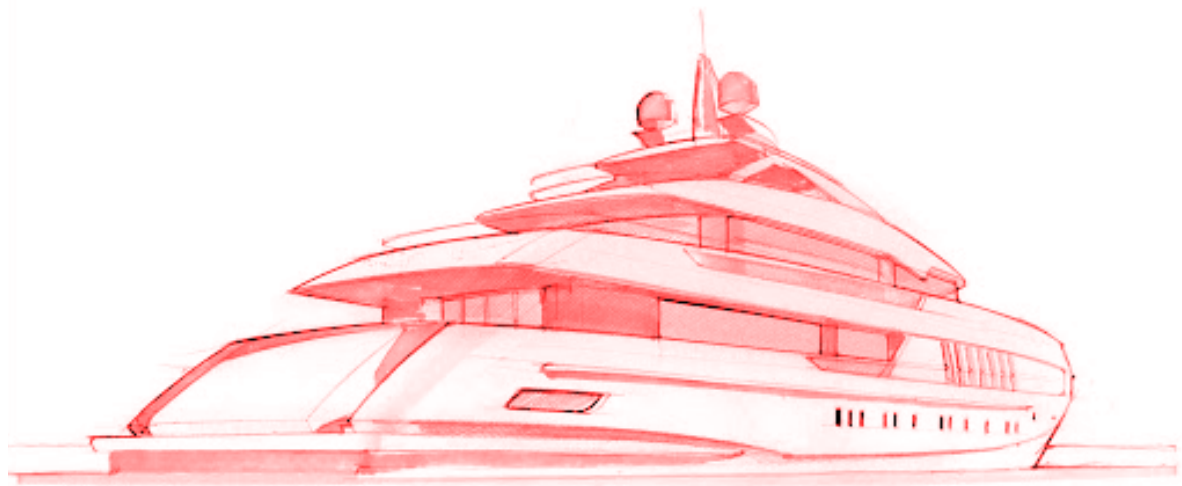


# VOLVO ocean Race



Gruppe 5  
Anders  
Nicklas  
Simon  
Tanya

TEC Praktik Center Ballerup  
P2 Case Opgave  
D.19/3-21

<b>1.0 Indledning</b>	<b>4</b>
<b>2.0 Opgavebeskrivelse</b>	<b>4</b>
<b>3.0 Løsningsforslag</b>	<b>5</b>
3.1 Vores Løsning	5
3.1 Anvendte teknologier	5
3.1.1 Xampp	5
3.1.2 Html, CSS og JavaScript	5
3.1.3 PHP	6
3.1.4 Bootstrap	6
3.1.6 ReactJS	6
3.1.6 MariaDB	6
3.2 Mockup	7
3.3 UML	9
<b>4.0 Navigationsbar</b>	<b>10</b>
4.1 Funktioner	10
4.2 Udfordringer	10
<b>5.0 Database</b>	<b>11</b>
5.1 Forord: MariaDB & PHPMyAdmin	11
5.2 Klassediagram	11
5.3 Skabelsen af databasen	12
5.4 Indsættelse af Data.	13
5.5 Eksport og Import	14
<b>6.0 Ræs</b>	<b>15</b>
6.1 Opbygning	15
6.2 Grafik	15
6.3 Udfordringer	16
<b>7.0 Videoafspiller</b>	<b>16</b>
7.1 Funktioner	16
7.2 Udfordringer	16
<b>8.0 Administrationsdel</b>	<b>16</b>
8.1 Funktioner	16
8.2 Udfordringer	16
<b>9.0 Mulige fremtidige tiltag</b>	<b>17</b>
9.1 Spil	17
9.2 WebGL	18
<b>10.0 Konklusion</b>	<b>19</b>

<b>11.0 Bilag</b>	<b>21</b>
11.1 Filstruktur	21
11.2 Database MySQL koder	22
11.3 Index.php	25
11.4 process.php	30
11.5 login.php	32
11.6 logout.php	32
11.7 mysqli_connect.php	32
11.8 race.js	33
11.9 getshipinfo.php	37
11.10 styles.css	38
11.11 race.js (WebGL)	40

## 1.0 Indledning

I denne øvelse skal der udarbejdes en hjemmeside, der illustrerer bådæret Volvo Ocean Race. Øvelsen er en gruppeøvelse.

## 2.0 Opgavebeskrivelse

1. **Ræs:** Der skal laves en hjemmeside, hvor der vises et ræs, med top 5 fra det seneste Volvo Ocean Race. Ræset skal illustrere hvem der vandt sidste Volvo Ocean Race.

Forslag til inspiration:



2. **Videoafspiller:** Vi skal vælge op til tre videoer fra [volvoceanrace.com](http://volvoceanrace.com), der skal afspilles i en youtube-afspiller, det skal være dynamisk i forhold til hvis vinduet eller siden størrelse ændres.
3. **Besøgstæller:** Der skal være en besøgstæller synligt i toppen af siden. Overvej og beskriv, om det er bedst med unikke besøgende eller samtlige.
4. **Administratorpanel:** Der skal være et administratorpanel til pointsystem i top 5.
5. **Footer:** Nederst på siden, skal der være en footer med generel info. Footeren skal være statisk og ligge i bunden af siden.

## 3.0 Løsningsforslag

### 3.1 Vores Løsning

Vi vil gøre det muligt for serer af Volvo Ocean Race at følge kapsejladsen på en hjemmeside, som vil vise hvem der ligger top 5, af de yachter som deltager i løbet, ved at vise deres points og navn og en beskrivelse af yacht og crew. Derudover, vil det være muligt at se videoer fra sejladsen på hjemmesiden.

Vi vil illustrere løbet af top 5. Da vi kun har data over placering og antal point, så kunne en simpel måde at illustrere løbet på, være at lade de enkelte bådes fart være konstant og afhængige af deres antal point. Dette er dog ikke en specielt spændende illustration, da det vil være tydeligt fra start af, hvem der vinder.

Vi vil udarbejde en grænseoverflade, som er brugervenlig og læsbar. Det skal være muligt for en administrator at tilgå de enkelt yachter i forhold til navn, points og beskrivelse, når en opdatering træder i kraft, skal top 5 på hjemmesiden opdatere sig så pointtavlen passer til de nye inputtede point. Det vil være muligt for administratoren at se antallet af besøgende på Volvo Ocean Race hjemmesiden ved hjælp af en besøgstæller. besøgstallet vil blive vist på administrator siden.

### 3.1 Anvendte teknologier

Til at løse denne øvelse, skal der anvendes en del forskellige teknologier. Der er stillet en række krav fra kunden.

#### 3.1.1 Xampp

Vi har anvendt Xampp til testmiljø, da vi arbejder hjemme grundet Corona-restriktionerne, har det ikke været muligt at anvende en af praktikcenterets servere til at køre vores løsning på. Vi har derfor måtte finde en måde at arbejde på hver især på egen maskine. Xampp har indbygget Apache server, som vi kan køre websiden fra og MariaDB, som vi kan anvende til database. Det er markant nemmere og mindre krævende for maskinen at opsætte, end at skulle installere en virtuel maskine, der både ville kræve mere at have kørende og fylde mere på harddisken. Xampp er relativt nemt at opsætte og kræver ikke det store af ens udstyr.

#### 3.1.2 Html, CSS og JavaScript

Vi vil anvende HTML, CSS og JavaScript. Det er en websides 'bread and butter'.

I denne sammenhæng vil det primært være ræset, der er opbygget i JavaScript og HTML Canvas. Ræset skal animeres og kunne startes, så der er det oplagt at anvende JavaScript.

Bootstrap gør dog, at vi ikke behøver at skrive så meget CSS som vi ellers ville.

### 3.1.3 PHP

Vi vil anvende PHP til at connecte hjemmesiden med databasen. På hjemmesiden vil vi bruge PHP på race-siden, til at læse fra databasen og ud fra det, lave en animeret illustration af top 5 placeringerne. Vi kommer nok til at bruge mest PHP til administrationsdelen, for at få en almindelig CRUD-funktionalitet, samt til at tjekke om login-oplysningerne er korrekte.

### 3.1.4 Bootstrap

Bootstrap er et framework til front-end, der tager sig af layout. Det gør det nemmere og hurtigere at bygge en webside op, så den får et professionelt look, uden at man skal opfinde en masse hyppigt anvendte teknikker fra bunden.

### 3.1.6 ReactJS

ReactJS er et javascript bibliotek til front-end. Det bruges til at kommunikere med DOM og gør det muligt at blande html og JavaScript på en mere dynamisk måde end normalt. React anvender en udvidet udgave af JavaScript JSX (JavaScript XML), men det er også muligt udelukkende at anvende almindelig JavaScript. Så selvom ReactJS er et front-end bibliotek er det altså ikke et bibliotek som tilføjer grafik eller layout. Det anvendes derfor normalt sammen med andre biblioteker som fx Bootstrap.

