**Shoppingapplikation för goldstarr trading**

**Kurs:**

Objektorienterad programmering 1

**Klass:** SYSM4

**Termin och år:** HT 2020

**Författare:**

Michael Sava  
 Carl Emil Lidbom

Caroline Jørgensen

Hanny Frank **Lärare:** Fredrik Jönsson

Sammanfattning

Rollen som systemutvecklare kan vara mångfacetterad med olika arbetsuppgifter och upplägg. För detta krävs kunskap och förståelse för att förstå hela kedjan från att ta emot ett uppdrag till att ha en färdig produkt. Hur denna process ser ut kan skilja sig från olika företag och vad för typ av uppdrag det rör sig om. Denna rapport kommer att innefatta alla de steg som vanligtvis ingår i ett mindre uppdrag. I detta innefattas uppstart av projektet, metodik för att bryta ner och planera arbetet samt verkställandet av projektet. För detta projekt har arbetsgruppen valt att anamma Scrum-metodik i sitt arbete, om än inte fullskaligt. Detta avvägande har gjorts då projektet varit kort och arbetsgruppen varit liten. Vi vill alltså ta del av den inkrementella metodiken från Scrum i arbetet för att kunna arbeta så effektivt som möjligt.

Innehållsförteckning

[Sammanfattning 2](#_Toc55811226)

[Inledning 4](#_Toc55811227)

[Tillvägagångssätt 4](#_Toc55811228)

[Slutsatser och avslutande diskussion 10](#_Toc55811229)

[Referenser 11](#_Toc55811230)

Inledning

Att arbeta med systemutveckling i all dess form ställer inte enbart krav på att ha god kännedom i de tekniska aspekterna. Arbetet sker nästan uteslutande i koalition med både arbetskamrater, chefer och externa aktörer. Många arbeten sker numera även i projektformat. En systemutvecklare behöver därför fler egenskaper än just de tekniska aspekterna för att kunna utföra sitt arbete. Det har under flera årtionden bedrivits forskning kring gruppdynamik i arbetsgrupper och projektarbete. Forskning visar på att olika metoder för att bedriva arbete kan fungera olika bra beroende på förutsättningarna. Viss typ av metodik och ledarskap lämpar sig olika beroende på storlek på exempelvis gruppen/företaget och vilken företagskultur som råder (Jacobsen & Thorsvik 2018). Det finns även viss best practice i vilket arbetssätt som passar vilken typ av arbete. Ett agilt arbetssätt lämpar sig till mindre projekt och ett av dessa arbetssätt är Scrum (Sommerville 2016). Syftet med detta arbete är att utföra given uppgift utefter de principer som tillämpas i ett agilt arbete, primärt kring hur Scrum appliceras. Detta då vi är en liten grupp med begränsad tid.  
  
Uppdragsgivaren för projektet är från det nyuppstartade företaget Goldstarr trading. Ägarna har avtal med en liten skara av kunder och leverantörer. Initialt kommer applikationen i fråga vara småskalig för att hantera basala funktioner för verksamheten såsom att kunna lägga order för kund, ha koll på lagerstatus och kunna ta emot leveranser. Företaget i fråga har ett intresse i att expandera sin verksamhet vilket på sikt kommer att leda till att fler krav ställs vad gäller funktionaliteten på applikationen. Vad gäller dessa typ av uppdrag brukar det finnas ett ömsesidigt intresse hos både uppdragsgivaren och de som utför uppdraget att upprätthålla en god relation för framtida uppdrag. Om än uppdraget enbart är avtalat att röra de krav som ställs är det viktigt att föra en diskussion kring skalbarheten i applikationen.

Tillvägagångssätt

För att utföra ett så bra arbete som möjligt har gruppen initialt diskuterat hur upplägget ska se ut vad gäller vilka spelregler vi har, vilka aktiviteter vi har samt tidsplaneringen för detta.  
  
  
SpelreglerVad gäller spelregler är det bestämt att vi sitter från 9-16 varje dag i samtal. 16:30 samlas alla dagligen i ett samtal för en sammanfattning på dagen och vad som ska

göras inför nästkommande dag. Uppgifterna är någorlunda uppdelade mellan oss, specifikt för de som inte kan delta i samtalet under dagen, annars sker grupparbetet till stor del i samlad grupp.   
Sedan tidigare har gruppen skapat spelregler som även är tänkta att efterföljas i detta arbete. Följande punkter togs då upp:  
  
**Kommunikation**Om en person i gruppen får svårigheter med sina uppgifter är det bra att kommunicera det till gruppen i tid.

**Hålla tider/anmäla frånvaro**Ha tålamod och förståelse för formatet vi jobbar i.Ha förståelse för att kommunikation inte är densamma vid ett fysiskt möte

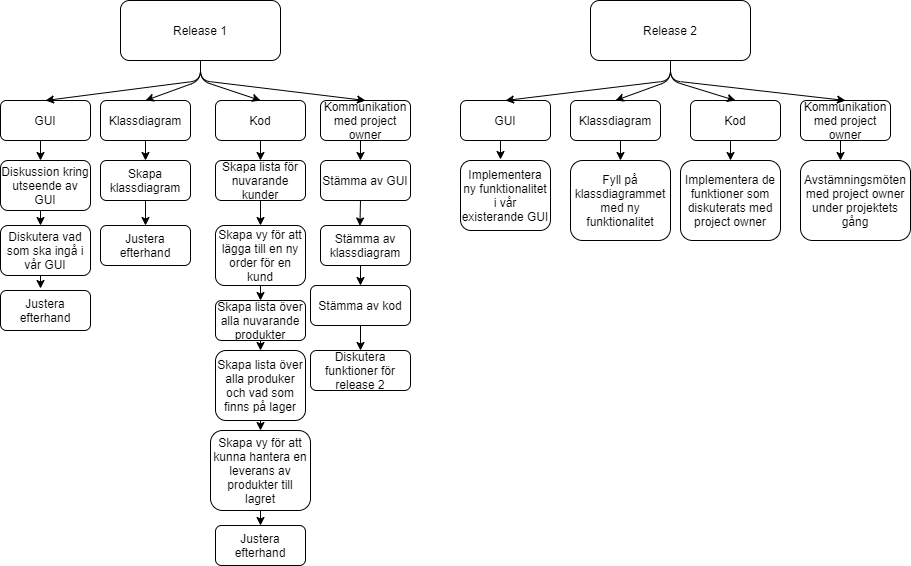
**Struktur**Att arbeta strukturerat och veta hur vi gör och vad vi gör, annars får vi stämma av med varandra

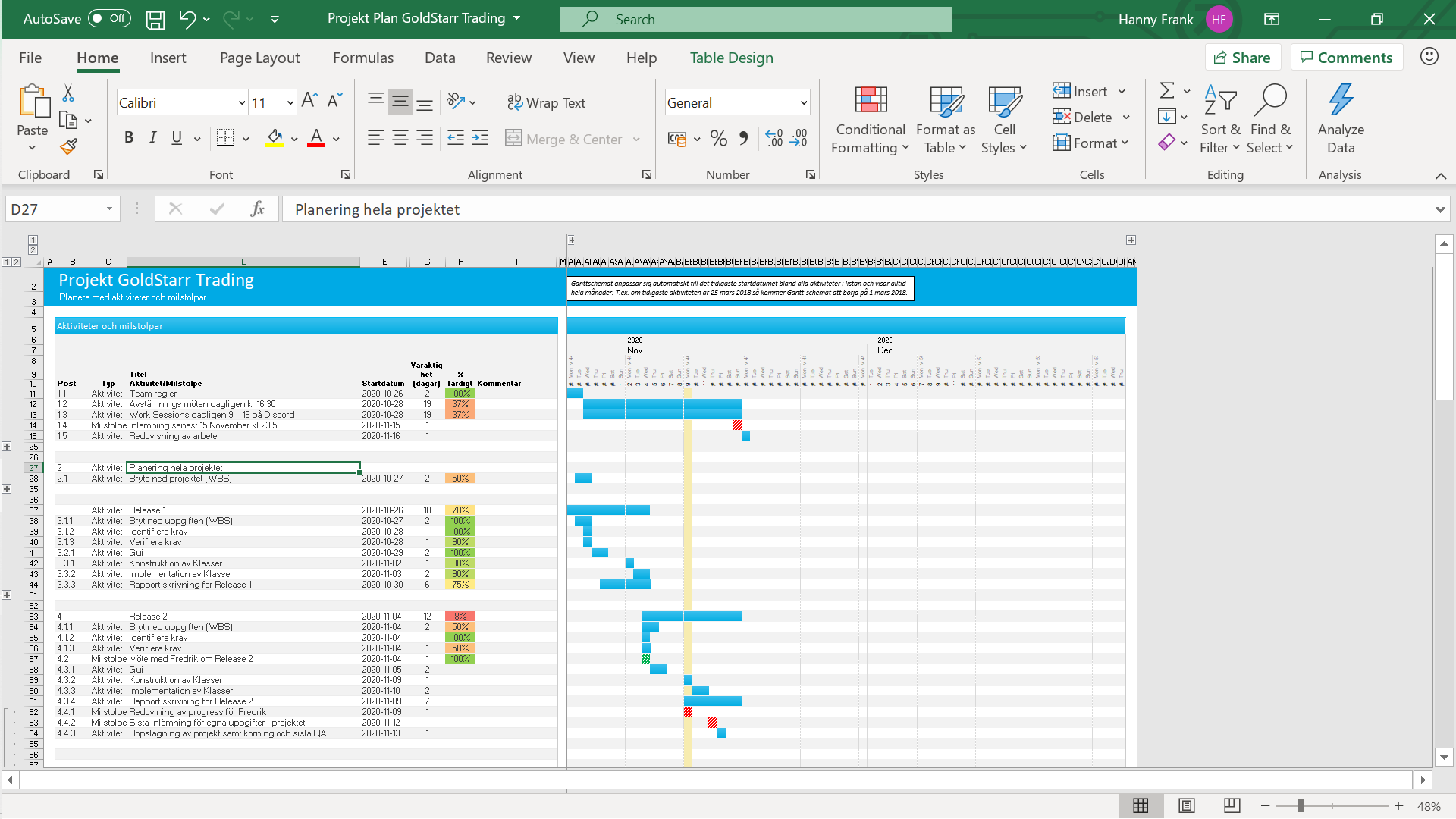
**Samarbeta**Dela information med varandra, förutsatt att detta sker inom rimliga gränser.

