FH University of Applied Sciences TECHNIKUM WIEN

Aufgabenstellung Abschlussprojekt WebTech1 – WS2020

BIF1DUAL

Allgemeine Informationen

a) Zum Abschluss der LV Webtechnologien muss ein Web-Projekt umgesetzt werden. Die Gewichtung der einzelnen Aufgaben und die Punkteverteilung finden Sie in der zugehörigen Bewertungsmatrix.

b) Verwenden Sie für die Umsetzung Teile Ihrer bereits abgegeben (und nach gemeinsamer Besprechung in den LVs verbesserten) Übungen wieder.

c) Die Bearbeitung erfolgt in 3er Teams.

d) Das Abnahmegespräch erfolgt dann ebenfalls im Team, jedes Teammitglied muss das Projekt (Funktionsumfang und Code) kennen und Fragen dazu beantworten können (ähnlich den Übungsabnahmen)
   i. Das Abnahmegespräch orientiert sich an einer Bewertungsmatrix. Diese steht vorab zur Verfügung und dient u.a. der Darstellung der Gewichtung der einzelnen Aufgaben sowie zur Kontrolle der Vollständigkeit.

e) **Für manche in grün markierte Tasks, gibt es Extrapunkte zu erreichen. Diese Aufgaben sind optional.**

f) Die Angabe lässt gewisse Gestaltungs- und Implementierungsfreiheiten zu. Folgende **Basis-Features** müssen abgebildet werden:
   i. Registrationsmöglichkeit und Profilbearbeitung für User
   ii. Userverwaltung durch den/die AdministratorIn
   iii. Posten eigener Beiträge
   iv. Verwaltung der Beiträge in einer Datenbank und übersichtliche Darstellung im Browser
   v. Liken, Disliken von Beiträgen
   vi. Kommentarfunktion für Beiträge

g) Die Umsetzung des Projekts wird **sämtliche LV-Inhalte** abdecken:
   i. Frontend: HTML5, CSS3, Bootstrap
   ii. PHP & OOP
   iii. Fileupload
   iv. MySQLi + Prepared Statements
   v. Authentifizierung
   vi. Verwendung von HTML/CSS Frameworks und PHP-Libraries

h) Achten Sie auf eine **gute User Experience** innerhalb der Applikation (keine Whitepages, keine Fehlerzustände, guter Gesamteindruck, …)

Projektstruktur und Grundlayout

a) Erstellen Sie eine Seite **index.php**, welches das **Grundlayout** enthält und die Hauptbereiche des sozialen Netzwerks abbildet sowie etwaige CSS und JS-Libraries einbindet. Die Inhalte aller Bereiche werden mittels include eingebunden. (Header, die Navigation, der Hauptinhaltsbereich ...).
b) Die Projektstruktur soll **Komponenten** klar **unterteilen.** Recherchieren Sie diesbezüglich wie führende Open-Source Projekte diese Aufgabe lösen und orientieren Sie sich an best-practice Empfehlungen bei der Strukturierung des Projektes.
c) Die Implementierung des sozialen Netzwerks ist mittels **OOP-Konzept** umzusetzen. Erstellen Sie Klassen für alle Entitäten, die im System gebraucht werden (bspw. User, Posts, …) und legen Sie eine DB-Service-Klasse an, welche zentral die Zugriffe auf die DB implementiert.
d) Achten Sie beim Implementieren darauf, dass **Sicherheitslücken** möglichst **geschlossen** werden.
e) Eine **Datenbank**, welche die Beitragsverwaltung abbildet, ist anzulegen.

