UPPGIFT 1.

Jag har delat upp uppgiften i olika steg som jag anser vara nödvändiga för att utföra den korrekt, jag har även satt upp en rimlig tid att klara av stegen på. Den här metoden kommer jag att använda för alla uppgifter i "Uppgift 1 - Tre enkla programmeringsuppgifter".

Planering av Uppgift 1a:
1. Börja med att skriva koden som läser in textraden som användaren matar in och spara in den i en sträng. (5min)

2. Läs sedan av varje bokstav/nummer (eftersom ingen felhantering ska implementeras) av strängen genom att använda en loop.(15min)

3. Implementera if-satser som kontrollerar hur många a/A som finns i strängen och skapa 2 int-variabler som ökas om så är fallet.(15min)

4. Skapa sedan en sträng med String.Format som skriver ut antalet a/A beroende på vad int-variablerna har för värde som skapades i steg 3.(5min)

Totaltid (45min)

Reflektion:
Steg 1: 5 min, inga problem med det här steget.

Steg 2: 5 min, utförandet blev inte som jag hade tänkt mig här, eftersom jag endast loopar igenom strängen som matas in behöver jag inte läsa av något i det här steget vilket gör att det gick snabbare än väntat.

Steg 3: 20 min, här fick jag fel när jag skulle jämföra textraden som lästes in, jag skapade en string array och jämförde vilket inte fungerade. Jag fick felsöka och leta och hittade att jag istället kunde jämföra med en char array. Tiden blev inte som väntat eftersom jag fick läsa en del om hur jag skulle lösa problemet.

Steg 4: 5 min, inga problem med det här steget.
Sluttid: 35 min.

Planering av Uppgift 1b:

1. Skriv kod som läser in nummer som matas in av användaren och spara det i en variabel av typen int. (5 min)

2. Läs av följande nummer genom att använda en for loop. (5 min)

3. Använd sedan if-satser för att ta reda på hur många nollor, udda och jämna tal det är. (10 min)

4. Presentera resultatet genom String.Format. (5 min)
Totaltid (~ 30 min)

Reflektion:

Steg 1: 5 min, fick ändra lite på sättet jag läste in siffror på från förra uppgiften, men planeringen höll ändå.

Steg 2: 15 min, For loopen blev lite problem med, eftersom jag läser in till int så kunde jag inte använda samma metod för att loopa igenom numret. Där löste jag det genom att konvertera den inlästa variabeln till string och efter det kunde jag använda samma metod. Gick över planering med 5 min.

Steg 3: 5 min, här gjorde jag klart if-satserna fortare, jag satte nog lite för bra tid eftersom beräkningen med modulus operatorn inte var så svår.

Steg 4: 5 min, presentationen var likadan som förra uppgiften vilket gjorde att det inte var några problem.

Jag fick även komplettera med en try catch sats för att programmet inte skulle krascha om man inte matade in endast siffror, vilket utökade tiden en aning.
Sluttid: 35 min.

Planering av Uppgift 1c:

1. Börja med att läsa in 2 nummer för att sedan kunna jämföra nummer när de resterande 8 läses in. (5 min)

2. Skapa sedan en if-stats för att se vilket tal som är störst respektive näst störst av de 2 inlästa. (5 min)

3. Skapa sedan en for-loop och läs in resterande tal och placera en if-stats i for-loopen som jämför med de 2 första talen. (10 min)

4. Presentera sedan resultatet med String.Format. Lägg även till en try catch sats för felhantering. (5 min)
Totaltid (~ 30 min)

Reflektion:

Steg 1: 5 min, inga problem.

Steg 2: 5 min, här fick jag dock skapa 2 nya int variabler störst respektive näststörst för att sedan kunna använda dem i nästa if-satser.

Steg 3: 15 min, här fick jag lite problem när jag skulle jämföra eftersom jag hade glömt att skapa de 2 nya variablerna i steg 2, det tog 5 min längre eftersom jag inte hittade felet direkt.

