GIK299 Projektrapport

Grundläggande information

Författare: Dennis löv
Fredrik Jansson

Kurskod: GIK299

Grupp: 38

E-post: h25denlo@du.se
h25frjan@du.se

Datum: 9 Jan 2026

Projekt: Alternativ C; Shoppingapp.

Syfte

Syftet med uppgiften är få lära oss jobba i grupp i ett lite större programmeringsprojekt och lära oss hur man använder simpla datastrukturer, kontrollstrukturer, klasser, UML-diagram och flödesscheman.

Projektval och målgrupp

Vi har valt att göra en shoppingapp där det säljs elektronik, målgruppen blir då elektronik intresserade och generellt 18 - 50 år. Vi utgår från att en grundläggande teknisk kompetens finns hos användaren, men vi ska se till att appen är så lätt och tydlig som möjligt att använda då webshoppen ska kunna användas av vem som helst.

User stories

Lägga till rabatt/hantera produkter som admin:

- Skapa och logga in på admin konto
- Redigera specifika varor
- Ange rabatt i procent beräknat programmet nytt pris automatiskt
- Originalpris ska fortfarande sparas och inte ändras

Kundinloggning:

- Databas över användare
- Spara leveransinformation

Sökfunktion:

- Funktionalitet för att kategorisera produkter
- Sökfunktion genom namn och kategori

Rekommenderade varor:

- Para ihop varor (om kund väljer A rekommendera B, inte vice versa)
- Presentera rekommenderade varor för kund i gränssnittet

Egna tolkningar, överväganden och val

Vi har tolkat de givna user stories i instruktionen som programmets grundfunktionalitet och kommer att implementera dem. Våra user stories ser vi sen som extrafunktioner. Vi har försökt välja praktiska funktioner som är vanliga i riktiga webshoppar idag. Vi har även försökt att inte välja alltför lätta funktioner att implementera, då syftet med denna uppgift både är att lära oss nya saker samt att visa upp de färdigheter vi har.

Användning av AI-verktyg (t.ex. ChatGPT, Copilot)

• Vilka frågor ställde ni? Vilka delar av uppgiften var svårast och hur hjälpte AI-verktyget er?

AI har använts till och från under projektets gång. Då syftet med projektet är att vi ska utveckla våra kunskaper så har vi försökt använda AI så lite som möjligt, och inte låtit AI verktygen sköta själva tänkandet eller problemlösandet åt oss. Vi har inte bett AI skriva koden åt oss, utan det används istället för att besvara specifika frågor samt som ett värdefullt verktyg vid debugging.
När vi har stött på olika fel har AI varit väldigt användbart för att förklara orsakerna bakom felen samt hur de kan förebyggas. Ofta har vi bett AI att tyda error loggarna, och förklara i simplare termer, samt bett om hjälp i att hitta mer svårupptäckta logikfel. Ofta är dessa väldigt svåra att hitta (ett exempel på ett logikfel var användandet av user istället för User i en rad som ledde till problem) så här har användandet av AI besparat oss mycket tid.

AI har även varit väldigt användbart för att förklara koden som ens partner har skrivit när man ska in och ändra kod som den andra skrivit, så man har full förståelse av hur allt sitter ihop vilket förebyggt mycket fel.

• Hittade ni fel eller brister i lösningarna? Hur rättade ni dem?

Vid många tillfällen har AI misstolkat vad man är ute efter och gett förslag som är helt inkorrekta. Vi har dock en god nog förståelse av programmering för att upptäcka dessa uppenbara fel och avfärda dem. Oftast har AI också kommit med ideer och lösningar som är nästan korrekta, men som krävt en del manuell justering för att passa in i programmet och fungera på just det sättet som vi velat.

• Jämförde ni med andra källor eller klasskamrater?

Lösningar och svar har jämförts med diverse information från google och stackoverflow, men inte med andra klasskamrater.

• Hur tror ni att användningen av AI påverkar er förmåga att lösa programmeringsproblem självständigt?

Användningen av AI har nog haft en positiv inverkan på förmågan att lösa programmeringsproblem självständigt då den under projektets gång förklarat mycket i simpla termer vilket gett förståelse som vi kommer att ta med oss i framtiden. Att analysera AIs lösningar och svar har även utvecklat vår förmåga att granska kod och se vad som kommer funka eller inte funka. Vi hade absolut klarat oss utan AI om det skulle behövas, men AI har snabbat upp processen för att hitta den information man behöver.

