OOP Challenge

Challenge A: OOP Login Systeem

datasource

? Inleiding, wat ga je doen?

De **OOP Challenge** is een fantastische kans om jouw vaardigheden in **objectgeoriënteerd programmeren** (OOP) naar een hoger niveau te tillen en direct toe te passen in de praktijk. Je combineert je **Cyber Security** vaardigheden met de **OOP** vaardigheden.

Je hebt de keuze uit twee boeiende projecten;

- je maakt een OOP Login Systeem, of
- je maakt een (mobiele) ToDo manager die je kan omzetten naar een mobiele applicatie, dit is lastiger maar ook veel leuker want je maakt een echte mobiele app!

? Leerdoelen

- Je past klassen, objecten, constructors en methodes toe in een realistisch scenario.
- Je gebruikt encapsulation om gebruikersgegevens te beveiligen.
- Je slaat gebruikers op met PDO en hashed wachtwoorden.
- Je maakt een registratie- en loginroutine in OOP-stijl.

? Opdrachtbeschrijving

Situatie: Je werkt bij een klein webbureau en een klant wil een eenvoudig maar veilig **login-systeem** laten bouwen in PHP. Je gaat dit project zelfstandig uitvoeren als een mini-challenge. Dit is een uitstekende kans om je **OOP-vaardigheden** te laten zien in je **GitHub portfolio**. Dat kan je helpen om sneller een stage of baan te vinden in de webdevelopment-sector!

Wat je gaat bouwen:

- Een registratiepagina waar nieuwe gebruikers zich kunnen aanmelden
- Een loginpagina waar gebruikers zich kunnen aanmelden
- Beveiligde opslag van wachtwoorden via `password_hash()`
- Gebruik van `PDO` voor communicatie met de database (bijv. SQLite of MySQL)

? Structuur en Klassen

- **Class User**
  - Properties: `private $id`, `$username`, `$passwordHash`
  - Constructor stelt naam + wachtwoord in (en hashed het wachtwoord)
  - Methode `verifyPassword($plainText)` vergelijkt met hash
  - Getters voor `getId()` en `getUsername()`
- **Class UserDatabase**
  - Maakt verbinding via `PDO` (gebruik SQLite of MySQL)
  - Methodes: `addUser(User $user)`, `findUserByUsername($name)`
  - Prepared statements verplicht!

? Technische vereisten

- Gebruik **password_hash()** en **password_verify()** voor wachtwoorden
- PDO met **prepared statements** (geen ruwe SQL!)
- Toon foutmeldingen bij fouten (bestaande user, fout wachtwoord, etc.)

?? Bestanden

- `User.php` – beschrijft de user zelf
- `UserDatabase.php` – regelt opslag en ophalen
- `register.php` – laat gebruiker registreren
- `login.php` – laat gebruiker inloggen
- `login-success.php` – pagina die je ziet na succesvol inloggen

? Reflectie

- Wat zijn de voordelen van wachtwoord hashing?
- Waarom gebruiken we een aparte klasse voor opslag en een voor gebruikers?
- Wat zou je uitbreiden als je dit systeem echt online wilde zetten?

? Bonus: Voor je portfolio

Zet dit project op **GitHub** met een korte uitleg in de README over hoe het werkt. Voeg schermafbeeldingen toe van het registratie- en loginproces. Dit is een mooi voorbeeldproject om te laten zien wat je kunt!

? Gebruik AI

AI gebruik op de juiste manier, zoals geleerd tijdens de lessen, is toegestaan en wordt zelfs aangemoedigd.

Het is wel verplicht om de AI log in te leveren en je moet je code kunnen uitleggen!

? Inleveren

- AI log - alle hulp van AI inleveren!
- SQL export.
- Alle bestanden nodig voor jouw website.
- Reflective in PDF
- (Optioneel) Link naar je GitHub repository

Challenge B: (mobiele) OOP ToDo Manager

? Inleiding, wat ga je doen?

De **OOP Challenge** is een fantastische kans om jouw vaardigheden in **objectgeoriënteerd programmeren** (OOP) naar een hoger niveau te tillen en direct toe te passen in de praktijk. Je combineert je **Cyber Security** vaardigheden met de **OOP** vaardigheden.

Je hebt de keuze uit twee boeiende projecten;

- je maakt een OOP Login Systeem, of
- je maakt een (mobiele) ToDo manager die je kan omzetten naar een mobiele applicatie, dit is lastiger maar ook veel leuker want je maakt een echte mobiele app!