Steg 4: 5 min, presentation va inga problem.
Totaltid (30min)

Reflektion kring planering:

Efter att ha tittat tillbaka på min planering och tittat igenom strukturen så ser jag att det finns förbättringar som jag kan implementera. Det första som slog mig var att jag kunde gjort planeringen mycket tydligare i stegen som jag satte upp för att kunna göra de olika uppgifterna. Exempel ” Implementera if-satser som kontrollerar hur många a/A som finns i strängen och skapa 2 int-variabler som ökas om så är fallet.” Jag har skrivit precis vad jag ska göra i det här fallet, men jag har inte planerat det på ett tydligt sätt. Det var nog därför jag fick det här slutresultatet på steget: ” Här fick jag fel när jag skulle jämföra textraden som lästes in, jag skapade en string array och jämförde vilket inte fungerade.” Jag tror det hade varit bättre om jag planerade HUR jag ska göra vissa saker istället för att planera endast vad jag ska göra. För om planeringen hade varit tydligare så kanske jag hade märkt att jag var tvungen att göra på ett annat sätt innan jag började göra uppgiften.

Ett annat fel som jag la märke till när jag jobbade var att planeringen blev ganska svår att hålla, eftersom de olika tiderna jag satte upp var såpass korta. Planeringen avvek ett antal gånger på grund av väldigt små fel. Det tror jag hade gått att undvika om jag kanske planerade UPPGIFTERNA istället för en och en. Men genom att dela upp planeringen som jag gjorde nu fick jag helt plötsligt egna steg som jag själv kunde följa när jag gjorde uppgiften och det underlättade en hel del. Innan när jag gjort uppgifter i C# kursen och Webbteknisk Introduktion så har jag endast satt mig ner, kollat på instruktionerna vi fått ut av lärarna och sedan gjort uppgiften. Jag märkte väldigt tydligt när jag arbetade med dessa (små) programmeringsuppgifter att allt var ganska lätt. Det har väl en hel del att göra med att inte så mycket är nytt, men det var heller inte mycket som var nytt i Webbtekninsk Introduktion för mig heller, men ändå kunde jag fastna ganska mycket på uppgifterna.

Det jag också noterade var att även om ”de små” programmeringsuppgifterna blev svårare så blev jag nästan klar snabbast med den sista uppgiften, just genomförandet. Eftersom jag blev bättre på att planera under uppgifterna som gick, i sista uppgiften skrev jag mer om hur jag skulle göra istället för vad jag skulle göra. Här är ett exempel på skillnaden. Uppgift 1, ” Börja med att skriva koden som läser in textraden som användaren matar in och spara in den i en sträng.”. Där har jag skrivit vad som ska göras, jag har även gjort liknande fel även i sista planeringen, men på vissa ställen så är planeringen mycket tydligare. ”Skapa sedan en for-loop och läs in resterande tal och placera en if-stats i for-loopen som jämför med de 2 första talen”. Eftersom planeringen här är mycket tydligare och mer genomtänkt blev det också mycket lättare att utföra, eftersom jag lättare kan hålla koll på vad som ska göras. Det blir också väldigt mycket lättare att planera tiden de olika stegen tar om jag lägger upp planeringen på det här sättet. Slutligen kan jag säga såhär, genom att ge tid till att planera uppgifterna så sparade jag mycket tid när jag genomförde dem. Jag kände att när jag hade planerat uppgifterna så fastnade jag inte på samma sätt som jag har gjort innan så fort ett litet fel dyker upp. Givetvis är det svårt att utföra en planering som håller bort alla fel men att ta sig tid till att planera uppgifterna mer noggrant och skriva hur de ska utföras istället för vad som ska utföras är en väldigt bra metod. Jag har väldigt länge tagit lätt på planering, men genom att lägga tid på att både förutse fel och möjligheter gör att man sparar väldigt mycket tid.

UPPGIFT 2.

Det finns väl många olika planeringsstrategier därute och alla har vi vår egen, den jag har använt mig av i första uppgiften är en strategi som funkar väldigt bra för mig och det är väl så jag har planerat mina uppgifter när jag gjort dem. När mindre projekt ska göras brukar jag inte skriva ner mina planeringar utan bara försöka ha allt i huvudet. Det är en strategi som funkar i många avseenden men när projekt blir större och uppgifterna blir svårare är det väldigt svårt att kunna hålla koll på allt utan att skriva ner saker. En annan planeringsstrategi som går att använda är att rita upp så kallade mindmaps för att hålla koll på saker. Det här är ett planeringssätt som jag använt mig av i vissa avseenden när uppgifter inte varit så tydliga och strikta. Det går ut på att sätta ut sitt slutgiltiga mål i mitten av mappen och så jobbar man ut, dvs när man kommer på en ide så drar man en linje från sitt slutgiltiga mål till nästa ide osv. På det här sättet kan man komma på väldigt många nya idéer för varje gång man skriver upp något nytt så öppnas nya möjligheter för nya idéer.

