

Projektuppgift 1 - Blogg

Grupp

Betygsnivå: IG/G/VG

Mål

Målet med uppgiften är att utvärdera dina kunskaper i följande moment:

- **Versionshantering och samarbete med GIT**
- **Arbeta mot databas med PDO i PHP**
- **Formulärhantering**
- **GET/POST**
- **Sessionshantering/Användarhantering**
- **Funktioner i PHP**
- **Objektorientering i PHP**
- **Planering av webbapplikationer.**
- **Hantering av personuppgifter och säkerhet.**

Fil och mappstruktur


Detta är en standard för hur strukturen för projektet ska se ut. Har ni flera includes eller andra klasser eller flera css-filer så är det ok att lägga till dessa, huvudsaken är att strukturen följer detta mappmönster. Ni får alltså ha fler filer än vad som listas nedan och de filer som listat är inte säkert att ni behöver ha, ni kanske t.ex. inte har en fil som heter Product.php.

Din inlämning måste följa strukturen nedan annars får ni automatisk komplettering. Din mapp som ni har er uppgift i **måste** heta enligt namnmönstret nedan. Er css ska ligga i en mapp som heter css och själva filen ska heta style.css. Om ni har några bilder kopplade till er inlämning ska de ligga i en mapp som heter images. Om ni har bilder så ska dessa vara i en rimlig storlek och ni ska inte skicka med en 10mb stor bakgrundsbild som inte behövs (tumregel är att bilderna kan vara max 1mb, men gärna max 500kb).

Inlämningen ska ha en fil som heter README.md, d.v.s. filen heter README med stora bokstäver och filformatet är .md. Som get en beskrivning av projektet samt innehåller information om den gemensamma kodstandarden, d.v.s. vilka regler och normer ni har för att skriva kod i gruppen.

index.php ska vara första sidan man kommer till. Resterande sidor som inte är index.php ska ligga i en mapp som heter views.

-  gruppnamn_project
 -  css
 -  style.css
 -  images
 -  image.png/image.jpg/image.svg
 -  includes/partials
 -  database_connection.php
 -  functions.php

-  classes
 -  Comments.php
 -  Posts.php
-  views
 -  login.php
 -  post.php
-  index.php
-  README.md
-  GUIDELINES.md / CODE_STANDARD.md
-  database.sql

Uppgiftsbeskrivning

Vi på Millhouse vill ha ett enklare blogg-system där användaren kan logga in, skapa inlägg samt kommentera på dessa inlägg. Systemet ska ha ett lättanvänt gränssnitt och rikta sig till personer med låg datorvana.

Företaget Millhouse

Millhouse är ett grossistföretag som säljer kläder, accessoarer och mindre inredningsartiklar till mode- och livsstilsbutiker. Millhouse planerar nu att starta en rad egna webbshopar med produkttyperna:

- klockor
- solglasögon
- mindre inredningsartiklar

För att skapa dialog med slutkund vill de skapa en blogg där kunderna kan kommentera, komma med önskemål med mera. Det är denna del man önskar köpa in från ett mindre bolag via en pitch. Millhouse omsätter 75 miljoner kronor årligen och har 50 anställda, främst inom administration, inköp och lagerhantering.

Millhouse har inte tidigare haft en direkt relation med slutkunderna, eftersom de varit grossister. De har ingen direkt tidigare erfarenhet av e-handel och att marknadsföra direkt mot kund. Det är därför sannolikt att Millhouse uppfattar det som positivt om ni bidrar med insikter om marknaden och kunderna inom det produktsegment ni väljer.

Krav

- Ni har implementerat alla egenskaper som är listade under Egenskaper
- Varje användare ska sparas till databasen och lösenordet ska vara krypterat.
 - Varje användare ska vara en av två roller: **Admin** eller **User**. En admin kan ändra på allt och göra allt medan en **User** enbart kan läsa informationen på sidan samt kommentera inlägg.

- För att se informationen på sidan måste man vara inloggad som antingen **User** eller **Admin**.
- Varje inlägg ska sparas till databasen
 - Varje inlägg ska ha en ett **unikt ID, titel, beskrivning, bild, kategori, datum för skapande** samt ett ID på den användaren som har skapat inlägget.
- Varje kommentar ska sparas till databasen
 - Varje kommentar ska ha ett **unikt ID, innehåll, datum för skapande, ID för inlägget som det tillhör** samt **ID på vem som har skapat kommentaren**
- All information lagras i databasen och databasen är uppdelad i flera tabeller.
- Ni använder er utav egenskrivna funktioner och klasser i projektet.

Egenskaper

Egenskap: Registrera användare

För att kunna använda sidan behöver vi logga in. För att logga in behöver vi kunna registrera en ny användare.

Scenario: Användaren vill registrera sig

Givet: att användaren besöker den separata sidan för registrering

Så: visas ett registreringsformulär nödvändiga inputfält för registrering, samt en knapp för att skicka formuläret.