Vi har valgt ikke at anvende React, da vi mener at det unødigt gør løsningen mere kompleks, ikke mindst i forhold til opsætning. Hvis det blev implementeret kunne det dog muligvis gøre dele af løsningen mere simpel og nemmere at vedligeholde.

### 3.1.6 MariaDB

Indtil videre har vi anvendt Microsoft SQL Server, når vi har arbejdet med database. Den er tung at anvende, så det er en fordel at anvende en database der er nemmere at sætte op og som er let. MariaDB er indbygget i Xampp, så det er oplagt at anvende her. Det er en meget lille datamængde som vi arbejder med, så man kunne godt argumentere for, at det er meningsløst overhoved at bruge en database i denne øvelse, men dels lærer vi noget nyt i processen, og dels kan man sige at løsningen ville være mere skalerbar med en database. MariaDB bygger på MySQL, og det er derfor det sprog, der skrives i, det minder langt hen ad vejen om MSSQL.

## 3.2 Mockup

Disse mock up viser vores designforslag til Volvo Ocean Race, vi har valgt at gå med maritime farver for at fremhæve at dette Race foregår på vandet. Vi har valgt vil lave en statistisk footer, det vil sige, at footeren med kontaktoplysninger på Volvo Ocean Race, altid er fremvist, det har vi valgt, for at gøre det lettere for sejladsens følgere at kunne finde kontaktoplysninger, skulle de ønske kontakt.

Forside

The mockup shows a website with a blue gradient background. At the top, the text "Ocean Race" is displayed in a green, italicized font. Below this is a navigation bar with links for "Race", "Video", and "Admin". To the right of the navigation bar is a login section with a "Login" button, a "Username" input field, and a "Password" input field. The main content area features the text "Current race" in a blue, italicized font, followed by the sentence "Below you can see the status of the current race." Below this text is a large white rectangular box labeled "Race window". At the bottom of the page is a blue footer bar containing the word "Footer".

## Videoside

**Ocean Race**

Race Video Admin

Login

Username

Password

### Videos

Here you can watch some of our videos about the Ocean Race.

Team view

Footer

## Adminside

**Ocean Race**

Race Video Admin

Login

Username

Password

### Team management

Here you can manage teams

<u>Team name</u>	<u>Vessel</u>	<u>Points</u>	<u>Previous placement</u>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

*\* / Already taken*

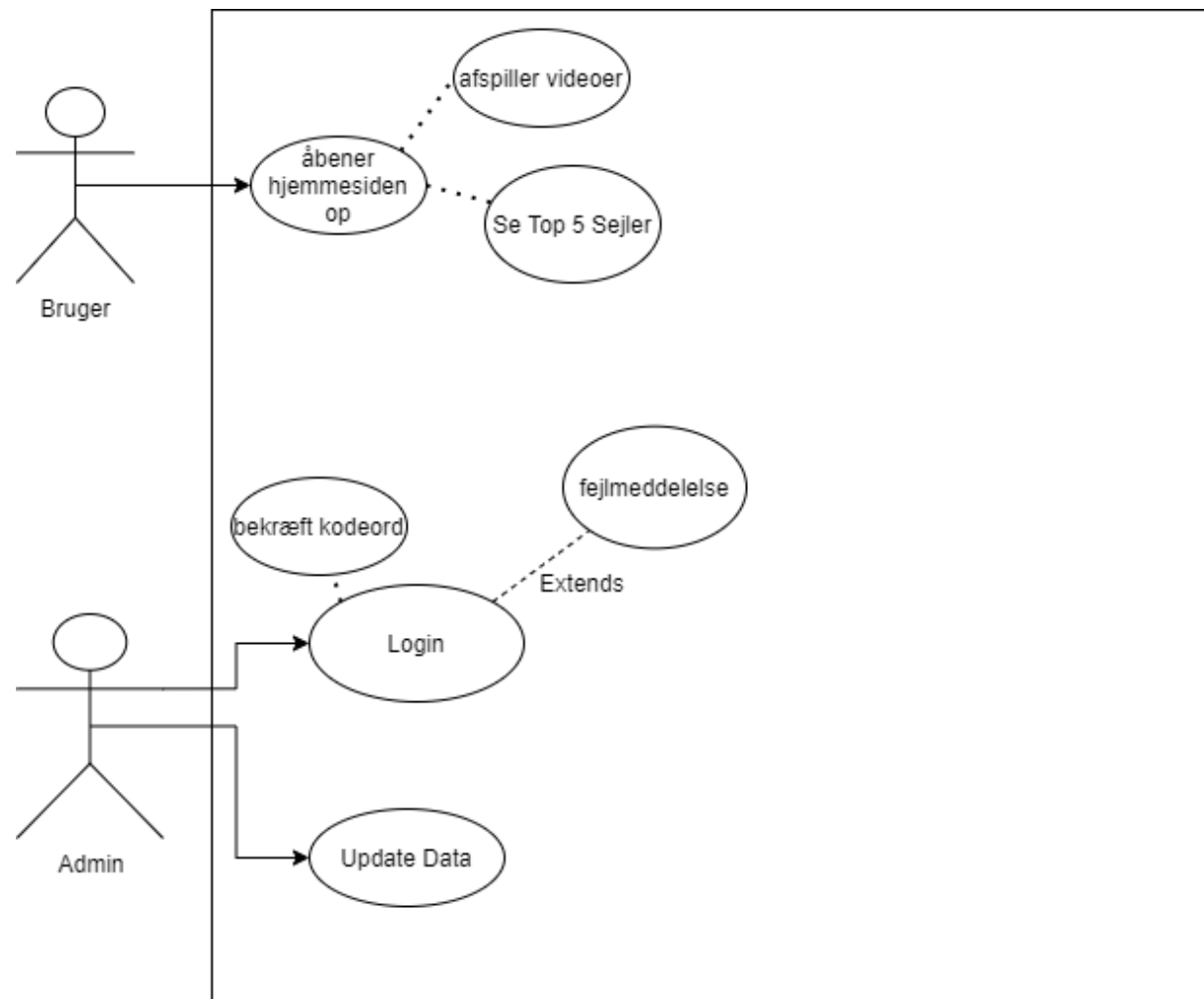
Team view

DeleteRegisterUpdate

Footer



### 3.3 UML



## 4.0 Navigationsbar

### 4.1 Funktioner

Vi har lavet en fane til ræset, en fane til video og en fane til administratorer. I navigationsbaren er der også en login-form samt logud. Når man ikke er logget ind, kan man ikke se fanen til administrator-siden. Der gives mulighed for at logge ind, ved at indtaste brugernavn og adgangskode og trykke på en "log in"-knap. Der bliver tjekket om login-oplysningerne er rigtige, altså om administratoren er i databasen, hvis ikke, gives der en fejlmeddelelse om at brugernavn og adgangskode er forkert. Hvis administratoren er i databasen, logges man ind. Når man er logget ind vises fanen til administrator-side og login-form ændres til at være en logud-knap. Når man trykker på "log ud"-knappen, vises en alert-box om at man er logget ud.

### 4.2 Udfordringer

Designet af navigationsbaren er ikke lykkedes helt som tænkt. Login-formen ligger helt i toppen af navigationsbaren frem for på linje med fanerne, det samme gælder log-ud-knappen.

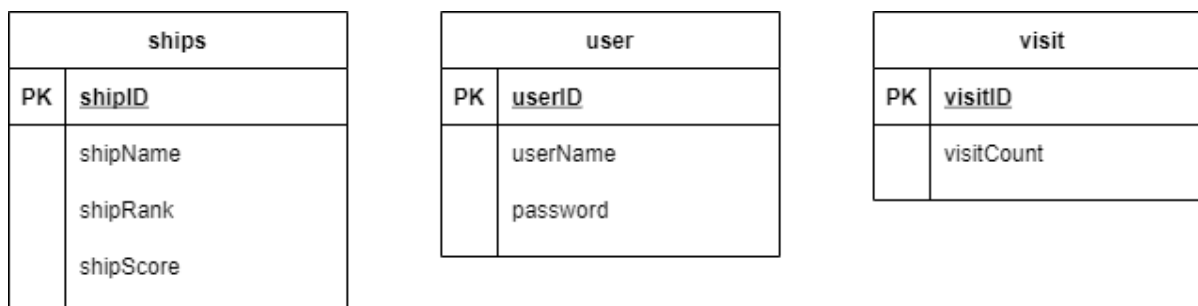
## 5.0 Database

### 5.1 Forord: MariaDB & PHPMyAdmin

I denne opgave skal der bruges MariaDB, som er en open source database software, og da vores opgave omhandler en hjemmeside, bruger vi PHPMyAdmin som er et administrations tool til MariaDB og MySQL.