Jag har väldigt länge upplevt att det är svårt att hålla en planering fullt ut som man sätter upp själv. Det blir ofta att jag skjuter upp saker för jag tänker att jag hinner med dessa ändå. Det finns många saker som kan komma i vägen, ofta handlar det om att mobilen ringer eller att kompisar skriver och vill spela något spel på datorn när dem är lediga. Det kan även vara dålig sömn, inte tillräckligt med vätska respektive dålig kost osv. Dessa är saker som jag trodde spelade väldigt mycket mindre roll när man sitter och arbetar framför datorn och arbetar än när jag tex spelar fotboll. Inför varje träning och match försöker jag vara seriös med kost och sömn och jag har upptäckt att jag även behöver vara det inför en tuff ”arbetsdag” i skolan. Därför har jag upptäckt att planering är en väldigt viktig del av vardagen numera. Om jag vet att jag ska göra en väldigt tuff uppgift försöker jag planera så jag får bra med sömn och oftast ha bra mat som jag kan värma, annars är det väldigt lätt att det blir någon dålig kost. Något jag alltid försöker göra när jag vet att jag ska jobba med skoluppgifter eller annat arbete, det är att lägga undan mobilen. Det känns som att mobilen är något som kan störa min planering väldigt lätt. Om jag hör att någon ringer eller får ett sms, whats app meddelande eller vad som helst blir jag väldigt nyfiken. Då går det lite tid till att titta på mobilen, oftast hittar man något mer intressant bland händelserna och så fortsätter det. Idéerna jag hade innan jag tog fram mobilen är oftast bortblåsta. Jag försöker så gott det går att koppla bort all kommunikation på datorn också, såsom facebook, steam och skype. För där vet jag att folk kommer skicka annars vilket gör att man kanske går med på att spela något spel med dem en stund, för jag intalar mig själv att jag hinner. Så det jag försöker göra är att skära av så mycket kommunikation som möjligt och sedan gå in i min ”egna” värld.

De två förbättringsåtgärderna jag skulle vilja implementera i mitt arbete är vissa saker jag redan tagit upp men att lyfta fram dem tydligare. När jag skriver planeringen ska jag skriva tänka ut och skriva mer hur jag ska göra saker istället för vad jag ska göra, eftersom jag då får en bättre blick om hur långt tid det kommer ta och vad som ska göras. När jag planerar ska jag lägga en större vikt vid vad som kan gå fel också och kanske implementera en felmarginal där jag tror att det kommer behövas i uppgiften.

UPPGIFT 3.

Jag har delat upp uppgiften i olika steg som jag anser vara nödvändiga för att utföra den korrekt, jag har även satt upp en rimlig tid att klara av stegen på. Den här metoden kommer vara lik den jag använde till första uppgiften men genom att ha analyserat metoden och gjort förbättringar kommer den skilja sig en aning från uppgift 1.

Uppgift 3a:

1. Börja med att presentera en text så att användaren vet vad som ska göras.

2. Skapa en string som blir värdet av vad användaren matar in.

3. Skapa en for-loop som läser av varje bokstav i strängen.

4. Implementera en if-sats som tar reda på om sista bokstaven är lika med första och efter det går framåt respektive bakåt i char-ordningen.

5. Implementera en els-sats som kastar ett undantag.

6. Implementera felhantering, try catch ihop med en while-loop.

7. Om while-loopen bryts presenteras resultatet true.

Uppgiften ska ta 30 min och då är lite felmarginal inräknad.

Reflektion:

Uppgiften tog ca 25 min att genomföra, felmarginalen var bra att jag la in eftersom jag kände att det skulle bli lite problem med if-satsen, för jag hade inte riktig koll på hur jag skulle skriva den. Felet jag stötte på var att jag inte kunde skriva str.Length inuti en char för att ta reda på sista bokstaven utan jag fick skapa en ny variabel av sträng-längden istället.