Frameworks und Libraries

a) Es bleibt Ihnen überlassen, ob Sie für die Implementierung diverse Frameworks oder Bibliotheken verwenden, wie z.B. Bootstrap, YAML, …
b) Die Verwendung von Fertiglösungen (wie CMS o.ä.) ist nicht erlaubt

Abgabemodalitäten

a) Das Projekt ist **vollständig** als zip/rar File in Moodle hochzuladen (maximal Dateigröße 20MB), inklusive einer Kopie der MySQL-Datenbank (sql-Anweisungen) und einem Hinweis auf die Teammitglieder im Dateinamen, bspw:
    i. **WEB-Projekt-WS2020-21 _Nachname_Nachname**.zip
        - Halten Sie sich unbedingt an die Nomenklatur, um die korrekte Zuweisung und somit Bewertung von gemeinsam abgegeben Projekten sicherzustellen
        - Verkleinern Sie Bilddateien/Videos ggf., um die Größe des Projekts für den Upload zu reduzieren
b) **Kommentieren** Sie komplexere Abläufe/Funktionen im Code
c) Falls weitere Konfigurationsschritte zur Inbetriebnahme der Applikation nötig sind, legen Sie eine **readme.md**-Datei bei (z.B. Zugangsdaten zur Datenbank).

Empfohlene Vorgangsweise bei der Projektumsetzung

Starten Sie zuerst mit den Oberflächen, die Sie benötigen. Beginnen Sie mit einem Basis Template. Überlegen Sie, welche Seiten Sie benötigen (einzeln, Unterseiten …). PHP-Funktionen ergänzen Sie dann nacheinander Schrittweise, bis zur DB-Anbindung.

1) *Beginnen Sie mit einem Basislayout*
2) *Überlegen Sie, welche Elemente in welcher Form dargestellt werden sollen*
    a. *Setzen Sie schon vorgefertigte Komponenten einer Library ein*
    b. *Fügen Sie statische Inhalte hinzu wie bspw. Hilfe/Impressum …*
    c. *Verwenden Sie vorerst statische HTML-Platzhalter für dynamische Inhalte*
3) *Ergänzen Sie das individuelle Design*
    a. *Eigene CSS-Datei*
    b. *Bilder*
    c. *…*
4) *Vergessen Sie nicht, den Code/die Oberfläche zu testen!*
5) *Erstellen Sie alle Eingabemasken/Uploadformulare*
6) *Legen Sie auch leere Files an, für noch zu implementierende Komponenten*
7) *Entwerfen Sie ein erstes DB-Modell (am Papier), das alle benötigten Daten aufnehmen und abbilden kann. Einzelne Tabellen können bereits ein Anhaltspunkt für den Entwurf zukünftiger PHP-Klassen sein (User, Beiträge, …)*
    a. *Sollte das DB-Modell noch unvollständig sein, schärfen Sie dieses sukzessive nach*
8) *Fügen Sie Funktionalität schrittweise hinzu:*
    a. *Ergänzen Sie die Auswertung der Eingabemasken (vorerst noch ohne DB-Anbindung), Prüfung auf Vollständigkeit/Validierung, Fehlermeldungen*
    b. *Überlegen Sie, wo Sessions und Cookies notwendig sind und implementieren Sie diese*
    c. *Fügen Sie die Upload-Funktionalität hinzu*
    d. *Vergessen Sie nicht, den Code zu Kommentieren und zu testen!*
    e. *Entwerfen Sie jene Klassen, die Sie zum Abbilden von Daten (bspw. User, …) und Funktionalitäten (DB-Klasse, ggf. Bildbearbeitung, …) benötigen*
    f. *Verwenden Sie das OOP-Konzept und ziehen Sie die Verwendung von Objekten dort nach, wo dies bisher nicht geschehen ist (bspw. User-Registrierung -> befüllt User-Objekt, …)*
    g. *Überlegen Sie, welche Daten Sie von der Oberfläche in die DB-speichern müssen und fügen Sie eine Funktion nach der anderen in der DB-Klasse hinzu*
        i. *Speichern von Usern*
        ii. *Speichern von Beiträgen*
        iii. *…*
    h. *Wo können sich Änderungen in der DB ergeben? Fügen Sie weiter Funktionen in der DB-Klasse hinzu:*
        i. *Aktualisierungen von Userdaten*
        ii. *Liken von Beiträgen*
        iii. *Hinzufügen von Kommentaren*
        iv. *…*
    i. *Welche Daten benötigen Sie aus der DB? Ergänzen Sie weitere Funktionen:*
        i. *Userdaten abrufen und an der vorbereiteten Oberfläche anzeigen*
        ii. *Beiträge abrufen*
        iii. *…*
    j. *Vergessen Sie nicht, den Code zu Kommentieren und zu testen!*
    k. *Ergänzen Sie etwaige Zusatzfeatures für Zusatzpunkte*