### 5.2 Klassediagram

Før vi gik i gang med at skrive MySQLén til vores database, skulle vi have en oversigt, over hvad der skulle være i vores database. Så vi udarbejdede et database klassediagram.



Vi fandt ud af, denne opgave kunne klares ved brug af tre tables.

En til skibene som deltager i løbet, den skal indeholde et ID, skibets navn, skibets placering i løbet og hvor mange point skibet har samlet sammen i løbet.

Dernæst skulle der være en table med bruger, den skal bruges til når administratorer logger på siden for at ændre i løbet, den skal indeholde et ID, et brugernavn og en kode.

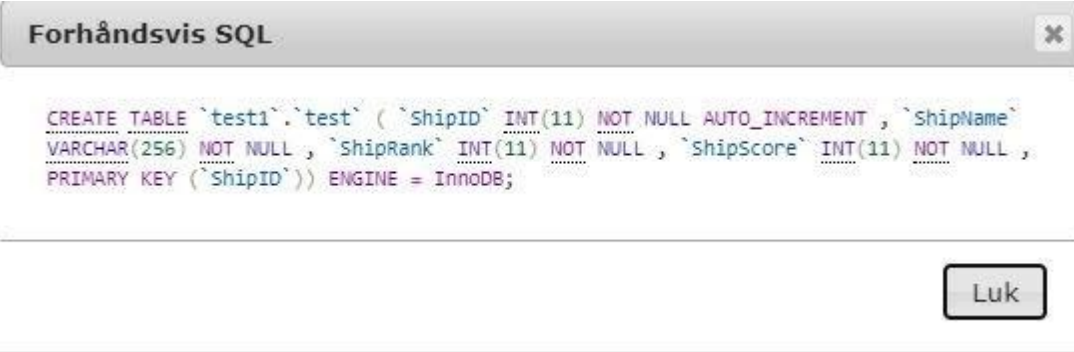
Til sidst skulle vi have en table med besøgsantal, her besluttede vi os for at den skulle have et ID og en besøgstæller, som ved hvert besøg på hjemmesiden, bliver lagt en til talet som ligger i besøgstælleren, ved brug af en update funktion.

## 5.3 Skabelsen af databasen

Efter at have udarbejdet et database klassediagram, var det nu blevet tid til at oprette databasen gennem PHPMyAdmin. Her følger en oprettelsesguide af vores database.

Efter man har åbnet PHPMyAdmin op, kommer man ind på en hjemmeside via sin browser. Her er en oversigt over de databaser, der allerede findes. I kolonnen til venstre, oppe over oversigten, står der "Ny", den klikker man på, og man bliver så ført over til oprettelsen af sin database. Her kan man vælge et navn til sin database og hvilket sprog man ønsker det skal kunne bruge. Vi kalder vores database for "ocean\_race" og giver den et dansk layout og afslutter ved at klikke på "opret".

Dernæst kommer man ind i databasen, og man skal nu oprette nogle tabeller inde i databasen. Her kommer vores klassediagram i brug, for nu ved vi, hvordan vores tabeller skal se ud, og hvilken form for data, der skal ind i dem. Så er det bare at give hver kolonne det rigtige navn og angive, hvilken værditype, den enkelte kolonne skal indeholde fx. varchar eller int, og om det er en primær nøgle. Når man har lavet en tabel, så kan man klikke "gem" eller "forhåndsvis SQL". Det smarte ved PHPMyAdmin, er at den skriver din MySQL for dig, så hvis vi bare klikker "gem", så skriver den koden, for at oprette tabellen og kolonnerne ud fra de data, vi har indsat. Hvis vi klikker på "forhåndsvis SQL", så viser PHPMyAdmin, det kode, som skal bruges for at oprette databasen og dens indhold.. Et eksempel på dette, kan være når jeg opretter denne testtabel.



```
CREATE TABLE `test1`.`test` ( `ShipID` INT(11) NOT NULL AUTO_INCREMENT, `ShipName` VARCHAR(256) NOT NULL, `ShipRank` INT(11) NOT NULL, `ShipScore` INT(11) NOT NULL, PRIMARY KEY (`ShipID`)) ENGINE = InnoDB;
```

Vi får oprettet alle 3 tabeller med de ønskede kolonner og værdier. Vores database er nu oprettet og klar til brug.

## 5.4 Indsættelse af Data.

Det er nu blevet tid til at fylde databasen med den data, som skal bruges, for at få den ønskede funktion af databasen. Vi går igen ud i PHPMyAdmin's venstre kolonne og klikker på det lille plus ved siden af vores databases navn, dette gør, at vi får vist alle tabellerne under denne database. Vi vælger så den tabel, vi ønsker at arbejde i lige nu. I dette eksempel tager vi ships tabellen.



Vi kommer nu ind i på ships siden, hvorfra vi kan arbejde med lige netop denne tabel. Vi kan ændre i den, oprette nye kolonner osv. Vi vælger "Indsæt" fra menubaren, dette fører os hen til en side, som ser sådan her ud:

Kolonne	Datatype	Funktion	Nulværdi	Værdi
shipID	int(11)			
shipName	varchar(256)			
shipRank	int(11)			
shipScore	int(11)			

☒ Ignorer

Kolonne	Datatype	Funktion	Nulværdi	Værdi
shipID	int(11)			
shipName	varchar(256)			
shipRank	int(11)			
shipScore	int(11)			

Udfør

Her er gjort klart til, at vi kan indtaste den ønskede data, ud fra hver af kolonne. Hvis vi nu skriver "TestSkib1" i kolonnen "shipName", læg mærke til, vi springer "shipID" over, dette er fordi, vi har sat den til selv at udfylde med en int, som går en op for hvert row der, bliver indsat i tabellen. Dernæst giver vi den plads 8 i kolonnen "shipRank" og siger, vi har 25 points i kolonnen "shipScore", og klikker så på "Udfør". Så kommer vi hen på en side, hvor vi kan se den SQL, som er brugt for at indsætte dataen, og vi får også at vide, at en row var tilføjet til vores tabel, så hvis vi klikker på "Vis" i menubaren, kommer vi hen til oversigten over "ships"-tabellen og der kan vi se alt den data, som vi har indsat.

←T→		shipID	shipName	shipRank	shipScore
<input type="checkbox"/>	 Ret  Kopi  Slet	1	DongFeng Race Team	1	73
<input type="checkbox"/>	 Ret  Kopi  Slet	2	MAPFRE	2	70
<input type="checkbox"/>	 Ret  Kopi  Slet	3	Team Brunel	3	69
<input type="checkbox"/>	 Ret  Kopi  Slet	4	team AkzoNobel	4	59
<input type="checkbox"/>	 Ret  Kopi  Slet	5	Vestas 11th Hour Racing	5	39
<input type="checkbox"/>	 Ret  Kopi  Slet	6	Turn the Tide on Plastic	6	32
<input type="checkbox"/>	 Ret  Kopi  Slet	7	Team Sun Hung Kai/Scallywag	7	31
<input type="checkbox"/>	 Ret  Kopi  Slet	8	TestSkib1	8	25

Og som man kan se, så er vores nye tilføjelse der også.

## 5.5 Eksport og Import

Efter at have fyldt data i vores database, er det nu blevet tid til, at eksportere databasen ud til os alle i gruppen, så hver enkelt af os, har adgang til den. Dette gøres ved at klikke på databasens navn ude i venstre kolonne på PHPMyAdmin-siden, når det er gjort, så vælger du "Eksporter" i menubaren. Det bringer dig hen til en side, hvor du har mulighed for, at vælge mellem en Hurtig eller en brugerdefineret eksport. Vi vælger den hurtige til vores gruppes database. Dernæst skal vi vælge hvilken slags format, vi ønsker at eksportere den i, her vælger vi SQL. Vi afslutter med at klikke "Udført". Vi får nu sendt en SQL-fil fra PHPMyAdmin til vores overførelsesmappe. Denne fil, sender vi så videre til alle deltagere i gruppen, som nu åbner deres PHPMyAdmin op og klikker på "ny" ude i venstre kolonne og giver deres nye database navnet "ocean\_race". Men i stedet for at oprette tabeller i den, klikker de på "import" i menubaren og vælger den SQL-fil, som er blevet delt med dem, og afslutter med at klikke "udfør". SQL-koderne fra filen, bliver nu læst af PHPMyAdmin, som udfører, hvad koderne skriver. Dette kan tage noget tid, afhængig af databasens størrelse. Når den er færdig, har hver enkelt deltager nu en tro kopi, af den originale ocean\_race database med tabeller, kolonner og data. Vi er nu klar til at bruge databasen på vores hjemmeside.

