Klausur „Entwicklung webbasierter Anwendungen“
Wintersemester 2024/25

Hinweise

- Bearbeitungszeit: 5+90+5 Minuten (je 5 min für Download und Upload).
- Insgesamt gibt es 100 Punkte in 6 Aufgaben.
- **Abgabe bis 19.02.2025 um 17:40 über GIT in Ihrem persönlichen Repository für die Prüfung**
- Erlaubte **Hilfsmittel**: Ein Notebook mit Internetzugang pro Person, sowie alle auf dem Notebook mitgebrachten Informationen und jede auf Papier gedruckte Quelle.
- Als **Täuschungsversuch** gelten:
    - Jegliche Formen der "Teamarbeit" z.B. mit Chats, GitHub oder Netzlaufwerken
    - Abonnierte oder gekaufte Tools wie z.B. ChatGPT4 oder AI-Assistant für PHPStorm
- **Lesen** Sie zuerst die gesamte Aufgabenstellung bevor Sie mit der Lösung der Aufgaben beginnen.
- **Bearbeiten** Sie die Aufgabe im Ordner **src/Exam** des Repositories, das ihnen für die Prüfung zugewiesen wurde. Legen Sie keine neuen Dateien an und benennen Sie die Dateien nicht um. Commiten bzw. pushen Sie ihren Code via Git regelmäßig (ca. alle 10 Minuten) in Ihr zugeteiltes Repository.
- Die **Abgabe** erfolgt ausschließlich über Git. Es wird nur bewertet, was sich bis zum Abgabetermin in dem für Sie angelegten Gitlab-Repository in den vorbereiteten Dateien unter src/Exam befindet.
- Schreiben Sie wartbaren und standardkonformen Code, der die **Regeln** der Veranstaltung für ordentlichen Code und die Vorgaben durch die Seitenklassen berücksichtigt. Verwenden Sie Exceptions zur Behandlung von ungewöhnlichen Fehlern. Entwickeln Sie eine barrierefreie, sichere Anwendung. **Achtung: Die Abgabe von größeren, unnötigen Code-Teilen - wie sie KIs gerne erzeugen - wird als nicht-wartbar (= falsch) gewertet, selbst wenn sie korrekte Teile enthalten. Außerdem werden nur Inhalte, die Teil der Veranstaltung waren, als korrekt anerkannt.**
- Diese Aufgabenblätter werden nicht abgegeben. Schreiben Sie also keine Lösungsteile darauf. Sie dürfen die Seiten natürlich auch trennen.
- In den PHP-Seitenklassen ist die strenge Typüberprüfung aktiviert. Diese Überprüfung darf nicht abgeschaltet oder umgangen werden. Ändern Sie nicht die Signaturen der vordefinierten Methoden.

Zulieferung

- In Ihrem GitLab-Projekt sind die Dateien, die Sie bearbeiten sollen, schon angelegt.
- Als Zulieferung erhalten Sie eine ZIP-Datei mit vorbereiteten Inhalten. Kopieren Sie zunächst diese Dateien in das Verzeichnis src/Exam und überschreiben Sie die vorhandenen Dateien.
- Wenn Sie die Datenbank mit dem SQL-Skript importieren wollen, melden Sie sich über phpMyAdmin mit dem **Benutzer: root** und **Passwort: ganzGeheim** an. Anschließend nutzen Sie die „Importieren“ Funktionalität, wie Sie es kennen. **Achtung: Datenbank kann in einer Untergruppe 2024 auftauchen!**
- **Page.php**: Die bekannte Basisklasse der Seitenklassen (zugeliefert, nicht bearbeiten)
- **Exam.php** – siehe Aufgabe 1
- **ExamApi.php** – siehe Aufgabe 2
- **Exam.js** – siehe Aufgabe 3
- **ExamService.php** – Siehe Aufgabe 4
- **Exam.txt** – siehe Aufgabe 4
- **Exam.css** – siehe Aufgabe 5

Die Aufgabe allgemein: „h_da – Registrierung“

In dieser Klausur sollen Sie eine Registrierung implementieren, die eine Funktion zur Passwortsicherheit beinhaltet. Darüber hinaus sollen weitere Funktionen zur Validierung und zur Anzeige von Rückmeldungen entwickelt werden, um dem Nutzer eine benutzerfreundliche Registrierung zu gewährleisten.