Uppgift 3b:

1. Börja med att skapa 2 privata int fält.

2. Skapa konstruktorn som ska ta två parametrar.

3. Skapa getNumerator och getDominator som ska returna täljaren respektive nämnaren som konstruktorn initierar.

4. Skapa isNegative som kollar om talet är positivt eller negativt med hjälp av en if-sats.

5. Skapa en Fraction metod som ska heta add, den ska ta två parametrar och ska skapa ett nytt Fraction objekt.

6. Skapa metoden multiply() som ska göra beräkningen.

7. Skapa en bool isEqualTo som testar nämnare respektive täljare på olika Fraction objekt med hjälp av en if-sats.

8. Skapa en string toString() som retunerar resultatet i en sträng.

9. I main metoden ska funktionen testas och där ska även en try catch sats finnas med.

Uppgiften ska ta 2h enligt min beräkning då jag är osäker på hur vissa moment ska utföras.

Reflektion:

Uppgiften tog ca 2 timmar och 30 minuter att utföra. Det var väldigt mycket svårare att skriva en utförlig planering när jag inte riktigt kunde föreställa mig hur programmet skulle se ut. Det blev några fel som jag fick leta upp och felsöka. Första felet var att jag inte hade implementerat felhantering på ett korrekt vis. Det löste jag genom att kasta ett undantag i multiply som sedan togs hand av try catch satsen i main metoden. isEqualTo metoden hade jag också problem när jag skulle jämföra eftersom jag endast jämförde objekten med varandra och då funkade ingenting. Jag löste detta genom att jämföra objekten i Fraction objektet istället och på så sätt kunde jag jämföra dem. Ett litet fel som också tog extra tid var att istället för att skriva ut toString i main metoden så kallade jag endast på metoden och ingenting skrevs ut. Det tog också ett tag innan jag hade listat ut det.

Reflektion kring planering:

Första uppgiften i Uppgift 3 gick väldigt bra med min nuvarande planering. Eftersom jag visste precis hur programmet skulle se ut och hur lång tid det skulle ta att göra momenten eftersom jag visste på ett ungefär vilken kod som skulle skrivas. Jag fick dock problem med den andra uppgiften, dels för den var större och skulle ta längre tid, men där fanns vissa saker jag inte hade 100% koll på. Vilket gjorde att planeringen blev ganska svår att genomföra på samma sätt som i tidigare uppgifter. Här fick jag istället fundera lite extra på felmarginal, eftersom jag var ganska säker på att jag var tvungen att kolla upp en del saker. Förbättringarna som jag införde ifrån uppgift 2 tycker jag fungerade bra på den första uppgiften i steg 3. Där tyckte jag själv att jag lyckades skriva en relativt kort planering men den hade all information som behövdes för att endast följa den. Det blev också lättare att utföra när jag bara kunde följa planeringen och på så sätt komma ihåg vad det var som skulle göras. Det jag tror att jag måste jobba på är hur jag ska lösa uppgifter som jag kanske inte riktigt har koll på hur de ska skrivas. För att planera utifrån felmarginal märkte jag var väldigt svårt eftersom du inte vet vart felen kommer dyka upp oftast och du vet inte hur lång tid det kommer ta att lösa dem. Jag tror att den mest hållbara planeringen är att istället planera in extra tid för att ta reda på saker som jag är osäker på inom programmet som jag skriver. Tex om jag inte har bra koll på hur konstruktorer fungerar så kanske jag ska planera in tid för att kunna ta reda på vissa saker när det gäller konstruktorer. Det går givetvis att ta reda på all information innan jag börjar med uppgiften men då kommer jag troligtvis inte ha den förståelsen som jag behöver sedan för att lösa de specifika uppgifterna. Då kanske jag får leta upp information ändå när jag väl kommer till det steget. Men när jag ska planera större uppgifter kanske det är bra att dela upp planeringen, i en del som kommer innan utförandet. Det vill säga jag sätter upp en planering för vad jag borde lära mig innan jag startar uppgiften efter jag delat upp den i vad jag ska göra. Är det någon del av programmet jag ska skriva som jag är osäker på så kan det vara bra att ha lite förkunskap och kanske även förstå mycket mer när jag väl börjar programmera. Innan när jag har gjort uppgifter i C# kursen så stötte jag på delar i uppgiften som jag inte förstod och efter jag hade lärt mig om vad de delarna betydde

så var jag tvungen att skriva om nästan hela min kod. När jag planerar större uppgifter till mig själv i fortsättningen så kommer jag nog att dela upp planeringen lite mer istället för bara steg.

UPPGIFT 4.