## 6.0 Ræs

### 6.1 Opbygning

Først indlæses alle billederne til både og skibe.

Der oprettes en klasse 'ship' med properties til data fra databasen.

Dataen hentes og skrives til skjulte felter på html'en via DOM. Dette er for at komme under om den udfordring, at fetch-kaldet i php er asynkront. Dette er nok ikke en løsning, man ville anvende i et professionelt miljø. Der oprettes et array med alle objekterne af både, hvor dataen hentes fra DOM.

### 6.2 Grafik



Grafikken er lavet ved at anvende en billedfil til bådene. Man kunne godt farve bådene via kode på forhånd, hvilket også ville give adgang til flere farver, men her er der i stedet lavet forskellige udgaver af det samme billede med forskellige farver.

Selve løbet visualiseres simpelt, ved at billederne placeres på et Canvas og deres position opdateres hver frame, når spillet er startet.

bådene flyttes en smule op og ned, for at simulere bølger i vandet. Det gøres ved at bruge sinusværdien af deres x-position og justere lidt. Dette blev efterfølgende skiftet ud, og timestamp anvendes i stedet, for at undgå at bølgerne stopper, når båden kommer i mål.

Ligeledes ændres hvor meget af billedets bund der skal tages med, på samme vis, for at simulere at bådene bevæger sig op og ned i vandet.

For yderligere at give idéen om vand, anvendes billeder af bølger. Billedet er farvet i forskellige versioner, som så placeres med den mørkeste øverst og bagerst. Bølgerne tegnes oven på bølgerne, så der gives et indtryk af perspektiv. Bølgerne flyttes op og ned med samme frekvens som bådene.

## 6.3 Udfordringer

Hvis man starter løbet forfra flere gange, bliver siden langsommere. Det er en alvorlig fejl, og det er ikke lykkedes at identificere hvad det skyldes. Det burde ikke være så tungt. Måske skyldes det at billederne har transparente områder.

## 7.0 Videoafspiller

### 7.1 Funktioner

Sidens funktion er at afspille en embedded youtube-video om Volvo Ocean Race.

### 7.2 Udfordringer

Der har været problemer med at få baggrunden til at udfylde hele siden, uden at generere scroll nedafd, når der ikke er behov for det. Vi har lavet en manuel lappeløsning, som kun virker på nogle pc'er.

Videoen fortsætter med at afspille i baggrunden, hvis man skifter over til administration eller ræset.

## 8.0 Administrationsdel

### 8.1 Funktioner

Man kan kun komme til administratorsiden, når man er logget ind. Som administrator kan man oprette, redigere og slette teams inklusiv deres point og rank.

Man kan ikke oprette det samme team-navn 2 gange. Når man vælger "edit", altså rediger, udfylder den inputfelterne med teamets info og "save"-knappen bliver til "update".

### 8.2 Udfordringer

Vi har stadig den udfordring at når man udfører noget på administratorsiden, så loader den tilbage til "race"-siden, som er vores forside. Vi har muligvis fundet løsningen på dette, men grundet tidspres har vi ikke nået at ændre det.



Den udskriver ikke meddelelser til brugeren. Fx hvis man prøver at oprette et teamnavn, som allerede eksisterer, så opretter den det ikke, men giver ikke en meddelelse om hvorfor. Fejlmeddelelsen kunne være noget i stil med "Dette team eksisterer allerede, hvis du ønsker at ændre noget, tryk på edit ud for team-navnet". Den giver heller ikke meddelelser, når et team er blevet oprettet, slettet eller redigeret. Vi har ikke nået at finde en løsning på dette grundet tidspres.

Inputfelterne ligger ikke så pænt og save/update knappen ligger ikke på linje med hinanden. Vi har ikke fået dette til at fungere grundet tidspres.

## 9.0 Mulige fremtidige tiltag

Vi har under arbejdet med denne opgave, kommet op med et tiltag der kunne fremme hjemmesiden i fremtiden. Her er hvad vi har snakket om.

### 9.1 Spil

Ide

Vi forestiller os at man, som besøger af hjemmesiden, kan oprette en bruger. Denne bruger tildeles et antal points, disse point bruges til at spille på de enkelte skibe i kapsejladsen fx. bruger: xXB0Xx har 100 points. han ser så at båd: Anne2 ligger tæt på førstepladsen, xXB0Xx klikker så på Anne2 og siger at han gerne vil ligge 50 points på at de ligger nr 1 ved næste opdatering af løbet. Og så venter man. Skulle det så ske, at xXB0Xx har ret, vinder han flere points, ellers mister han dem han har satset. Efter at have vundet eller tabt vil han være i stand til at tjekke, hvor han ligger med sine points på Pointtavlen, der viser hans og andre brugers navn og hvor mange points de har.

Fordel:

Det ville gøre siden mere spændende og interaktiv for brugeren og ville på sigt gøre at nye brugere ville komme til for at følge løbet og smide points på deres yndlingsskib.

Ud fra det billede som er givet som eksempel, ligner ræset et hestevædeløbsspil.



Eksempel på hestevæddeløbsspil i Tivoli.

<https://www.youtube.com/watch?v=Wzs-mTmKGVQ>

Et simpelt spil - Man spiller på en hest, og hestene bevæger sig i forskelligt tempo mod målstregen på skinner. Hesten der når først frem har vundet.

Det kunne være spændende, hvis vi havde live data over Volvo Ocean Race, eller fx havde deltagerens GPS-koordinator gennem sejladsen, således at vi kunne lave en grov illustration af løbets forløb.

Hvis vi lavede et spil som hestevæddeløbsspillet i Tivoli, hvor man antageligvis ikke kender vinderen på forhånd, så kunne vi lade den enkelte hests hastighed ændre sig tilfældigt gennem løbet, for at skabe noget spænding. Dette system kunne udvikles på forskellig vis, fx kunne man ønske, at der især var spænding omkring slutningen af løbet, ligesom man kunne lave forskellige systemer som styrede tilfældigheden. I Tivolis version af spillet fungerer det således, at en spiller får et antal bolde, som han skal skyde ned i nogle huller. Hullerne giver forskellige antal point. Og hestens hastighed får et 'boost' afhængigt af antallet af point.

Hvis vi så forestiller os, at vi kombinerer de to systemer, således at vi på forhånd har bestemt hvem, der vinder, og de andre placeringer, ud fra vores data, men samtidig ønsker at simulere et løb med drama og overraskelser undervejs, så burde dette kunne gøres ved at den enkelte deltageres point stadig afgør farten, men i stedet for at være konstant, så deles ud over forskellige steder på banen, som en form for hastigheds 'boost'.

## 9.2 WebGL

JavaScript-animationen kører ikke specielt flydende, og især, hvis man starter den forfra gentagne gange, får den browseren til brokke sig. Det er ikke optimalt. Derfor forsøgte vi at lade animationen køre i WebGL i stedet. Det betyder at grafikortet anvendes til at render grafikken i stedet for på CPU som normalt med Canvas. Den rimeligt simple animation burde ikke være specielt tung, men anvendelsen af WebGL ville også åbne op for at kunne gøre mere avancerede ting.

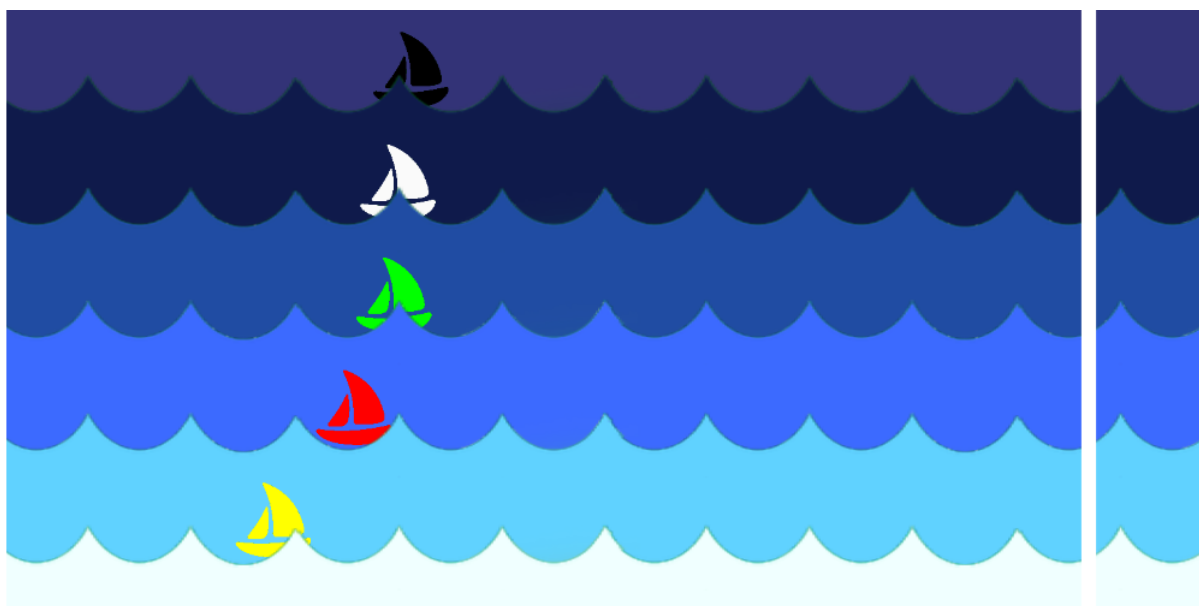
Vi anvendte biblioteket P5 for at gøre implementeringen af WebGL lidt nemmere, og for at lave en light-løsning. Vi skulle alligevel stadig arbejde med et fladt rum og ikke benytte 3D. Implementeringen af WebGL gav et markant bedre resultat. Udover GPU'en kan det delvist skyldes måden P5 er bygget op omkring loops, men der manglede stadig at blive overført enkelte elementer som tekst.