Grundidee

User der Website haben die Möglichkeit Beiträge zu erstellen (Text und/oder Bilder), und mit unterschiedlichen Rechten freizugeben. Freigegebene Beiträge anderer User können in der eigenen Ansicht dargestellt werden. Schwerpunkte sollen sein:

- **User**:
  Userverwaltung (2-3 Zustände: anonym (BesucherIn), User (registrierter) User, Admin). Admins können auch User verwalten.
- **Datenbank**:
  Grundlegende Verwaltung aller Daten in einer Datenbank (User, Posts, Likes, Kommentare, …) → Aufbau einer geeigneten Datenbankstruktur, Zugriff mittels entsprechender DB-Zugriffs-Klassen
- **Upload / eigene Beiträge**:
  User können eigene Beiträge erstellen, diese verwalten (bearbeiten, löschen) und mit Freigaben versehen (öffentlich oder nur registrierte Benutzer)
- **Liken / Disliken und Kommentieren von Beiträgen**:
  Beiträge können von registrierten Benutzern geliket, disliket und/oder kommentiert werden.
- weitere Bereiche der Webseite:
  - **Hilfe** (Benutzeranleitung)
  - **Impressum** (Standard-Impressum), **Namen und Bilder der Ersteller** der Website

Die gesamte Website muss **responsive** sein und somit für Smartphones, Tablets und Desktop-Rechner verwendbar sein.

Für eine außergewöhnlich gute **Usability**, Grundzüge der **Accessibility** und **Suchmaschinenoptimierung** werden Zusatzpunkte gewährt.

Userverwaltung

Es gibt 3 verschiedene Usertypen: **BesucherIn,** (registrierter) **User, Admin**

Alle Usertypen sehen

- Hilfe und Impressum
- Anzeigen von frei verfügbaren (freigegebenen) Beiträgen
- Suchen/Filtern von Beiträgen

Was kann der **anonyme User**:

- Registrierung und Login sind verfügbar
  - Hinweis: eine Registration über das Webinterface speichert immer normale User, AdministratorInnen müssen direkt in der Datenbank als solche gekennzeichnet werden
- KEINE eigenen Beiträge oder Uploadmöglichkeit
- Es werden aber alle öffentlichen Beiträge dargestellt.

Was kann der **registrierte User** zusätzlich zum anonymen User:

- statt Registrierung und Login kann sich ein eingeloggter User wieder ausloggen
- Bearbeitung des eigenen Profils
- Benutzerbild als Thumbnail erstellen und ändern
- Eigene Beiträge erstellen
- Eigene Beiträge verwalten (Freigabe, Löschen, Ändern)
- Öffentliche und Beiträge anderer registrierter Benutzer liken/disliken und/oder kommentieren

Was kann der/die **AdministratorIn** zusätzlich zu registrierten Usern:

- Verwaltung ALLER Beiträge (also nicht nur eigener): Erstellen, Liken/Disliken, Kommentieren, Löschen
- Userverwaltung (Übersicht bestehender und deaktivieren einzelner User)

Aufbau der Datenbank

Datenbankmodell

- Entwickeln Sie ein **geeignetes Datenbankmodell**
  - Hinweise:
    Sie müssen User und ihre Rolle speichern, Beiträge verwalten (etwaige Bilder selbst liegen im Filesystem des Webservers, die Datenbank verwaltet nur Referenzen auf die Bilder), wer Ersteller eines Beitrags ist und ob er öffentlich oder nur für registrierte Benutzer freigegeben wird. Wer etwas liket/disliket. Wer kommentiert hat und den Inhalt des Kommentars.

