber auch optional weitere Infos anzuzeigen.

Check-Modus:

Eine Überprüfung des gewonnenen Wissens oder bereits schon vorhandenen Wissens soll in einem Check-Modus erfolgen. Im Check-Modus sollen Fragen einzeln bearbeitet werden.

Dabei sollen zwei Möglichkeiten gegeben werden:

- überspringen einer Fragen, weil man sie noch nicht zu „200%“ sicher beantworten kann,
- oder die Fragen entsprechend zu „200%“ richtig beantworten.

Wird die Frage nicht richtig oder nur unvollständig beantwortet, z.B. bei MultipleChoice Fragen, soll ein Warnung auftauchen und anschließend eine Frage zurückgesprungen werden. Diese muss zunächst nochmal beantworten werden, ehe die gerade falsch beantwortete Frage wieder angezeigt wird. Ziel ist es, damit den Anwender zum sicheren Antworten zu bewegen und nach Möglichkeit das Raten ihm zu verderben :). Jede Frage kann auch übersprungen werden, wenn man unsicher ist. Jede falsch beantworte Frage in diesem Modus wird aber gezählt.

Sind insgesamt 7 Fragen falsch beantwortet worden, soll der Check-Modus abgebrochen werden mit einem Hinweis den Learn-Modus zu nutzen.

Wurde der Check-Modus beendet (entweder letzte Frage oder Button „beenden“ gewählt), soll eine Übersicht die Anzahl gewusste, Anzahl übersprungener und Anzahl falsch beantworteter Fragen angezeigt werden.

Exam-Modus:

Der Exam-Modus entspricht dem Modus des lpisim, Fragen zu durchlaufen und am Ende (entweder letzte Frage oder Button „beenden“ gewählt) das Ergebnis, Anzahl gewusste, Anzahl nicht gewusste und Anzahl „geratener“ (=falsch beantworteter) Fragen, auszugeben. Der Prüfmodus soll abgebrochen werden, wenn 20% der gewählten Anzahl an Fragen falsch beantwortet wurden (z.B.: 20, 40, 60, ...) .

Hintergrund zur App

Mit diesem Projekt soll zunächst eine browserbasierte App für den Kunden, die **AraCara GmbH**, geschaffen werden. Sie soll von Teilnehmern einer Ausbildung zum Fachinformatiker (FIAE & FISI) genutzt werden können, um sich auf die internationale Zertifizierungsprüfungen für Prüfungen des Linux Professional Institut – LPI vorzzubereiten (hier nur LPI-1 101 Prüfung).

Als Benutzer der App kommen aber auch Personen in Frage, die sich allgemein auf Zertifizierungsprüfungen (hier LPIC) vorbereiten wollen, egal, ob sie dies in einem schulischen Umfeld, am Arbeitsplatz oder rein privat machen wollen. Die **AraCara GmbH** will später die App freigeben.

Bei normalen Prüfungssimulatoren besteht in der Regel keine Möglichkeit den Stoff zunächst zu lernen, man kann nur direkt in die Prüfungssimulationen einsteigen.

Dadurch besteht die Gefahr, dass sich beim Beantworten von Fragen, wenn man ins Raten kommt - und die Erfahrung lehrt, das es viele tun - sich „geratenes“ = “falsches“ Wissen verfestigt. Dies führt, unter Stress, wie in den echten Prüfungen, sofern man nur „ratent“ gelernt hat, zu großer Unsicherheit und damit oft zu schlechten Ergebnissen mit weiterhin unsicherem Wissen.

Daher soll die App die Möglichkeit bieten, sich die korrekten Antworten zunächst auch nur anzeigen zu lassen, ohne vorher selbst eine Antwort auswählen zu müssen/können. Dies soll der **Learn-Modus** sein.

Der **Check-Modus** soll bei jeder falscher Beantwortung einer Frage ein negatives Feedback geben, um den Anwender noch mehr zu motivieren, den Stoff vorher intensiv zu lernen, ehe Fragen nur unsicher beantwortet werden.

Der **Exam-Modus** soll erst nach Beantwortung aller Fragen ein Ergebnis anzeigen, aber er soll abgebrochen werden, wenn mehr als 20% der Fragen falsch beantwortet wurden, da nicht mehr davon ausgegangen werden kann, das der Anwender seinen Stoff beherrscht. Die Empfehlung sollte dann sein, den **Learn-Modus** und **Check-Modus** lieber nochmal zu benutzten.

Datenstruktur / Datengrundlage / Datenzugriff

Datengrundlage zur App sollen die Fragen , des mit dem „**lpisim**“ mitgegebenen LPIC-1-FragenKatalog LPI-2019-1-101d-QA oder LPI-2019-1-102d-QA, sein.

a. Datenstruktur Ermittlung: **sehr wichtig**

Ähnlich dem BookMonkey können zunächst wenige Bücher=Fragen (7) intern der App zur Verfügung gestellt werden, um die Datenstruktur (Interface) des jeweiligen Fragentyps zu entwickeln, siehe A.: **Multiple Choice**, Single Choice, Fill In.

(die Fragen liegen in einem JSON Format mit einer dadurch vorgegebenen Datenstrukturen vor. Es wird empfohlen diese zu verwenden.)

b. Datenzugriff:

1. Im nächsten Schritt soll dann ein Fragenkatalog mit vielen Fragen (pro Fragentyp mindestens 20 Fragen) in ein passendes Format umgewandelt werden und der Einfachheit halber, in der App selbst gespeichert werden, ähnlich „**book-store-service**“ in lokaler Version.

Restliche Fragen sollen später nachgetragen werden.
(vorliegenden JSON Dateien vereinfacht dies).

2. Datenzugriff (optional):

alternativ kann der Zugriff auf Fragenkataloge auch per HTTP „in Memory“ Verfahren (siehe TourOfHeroes) per HTTPClient „geladen“ werden, oder per HTTP/HTTPS Zugriff

realisiert, wie es im BookMonkey mit dem “api5.angular-buch.com“ auf einem externen DataStore realisiert wurde (schon sehr aufwendig: siehe swagger API).

Projektplanung

Das Projekt soll ***iterativ/dynamisch*** entwickelt werden, die Projekt-Dokumentation soll allerdings nur den fertigen Zustand dokumentieren und nicht auf evtl. iterative Zwischenschritte wie z.B. Änderung der StoryBoard/Funktionen/... eingehen, weil diese ergänzt/verändert werden musste. a. Begonnen werden soll zunächst mit einem ***Story-Board***, ähnlich dem des BookMonkey, in dem skizzenhaft die Ansichten und damit die Funktionsweise der

LLCE App gezeigt werden soll (Blatt Papier und Stift; später dann evtl. draw.io, ...).

b. Eine ***Funktion/Aktion****-Planung* soll Funktionen und Aktionen den, im Story-Board gezeigten, Elementen zuordnen.
(entspricht der Liste der gewünschten Funktionen in einem Lastenheft und den zugeordneten geplanten Umsetzungen in einem Pflichtenheft)

c. Eine Datenstruktur-Planung soll die Punkte a. und b. ergänzen, damit klar ist, welche Daten wie im Story-Board angezeigt und mit welchen Funktionen/Aktionen verarbeitet werden. (die Datenstruktur kann hier aus den JSON Dateien der Fragenkataloge übernommen werden; Planung dafür dann nicht mehr erforderlich)