Det fungerer således, at alle billeder anvendes som textures på et 'plane' altså et fladt plan. Det vil sige, at de eksisterer i et 3dimensionelt rum, men at vi ikke gør brug af den tredje dimension. Resultatet er således identisk med løsningen uden WebGL.

Canvas oprettes i Javascriptet

Løsningen er vedlagt i bilag, og kræver, at der er referencer til p5 biblioteket i index-filen, enten som filer eller på nettet således:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/p5.min.js"></script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/addons/p5.dom.min.js"></script>  
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/addons/p5.sound.min.js"></script>
```



Bådræset nu i tre dimensioner med WebGL:

## 10.0 Konklusion

Det har ikke været den nemmeste opgave, og der er anvendt mange nye forskellige teknologier, som har givet udfordringer. Dette har også gjort, at vi ikke har formået at lave siden helt efter vores mockup.

Det er vigtigt at udarbejde en fælles forståelse for opgaven inden, man går i gang, vi havde en del misforståelser, som kunne være undgået. Hertil er det oplagt, at det klares i forbindelse med udarbejdelsen af løsningsforslaget.

Under processen er vi blevet klar over, at der er brug for yderligere planlægning, når man arbejder i gruppe på den måde. Det at vi har arbejdet online, gør også at der i endnu højere grad er brug for planlægning, da der ikke er den almindelige løbende sparing, som der ville være hvis man arbejdede fysisk det samme sted.

Det viser sig at nogle fejl, først opstår og / eller bliver tydelige når de enkelte dele sættes sammen. Så det er nok en god ide fremover, at forsøge at have en form for prototype færdig tidligere i forløbet, for netop at fange disse.

Efter snak med andre elever, lader det til at Wamp skulle være en bedre løsning en Xampp; det kunne være godt at undersøge en anden gang.

## 11.0 Bilag

### 11.1 Filstruktur

Filstrukturen for websitet ser således ud:

```
mysqli_connect.php  
/htdocs/img/   *billedfiler*  
/htdocs/index.php  
/htdocs/process.php  
/htdocs/login.php  
/htdocs/logout.php  
/htdocs/getshipinfo.php  
/htdocs/styles.css  
/htdocs/jace.js
```

\* htdocs såfremt projektet køres i Xampp

## 11.2 Database MySQL koder

```
--  
--  
-- Database: `ocean_race`  
-- Denne database er tilgængelig på github som en eksporteret fil fra PHPMysqlAdmin  
--  
--  
-- Oversigt  
-- 1. Oprettelse af ships table  
-- 2. indsættelse af data i ships table  
-- 3. Oprettelse af user table  
-- 4. indsættelse af data i user table  
-- 5. Oprettelse af visit table  
-- 6. indsættelse af data i visit table  
-- 7. Oprettelse af primary keys  
-- 8. Oprettelse af AUTO_INCREMENT  
--  
--
```

```
-- -----  
-- 1. Oprettelse af ships table  
-- -----
```

```
CREATE TABLE `ships` (  
  `shipID` int(11) NOT NULL,  
  `shipName` varchar(256) COLLATE latin1_danish_ci NOT NULL,  
  `shipRank` int(11) NOT NULL,  
  `shipScore` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_danish_ci;
```

```
-- -----  
-- 2. indsættelse af data i ships table  
-- -----
```

```
INSERT INTO `ships` (`shipID`, `shipName`, `shipRank`, `shipScore`) VALUES  
(1, 'DongFeng Race Team', 1, 73),  
(2, 'MAPFRE', 2, 70),  
(3, 'Team Brunel', 3, 69),  
(4, 'team AkzoNobel', 4, 59),  
(5, 'Vestas 11th Hour Racing', 5, 39),  
(6, 'Turn the Tide on Plastic', 6, 32),  
(7, 'Team Sun Hung Kai/Scallywag', 7, 31);
```

```
-- -----
```

-----  
-- 3. Oprettelse af user table  
-----

```
CREATE TABLE `user` (  
  `userID` int(11) NOT NULL,  
  `userName` varchar(25) COLLATE latin1_danish_ci NOT NULL,  
  `password` varchar(25) COLLATE latin1_danish_ci NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_danish_ci;
```

-----  
-- 4. indsættelse af data i user table  
-----

```
INSERT INTO `user` (`userID`, `userName`, `password`) VALUES  
(1, 'Admin', 'Passw0rd');
```

-----  
-- 5. Oprettelse af visit table  
-----

```
CREATE TABLE `visit` (  
  `visitID` int(11) NOT NULL,  
  `visitCount` int(11) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_danish_ci;
```

-----  
-- 6. indsættelse af data i user table  
-----

```
INSERT INTO `visit` (`visitID`, `visitCount`) VALUES  
(1, 0);
```

-----

```
-----  
-- 7. Oprettelse af primary keys  
-----
```

```
--  
-- Indeks for tabel `ships`  
--  
ALTER TABLE `ships`  
  ADD PRIMARY KEY (`shipID`);
```

```
--  
-- Indeks for tabel `user`  
--  
ALTER TABLE `user`  
  ADD PRIMARY KEY (`userID`);
```

```
--  
-- Indeks for tabel `visit`  
--  
ALTER TABLE `visit`  
  ADD PRIMARY KEY (`visitID`);
```

```
-----  
-- 1. Oprettelse af AUTO_INCREMENT  
-----
```

```
--  
-- Tilføj AUTO_INCREMENT i tabel `ships`  
--  
ALTER TABLE `ships`  
  MODIFY `shipID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=8;
```

```
--  
-- Tilføj AUTO_INCREMENT i tabel `user`  
--  
ALTER TABLE `user`  
  MODIFY `userID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;
```

```
--  
-- Tilføj AUTO_INCREMENT i tabel `visit`  
--  
ALTER TABLE `visit`  
  MODIFY `visitID` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;  
COMMIT;
```



## 11.3 Index.php

```
<?php session_start();?>
<!doctype html>
<html lang="en">
    <head>
        <title>ocean volvo race</title>
        <!--down thereV is standard bshtml (b4-$)-->
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

        <!-- Bootstrap CSS -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT
2MZw1T" crossorigin="anonymous">
        <!--up there^ is standard bshtml (b4-$)-->

        <link rel="stylesheet" href="styles.css">

    <header><div class="header"><h1>Volvo Ocean Race</h1></div> <!--header of the
website--></header>

    </head>
    <body>
        <?php require_once 'process.php';?> <!--index.php now knows the file
process.php-->
        <?php if (isset($_SESSION['message'])): ?>
        <div class="alert alert-<?=$_SESSION['msg_type']?>">
            <?php
            echo $_SESSION['message'];
            unset($_SESSION['message']);
            ?>
        </div>
        <?php endif; ?>
        <div id="navbar"> <!--navigation-bar-->
        <ul class="nav">
            <li id="Racetab" class="active"><a data-toggle="pill"
href="#Race">Race</a></li>
            <li><a data-toggle="pill" href="#Video">Video</a></li>
            <li id="Admintab"><a data-toggle="pill" href="#Admin">Admin</a></li>
            <?php if(isset($_SESSION['user'])): ?> <!--If you are logged in-->
            <form class="justify-content" action="logout.php" method="POST">
<!--executes code from logout.php-->
            <button onclick="alert('you are logged out')" type="submit" class="btn
btn-success">Log out</button> <!--Log-out button-->
```