Om vi märker att dessa inte går att upprätthålla så är det viktigt att i tidigt skede kommunicera detta och agera enligt Newtons riktlinjer för projektarbete.

**Roll och ansvarsfördelning**Vad gäller roller och ansvarsfördelningså har gruppen valt att arbeta utefter ett agilt arbetssätt. Delar ur scrum-metodiken har applicerats i arbetet. Vi har dagliga möten vid samma tid där vi går igenom vad som har gjorts och vad som behöver göras för att hinna med vår sprint. Arbetet är inte strikt uppdelat efter rollen utan alla i gruppen gör det som behöver göras efter behov.  **Release 1**Arbetet är uppdelat i två releaser. Efter att första releasen är fullständig kommer möte hållas med uppdragsgivaren i huruvida det finns tid för att fortsätta med release 2 under projekttiden. Det är av större vikt att ha en fullständigt fungerande funktion än att ha ett projekt som har funktionalitet som inte är helt testad. Detta är alltså skälet till att release 1 ska vara fullständig innan planering och implementation av release 2 börjar.

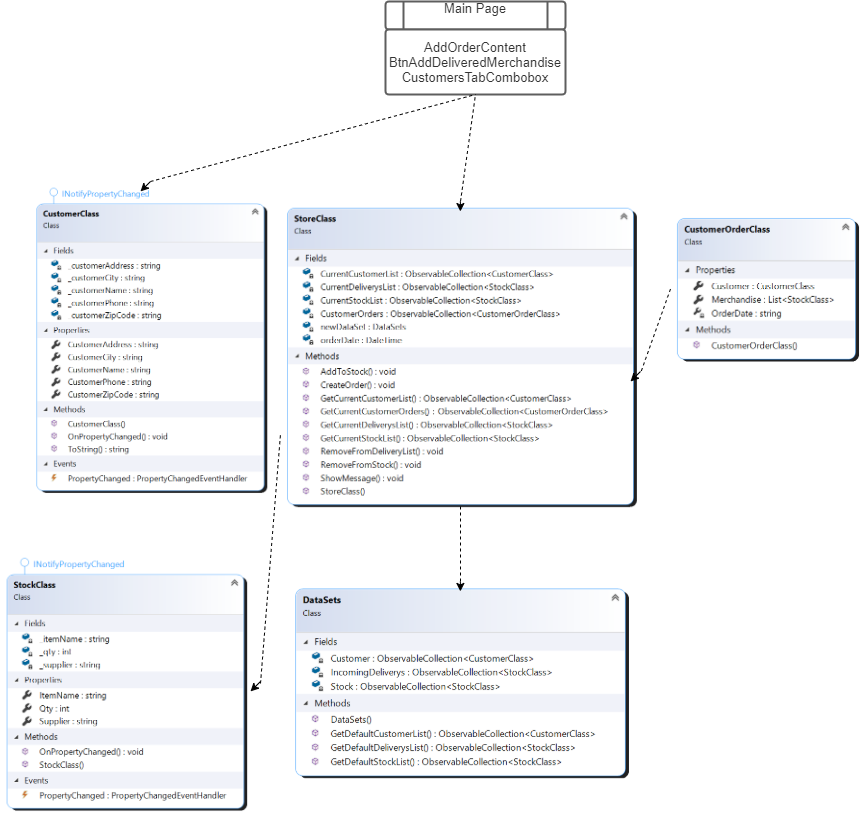
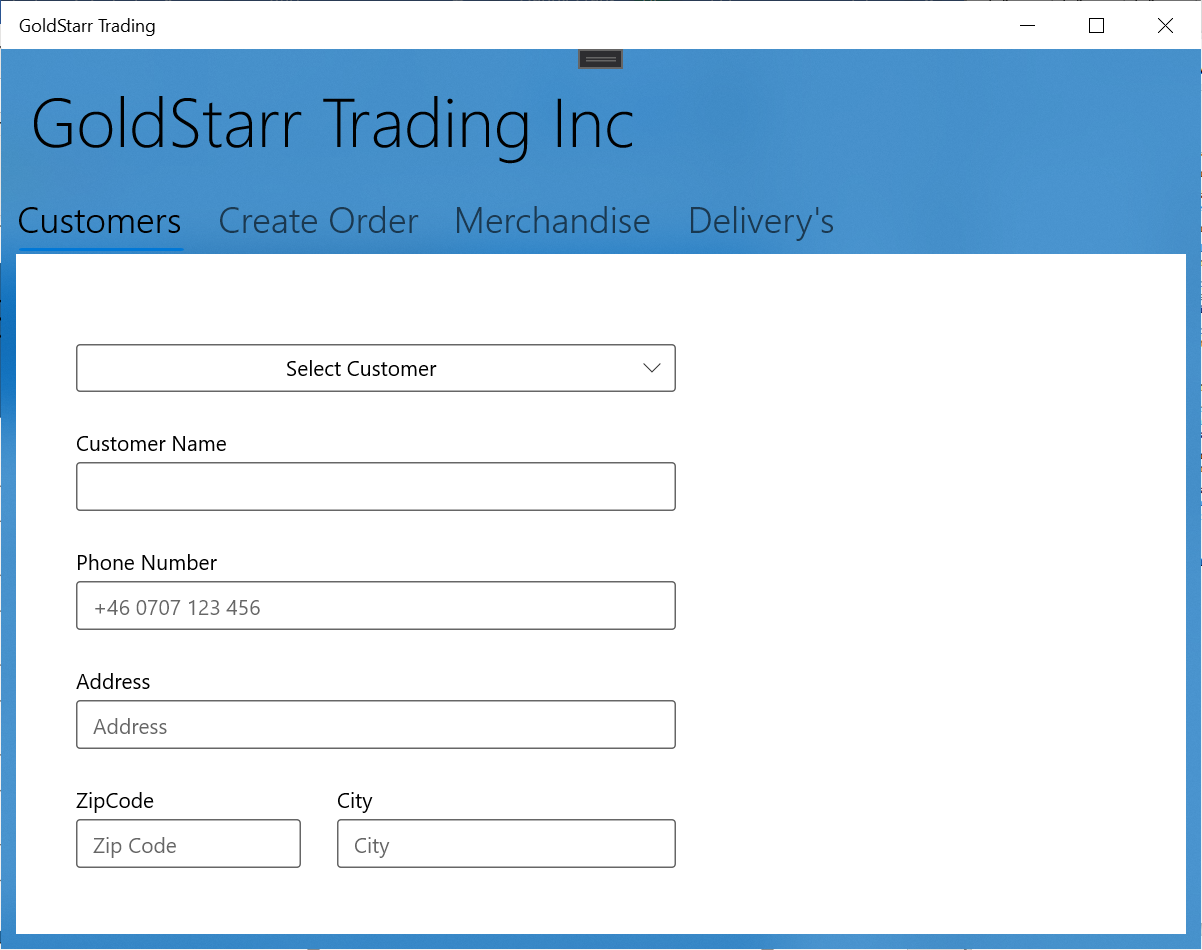
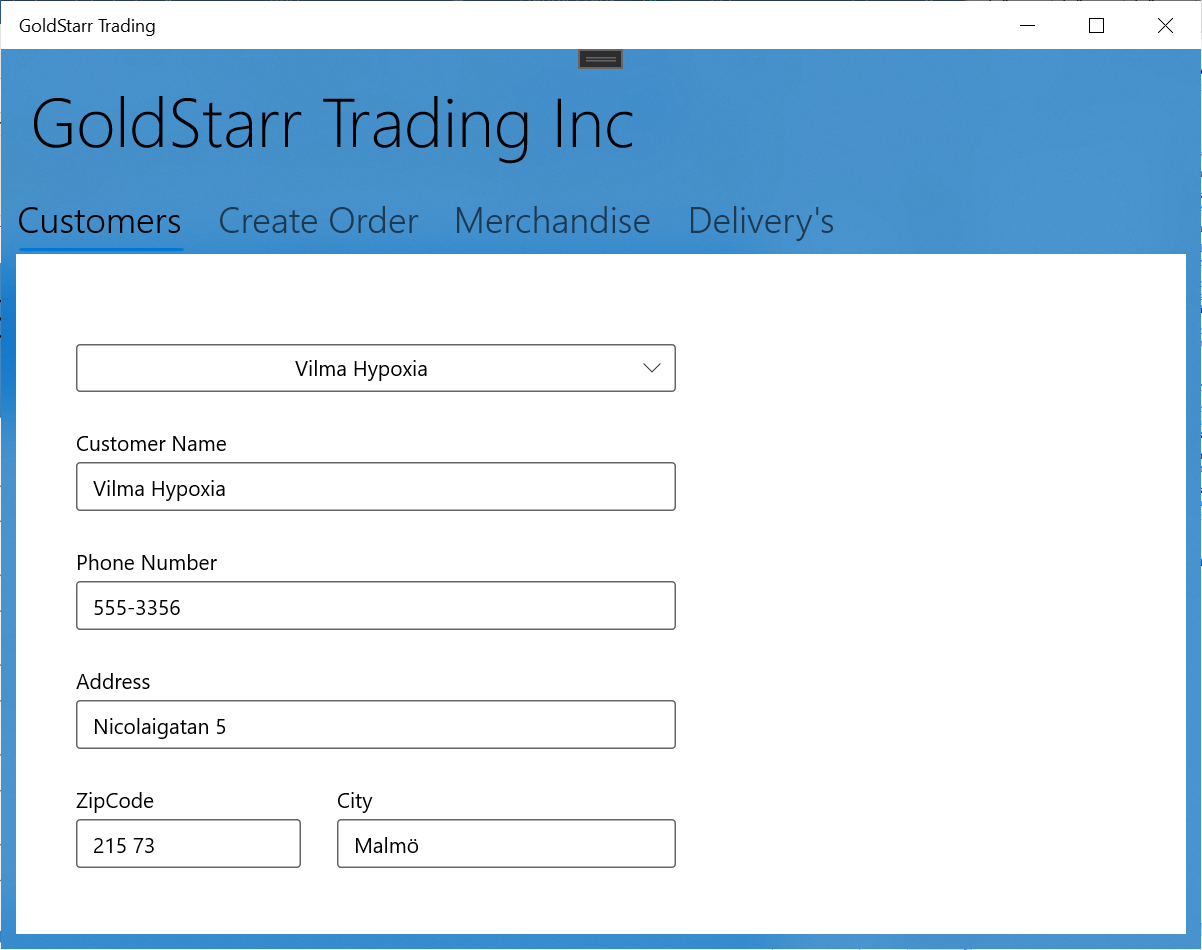
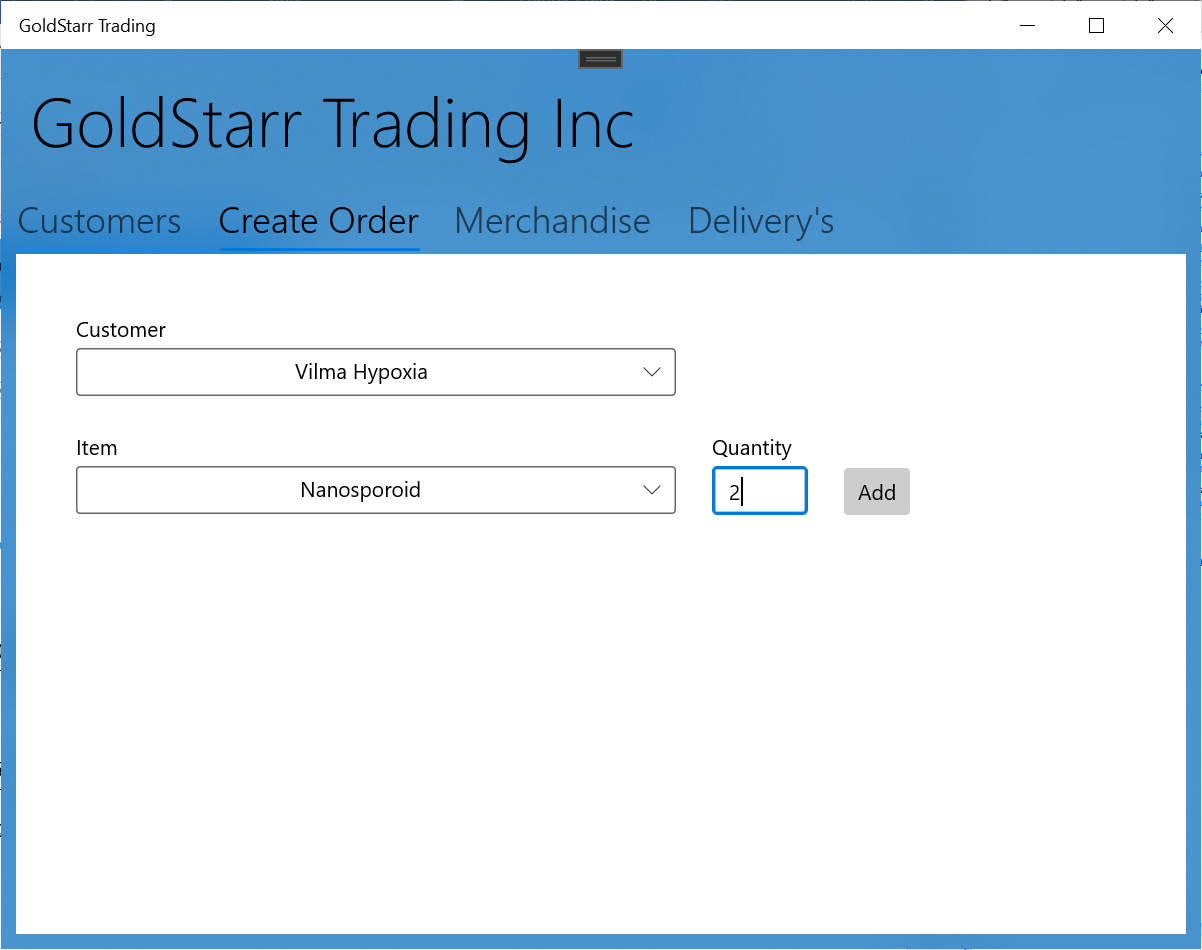
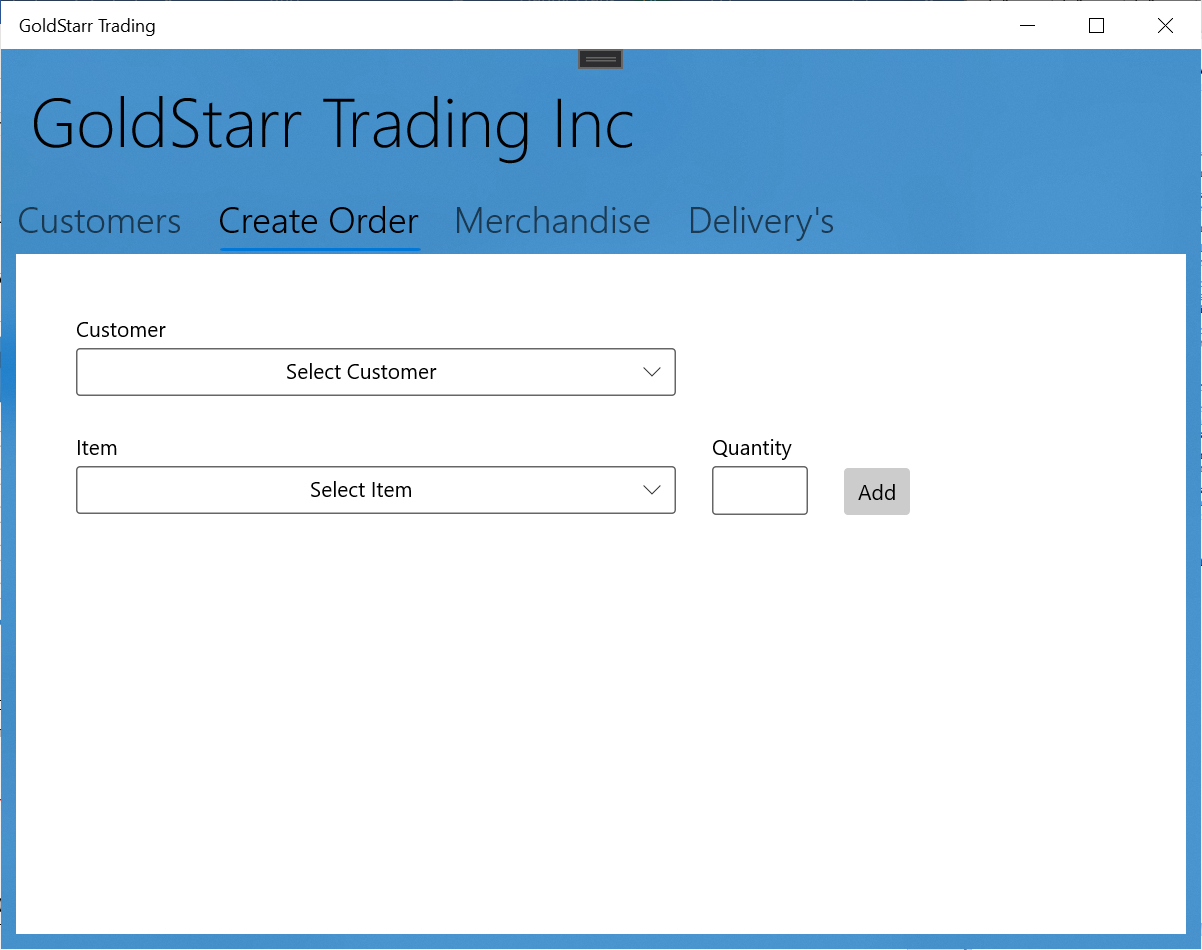
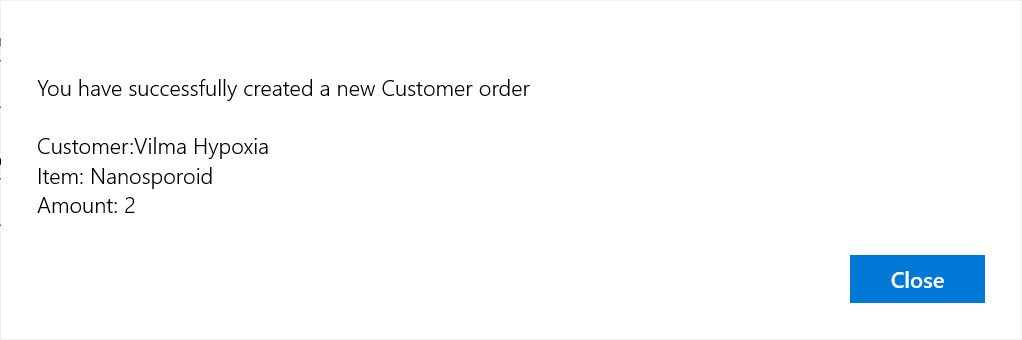
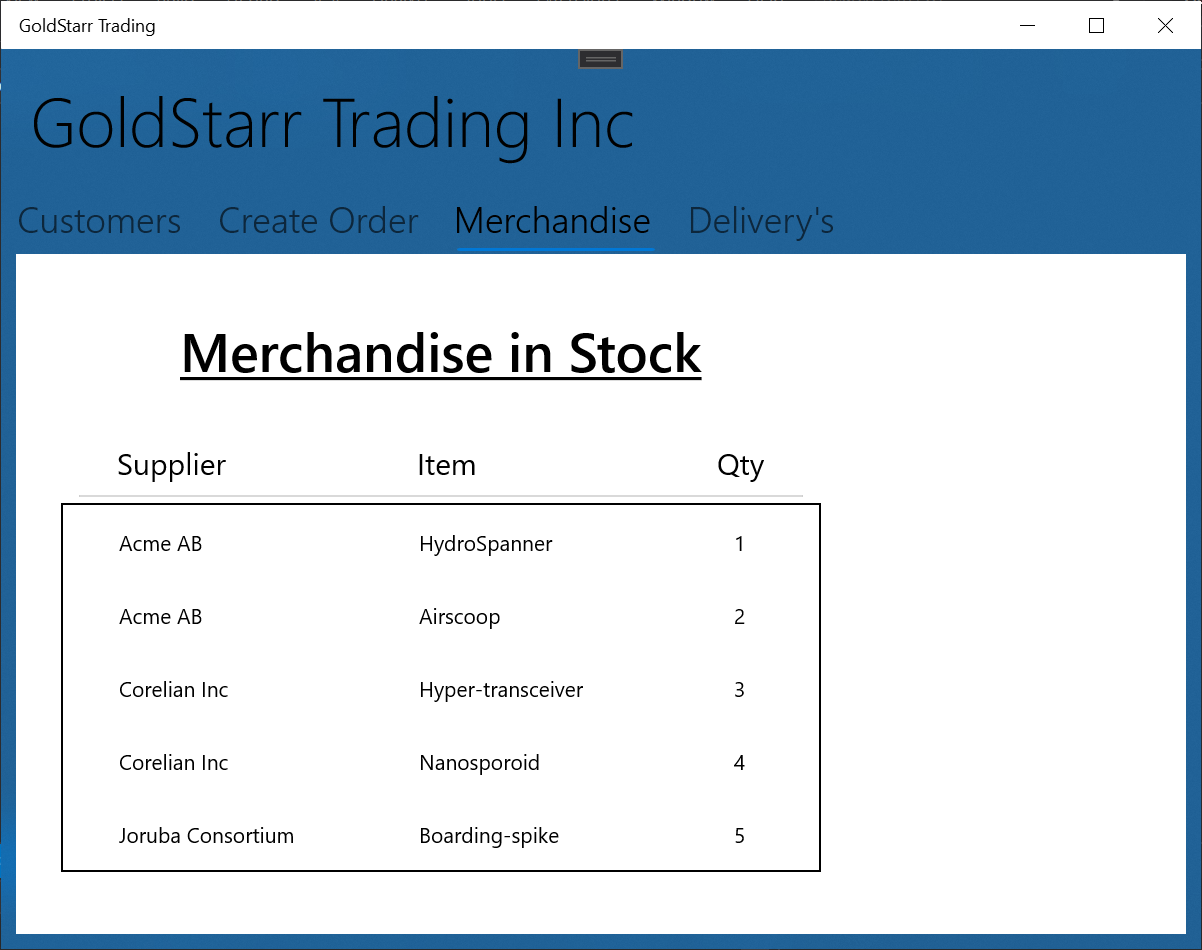
**Nedbrytning av projektet**För att få en överskådlig bild av omfattningen av projektet är det viktigt att dela upp projektet i mindre delar. För detta har gruppen använt sig av en WBS för att få förståelse för vilka delar som ska ingå.