Diese umfasst:

1. Einen Kopf-Bereich mit Navigationselementen und einer Hauptüberschrift.
2. Ein Formular zur Erstellung eines Benutzers, einschließlich serverseitiger Validierung und Echtzeit-Feedback zur Verfügbarkeit des Benutzernamens.
3. Einen Abschnitt, der clientseitig die Stärke vom eingegebenen Passwort ermittelt.

Anmerkung zum Thema Sicherheit: In dieser Klausur liegt der Schwerpunkt auf der Implementierung der Registrierung und der Echtzeitbewertung der Passwortstärke. Für den Einsatz in einer realen Anwendung wären jedoch zusätzliche Sicherheitsmaßnahmen wie eine umfassendere Passwortvalidierung und die sichere Speicherung von Passwörtern erforderlich. Diese Aspekte werden hier bewusst vereinfacht, um die Aufgabe übersichtlich und minimalistisch zu gestalten.

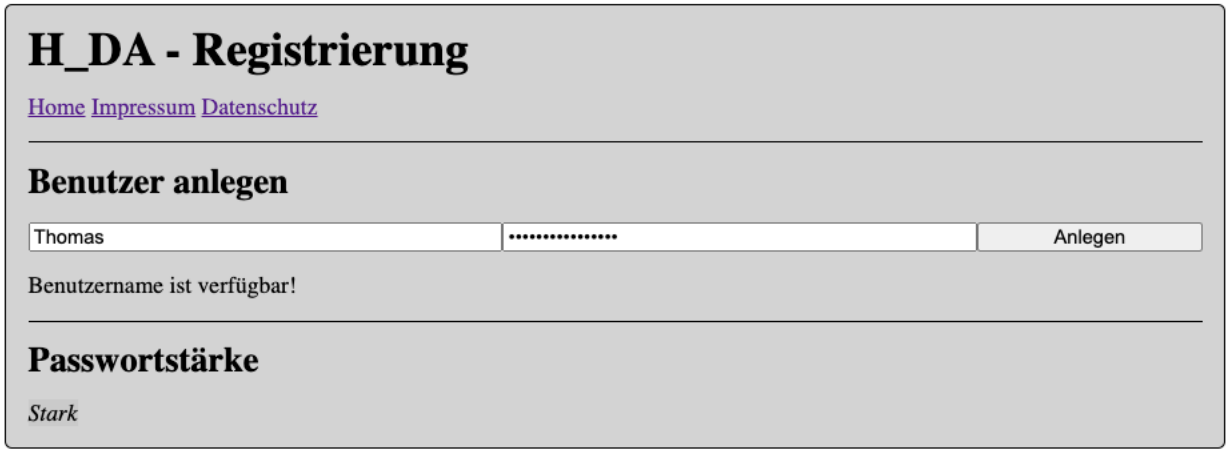

Abbildung 1: Registrierung mit beispielhafter Ausgabe

Datenbankstruktur

Die Datenbank heißt "*2024_password*" und besteht aus der Tabelle *user*. Es existieren schon vordefinierte Benutzer. Weitere Informationen können Sie bei Bedarf der zugelieferten SQL-Datei entnehmen.
Achtung: Bei den vorgegeneben Daten sind die Passwörter zur besseren Verständnis nicht verschlüsselt.