? Leerdoelen

- Je past OOP toe in een realistisch webproject.
- Je maakt meerdere klassen die met elkaar samenwerken.
- Je gebruikt PDO en password hashing op een veilige manier.
- Je bouwt een werkend mini-systeem dat je op GitHub kunt zetten.

? Opdrachtbeschrijving

Situatie: Een start-up wil een prototype van een eenvoudige "ToDo Manager" waarin gebruikers taken kunnen aanmaken, afvinken en verwijderen. Jij bouwt dit systeem in PHP met OOP. Dit is een mooie kans om een compleet mini-project te maken voor je **GitHub portfolio** — ideaal om te laten zien bij een sollicitatie voor stage of werk!

? Functionaliteiten

- Registratiepagina: nieuwe gebruikers kunnen zich aanmelden
- Loginpagina: gebruikers kunnen inloggen met hun wachtwoord
- Taakpagina: ingelogde gebruikers kunnen taken toevoegen, afvinken en verwijderen
- Alle data wordt opgeslagen in een database met **PDO** (SQLite of MySQL).

? Mobiele App (optioneel)

- Je kan de ToDo manager omzetten naat een **Progressive Web App**. Je krijgt de beste gebruikers ervaring als je je ToDO manager zoveel mogelijk omzet naar een **Single Page Web Application** (SPA). Dat kan met een Framework, maar ook met JavaScript.

Wat is een SPA en een Progressive Web App?

Een **one page applicatie** (of **Single Page Application**, afgekort **SPA**) is een webapplicatie of website die uit **één HTML-pagina** bestaat en **dynamisch inhoud laadt**, zonder de hele pagina opnieuw te laden bij navigatie of interactie.

? Hoe werkt het?

Bij een SPA:

- Wordt bij het eerste bezoek één HTML-pagina geladen.
- Daarna worden alleen **delen van de pagina aangepast via JavaScript** (meestal met behulp van frameworks zoals React, Vue of Angular).
- Communicatie met de server gebeurt via **AJAX** of **fetch()-aanroepen** om data op te halen of op te slaan, vaak in JSON-formaat.

Je kunt een SPA omzetten in een PWA, Progressieve Web App.

? Wat is een PWA?

Een **PWA** is een **website die aanvoelt als een app**. Je opent hem in de browser, maar je kunt hem ook **op je telefoon zetten als icoon**, net zoals een echte app uit de App Store of Play Store.

? Hoe werkt het?

Een PWA:

- Start gewoon via je browser (zoals Safari of Chrome)
- Kan op je **beginscherm gezet worden** als een app-icoon
- **Werkt ook offline** als je het goed instelt
- Ziet eruit en werkt zoals een gewone app

Wil je een Mobiele app op deze manier maken, laat AI je dan de details verder uitleggen,

? Klassen

- **Class User**
  - Properties: `$id`, `$username`, `$passwordHash`
  - Constructor en `verifyPassword($plainText)`
- **Class UserDatabase**

- Maakt verbinding via `PDO`
- Methodes: `addUser()`, `findUserByUsername()`

- **Class Task**
  - Properties: `$id`, `$user_id`, `$description`, `$completed`
  - Methodes: `toggleCompleted()`, `getDescription()`, `isCompleted()`

- **Class TaskManager**
  - Methodes: `addTask()`, `getTasksByUser()`, `deleteTask()`, `toggleTask()`
  - Alle database-interacties via `PDO` met prepared statements

? Reflectie

- Waarom werken we met meerdere klassen in plaats van één grote?
- Wat gebeurt er als een niet-ingelogde gebruiker naar de takenpagina navigeert?
- Hoe zorgt jouw systeem ervoor dat taken niet tussen gebruikers verwisseld kunnen worden?

? Bonus voor je portfolio

Zet dit project op **GitHub**. Voeg screenshots toe en schrijf een README-bestand waarin je uitlegt wat het systeem doet. Dit laat aan stagebedrijven of werkgevers zien dat jij zelfstandig een werkend OOP-project kunt bouwen!

? Gebruik AI

AI gebruik op de juiste manier, zoals geleerd tijdens de lessen, is toegestaan en wordt zelfs aangemoedigd.

Het is wel verplicht om de AI log in te leveren en je moet je code kunnen uitleggen!

? Inleveren

- AI log - alle hulp van AI inleveren!
- SQL export.
- Alle bestanden nodig voor jouw website.

- Reflective in PDF
- (Optioneel) Link naar je GitHub repository

--

Revision #14
Created 2025-06-18 14:34:02 UTC by Max
Updated 2026-07-10 09:29:10 UTC by Max