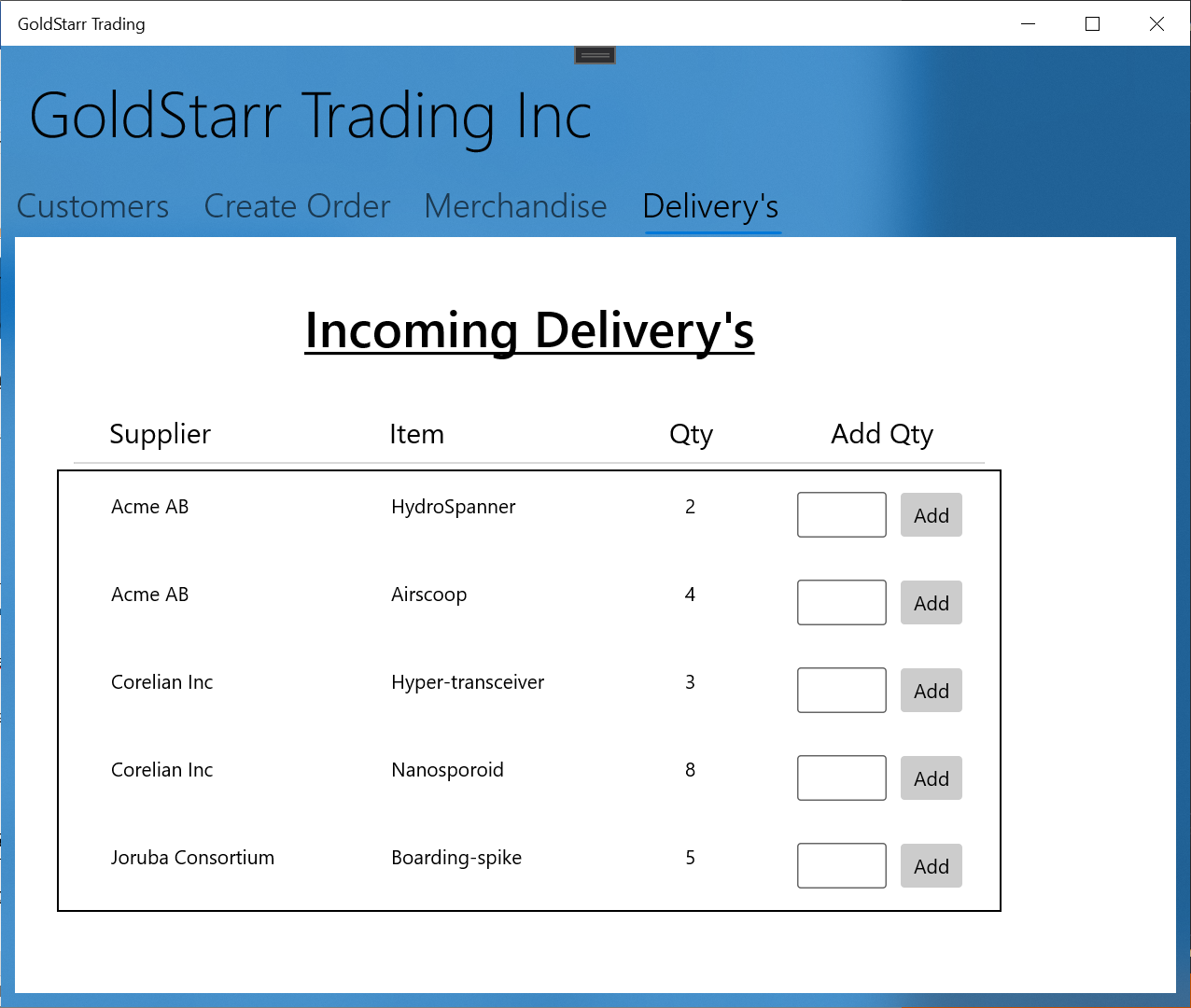
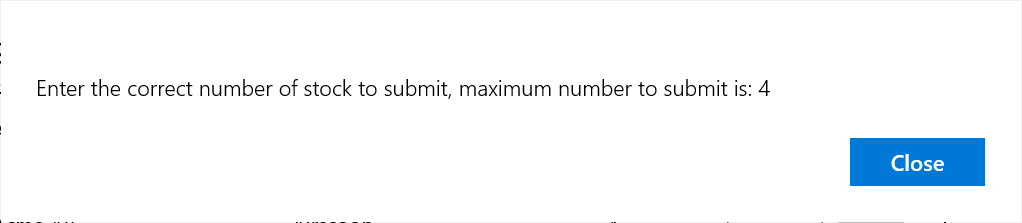
**WBS**Utifrån den diskussion som gruppen har haft i samråd med uppdragsgivare har följande WBS skapats för den första releasen. Att skapa en WBS över de aktiviteter som ska utföras underlättar för att bryta ner uppgiften i hanterliga moment. Med hjälp av WBS:en går det även att få en tidsuppfattning för respektive moment.  
  