```

        </form>
        <?php else: ?> <!--If you are logged out-->
        <style type="text/css">#Admintab{display:none;}</style> <!--The admin tab is
only shown, if you are logged in.-->
        <form class="justify-content" action="login.php" method="POST">
<!--executes code from login.php-->
        <div class="form-group">
            <input type="text" placeholder="Username" class="form-control"
name="username"> <!--Input field for username-->
        </div>
        <div class="form-group">
            <input type="password" placeholder="Password" class="form-control"
name="password"> <!--Input field for password. type=password hides the text and only
writes dots, when you write a char.-->
        </div>
        <button type="submit" class="btn btn-success">Log in</button> <!--Log-in
button-->
        </form>
        <?php endif; ?>
    </ul>
</div>
<div class="tab-content"> <!--content in the tabs from the navigationbar-->
<div id="Race" class="tab-pane active"> <!--The race-tab. "active" means the
website opens the admin-tab by default-->
    <!--Anders' kode her-->
    <h3>Race</h3>
    <div class="overflow-hidden">

        <div class="row pt-4">

            <div class="col-sm-2"></div>

            <div class="col-sm-8">

                <canvas id="race_canvas" width="800"
height="400"></canvas>

                <button type="button" size="30" onclick="start()">Start
Race</button>

                <p hidden id="shipName0"> </p>
                <p hidden id="shipName1"></p>
                <p hidden id="shipName2"></p>
                <p hidden id="shipName3"></p>
                <p hidden id="shipName4"></p>

                <p hidden id="shipRang0"></p>

```

```

        <p hidden id="shipRang1"></p>
        <p hidden id="shipRang2"></p>
        <p hidden id="shipRang3"></p>
        <p hidden id="shipRang4"></p>

        <p hidden id="shipScore0"></p>
        <p hidden id="shipScore1"></p>
        <p hidden id="shipScore2"></p>
        <p hidden id="shipScore3"></p>
        <p hidden id="shipScore4"></p>

    </div> <!--end col -->

    <div class="col-sm-2"></div>

</div> <!--end row -->

<!-- ræs JavaScript -->
<script src="race.js"></script>

    <p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem
    accusantium doloremque laudantium, totam rem aperiam.</p>
    </div>
</div>
    <div id="Video" class="tab-pane fade"> <!--Video-tab. Problem: video is behind footer
    and header-->
        <!--Simons kode-->
        <div class="overflow-hidden">
            <div class="row" id="videoen">
                <div class="col-sm-2"></div> <!--Placing the video in the middel of the
    screen-->
                    <div class="col-sm-8 p-5">
                        <div class="embed-responsive embed-responsive-21by9">
                            <iframe class="embed-responsive-item"
    src="https://www.youtube.com/embed/videoseries?list=PL8z1Z17Ds-C4LtWQT55QRbCaqa-
    dQyOoT&autoplay=0&mute=1loop=1controls=0"></iframe>
                                </div>
                            </div>
                        <div class="col-sm-2"></div>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <div id="Admin" class="tab-pane fade"> <!--the admin-tab.-->
        <h2>Team management</h2>
        <form class="justify-content" action="process.php" method="POST">

```

<!--The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute).

The form-data can be sent as URL variables (with method="GET") or as HTTP post transaction (with method="post").

-->

```
<?php $result = $mysqli->query("SELECT * FROM ships") or  
die($mysqli->error); /*selects all data from the table named "ships"*/ ?>
```

<input type="hidden" name="id" value="<?php echo \$id; ?>"> <!--This is for the functionality of the updatebutton. This is hidden from the user-->

<div class="form-group">

<label for="name">Team name</label> <!--label is a text that typically belongs to an input"box"-->

<input type="text" name="name" class="form-control" value="<?php echo \$name; ?>" placeholder="Enter team name"> <!--input is a little box, where the user can input information. placeholder is text in inputfield, you dont have to delete. It dissapears, when you write in the field-->

</div>

<div class="form-group">

<label for="points">Points</label>

<input type="text" name="points" value="<?php echo \$points; ?>" class="form-control" placeholder="Enter points">

</div>

<div class="form-group">

<label for="rank">Rank</label>

<input type="text" name="rank" value="<?php echo \$rank; ?>" class="form-control" placeholder="Enter rank">

</div>

<div class="form-group">

<!--If edit is pressed the update varibele is true then let the save button be an update button-->

<?php if (\$update == true): ?>

<button type="submit" class="btn btn-primary" name="update">Update</button>

<?php else: ?>

<button type="submit" class="btn btn-primary" name="save">Save</button> <!--type= submit means the button submits form-data. btn btn-primary makes the button blue-->

<?php endif; ?>

</div>

</form>

<!--table to view data - has to be in a div, to make it a scrollbar -->

<div id="tablecontent">

<table>

```

        <tr>
            <th>shipID</th>
            <th>shipName</th>
            <th>shipRank</th>
            <th>shipScore</th>
            <th>Action</th>
        </tr>
        <?php while($row = $result->fetch_assoc()): ?>
        <tr>
            <td><?php echo $row['shipID']; ?></td> <!--standard data cell-->
            <td><?php echo $row['shipName']; ?></td>
            <td><?php echo $row['shipRank']; ?></td>
            <td><?php echo $row['shipScore']; ?></td>
            <td>
                <a href="index.php?edit=<?php echo $row['shipID']; ?>" class="btn
btn-info">Edit</a> <!--edit-button-->

                <a href="index.php?delete=<?php echo $row['shipID']; ?>" class="btn
btn-danger">Delete</a><!--delete-button-->
            </td>
        </tr>
        <?php endwhile; ?>
    </table>
</div>
</div><!--admin-tab is done-->
</div><!--tabcontent done-->

<!--down thereV is standard bshtml (b4-$)-->
<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi
6jizo" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJK6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86
dIHNDz0W1" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIlly6OrQ6VrjIEaFf/nJGzlxFDsf4x0xIM+B07j
RM" crossorigin="anonymous"></script>
    <!--up there^ is standard bshtml (b4-$)-->
</body>
<footer><div class="footer"><p>Copyright&copy;<?php echo date("Y");
?></p></div></footer><!--footer of the site. copyright-tegn efterfulgt af php, som udskriver
det nuværende år.-->
</html>

```

## 11.4 process.php

```
<?php
```

```
$mysqli = new mysqli('localhost','root','','ocean_race') or die(mysqli_error($mysqli));  
/*Connect mysql database.('hostname', 'username', 'password', 'database-name').  
   "or die(mysqli_error($mysqli))" shows error, if it can't connectes to the database.  
*/
```

```
/*//checking connection:  
if ($mysqli->error)  
{  
    echo "not Connected";  
}  
echo "Connected successfully";*/
```

```
//default values  
$id = 0;  
$name = ""; //there isn't anything in the input box  
$points = ""; //there isn't anything in the input box  
$rank = ""; //there isn't anything in the input box  
$username = "";  
$password = "";  
$update = false; //update button isn't shown
```

```
//if the button named save has been pressed:
```

```
if (isset($_POST['save']))
```

```
{  
    //store everything inside variables  
    $id = $_POST['id'];  
    $name = $_POST['name'];  
    $points = $_POST['points'];  
    $rank = $_POST['rank'];
```

```
    $checker = "SELECT shipName FROM ships WHERE shipName='$name';"  
    $check = mysqli_query($mysqli,$checker);  
    if (mysqli_num_rows($check)>0) //If you try to add a team, that already exists  
    {  
        $_SESSION['message'] = "Team name already exists!";  
        $_SESSION['msg_type'] = "danger";  
    }  
    else
```

```
    {  
        $mysqli->query("INSERT INTO ships (shipName, shipScore, shipRank)  
VALUES('$name', '$points', '$rank')") or die($mysqli->error);
```

```

        $_SESSION['message'] = "Record has been saved!";
        $_SESSION['msg_type'] = "success";
    }
    header("location: index.php"); //return to page when processed
}

//edit
if (isset($_GET['edit']))
{
    $id = $_GET['edit'];
    $update = true;
    $result = $mysqli->query("SELECT * FROM ships WHERE shipID=$id") or
die($mysqli->error);

    $row = $result->fetch_array(); //this returns the data from the record(database)
    $name = $row['shipName'];
    $points = $row['shipScore'];
    $rank = $row['shipRank'];
}

//update
if (isset($_POST['update']))
{
    //store everything inside variables
    $id = $_POST['id'];
    $name = $_POST['name'];
    $points = $_POST['points'];
    $rank = $_POST['rank'];
    $mysqli->query("UPDATE ships SET shipName='$name', shipScore='$points',
shipRank='$rank' WHERE shipID=$id") or die($mysqli->error);

    $_SESSION['message'] = "Record has been updated!";
    $_SESSION['msg_type'] = "warning";
    header("location: index.php"); //return to index page
}

//delete
if (isset($_GET['delete']))
{
    $id = $_GET['delete'];
    $mysqli->query("DELETE FROM ships WHERE shipID=$id") or die($mysqli->error);