id	name	password
1	Max	passwort123
2	Heike	heike1992
3	Benjamin	test

#	Name	Typ	Kollation
1	id	int(255)	
2	name	varchar(255)	utf8mb4_general_ci
3	password	varchar(255)	utf8mb4_general_ci

Aufgabe 1: HTML/PHP (Exam.php) (30 Punkte)

Implementieren Sie in der Datei **Exam.php** unter Verwendungen der **Seitenklassenstruktur** sowie **PHP** und **semantischem HTML,** die in **Abbildung 1** dargestellte Webseite. Folgendes ist zu beachten:

- Nutzen Sie hierbei den vorgegebenen Code aus **Exam.php** und erweitern Sie diesen.
- In dieser Aufgabe soll noch kein JavaScript verwendet werden.
- Das HTML-Attribut für ein Passwort-Eingabefeld lautet: type="password“.
- Falls es beim Passwortfeld zu Problemen mit der AutoFill-Funktion kommt, kann das HTML-Attribut autocomplete="new-password" Abhilfe schaffen.
- Funktion zur Passwortverschlüsselung: password_hash($password, PASSWORD_DEFAULT);

Realisieren Sie folgende Struktur und Funktionalität:

- Kopfteil mit Überschrift und Navigation
  - Die Links haben keine Funktionalität
- Das „Benutzer anlegen“ Formular
  - Mit diesem Formular soll beim Abschicken ein neuer Nutzer in die Datenbanktabelle: user eingetragen werden.
    - Ein Nutzer darf nur dann in die Datenbank gespeichert werden, wenn sein Name noch nicht vorhanden ist. Stellen Sie sicher, dass dieses Verhalten serverseitig validiert wird, um doppelte Benutzernamen zu verhindern.
    - Die clientseitige Validierung wird in Aufgabe 3 programmiert. Bereiten Sie dafür bereits die entsprechende Struktur vor.
    - Nach dem Absenden des Formulars soll der Nutzer unterhalb des Formulars eine Rückmeldung erhalten, ob die Registrierung erfolgreich war oder nicht.
    - Das Passwort soll sicher verschlüsselt und anschließend in der Datenbank gespeichert werden.
- Ein Passwortstärke Abschnitt
  - Dieser Abschnitt hat zu diesem Zeitpunkt noch keine Funktionalität, diese wird dann in Aufgabe 3 realisiert. Sie sollten dennoch schon die nötige HTML-Struktur realisieren. Diese soll später mit JavaScript angesteuert werden können

Aufgabe 2: PHP (ExamApi.php) (10 Punkte)

Implementieren Sie in der Datei **ExamApi.php** unter Verwendung unserer **Seitenklassenstruktur** ein **PHP-Skript** welches als **Response** einen **JSON-String** zurückliefert. Folgendes ist zu beachten:

- Nutzen Sie hierbei den vorgegebenen Code aus **ExamApi.php** und erweitern Sie diesen.

Realisieren Sie folgende Funktionalität:

- Hierbei handelt es sich um eine API, welche überprüft ob ein Benutzername bereits in der Datenbanktabelle: user vorhanden ist oder nicht.
- Der Benutzername wird der API beim Request als Parameter zur Verfügung gestellt.
- Je nachdem ob der Name vorhanden ist oder nicht soll die API entsprechen eine sinnvolle Antwort zurückliefern.

Aufgabe 3: JavaScript (Exam.js) (25 Punkte)

Implementieren Sie in der **Exam.js** Datei via JavaScript die Bewertung der Passwortsicherheit sowie die Verfügbarkeit vom Benutzernamen. Folgendes ist zu beachten:

- Nutzen Sie hierbei den vorgegebenen Code aus **Exam.js** und erweitern Sie diesen.
- Auf Events soll ausschließlich in der **Exam.js** gehört werden.
- Achten Sie drauf, dass der JavaScript-Code erst ausgelöst wird, wenn das DOM vollständig geladen ist.