WBS över nedbrytningen av projektet. Projektet har delats upp efter arbete med GUI:t, klassdiagram, kod samt kommunikation med projekt owner

**Tidsplanering**Projektet pågår under 3 veckors tid. Efter den WBS som har gjorts har ett gantt-schema skapats över de aktiviteter som skall utföras och den estimerade tiden för dessa. Detta kan i efterhand behöva ändras under projektets gång om estimeringen inte stämmer. Gannt-schemat är dock en bra indikation för att hålla sig till en planering inom rimlig tidsram. Med den nedbrytning som gjorts och tidsplaneringen från gantt-schemat går det även att få en översikt för om projektet är genomförbart på den tid som ges. Skulle det vara så att tiden inte räcker till då ett moment skulle ta längre tid är det därför lättare att prioritera bort aktiviteter. Dels för att vi vet vad som har lägre prioritet, dels att aktiviteterna är definierade och att det går att urskilja ett beroende mellan olika aktiviteter.  
  


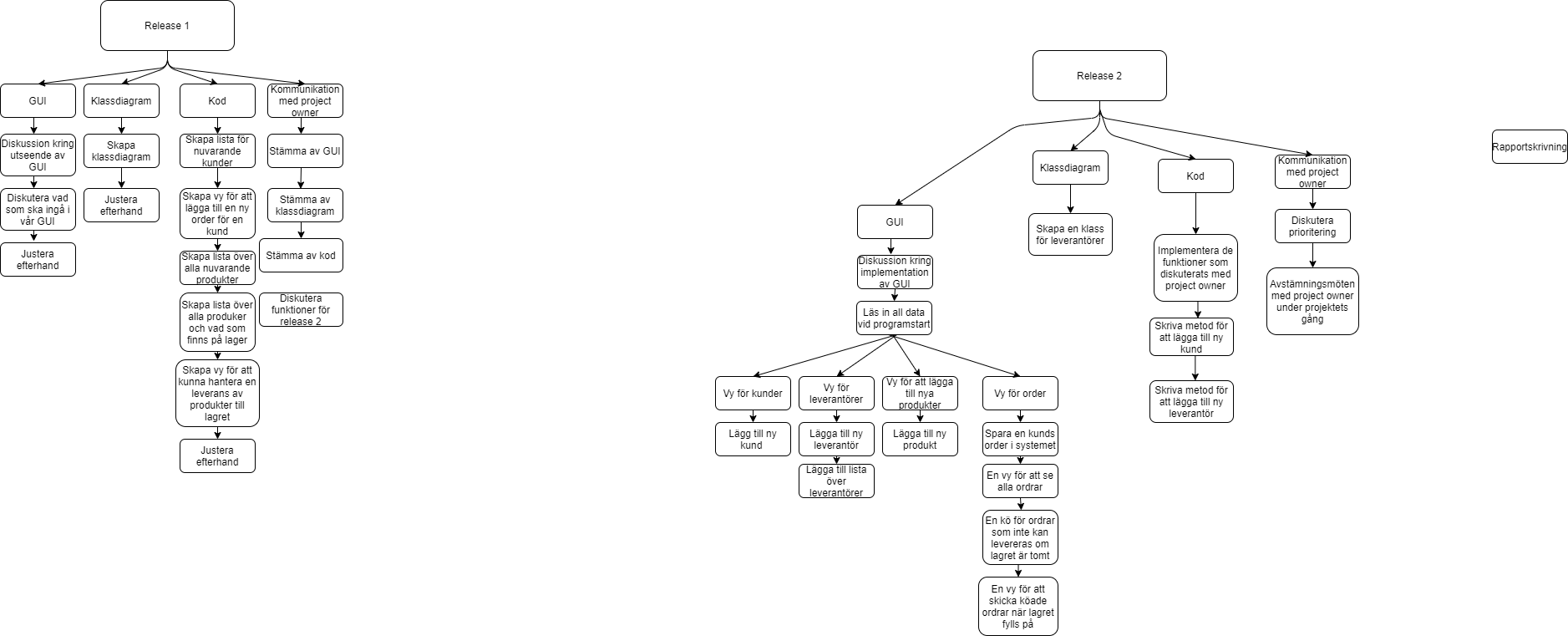
Ögonblicksbild över det gantt-schema som följdes.Diagrammet fylldes i dagligen för att skapa en uppfattning i vart vi var i projektet. **Utförande release 1**Uppdragsgivaren (project owner) önskar en applikation för att kunna hålla koll på sitt butikslager för en nystartad butik. Det finns en liten kundbas och leverantörer. På längre sikt finns det ett intresse av att expandera denna butik, bland annat med att även utveckla en onlinebutik. Initialt ställs följande krav på applikationen.

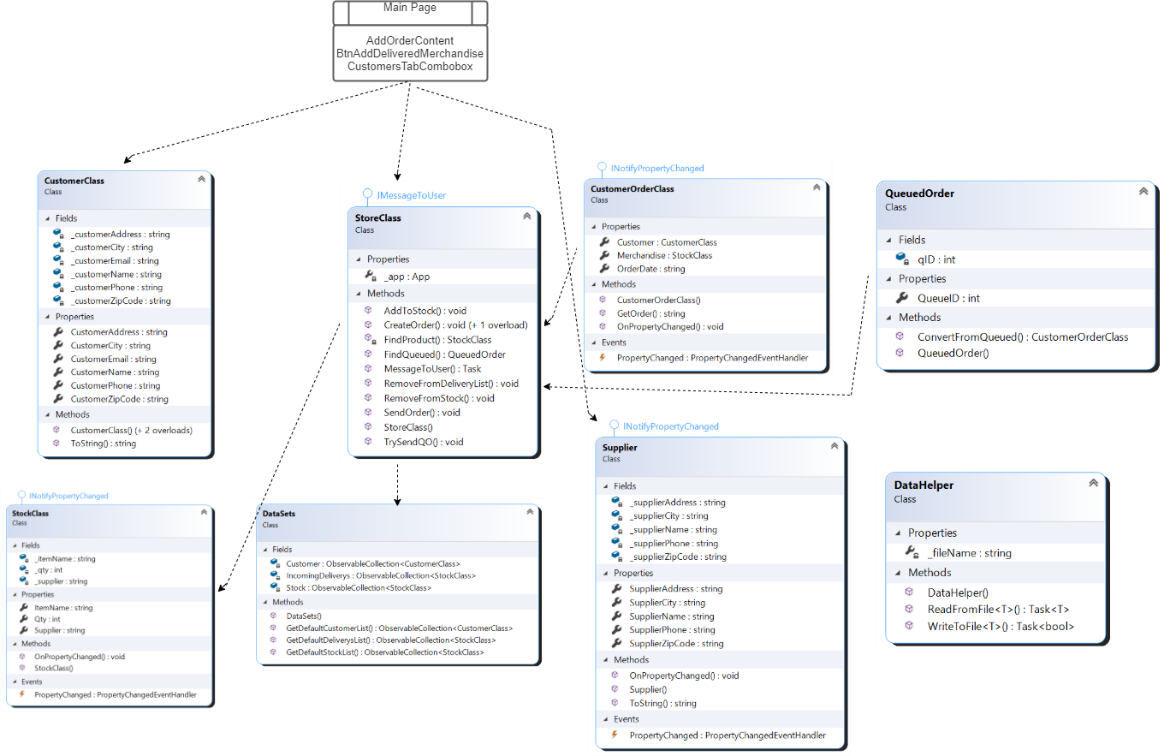
* En lista över nuvarande kunder
* En vy för att lägga till en ny order för en kund
* En lista över alla nuvarande produkter
* En vy som listar alla produkter och hur mycket som finns på lager
* En vy för att kunna hantera en leverans av produkter till lagret