Datenbankzugriff mittels PHP

- Der Datenbankzugriff hat über mySQLi zu erfolgen (alternativ auch PDO möglich)
- Die Zugangsdaten sollen in einer zentralen config-Datei gespeichert werden, damit sie an zentraler Stelle leicht zu ändern sind.
- Die Kommunikation zwischen Datenbank und PHP muss über PHP-Klassen (objektorientierte Programmierung) realisiert werden.
- Explizite Vorkehrungen gegen SQL Injections werden mit Extrapunkten belohnt.

Beschreibung der einzelnen Komponenten

User-Registrierung

a) Beim Erfassen eines Users sind (zumindest) folgende Daten anzugeben:
   i. Anrede
   ii. Vorname
   iii. Nachname
   iv. Emailadresse
   v. Benutzername

vi. Passwort

b) Überprüfen Sie alle eingegebenen Daten sowohl clientseitig (mit speziellen HTML5 Tags voreinschränken) als auch serverseitig auf Vollständigkeit und Richtigkeit. Das Passwort muss immer 2x angegeben werden und die Werte werden auf Übereinstimmung geprüft. Der Username muss eindeutig sein (auch in der Datenbank). Erst wenn alle Daten validiert wurden, wird der neue User in der DB angelegt. Das Passwort wird verschlüsselt in der DB abgelegt.
c) Legen Sie manuell in der Datenbank einen eigenen User an, der als AdministratorIn gekennzeichnet wird. Die Rechte des/der AdministratorIn unterscheiden sich von herkömmlichen UserInnen.

User-Login

a) Erstellen Sie ein Login-Formular mit Eingaben für Usernamen und Passwort
b) Überprüfen Sie die Werte gegen die Datenbank. Nur wenn es genau eine/n UserIn in der DB mit den Daten gibt, gilt der Login als ok und der Username wird am Bildschirm angezeigt. Der Login-Status ist permanent auf der Seite sichtbar. Eingeloggte User sehen einen Logout-Button.
c) Auf Basis des eingeloggten Users werden Features der Webseite freigeschaltet oder deaktiviert.

Profilverwaltung

a) Eingeloggte User können eigene Registrationsdaten einsehen und bearbeiten (Stammdaten bearbeiten, Passwort neu setzen, …)
b) Achten Sie darauf, dass sensible Informationen (Passwort) nicht vollständig angezeigt werden und **beim Ändern des Passwortes** auch das **alte Passwort** eingegeben und überprüft werden muss.

Hilfe / Impressum

a) Erstellen Sie eine Hilfe-Seite, auf der erklärt wird, wie die Website zu verwenden ist (kurze Benutzeranleitung)
b) Fügen Sie ein Impressum ein (→ recherchieren Sie, was aktuell ein Impressum enthalten muss).
c) Das Impressum enthält zusätzlich Bilder und Namen der am Projekt beteiligten Personen
d) Sowohl Hilfe als auch Impressum müssen jederzeit einfach erreichbar sein

Beitragsverwaltung generell

a) **Beitragsanzeige**:
Anzeige der Beiträge im Browser übersichtlich gruppiert und voneinander getrennt (zusammengehörige Elemente wie Bild und Text sind gut sichtbar sowie eine klare Abgrenzung zu anderen Beiträgen), die neuesten Beiträge sollen immer ganz oben angezeigt werden
b) **Zusatzinfos der Beiträge**:
Kommentare, Likes/Dislikes, Tags werden unter den Beiträgen abgebildet (z.B.: eingerückt als “Thread”)
c) **FileUpload**:
Hochladen eigener Bilder die – serverseitig - verkleinert in den Beiträgen angezeigt

werden

d) **Freigabe von Beiträgen:**
   Beiträge können öffentlich oder nur für registrierte Benutzer freigegeben werden
e) **Kommentieren & Liken**:
   Kommentare und Likes müssen für Beiträge anderer User erstellbar sein.