Metod

Vi bestämde oss för att programmet skulle vara en konsolapplikation för att det är vad vi är mest bekväma med, vilket lät så stor andel av vår tid som möjligt att gå till logiken och strukturen av programmet istället för att krångla med någon fancy GUI. Vi har sen försökt att separera ut koden i klasser på ett logiskt sätt, i den mån det gått.

Design

Programmet består av en huvudmeny där man som användare kan välja att se alla produkter, söka på produkter, logga in, se din varukorg, visa din information och uppdatera din information. Som admin så kan man också välja att uppdatera andras information, skapa nya användare, ta bort användare, och visa alla användare.

Klassen ShopUI sköter all UI, till en början skulle den bara sköta det men det blev svårt att separera UI från logiken och om man ska göra ändringar i UI så kan man behöva hoppa en del i koden så ShopUI blev det som också skötte hela program loopen. De andra klasserna vi delat upp programmet i är följande:

Program:
Huvudprogrammet. Startar applikationen, initialiserar alla nödvändiga komponenter (produkt inventarie, inloggningssystem, användarhantering) och kör huvudmenyloopen (ShopUI).

LoginSystem:
Sköter allt som har med själva inloggningen av användare att göra, autentisering av användare, samt registrering av nya användare och uppdatering av användarinfo.

User:

Representerar en användare i systemet. Innehåller användarinformation som användarnamn, lösenord, leveransadress och en varukorg. Har metoder för att kontrollera om användaren är admin, lägga till produkter i varukorgen och kontrollera om en produkt redan finns i varukorgen.

UserRepository:
Lagrar och hämtar alla registrerade användare. Fungerar som en enkel databas för användardata.

UserManager:
Innehåller all funktionalitet för att låta admins skapa nya användare, ta bort befintliga användare och uppdatera användarinformation.

Product:
Representerar en produkt i butiken. Lagrar produktegenskaper som namn, pris, lagersaldo och kategori. Innehåller även funktioner för att hantera rabatter (reducerat pris) och rekommenderade sammankopplade produkter.

ProductCategory:
En enum som definierar de tillgängliga produktkategorierna: Gaming, Audio, Storage och Accessories.

ProductManager:
Ett administratörsverktyg för att hantera produkter. Låter admins skapa nya produkter, ändra befintliga produkters egenskaper (namn, pris, kvantitet, kategori) och tillämpa rabatter på produkter.

Samarbete

Vi samarbetade över discord och jobbade mestadels tillsammans i ett samtal med lite ändringar utanför och Git för versionshantering.
Finns inga Trello skärmdumpar för projektet tog 3 dagar att göra klart och vi såg det här när projektet va i princip klart. Vi har istället haft en löpande kommunikation genom discord där vi tillsammans sett över vad som behövt göras, samt varit tydliga med vad vi jobbar/tänkt jobba med för att inte vara och ändra i samma del. Har även stämt av efter varje git push över discord för att säkerställa allt fungerar som det ska samt att förklara de ändringar eller additioner som gjorts till programmet.