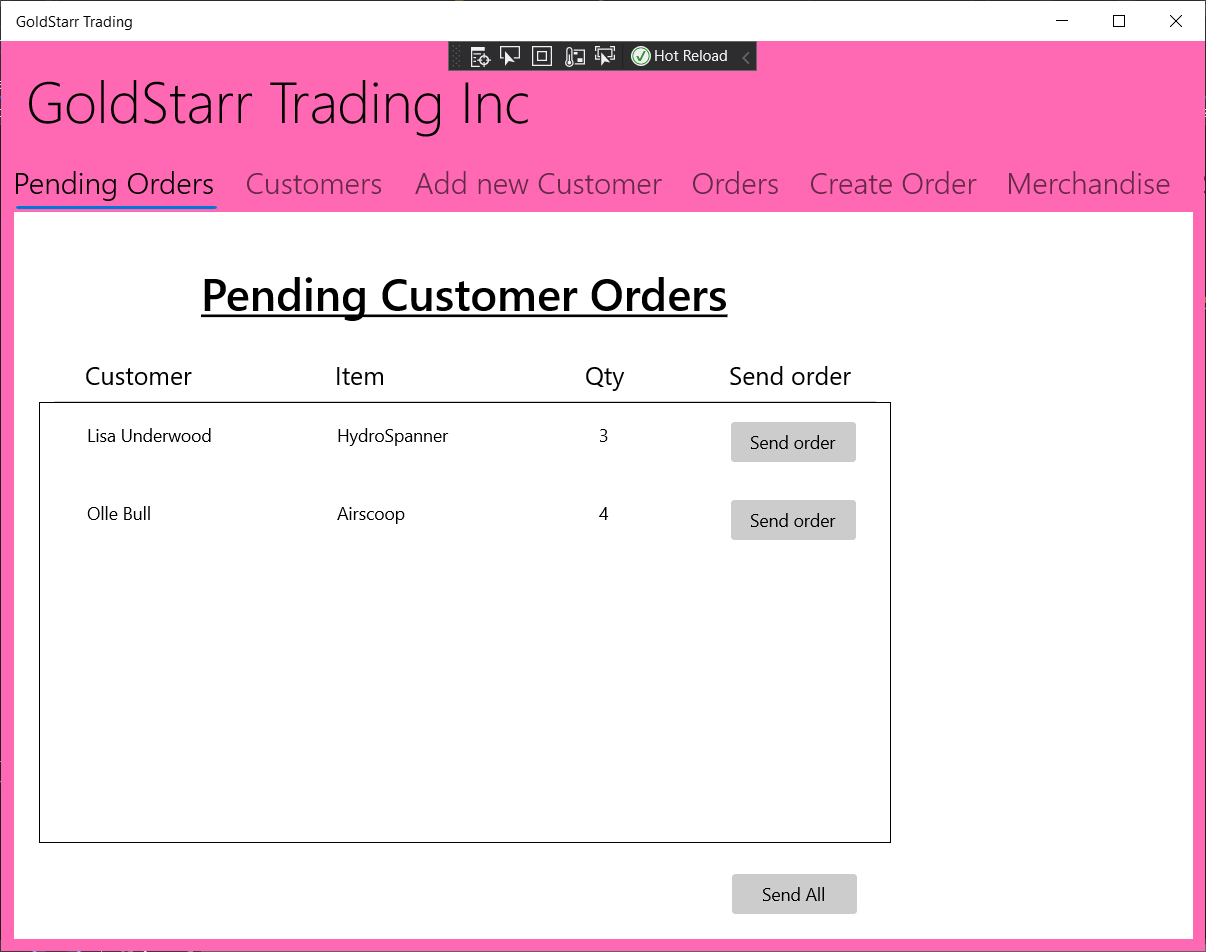
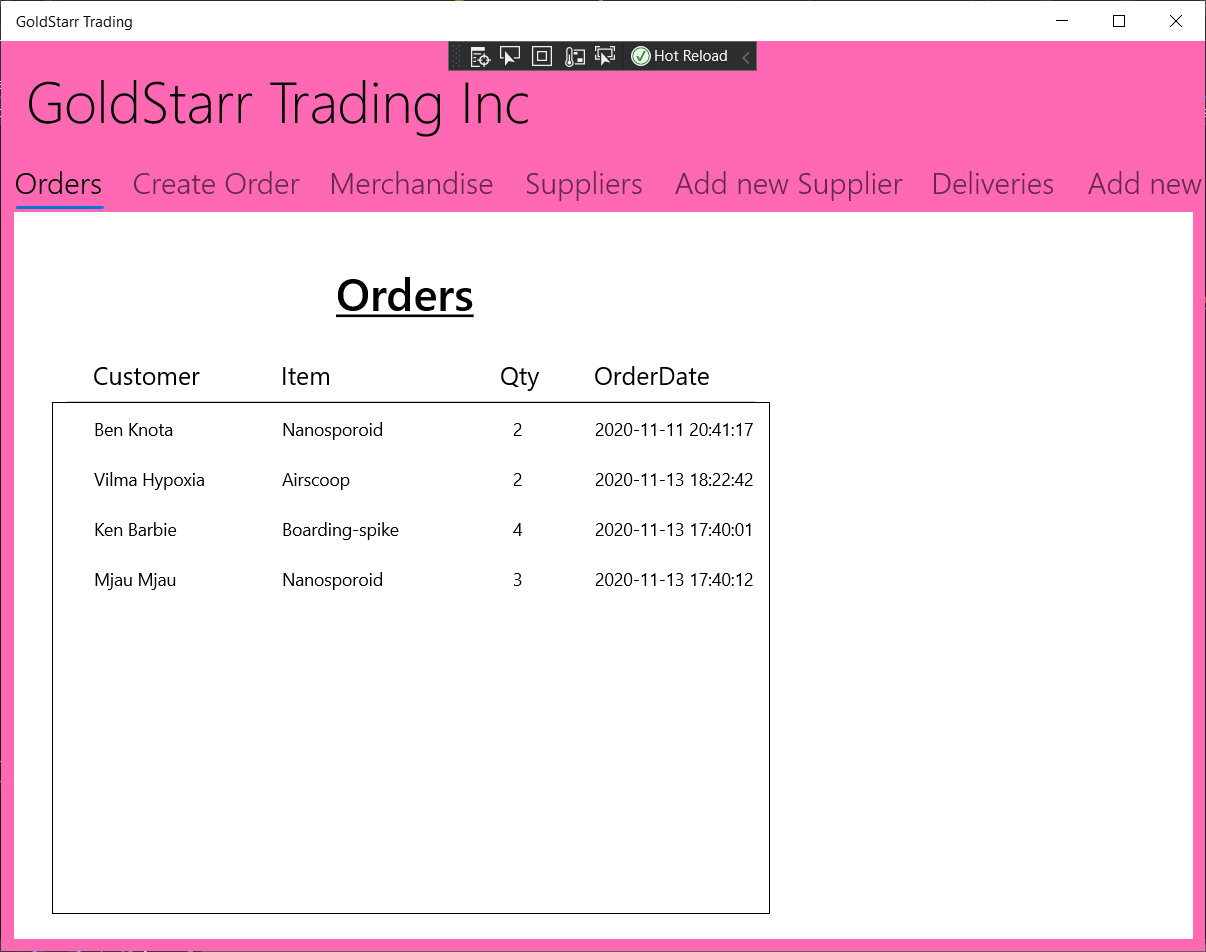
Då företaget har för intention att expandera i framtiden finns en del element att ta hänsyn till. Den applikation som utvecklas bör rimligtvis kunna skalas upp allteftersom företaget expanderar och fler funktioner behövs för applikationen. Det är därför av stor vikt i planeringsfasen att skapa ett användargränssnitt och klassindelning där det tagits hänsyn till att fler funktioner kan tillkomma. Det är av stor vikt att klassindelningen och hur koden byggs upp underlättar för att kunna addera funktionalitet till applikationen.  
  