Realisieren Sie folgende Funktionalität:

a) Passwortsicherheit
- Je nach Länge vom Passwort soll folgender farblicher Text im Passwortstärke Abschnitt erscheinen:
  - **Weniger als 4 Zeichen** → Text: Schwach, Farbe: rot
  - **4 bis 8 Zeichen (einschließlich 4 und 8)** → Text: Ok, Farbe: orange
  - **Mehr als 8 Zeichen** → Text: Stark, Farbe: grün
- Die Farben sollen mit separate CSS-Klassen realisiert werden.
- Bei jeder Eingabe in das Passwortfeld, soll der Stärke-Text entsprechend aktualisiert werden.
- Der Nutzer kann auch schwache Passwörter abschicken.

b) Verfügbarkeit vom Benutzernamen
- Bei jeder Eingabe in das Benutzernamenfeld soll mithilfe von **ExamApi.php** geprüft werden, ob der Name bereits existiert. Das Ergebnis der Überprüfung soll dem Nutzer unterhalb des Formulars angezeigt werden. → Benutzername ist vergeben / Benutzername ist verfügbar
- Außerdem soll der „Anlegen“ Knopf nur aktiv sein, wenn der Benutzername verfügbar ist, ansonsten ist er deaktiviert. Passen Sie gegebenenfalls hierfür das HTML aus Aufgabe 1 an.

Aufgabe 4: Alles zusammen (Exam.txt und ExamService.php) (15 Punkte)

Wie würden Sie die in der **Abbildung 2** abgebildete Seite, mit dem Wissen welches Sie nun über unsere Klausur und die Struktur haben, realisieren?

- Beschreiben Sie ihr Vorgehen so detailliert wie möglich in der **Exam.txt** und realisieren Sie es anschließend in **ExamService.php**.
- Achtung: Für die Beschreibung gibt es auch bereits Punkte.

Abbildung 2: Adminseite mit beispielhafter Ausgabe

Aufgabe 5: CSS (Exam.css) (10 Punkte)

Implementieren Sie in der **Exam.css** Datei via **CSS** das (grobe) Styling der Webseite. Die **Abbildung 1** sollte Ihnen als Vorlage dienen. Folgendes ist zu beachten:

- Sie müssen nicht alle Abstände pixelgenau realisieren.
- Es muss nur das Styling aus Abbildung 1 realisiert werden.
- Nutzen Sie insbesondere CSS Flexbox

Außerdem Achten Sie drauf, dass die Seite responsive sein soll.

- Die Links der Navigation und die Elemente vom „Benutzer anlegen“ Formular sind auf dem Handy untereinander
- Die ersten zwei Elemente vom „Benutzer anlegen“ Formular wachsen doppelt so stark wie das dritte.

Aufgabe 6: Lauffähigkeit/Vollständigkeit (10 Punkte)

Gehen Sie diese Aufgabe erst an, wenn Sie die übrigen Aufgaben nach bestem Wissen gelöst haben.

- **Achtung: Aufgabe 4 muss hier nicht berücksichtigt werden!**

Prüfen Sie die Lauffähigkeit ihrer Anwendung indem Sie das Docker-Setup starten und im **localhost** ihre Anwendung begutachten. Folgendes soll geprüft werden:

- Prüfen Sie ob Ihr erzeugter HTML-Code valide und barrierefrei ist und den EWA-Regeln für ordentlichen Code entspricht.
- Sieht das Styling ihre Lösung ungefähr so aus wie in Abbildung 1?
- Rufen Sie die **ExamApi.php** im Browser auf und prüfen Sie das Ergebnis
- Vergewissern Sie sich ob die Anwendung funktionsfähig ist, indem Sie die Registrierung testen.

Abgabe

Laden Sie Ihr Endergebnis in **Git** hoch und prüfen Sie über **GitLab (https://code.fbi.h-da.de)**, ob die Dateien auch wirklich angekommen sind. Bei Problemen bewahren Sie bitte Ruhe und informieren Ihren Betreuer.