Beitragsverwaltung / Beitragsanzeige

Die Beitragsanzeige ist ein zentraler Bestandteil des Projekts. Die Darstellung soll sehr flexibel sein und Benutzern vielfältige Einstellmöglichkeiten bieten

a) Bilder werden **als Thumbnails** konstanter Größe dargestellt. Achten Sie darauf, dass die Bilder, dort wo sie als Thumbnails benötigt werden, tatsächlich nur in Thumbnail-Größe vom Server zum Browser übertragen werden. Es empfiehlt sich die Thumbnails als fertige Bilder am Server in einem eigenen Verzeichnis vorzubereiten (automatische Verkleinerung mittels PHP).
b) Die Darstellung soll übersichtlich sein, ein Beitrag ist klar als solcher erkennbar und von anderen unterscheidbar, kann – wenn vorhanden auch Bilder - darstellen und es soll klar erkennbar sein, wer Autor des Beitrags ist.
c) Beiträge (eigene, öffentliche, für registrierte Benutzer) werden auf der Seite in Reihenfolge vom aktuellsten bis zum ältesten abgebildet.
d) **Zusatzinformationen** zu Beiträgen werden angezeigt, das sind z.B. i. Datum
    a. Autor des Beitrags
    b. Likes/Dislikes (Anzahl muss auch dargestellt werden)
    c. Tags
    d. Freigabestatus (öffentlich oder registrierte Benutzer)
e) **Sortierung** nach mindestens zwei verschiedenen Kriterien entsprechend der Zusatzinformationen (z.B.: Datum, Beiträge mit den meisten Likes, mit/ohne Bild, Beiträge von User xyz, …)
f) (nur) **eigene Beiträge können** auch wieder **gelöscht werden**.
g) **Fancybox/Lightbox**: durch Klick auf ein Bild wird dieses im Großformat in einer Fancybox/Lightbox dargestellt. (zusätzliche JS Library nötig)

Beitragsverwaltung / Fileupload

Es können neue (eigene) Bilder aus dem Filesystem zum Server hochgeladen werden. Dabei ist zu beachten, dass nur Bilddateien für den Upload erlaubt sein sollen.

Serverseitig müssen hochgeladene Bilder weiterbearbeitet werden:

a) Hochgeladene Bilder werden immer dem gerade erstellten Beitrag als **Besitzer** zugeordnet
b) **Speichern der Bilder** erfolgt in einem eigenen Bilderverzeichnis.
c) Gleichzeitig wird die **Erstellung von Thumbnails** in einem eigenen Verzeichnis, für die spätere Verwendung in der Bildanzeige empfohlen.
d) Empfohlen wird weiters die **Speicherung von häufig benötigten Informationen in der Datenbank** (Bildname, …)

Beitragsverwaltung / Suche
Beiträge können geliket, disliket und kommentiert werden:

a) Likes/Dislikes sollen mit eindeutigen Icons dargestellt werden.
b) In der Anzeige gibt es die Möglichkeit **nach Tags zu filtern**. Es werden dann nur solche Beiträge angezeigt.
c) Man kann auch nach mehreren Tags gleichzeitig filtern.
d) Weiters gibt es ein **Suchfeld**. Eingegebene Begriffe werden in den Beiträgen, Kommentaren und – wenn vorhanden - Bildnamen gesucht.

Beiträge freigeben

a) Bei jedem Beitrag gibt es die Option diesen öffentlich oder nur für registrierte Benutzer freizugeben
b) Beiträge, die freigegeben wurden, werden entsprechend gekennzeichnet (mit den Erstellernamen oder öffentlich zur Verfügung gestellt werden)
c) Die Freigabe kann jederzeit wieder geändert werden

Userverwaltung für AdministratorInnen
Nur AdministratorInnen haben die Möglichkeit User zu verwalten. Dazu gehört:

a) Anzeige aller User und Auswahlmöglichkeit
b) Zu jedem User eine Liste aller Beiträge und der Freigaben
c) Verändern des Userstatus: aktiv, inaktiv
d) Inaktive User können sich nicht mehr einloggen, bleiben aber in der Datenbank erhalten.
e) Reset des Passwortes eines bestimmten Users und Zusenden des neuen Passwortes an die eingetragene Emailadresse (Achtung: Email-Versand bei XAMPP erfordert mitunter Anpassung der PHP-Settings).