För att få översikt på fördelningen av metoder i applikationen har ett UML-diagram skapats för att se över beroenden i klasserna. Ingen av klasserna har ärvt från varandra men det går att urskilja beroenden från de olika klasserna vars metoder tar del av andra klassers innehåll.  
  
  
UML-diagram över applikationen  
  
  
För att skapa ett så användarvänligt gränssnitt som möjligt har vi valt att skapa flikar som tydligt är benämnda med vad som finns i respektive flik. Detta för att minimera antalet klick för användaren att utföra sin uppgift och enkelt kunna navigera sig genom applikationen. Nedan syns användargränssnittet för applikationen  
  
   
Meny för att se befintliga kunder  
  
Första fliken *customers* visar företagets kunder. Överst finns en kombobox där användaren kan se de kunder som finns. Genom att klicka på respektive kund kan användaren få fram uppgifter om kunden i fråga.  
  
Under fliken *create order* kan användaren genom två komboboxar välja en kund samt välja den produkt som kunden önskar köpa. Till höger om denna kombobox kan användaren skriva in det antal som kunden önskar köpa. När användaren trycker på *add* kommer ett pop up fönster dyka upp som berättar vilken kund ordern lagts på, vilken produkt och antal. Skulle det vara så att lagret inte är tillräckligt så kommer ett felmeddelande berätta att ordern inte går att genomföra.  
  
  
Meny för att lägga en order.  
  
Pop-up meddelande för att bekräfta att orden har gått igenom.  
  
Under fliken *merchandise* visas lagret för de produkter som företaget säljer. När en order skapas så uppdateras lagret utefter den order som skapats.  
  
  
Lagerstatus för alla varor  
  
Under fliken *Delivery´s* visas de produkter som kommit in som ska läggas till i *merchandise*. Under *Qty* visas det hur många produkter som levererats och under *add Qty* adderar användaren dessa till sitt lager, vilket uppdaterar antalet i *merchandise*. De kan vara så att företaget fått fler produkter än vad de beställt eller av någon anledning inte vill ta in alla inkommande produkter i lagret. Användaren har då möjlighet att inte addera hela inkommande beställningen till lagret. Skulle användaren försöka lägga till fler produkter till lagret än vad som kommit in så kommer ett felmeddelande att dyka upp som visas nedan.

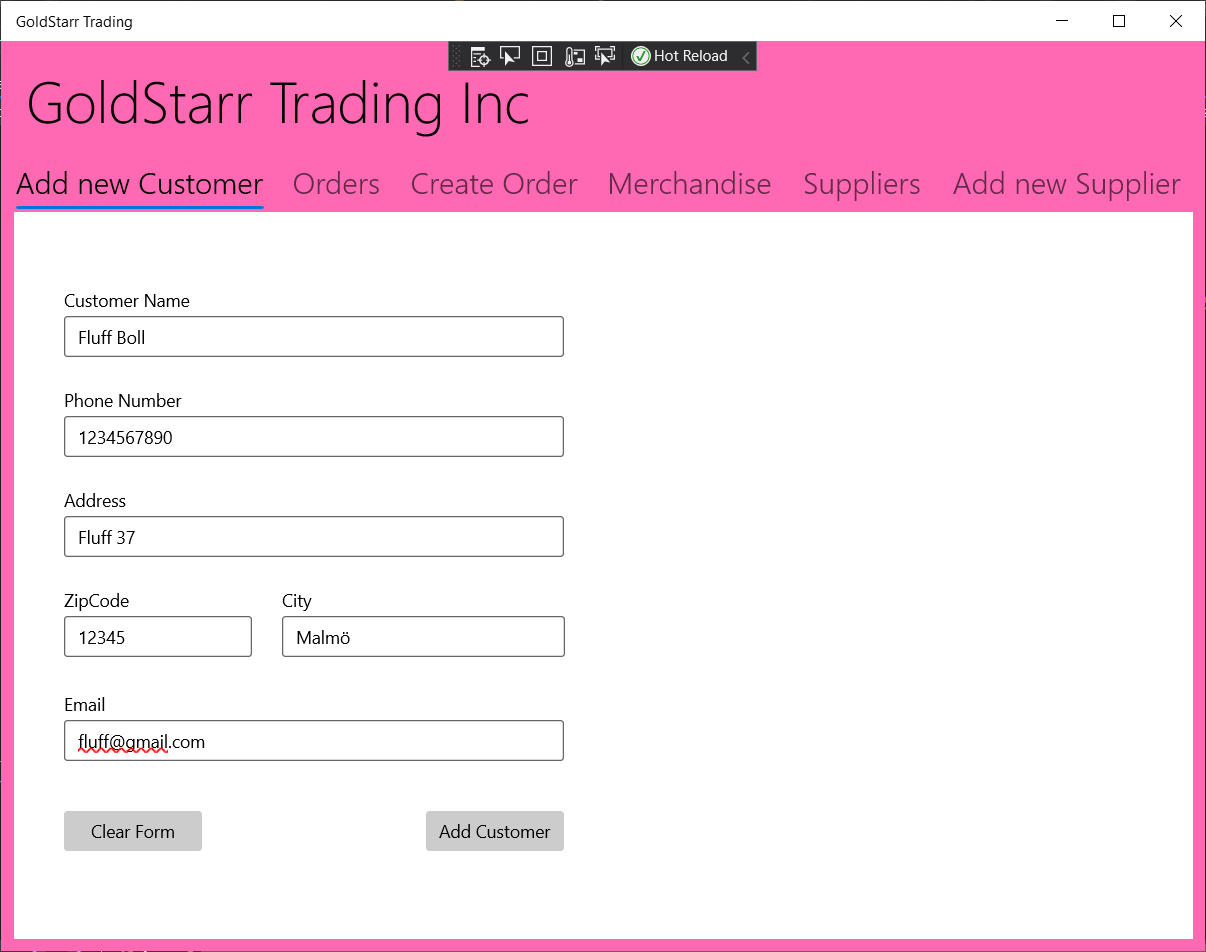
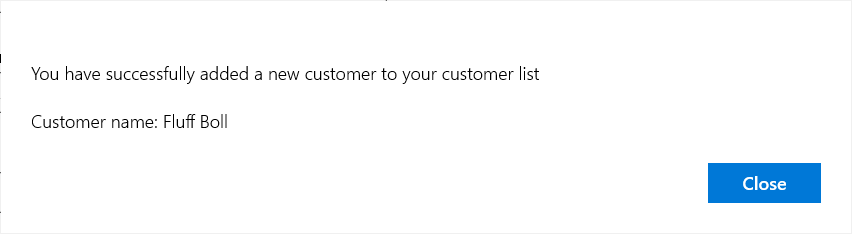
   
Menyn för att ta emot de leveranser som kommit in och lägga in dem i lagret.  
  
Felmeddelande då användaren försöker lägga till fler produkter än de som har levererats  
  
  
**Release 2**Inför release 2 har arbetsgruppen fört en dialog med Project owner vad gäller möjligheterna för att bygga vidare på applikationen. Efter kontinuerlig kommunikation beslutades det att följande punkter skulle implementeras inför release 2.