Scenario: Användaren registrerar sig

Givet: att användaren befinner sig på registreringssidan

Och: fyller i alla fält som är nödvändiga att fylla i

När: användaren klickar på *“Registrera”/“Register”*

Så: registreras användaren och tas till den separata inloggningssidan eller loggas in direkt och tas till förstasidan

Scenario: Användaren misslyckas med registrering

Givet: att användaren besöker registreringssidan

Och: missar att fylla i ett nödvändigt fält

När: användaren klickar på *“Registrera”/“Register”*

Så: omdirigeras användaren tillbaka till samma sida med ett errormeddelande

Och: ingen användare skapas i databasen

Scenario: Användarnamn/email finns redan

Givet: att användaren besöker registreringssidan

Och: fyller i en redan existerande användares mail eller användarnamn

När: användaren klickar på *“Registrera”/“Register”*

Så: omdirigeras användaren tillbaka till samma sida med ett errormeddelande

Och: ingen användare skapas i databasen

Egenskap: Användarinloggning

För att kunna skapa och redigera innehåll på sidan så måste vi ha loggat in.

Scenario: en användare vill logga in

Givet: att användaren besöker den separata inloggningssidan

Så: visas ett inloggningsformulär med inputfält där man kan fylla i all nödvändig information för att logga in

Scenario: en användare loggar in

Givet: att användaren besöker inloggningssidan

Och: fyller i alla fält som är nödvändiga

När: användaren klickar på *“Logga in”*/*“Log in”* eller liknande knapp

Så: loggas användaren in och tas till förstasidan

Och: användare förblir inloggad via en session

Scenario: en användare misslyckas med inloggning

Givet: att användaren besöker inloggningssidan

Och: missar att fylla i ett nödvändigt fält

När: användaren klickar på *“Logga in”*/*“Log in”*

Så: omdirigeras användaren tillbaka till samma sida med ett errormeddelande

Och: ingen användarsession skapas

Egenskap: Logga ut användare

Vi behöver också kunna logga ut en specifik användare utifall vi vill byta användare eller liknande

Scenario: en användare vill logga ut

Givet: att användaren är inloggad och befinner sig på valfri sida

Och: användaren klickar på *“Log out”*/*“Logga ut”* eller liknande menyval

Så: Loggas användaren ut från sidan och användarens session förstörs

Och: sidan laddas om

Egenskap: Skapa nytt inlägg

Vi behöver kunna lagra ett inlägg under en längre tid och låta andra användare se innehållet. För att låta användaren kunna kolla innehållet på flera olika enheter och tidpunkter behöver vi lagra inlägget i en databas.

Scenario: en användaren vill skapa ett nytt inlägg

Givet: att en användare besöker förstasidan

Och: trycker på *“Skapa inlägg”* eller liknande menyval

Och: användaren är inloggad

Och: användaren har rättigheter att skapa ett nytt inlägg

Så: omdirigeras användaren till den separata sidan för att skapa nytt inlägg

Scenario: en användare skapar nytt inlägg

Givet: användaren befinner sig på den separata sidan för att skapa nytt inlägg

Och: användaren har fyllt i alla nödvändiga fält

Och: trycker på knappen "*Skapa*" / "*Create*" eller liknande knapp

Så: Skapas ett nytt inlägg i databasen

Och: användaren blir omdirigerad till förstasidan

Scenario: en användare misslyckas med att nytt inlägg

Givet: användaren befinner sig på den separata sidan för att skapa nytt inlägg

Och: användaren har missat att fylla i ett nödvändigt fält

Och: trycker på knappen "*Skapa*" / "*Create*" eller liknande knapp

Så: Skapas **inte** ett nytt inlägg i databasen

Och: användaren blir omdirigerad till samma sida med ett errormeddelande

Egenskap: Lista inlägg

När vi kommer till förstasidan vill vi ha en lista på vilka inlägg som finns skapade.

Scenario: Visa sidans senaste skapade inlägg

Givet: att användaren besöker förstasidan

Och: användaren är inloggad

Så: visas en lista med de senaste inläggen

Scenario: Visa inte sidans senaste inlägg vid utloggat tillstånd

Givet: att användaren besöker förstasidan

Och: användaren **inte** är inloggad

Så: omdirigeras användaren till inloggningssidan

Egenskap: Ta bort inlägg

En admin ska kunna ta bort inlägg på förstasidan samt på ett individuellt inlägg

Scenario: en användaren vill ta bort ett inlägg på förstasidan

Givet: att användaren besöker förstasidan

Och: användaren är inloggad

Och: användaren har rätt rättigheter att ta bort ett inlägg

Och: användaren klickar på "*Ta bort inlägg*" / "*Remove*" på ett specifikt inlägg