    $_SESSION['message'] = "Record has been deleted!";
    $_SESSION['msg_type'] = "danger";
    header("location: index.php"); //return to page when processed
}

```

?>

## 11.5 login.php

```
<?php
    session_start();
    include('process.php');
    $username = $_POST['username'];
    $password = $_POST['password'];

    $result = $mysqli->query("SELECT * FROM user WHERE UserName='$username'
AND Password ='$password'") or die($mysqli->error);

    $check = mysqli_num_rows($result);

    if($check > 0)
    {
        $_SESSION['user'] = $username;
        header('location: index.php');
    }
    else
    {
        echo "Username or password is wrong";
    }
?>
```

## 11.6 logout.php

```
<?php
    session_start();
    include('process.php');
    session_destroy();
    header('location: index.php');
?>
```

## 11.7 mysqli\_connect.php

```
<?php
DEFINE ('DB_USER', 'root');
DEFINE ('DB_PASSWORD','');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'ocean_race');

$dbc = @mysqli_connect(DB_HOST,DB_USER, DB_PASSWORD,DB_NAME)
OR die('could not connect to MySQL' .
    mysqli_connect_error());
?>
```



## 11.8 race.js

// indlæs billeder:

```
black = new Image;
white = new Image;
green = new Image;
red = new Image;
yellow = new Image;
white.src = './img/boat_white.png';
green.src = './img/boat_green.png';
black.src = './img/boat_black.png';
red.src = './img/boat_red.png';
yellow.src = './img/boat_yellow.png';
```

```
const boats = [black, white, green, red, yellow];
```

```
wave1 = new Image;
wave2 = new Image;
wave3 = new Image;
wave4 = new Image;
wave5 = new Image;
wave1.src = './img/waves_1.png';
wave2.src = './img/waves_2.png';
wave3.src = './img/waves_3.png';
wave4.src = './img/waves_4.png';
wave5.src = './img/waves_5.png';
```

```
const waves = [wave1, wave2, wave3, wave4, wave5];
```

```
class Team{
  constructor(name, ranking, score, color){
    this.name = name;          // skal hentes fra database
    this.rank = ranking;       // skal hentes fra database, men kunne i princippet beregnes på
    baggrund af score
    this.score = score;        // skal hentes fra database

    this.boost = score;        // resettes ved ny kørsel (kopi af score)
    this.position_x = 20;      // resettes ved ny kørsel
    this.speed = 20;           // resettes ved ny kørsel
  }
}

class Game{
  constructor(){
    this.canvas = document.getElementById("race_canvas");
    this.context = this.canvas.getContext("2d");
    this.context.fillStyle = "#333377";
```

```

        this.context.fillRect(0, 0, this.canvas.width, this.canvas.height);
        this.running = true;
    }
    update(deltaTime){

        for (let i = 0; i < 5; i++){
            var addBoost = Math.random() * teams_2018[i].boost;
            teams_2018[i].boost -= addBoost;
            teams_2018[i].speed += addBoost;

            this.running = false;

            // så længe et skib stadig flyttes kører løbet!
            if (teams_2018[i].position_x <= this.canvas.width - 60){
                teams_2018[i].position_x += teams_2018[i].speed/100;
                this.running = true;
            }
        }
    }
    draw(){
        this.context.fillStyle = "#333377";
        this.context.fillRect(0, 0, this.canvas.width, this.canvas.height);

        for (let i = 0; i < teams_2018.length; i++){

            this.context.fillStyle = 'white';
            this.context.font = '20px sans-serif';
            this.context.fillText(teams_2018[i].name,5, 75*i + 20);

            // bølger
            let yOffset = Math.sin(teams_2018[i].position_x/8);

            let yOffsetBoat = Math.cos(teams_2018[i].position_x/8);

            // bare for at huske parametrene
            // context.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);

            // tegn både
            this.context.drawImage(boats[i],
                0,
                0,
                boats[i].width,
                boats[i].height + yOffsetBoat*50,
                teams_2018[i].position_x,
                75*i + 20 + yOffset,
                50,
                50 + yOffsetBoat*2// 50
            );
        }
    }
}

```

```

    );
    // tegn bølger
    this.context.drawImage(waves[i],
        0,
        90,
        waves[i].width,
        waves[i].height,
        0,
        75*i + 20 + yOffset,
        waves[i].width*1.5,
        waves[i].height*1
    );

    // tegn målstreg
    this.context.strokeStyle = "white";
    this.context.rect(this.canvas.width - 60, 0, 1, this.canvas.height);
    this.context.stroke();
}
}
}

// problem dataen hentes asynkront! så det er ikke tilgængeligt udenfor funktionen
// en løsning: skriver til DOM
// TODO: lige nu sorterer den ikke skibene, hvis der ligger flere end 5 i tabellen skal vi
sortere

function getData(){
    // hent data fra databasen. AJAX-metode?
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            myObj = JSON.parse(this.responseText);

            // skriv data til DOM for at gemme til senere indlæsning:
            document.getElementById("shipName0").innerHTML = myObj[0][0];
            document.getElementById("shipName1").innerHTML = myObj[1][0];
            document.getElementById("shipName2").innerHTML = myObj[2][0];
            document.getElementById("shipName3").innerHTML = myObj[3][0];
            document.getElementById("shipName4").innerHTML = myObj[4][0];

            document.getElementById("shipRang0").innerHTML = myObj[0][1];
            document.getElementById("shipRang1").innerHTML = myObj[1][1];
            document.getElementById("shipRang2").innerHTML = myObj[2][1];
            document.getElementById("shipRang3").innerHTML = myObj[3][1];
            document.getElementById("shipRang4").innerHTML = myObj[4][1];

            document.getElementById("shipScore0").innerHTML = myObj[0][2];

```

```
        document.getElementById("shipScore1").innerHTML = myObj[1][2];
        document.getElementById("shipScore2").innerHTML = myObj[2][2];
        document.getElementById("shipScore3").innerHTML = myObj[3][2];
        document.getElementById("shipScore4").innerHTML = myObj[4][2];

        load();
    }
};
xmlhttp.open("GET", "getshipinfo.php", true);
xmlhttp.send();
}

getData();

let teams_2018 = [];

game = new Game();

function load(){
// hent data fra DOM
teams_2018.push(new Team(document.getElementById("shipName0").innerHTML,
document.getElementById("shipRang0").innerHTML,
document.getElementById("shipScore0").innerHTML, "red"));
teams_2018.push(new Team(document.getElementById("shipName1").innerHTML,
document.getElementById("shipRang1").innerHTML,
document.getElementById("shipScore1").innerHTML, "green"));
teams_2018.push(new Team(document.getElementById("shipName1").innerHTML,
document.getElementById("shipRang2").innerHTML,
document.getElementById("shipScore2").innerHTML, "yellow"));
teams_2018.push(new Team(document.getElementById("shipName3").innerHTML,
document.getElementById("shipRang3").innerHTML,
document.getElementById("shipScore3").innerHTML, "pink"));
teams_2018.push(new Team(document.getElementById("shipName4").innerHTML,
document.getElementById("shipRang4").innerHTML,
document.getElementById("shipScore4").innerHTML, "black"));

game.draw();

}
function start(){
// reset og start gameloop:

restart();

gameLoop();
}
```

```
function restart(){
    // bruges også til start
    for (i = 0; i < 5;i++){
        teams_2018[i].speed = 20;
        teams_2018[i].position_x = 20;
        teams_2018[i].boost = teams_2018[i].score;
    }
}

let lastTime = 0;

function gameLoop(timestamp){
    let deltaTime = timestamp - lastTime;
    lastTime = timestamp;

    game.update(deltaTime);
    game.draw();

    // stop loopet når bådene er i mål
    if (game.running == true){
        requestAnimationFrame(gameLoop);
    } else {
        // kan gøres smartere...
        for (i = 0; i < teams_2018.length;i++){
            if (teams_2018[i].rank == 1){
                // bør gøres mere dynamisk
                game.context.fillStyle = 'white';
                game.context.font = '30px sans-serif';
                game.context.fillText(teams_2018[i].name + " har
vundet!",game.canvas.width/4, game.canvas.height/2);
            }
        }
    }
}
```

## 11.9 getshipinfo.php

```
<?php

require_once('..mysqli_connect.php');

$query = "SELECT shipName, shipRank, shipScore FROM ships";

$response = @mysqli_query($dbc, $query);

$myObj;
```

```
if($response){
$rows = array();
    while($row = mysqli_fetch_array($response)){
        $rows[] = $row;
    }

    $myJSON = Json_encode($rows);

    echo $myJSON;
} else {
    echo "Couldn't issue database query";
    echo mysqli_error($dbc);
}
mysqli_close($dbc);