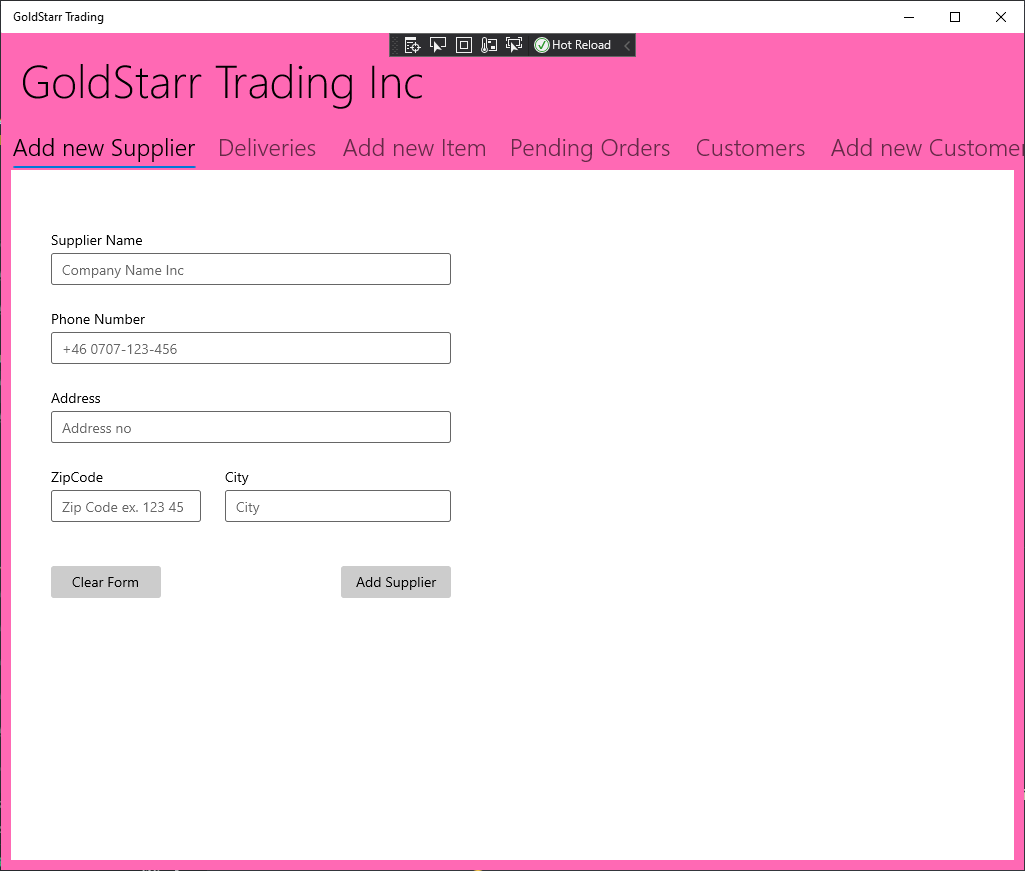
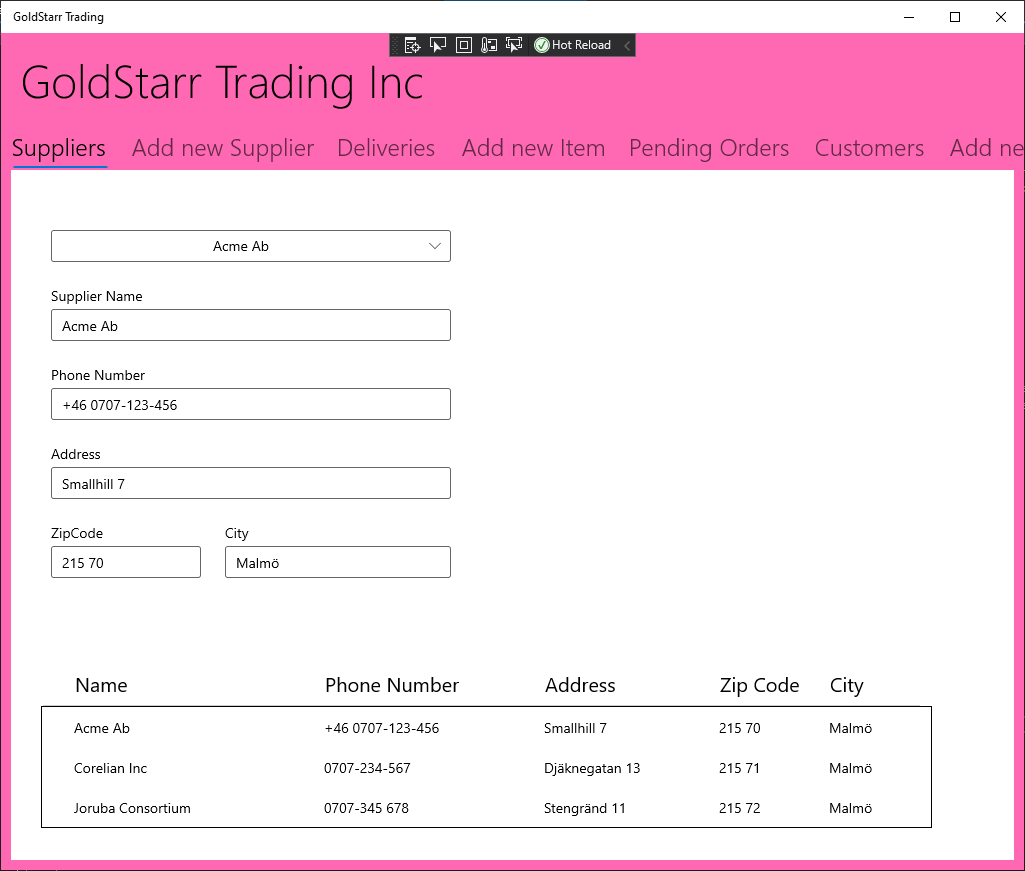
* En kunds ordrar ska sparas i systemet
* En vy för att se alla ordrar
* All data ska spara på fil och läsas in vid programstart
* En kö för ordrar som inte kan levereras om lagret är tomt
* En vy för att skicka köade ordrar när lagret har fyllts på
* En vy för att lägga till nya kunder
* En lista över leverantörer
* En vy för att lägga till nya leverantörer
* En vy för att lägga till nya produkter
* Utöka kundlistan med onlinekunder för den kommande online-affären

Under projektets gång insåg arbetsgruppen att tiden inte räckte för att skapa en *vy för att lägga till nya produkter*. Då denna inte går att kvalitetssäkra inför leverans bestämde arbetsgruppen att denna inte kommer med i release 2, om än den var någorlunda färdig. Denna funktionalitet skulle i sådana fall bli aktuell att ta med i en senare sprint om detta vore ett större projekt.   
  
Upplägget för release 2 var snarlikt upplägget i release 1 med att skapa en WBS över de aktiviteter som skulle utföras för att sedan använda oss utav ett gantt-schema för att få en uppfattning om hur lång tid som behövdes för respektive moment.  
  
  
WBS över release 2  
  
  
Efter att WBSen skapats och aktiviteterna planerats skapades det nya klasser som behövdes för release 2. Nedan visas klassindelningen för applikationen. Denna är i stora drag densamma som release 1, med ytterligare 3 klasser; en klass för att hantera köade ordrar, en för att hantera leverantörslistan och lägga till leverantörer samt data helper för att spara ner programmet till fil. De befintliga klasserna har även de fyllts med nya metoder, bland annat för att kunna lägga till nya kunder.

  
UML-diagram för release 2

Vad gäller de vyer som implementerats så har vi valt att använda samma koncept som i release 2 med flikar för att hålla programmet så enhetligt som möjligt.  
  
Ny funktionalitet för release 2 är bland annat en *orderflik.* I denna flik samlas ordrar där det står vem kunden är som ordern lagts på, vad för produkt denne har beställt, antal av produkten samt klockslag. Skulle det vara så att det inte finns tillräckligt många produkter i lagret kommer ordern skickas iväg till *pending orders*. I *pending orders* kan användaren skicka beställning till leverantör för den produkt som kunden önskar.  
  


Vy för skickade ordrar till vänster. Till höger visas de ordrar som inte kan skickas då lagret inte är tillräckligt.  
  
Nedanstående bild visar Fliken *Add new Customer* är en påbyggnad av *Customer* från release 1 där det även går att lägga till nya kunder. I denna går det även att lägga till en mailadress för att kunna ha med onlinekunder i listan.  
 

Det går även att lista de leverantörer företaget köper in produkter av samt lägga till nya leverantörer  
  
  
Vänster bild visar lista över leverantörer. Vyn har även en drop down meny för att kunna söka upp rätt leverantör. Höger bild visar vyn för att lägga till leverantör.

Slutsatser och avslutande diskussion

Under projektets första fas har gruppen levererat en applikation för Goldstarr trading att kunna lista sina kunder och hantera kundernas orders. Applikationen innehåller en lagerstatus som uppdateras utifrån den order som lagts och de leveranser som kommer in. Arbetet har utförts genom att förhålla sig till ett agilt arbetssätt där alla moment brutits ned i mindre delar. Gruppen försökte arbeta utifrån scrum-metodiken med dagliga möten och jobba agilt. Då vi inte är helt insatta i metodiken kring Scrum så har inte arbetssättet fungerat fullt ut. Projektets omfattning har inte heller varit optimalt för att använda Scrum då projektet bara varit 3 veckor och en sprint optimalt är 2-4 veckor. Det har dock varit givande att ha tankesättet kring Scrum i arbetet för att arbeta agilt. Arbetet har varit lärorikt för att få förståelse för alla moment i utveckling av ett projekt alltifrån bryta ner uppgiften, skapa ett användarvänligt GUI samt implementera kod. Det hade varit önskvärt att utveckla en planering kring testing av programmet, vilket vi inte har gjort. programmet har testats sporadiskt utan mätbara och specifika krav på testningen. Vi har även fått större förståelse för hur viktigt samtalet med Project owner är då en kravlista är statisk. Det kan finnas svårigheter i att tolka dessa och det är därför viktigt att ha samtal kontinuerligt med project owner för att säkerställa att kunden får det denna efterfrågar och önskar.  
  
  
Från release 2 tog vi lärdom av att det inte alltid går att estimera hur mycket som kommer att hinnas med på utsatt tid. Oförutsägbarheter kan uppstå och aktiviteter kan både ta längre tid än tänkt eller kortare tid än vad som är estimerat. Det viktigaste här är att hålla en dialog med project owner. Krav kan även ändras under projektets gång och då är det viktigt att lyssna på dessa.

Referenser

Jacobsen D. I. & Thorsvik J. (2018). *Hur moderna organisationer fungerar*. Studentlitteratur, Lund

Sommerville, I. (2016). *Software Engineering* (10:th Edition), Addison-Wesley