|  |
| --- |
| **Projekt:**  Webshop (Melon AB) |

## **Hur har projekt sett ut?**

### **Delmål**

* Bli klar med skissen utav hemsidan
* Bli klar med baksidan av hemsidan
  + **Användare**
    - Login modulen
    - Sign up modulen
  + **Admin**
    - Skapa produkter
    - Ta bort produkter
    - Modifera produkter
  + **Produkter**
    - Lägga till produkter i en kundvagn
    - Ta bort produkter från kundvagnen
    - Lägga en beställning
* Bli klar med framsidan av hemsidan
  + **Huvuddelar**
    - Landing Hemsidan
    - Produkt sidan
    - Skapa konto
    - Logga in
  + **Admin**
    - Admin panel
  + **Produkt**
    - Kundvagn
    - Betala

### **Mål**

Målet med projekt är att skapa en webshop som är lättanvänt från ett användarperspektiv. Projektets mål är att jag ska lära mig nya tekniker och skapa något som när man laddar om sidan ska inte allt man gjort innan försvinna. Jag vill även säkra hemsidan från olika former av attacker mot hemsidan.

Vi uppnår detta målet genom att använda oss av en server i bakgrund och sparar information i cookies (från och med nu kommer jag kalla detta sessions) och i en eventuell databas.

### **Problem jag stött på**

Det största problem jag stötte på var att alla använder kunna genom att skicka en POST request till *localhost/admin/remove/<id>* med då ett id på en produkt som fanns. Detta POST kommando tog då bort produkt helt utan någon form av validering, som jag trodde redan existerade. Detta hände på grund att den PATHen inte var tillräcklig säkrade och jag hade icke implementerad ett system för att stoppa normala använder från att skicka den POST request.

Jag fann senare detta problemet på följande ENDPOINTs:

*localhost/admin/add/<item>*

*localhost/admin/modify/<item>*

### **Hur löser jag eventuella problem?**

Jag löst detta problemet genom att sätta dit en spärr på requesten som gjorde det enbart möjligt för inloggade användare med account level 2 eller högre. Om användare inte skulle nå upp till detta säger vi till webbläsaren att hemsidan inte finns för att öka säkerheten på hemsidan.

De senare ENDPOINTsen fixades på samma sätt.

### **Säkerhet**

Sedan början av projektet har jag suttit säkerheten av projektet högt för att minska risken med möjliga attacker. Följande attacker har jag fixat:

* XSS
* SQL Injection
* Business Logic Errors
* Missing Authentication for Critical Function

XSS kan enkelt förklaras som javascript som kan exekvera på hemsidan utan en användares tillåtelse, detta kan vara farligt då angripare kan smuggla session cookien vilket användes för att verifiera att du är du som är på hemsidan. Detta kan resultera i att använder som blir utsatt för sitt konto övertaget av en angripare.

SQL Injection, detta är när man kan exekvera sql kod på hemsidan för att få ut information från databasen, detta kan göra det möjligt för en angripare att sno database inlogg på användare, användarnamn och lösenord. Lösenorden i databasen är dock krypterade men detta är en mycket farlig attack om det är möjligt på hemsidan.

Business Logic Errors och Missing Authentication for Critical Function, har jag redan gått igenom men anledning till att detta är farligt är att alla använder kan exekvera admin only kommandon som enbart ska kunna användes utav speciellt tillåtna personer.

## **Slutprodukt**

### **Teknik**

I detta projektet har jag använt mig utav följande tekniker:

Programmeringspråk:

* Python3
* Html

Moduler:

* Flask
* Flask SqlAlchemy
* Cryptography

Hosting och version control

* Heroku
* Git
* Github

### **Kod**

Servern som hemsidan bygger på är byggd med programmeringsspråket Python, vilket är ett simpelt skritpingspårkt, men stora möjligheter. I koden har jag även använt mig utav ett par moduler för att underlätta processen, de tre mest notvärda är Flask, Flask SQLAlchemy och Cryptography.

**Flask**

Flask är ett lätt WSGI web applications ramverk. Som är gjort för att enkelt och snabbt kunna slänga upp en komplicerad och kraftfull hemsida. I denna applikation håller flask hand om följande punkter:

* Rendera en dynamisk sida baserat på olika parametrar.
* Hantera kopplingen mellan framsidan och baksidan utav hemsidan.

**Flask SQLAlchemy**

Flask SQLAlchemy är ett tillägg till Flask som lägger till support för att kunna använda modulen SQLAlchemy. Det som SQLAlchemy gör att det lägger till support för att kunna använda oss utav olika typer av databaser. I detta fallet kommer vi att använda oss utav Sqlite3. Som är en simpel och effekt databastyp.

**Cryptography**

Denna modul används för att kryptera lösenorden innan vi sätter dem i databasen. Detta är pågrund utav säkerhetsskäl att vi inte vill spara lösenord i klartext, på grund att motivera olika former av attacker. Då om en angripare skulle kunna komma åt databas innehållet så skulle angriparen inte ha tillgång till lösenorden då dem är krypterade.

### **Utvecklingsområden**

De områden jag skulle vilja utveckla är hantera validering bättre, städa upp koden och bryt ut delar i funktioner. Jag skulle vilja bygga vidare på användarprofilen och visa lite mer relevant information och vissa bland annat gamla beställningar från användaren. Jag skulle även vilja lägga till en metod för att byta lösenord om du har glömt ditt tidigare genom en *forgot-password* metod eller liknande.

### **Användarfel**

Jag har tänkt väldigt mycket på det och har använt mig utav *flash* funktionen i Flask för att vissa använder om den har gjort något fel eller om dem behöver göra något innan dem kommer åt en viss sida. Exempel på detta till exempel när användaren försöker att att lägga till en produkt i varukorgen utan att vara inloggad. Då skickar vi dem till login sidan och säger att dem måste logga in innan dem fortsätter med sitt handlande.

### Användningsområden

Denna hemsidans användningsområde är för alla världens melon älskare. Nej men skämt o sido, användningsområdet för denna hemsidan är ganska minimal då det är typ enbart en prototyp för hur man skulle kunna bygga en webshop.

## 

## **Övrigt**

### Källkod

Github: <https://github.com/y3ll0wlife/LBS-Project/tree/main/Projekt/Webshop>

Zip med kod: <https://github.com/y3ll0wlife/LBS-Project/raw/main/Projekt/Webshop/Webshop.zip>

### Live Hemsida

Kanske tar lite tid att starta upp. Men vänta typ 30 sekunder sedan borde den vara öppen.

<https://melon-webshop.herokuapp.com/>