Så: tas inlägget bort från databasen

Och: sidan laddas om

Scenario: en användare vill ta bort ett inlägg på ett specifikt inlägg

Givet: att användaren besöker ett specifikt inlägg

Och: användaren är inloggad
Och: användaren har rätt rättigheter att ta bort ett inlägg
Och: användaren klickar på *“Ta bort inlägg”/“Remove”*
Så: tas inlägget bort från databasen
Och: användaren omdirigeras till förstasidan

Egenskap: Redigera inlägg

Vi vill kunna i efterhand redigera ett visst inlägg när någon har skapat det. Enbart admins ska kunna redigera ett inlägg

Scenario: en användare vill redigera ett inlägg
Givet: att användaren besöker förstasidan
Och: användaren är inloggad
Och: användaren har rätt rättigheter att redigera ett inlägg
Och: användaren klickar på *“Redigera”/“Edit”* på ett specifikt inlägg
Så: omdirigeras användaren till redigeringssidan för det enskilda inlägget

Scenario: en användare redigerar ett inlägg
Givet: användaren befinner sig på den separata sidan för att redigera ett inlägg
Och: användaren har fyllt i och uppdaterat alla nödvändiga fält
Och: trycker på knappen *“Uppdatera”/“Update”* eller liknande knapp
Så: uppdateras det existerande inlägget i databasen med den nya informationen
Och: användaren blir omdirigerad till förstasidan
Och: inlägget är uppdaterat på förstasidan

Egenskap: Visa kommentarer

Vi måste även kunna se alla kommentarer som finns kopplat till ett specifikt inlägg för att kunna läsa all feedback till ett specifikt inlägg

Scenario: en användare vill se kommentarer för ett inlägg
Givet: att en användare besöker förstasidan
Och: trycker på *“Kommentera”/“Comment”* på valfritt inlägg
Så: omdirigeras användaren till en separat sida som visar den enskilda produktens information
Och: det visas en lista på de senaste kommentarerna
Och: det visas ett fält för att kommentera inlägget

Egenskap: Kommentera inlägg

Användaren ska kunna kommentera ett valfritt inlägg för att kunna ge feedback på inlägget

Scenario: en användare vill kommentera ett inlägg
Givet: att en användare besöker förstasidan
Och: trycker på *“Kommentera”/“Comment”* på valfritt inlägg

Så: omdirigeras användaren till en separat sida som visar den enskilda produktens information samt ett kommentarsfält

Scenario: en användare kommenterar ett inlägg

Givet: att användaren befinner sig på sidan för den enskilda produkten

Och: har fyllt i alla nödvändiga fält

Och: trycker på knappen *“Kommentera”/“Comment”*

Så: sparas kommentaren till det specifika inlägget i databasen tillsammans med den inloggade användarens id/användarnamn

Och: sidan laddas om och kommentaren visas då för användaren i listan av kommentarer

Egenskap: Ta bort kommentar

En admin ska kunna ta bort olämpliga kommentarer från ett inlägg

Scenario: en användare tar bort en kommentar

Givet: att användaren befinner sig på sidan för den enskilda produkten

Och: användaren är inloggad

Och: användaren har rätt rättigheter att ta bort en kommentar

Och: trycker på knappen *“Ta bort kommentar”/“Remove Comment”*

Så: tas kommentaren bort från inlägget

Och: sidan laddas om och kommentaren är då borta

Bedömning

Bedömning sker individuellt och alla i gruppen måste bidra till projektet i lika hög grad. Detta visar ni lättast genom att göra commits då den som bedömer lättare kan se vem som har gjort vad. Detta går även att se om ni för loggbok eller en board (Trello) över den uppgifter som ska göras under projektets gång och också visar vem som har slutfört uppgiften.

G/VG baseras på hur väl upplagd koden är och i vilken grad du utnyttjar programmeringsspråket. Vidare baseras det till stor del på att man självständigt implementerar en stor del av lösningen. Du ska också visa att du kan på egen hand tänka på vilka säkerhetsrisker som finns i applikationen samt hur dessa ska åtgärdas.

- Väl upplagd kod. Detta innefattar konsekvent namngivning av variabler, korrekt intabning av koden och en logisk kodföljd.
- Förmåga att välja och använda rätt funktionalitet för rätt ändamål gällande variabler, loopar, statements och funktioner på ett smart sätt.
- Förmåga att kommentera kod när det behövs och att kommentera på ett tydligt sätt som hjälper läsaren av koden att förstå vad koden ska utfärda.
- Förmåga att lägga upp en databasstruktur som passar ändamålet.
- Förmåga att samarbeta med kod i grupp via GIT

Inlämning

- **Inlämningsformat:** .zip (alla andra format resulterar i försenad inlämning och komplettering)
- **Namnformat för inlämning:** gruppnamn_project.zip
- **Inlämningstid:**