Planering av en CloudPortifolio produkt.

1. Först måste ett grundläggande gränssnitt skapas så att alla kan börja jobba med sina olika delar i projektet.

2. Sedan måste inloggningsinterfacet skapas, som gör att användare kan logga in, skapa nya konton på sidan och att deras användarinformation sparas och kan återupptas nästa gång de loggar in. Här ska även finnas en panel som hanterar olika nivåer av användare, dvs admins och vanliga användare. Det ska även skapas en funktion som gör att admins har rätt till att ta bort och ändra åtkomst till dokument, annars ska det bara vara ägaren av dokumentet som ska kunna dela och redigera.

3. Applikationen ska också sättas upp mot servern där alla dokument och all information ska sparas och lagras. Det ska även skapas en uppladdningsfunktion som gör att användaren kan ladda upp och spara sina filer via ”molnet” med andra ord, servern.

4. Användarvänligheten måste också skapas, dvs ordningen där alla filer ska vara, alla så kallade ”filträd”. Det ska också finnas en koppling mellan användarinformationen och filerna. Systemet ska alltså fungera såhär: Varje användare ska få ett unikt filträd beroende på vilka filer de har redigerat vart de har sparat dem osv. Layoten av alla filer ska alltså vara unik beroende på vilken användare som loggar in.

5. Det måste också skapas en ”overall” layout som tar hänsyn till vad andra har redigerat för dokument. Om en användare har varit inne och redigerat ett dokument så ska det tydligt visas vilka dokument som har blivit redigerade.

6. Det måste skapas en kontrollpanel för admins så de kan hantera alla olika användare. Om en användare skulle vilja ta bort sitt konto så ska en funktion ta bort alla mappar och dokument tillhörande användaren. Det ska även finnas en funktion som gör att användarens delade dokument tas bort om de andra användarna som delar denna information samtycker, detta kommer att skötas genom ett frågeställande dokument som skickas via epost.

Hur ska detta utföras?

Jag skulle tilldela olika områden till alla 5 personer.

Person 1: Skapa inloggningsinterface och även skapa användarinformationen så personer enkelt kan registrera sig, dessutom ordna admin-panelen och tilldela admins full rättighet.

Person 2: Den här personen skulle sköta användarvänligheten, så att saker är på rätt ställe och att det ska finnas ett enkelt sätt att hitta dokumenten på.

Person 3: Ska sköta all serverhantering, dvs om användare laddar upp sina dokument mot sida så ska de sparas i molnet på servern som vi hyr.

Person 4: Personen här skulle sköta funkonaliteten på sidan, dvs det ska finnas en enkel textredigerare så man lätt kan ändra dokument och att sortering av dokument blir användarvänlig.

Person 5: Den här personen skulle jag sätta som en overall designer, istället för att vissa ska ordna sina egna designer så ska det finnas en tydlig linje över hela sidan och det passar nog väldigt bra om en person sköter det.

Projektet räknar jag med att det kommer ta ca 2 veckor att utföra, det behövs dock en del tid för att bugtesta och se hur bra servern svarar när det är många filer som hanteras samtidigt. Första versionen av produkten kommer att kunna släppas efter 3 veckor räknar jag med.

Reflektion kring uppgift 4:

Efter att ha planerat en mycket större uppgift som jag verkligen inte har koll på hur man ska göra allt så måste jag säga att det är väldigt svårt. Jag tror att det hade krävts mycket mer tid för att göra en riktigt bra planering på en sådan här uppgift om jag sedan skulle genomföra den. Eftersom jag hade fått planera in många timmar då jag tar reda på nya saker som ska implementeras osv. Det som också är väldigt svårt när man planerar en såpass stor uppgift är att veta hur lång tid alla de olika momenten tar. Eftersom det är svårt att veta i detalj vad som ska göras är det också svårt att sätta en bra tidsuppfattning på det hela. Jag tror att om man ska göra en riktigt bra planering så bör man ha hyfsad koll på vad som ska göras och vilka fel som kan tillkomma osv. I de första uppgifterna så hade jag en bra bild om vad som skulle göras och på så sätt kunde jag hålla min planering på ett bra sätt, jag visste ungefär hur lång tid alla moment skulle ta. När man nu fått en uppgift (som jag antar kommer från en kund tex) så blir det helt plötsligt mycket mer komplicerat och svårare att planera. Nu var jag tvungen att försöka ta reda på hur lång tid vissa moment skulle ta istället för att ha en egen uppfattning om hur lång tid det skulle ta. Det blir ju också väldigt svårt att förebygga några fel eftersom jag inte har koll på hur allting ska byggas upp heller.