Testning

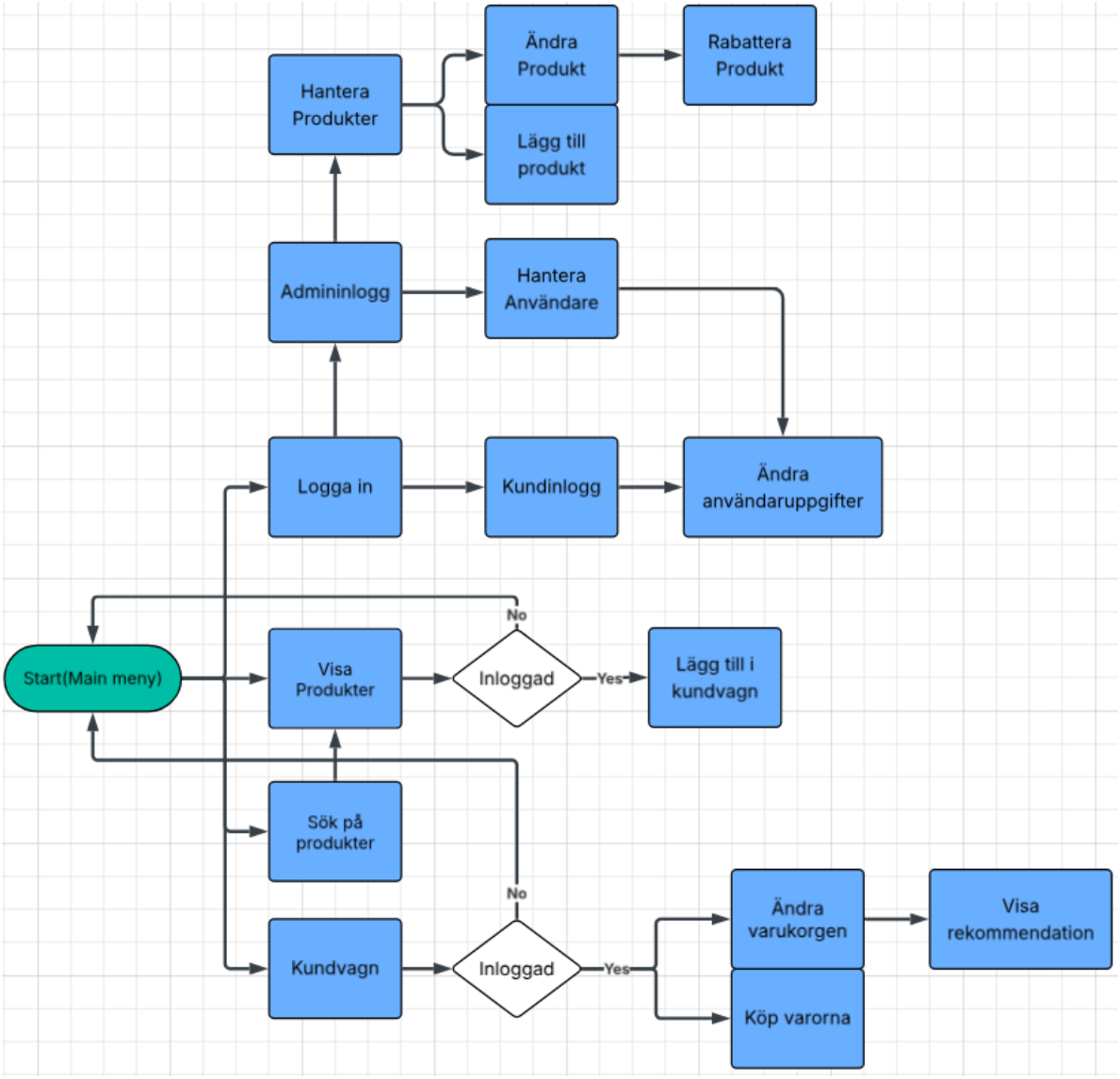
1. Normal användare – korrekt användning enligt instruktion.
Korrekt användning testades under hela utvecklingen för att säkerställa att allt fungerade.

2. Djävulsk användare – medvetet felaktig användning.
Medvetet felaktig input testades också under utvecklingen med mer djupgående och helhetstäckande testning i slutet av projektet.

3. Mormor-test – låt en utomstående testa appen och notera användbarhetsproblem.
Kompisar fick tillgång till koden och körde den själva medans vi satt i discord och kollade på, inga stora problem hittades förutom att det är lite svårt att läsa alla produkter samtidigt men det blir lite så när man kör i kommandotolken.

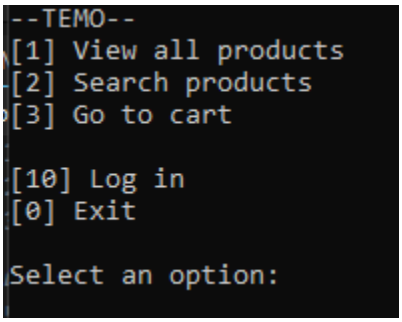
Arkitektur och UML/Flödesscheman

Vi började första dagen med att planera ut grovt hur vårt program skulle se ut/fungera genom att skapa följande UML samt flödesscheman. Detta hjälpte enormt då det gav oss en slags mall eller checklista att jobba efter, och det var alltid tydligt vad man kunde jobba med samt hur man skulle gå tillväga. Programmet har följt dessa scheman ganska precist, och vi kan inte se några tydliga sätt som programmets funktionalitet avviker från hur dessa scheman är upplagda.