?>
```

## 11.10 styles.css

```
html, body
{
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
    background: linear-gradient(rgb(255, 255, 255), rgb(50, 190, 235));
    display: flex;
    flex-direction: column;
}
header
{
    height: 10%;
    width: 100%;
    background: linear-gradient(rgb(10, 150, 195), rgb(210, 235, 240));
    display: flex;
}
#videoen
{
}
.header h1
{
    color: rgb(0, 255, 230);
    font-size: 30px;
    font-weight: bold;
    font-style: italic;
}
```

```
#navbar
{
    background-color: white;
    text-align: center;
}
ul
{
    border: solid black 1px;
}
li a /*items/links in the navigationbar */
{
    display: block;
    color: rgb(10, 150, 195);
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}
li a:hover /*when you hover the mouse over the items in the navigation bar */
{
    background-color: rgb(210, 235, 240);
}
li a.active /*when link is open */
{
    font-weight: bold;
    text-decoration: underline blue;
}
#username, #password
{
    width: 10%;
}
#logout, #login /*log out- and log in-button */
{
    align-self: center;
    /*float: right;*/
}
#Admin h2
{
    font-family: Rockwell/*Gabriola*/;
    font-weight:bold;
    text-align: center;
    font-style: italic;
    color: rgb(45, 165, 245);
    padding: 20px;
}
.form-group
{
    float: left;
```

```
padding-left: 6%;
padding-right: 6%;
text-align: center;
}
footer
{
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    height: 7%;
    background: linear-gradient(rgb(210, 235, 240), rgb(10, 150, 195));
    color: black;
    display: flex;
}
/*table*/
table
{
    width: 100%;
}
#tablecontent
{
    width: 100%;
    height: 250px;
    line-height: 3em;
    overflow: auto; /*makes t a scrollbar, when there is more content in the table than
there's room for*/
    border: solid black 2px;
}
th
{
    background-color: #588c7e;
    color: white;
}
tr:nth-child(even) {background-color: #f2f2f2}
```

## 11.11 race.js (WebGL)

// indlæs billeder:

```
var teams_2018 = [];
```

```
let boats = [];
```

```
let waves = [];
```



```
function preload(){

getData();

white = loadImage('./img/boat_white.png');
green = loadImage('./img/boat_green.png');
black = loadImage('./img/boat_black.png');
red = loadImage('./img/boat_red.png');
yellow = loadImage('./img/boat_yellow.png');

wave1 = loadImage('./img/waves_1.png');
wave2 = loadImage('./img/waves_2.png');
wave3 = loadImage('./img/waves_3.png');
wave4 = loadImage('./img/waves_4.png');
wave5 = loadImage('./img/waves_5.png');

boats = [black, white, green, red, yellow];
waves = [wave1, wave2, wave3, wave4, wave5];
}

class Team{
  constructor(name, ranking, score, color){
    this.name = name;           // skal hentes fra database
    this.rank = ranking;        // skal hentes fra database, men kunne i princippet beregnes på
    baggrund af score
    this.score = score;         // skal hentes fra database

    this.boost = score;         // resettes ved ny kørsel (kopi af score)
    this.position_x = 20;       // resettes ved ny kørsel
    this.speed = 20;            // resettes ved ny kørsel
    // bruges ikke da der bruges billeder i forskellige farver
    this.team_color = color;    // kan bare være tilfældig, men der skal helst ikke være to der
    har samme farve.
  }
}

class Game{
  constructor(){
    createCanvas(800, 400, WEBGL);
    fill("#333377");
    this.running = false;
  }
  update(){
    // Der er ikke taget højde for deltatime.

    this.running = false;
  }
}
```

```

    for (let i = 0; i < 5; i++){
        var addBoost = Math.random() * teams_2018[i].boost;
        teams_2018[i].boost -= addBoost;
        teams_2018[i].speed += addBoost;

        // så længe et skib stadig flyttes kører løbet!
        if (teams_2018[i].position_x <= width - 60){
            teams_2018[i].position_x += teams_2018[i].speed/100;
            this.running = true;
        }
    }
}
draw(){

    // clear screen
    background("#333377");

    for (let i = 0; i < 5; i++){

        // bølger
        let yOffset = Math.sin(frameCount/8)*2;
        let yOffsetBoat = Math.cos(frameCount/8)*4;

        ambientLight(125,125,125);

        texture(boats[i]);

        push();
        translate(teams_2018[i].position_x - width/2, 75*i + 40 + yOffsetBoat -
height/2);
        plane(50,50);
        pop();

        texture(waves[i]);

        push();
        translate(0, 75*i + 40 + yOffset - height/2 + 40);
        plane(waves[i].width*1.5, waves[i].height);
        pop();

        push();
        translate(width/2 - 80, 0)
        ambientMaterial(250);
        box(10, height, 0)
        pop();

        //texture(winner);
    }
}

```

```

        //plane(100,100);
    }
}

// problem dataen hentes asynkront! så det er ikke tilgængeligt udenfor funktionen
// en løsning: skriver til DOM
// TODO: lige nu sorterer den ikke skibene, hvis der ligger flere end 5 i tabellen skal vi
sortere

function getData(){
    // hent data fra databasen. AJAX-metode?
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            myObj = JSON.parse(this.responseText);

            // skriv data til DOM for at gemme til senere indlæsning:
            document.getElementById("shipName0").innerHTML = myObj[0][0];
            document.getElementById("shipName1").innerHTML = myObj[1][0];
            document.getElementById("shipName2").innerHTML = myObj[2][0];
            document.getElementById("shipName3").innerHTML = myObj[3][0];
            document.getElementById("shipName4").innerHTML = myObj[4][0];

            document.getElementById("shipRang0").innerHTML = myObj[0][1];
            document.getElementById("shipRang1").innerHTML = myObj[1][1];
            document.getElementById("shipRang2").innerHTML = myObj[2][1];
            document.getElementById("shipRang3").innerHTML = myObj[3][1];
            document.getElementById("shipRang4").innerHTML = myObj[4][1];

            document.getElementById("shipScore0").innerHTML = myObj[0][2];
            document.getElementById("shipScore1").innerHTML = myObj[1][2];
            document.getElementById("shipScore2").innerHTML = myObj[2][2];
            document.getElementById("shipScore3").innerHTML = myObj[3][2];
            document.getElementById("shipScore4").innerHTML = myObj[4][2];

            load();

        }
    };
    xmlhttp.open("GET", "getshipinfo.php", true);
    xmlhttp.send();
}

function setup(){

```

```
/*
  // text
  graphics = createGraphics(200,200);
  graphics.background(255);

  winner = createGraphics(300,100);
  winner.fill(255);
  winner.textAlign(CENTER);
  winner.text(teams_2018[0].name + " har vundet!",150,50);
*/

game = new Game();

load();

}

function load(){
  // hent data fra DOM
  teams_2018.push(new Team(document.getElementById("shipName0").innerHTML,
  document.getElementById("shipRang0").innerHTML,
  document.getElementById("shipScore0").innerHTML, "red"));
  teams_2018.push(new Team(document.getElementById("shipName1").innerHTML,
  document.getElementById("shipRang1").innerHTML,
  document.getElementById("shipScore1").innerHTML, "green"));
  teams_2018.push(new Team(document.getElementById("shipName1").innerHTML,
  document.getElementById("shipRang2").innerHTML,
  document.getElementById("shipScore2").innerHTML, "yellow"));
  teams_2018.push(new Team(document.getElementById("shipName3").innerHTML,
  document.getElementById("shipRang3").innerHTML,
  document.getElementById("shipScore3").innerHTML, "pink"));
  teams_2018.push(new Team(document.getElementById("shipName4").innerHTML,
  document.getElementById("shipRang4").innerHTML,
  document.getElementById("shipScore4").innerHTML, "black"));

  //game.draw();

}

function start(){
  // reset og start gameloop:

  restart();
}

function restart(){
  for (i = 0; i < teams_2018.length;i++){
```

```
        teams_2018[i].speed = 20;
        teams_2018[i].position_x = 20;
        teams_2018[i].boost = teams_2018[i].score;
    }
}

function draw(){

    game.update();
    game.draw();
    /*
    // stop loopet når bådene er i mål
    if (game.running == true){
        requestAnimationFrame(gameLoop);
    } else {
        // kan gøres smartere...
        for (i = 0; i < teams_2018.length;i++){
            if (teams_2018[i].rank == 1){
                // bør gøres mere dynamisk
                //game.context.fillStyle = 'white';

//game.context.rect(game.canvas.width/5,game.canvas.height/2.5,500,100);
                //game.context.fill();
                game.context.fillStyle = 'white';
                game.context.font = '30px sans-serif';
                game.context.fillText(teams_2018[i].name + " har
vundet!",game.canvas.width/4, game.canvas.height/2);
            }
        }
    }
    */
}
```