Det är dock väldigt intressant om hur olika strategier man måste ha beroende på hur stor planeringen av uppgiften är. När jag började planera de ”små programmeringsuppgifterna” så kunde jag planera i väldigt mycket detalj. Jag kunde skriva ganska kort men ändå kunde jag lägga in väldigt mycket om hur jag skulle göra uppgiften i planeringen. Det blev väldigt svårt att göra det här för då hade min planering blivit alldeles för lång och väldigt rörig tror jag. Eftersom jag inte har koll på vad som är bäst att göra först eller vad som är bättre att göra senare när man programmerar applikationen. Sedan tror jag att när man planerar för en grupp, i detta fall 5 personer så tror jag att det är lättare om personen som håller i projektet gör en större planering över hela projektet och de som sedan ska arbeta med uppgifterna gör sin egen planering utifrån den. För jag tror det skulle vara väldigt svårt att ha en anställd person som sköter planeringen åt 5 stycken helt individuella programmerare. Utan jag tror att som i uppgiften att man har en person som delar ut vad dessa 5 personer ska göra sen får de göra en egen planering och utifrån deras planering så kan man sedan bestämma en rimlig tidpunkt när projektet ska vara klart. Den metoden hade jag använt om jag var projektledare för utvecklingen av en sådan applikation. Dessutom tror jag att det gynnar arbetssättet om alla får vara delaktiga i planeringen och inte bara sitta och koda och aldrig se vad den slutgiltiga produkten egentligen är. Jag har haft en egen erfarenhet av att inte få vara med och planera alls när jag jobbade på ett företag och det blir väldigt svårt att motivera sig fullt ut om man inte får vara med och sätta mål men framförallt om man inte ser några mål alls. Så jag tror att om man planerar en uppgift för en stor grupp så borde man dela ut ansvaret till folket som ska skapa produkten, det tror jag gynnar de flesta i arbetsgruppen.

En annan bit som jag också tyckte var väldigt svårt att tidsbestämma är hur lång tid det tar att bugtesta en såpass stor applikation. Eftersom jag inte visste vilken server det skulle ligga till eller om vi skulle skapa en egen databas till allt så är det väldigt svårt att förutse några fel där. Jag tror dock att användarvänligheten och användarsystemet skulle nog inte ta alltför lång tid att testa så att det håller till en rimlig nivå men att testa så att servern klarar av att hantera många dokument och vad som händer när flera personer redigerar dokument osv kan nog ta ett tag. Jag tycker att det var väldigt svårt att sätta en tid på hur långt allt ska ta och jag tror det visar på hur mycket erfarenhet spelar in när det gäller planering. För när jag tittar tillbaka på uppgifterna så kunde jag se skillnad i mitt planeringssätt bara efter några enstaka uppgifter och jag tror att om man får chans att planera och utföra en stor applikation kommer nog nästa uppgift man tar sig an att planera vara enklare. Även om erfarenhet spelar in väldigt mycket så tror jag att när man jobbar i grupp så är nog den viktigaste aspekten att man håller en väldigt tydlig dialog när det gäller planeringsbiten och att man ser till så att alla är på ett och samma spår. Jag har märkt att det är svårt att planera uppgifter när de börjar bli större men ju mer erfarenhet jag samlar på mig så kommer nog planeringen bli lättare och lättare.

Tidslogg för laboration:

13/11: Planerat de 3 första små programmeringsuppgifterna. (2h)
17/11: Genomfört de 3 små programmeringsuppgifterna samt reflekterat, uppgift 1 klar. (3h)
18/11: Gjort klart uppgift 2, reflekterat kring alternativa strategier och implementerat förbättringsåtgärder. (3h)
19/11: Planerat uppgifterna i uppgift 3. (2h)
20/11: Genomfört båda uppgifterna i Uppgift 3, samt reflekterat kring planering och utförande. (4,5h)
23/11: Planerat uppgiften i uppgift 4 och delat upp vilka som ska utföra vad i projektet. (1,5h)
24/11: Reflekterat kring uppgift 4. (2h)

Av: Kim Hallgren (kh222mj) WP