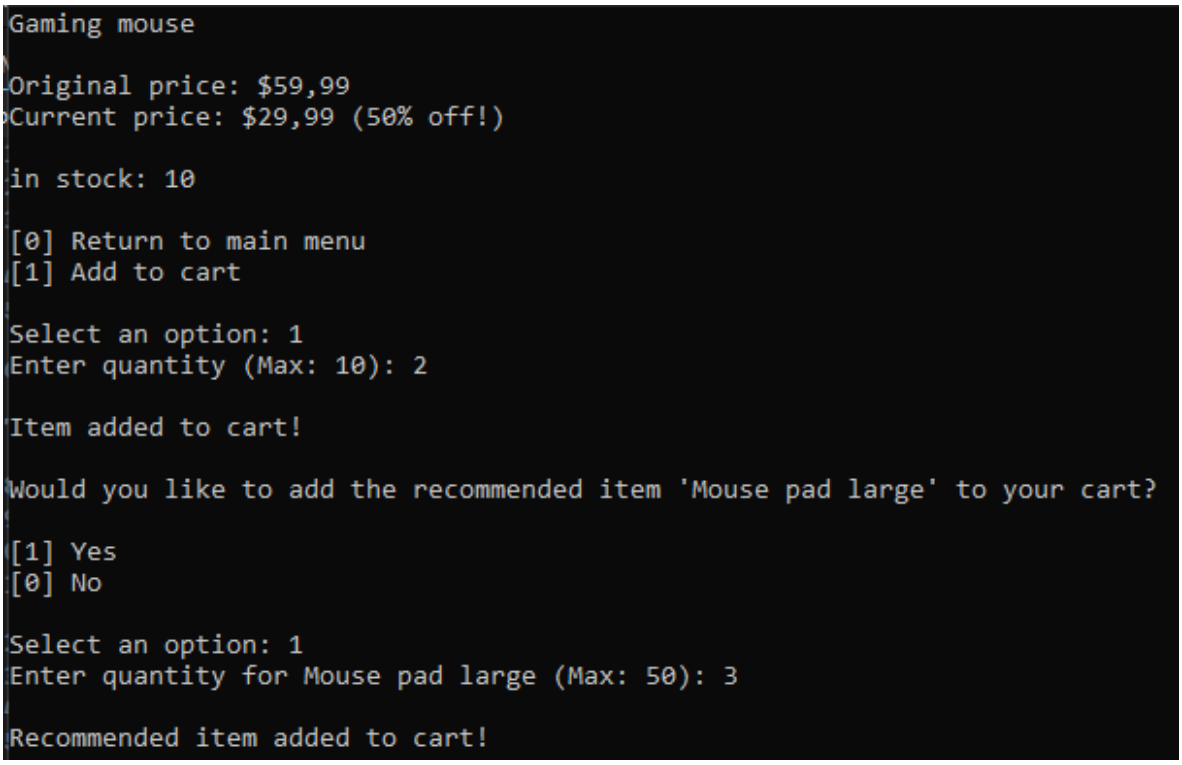

Hantera Produkter
Ändra Produkt
Rabattera Produkt
Lägg till produkt
Admininlogg
Hantera Användare
Logga in
Kundinlogg
Ändra användaruppgifter
Start(Main meny)
Visa Produkter
Inloggad
No
Yes
Lägg till i kundvagn
Sök på produkter
Kundvagn
Inloggad
No
Yes
Ändra varukorgen
Visa rekommendation
Köp varorna

Resultat

Huvudmenyn



Vara och rekommendation läggs till i kundvagnen



Varukorgen

Reflektion och slutsats

Det gick bra att samarbeta över discord fast vi hade lite problem med git och branches i början. Första tanken var att göra programmet med windows forms men denna idé slopades snabbt då det kändes som att mer tid skulle gå till att krångla med forms än till själva programmeringen och problemlösningen.

En simpel state machine hade gjort det enklare att separera ut koden mer så inte allt ligger i ShopUI. Användardatan skulle även kunna hanteras säkrare för bättre sekretess, men det kändes lite överkurs för detta projekt.

Appendix

Parprogrammeringslogg

Namn och Datum	Tid (h)	Ensam eller i par	% vid tangentbordet
Dennis och Fredrik 12/1	6	Ensam (I discordsamtal)	båda 100%
Dennis och Fredrik 13/1	3	Ensam (I discordsamtal)	båda 100%
Dennis 14/1	7	Ensam	100%
Fredrik 14/1	8	Ensam	100%