Kaisar Lafi Alhasson, presenterar denna redovisning och dess innehåll omfattar min deltagande roll i projektet. Uppgiften har gjort den 20-12-2023.

Repo länk : https://github.com/SturessonAdam/HangMan.git

Alla urklippar , är med texten som hyperlänk.

I denna skriftliga redovisning reflekterar jag över min roll i Iron pants gruppen i en agil utvecklingsprocess med fokus på kundvisionen på en variant av Hänggubbe spel. Utvecklingen sker Agilt genom veckovisa kundmöten.
Genom denna redovisning delar jag insikter om projektet, inklusive kundens vision, utmaningar, och min personliga reflektion kring den Agila strukturen.

Kundens vision ger projektet riktning och kontext, i kundens vision strävade vi efter att digitalisera det traditionella hänggubbe spelet och ge det en multiplayer funktion.
Initialt utvecklade vi en prototyp som innehöll alla de klassiska hänggubbe funktionerna. Vår tanke var att skapa en upplevelse där flera spelare kan öppna spelet med olika skärmar, och utmana varandra genom att spela mot varandra, och spelet har möjligt att förslå ett ord att spela med.

Efter ett kundmöte och en demonstration av prototypen insåg vi möjligheten att förbättra spelet, visionen ändrades till att inkludera sex identiska fönster inom samma integrerade utvecklingsmiljö (IDE). Detta skulle göra möjligt för samtidiga matcher och skapa en mer intensiv och samarbetsinriktad spelupplevelse. Den omformulerade visionen var ett resultat av strävan att optimera spelupplevelsen och göra det möjligt för fler deltagare att njuta av spelet samtidigt.

Tre utvalda users stories belyser projektets dynamik. Här beskrivs vem som implementerade dem:

Den första, User story, visualisering av hänggubbe gubben, som användare vill jag kunna se hur många försök jag har kvar och se gubben hängas på skärmen.
Gaukhar, som har kodat och implementerat denna metoden med koden. Här är en länk till metoden, straff. Vi behövde göra en justering efter code review.
Vi bytte från 0–9 till 1-10 för att få det att stämma överens med försök räknare och gubben.

Den andra, User story multiplayer funktion, implementerat via Adam och Kaisar har ändrat lite att metoden kommer att fungera fel fritt.
Som användare ska jag möta kundens behov av att kunna spela spelet mot andra i en multiplayer funktion, multiplayer.png, här är ändrade och nya metoder, initMultiplayerGame, multiplayer, och att skapa en map.

Den tredje, User story välja ett eget ord eller slumpa från lista, som användare vill jag kunna välja ett eget ord att spela med (och göra det dolt) eller låta spelet slumpa ett ord till mig, denna har implementerat by Kaisar,wordList.png, list.png.

på varje möte och code review diskutera vi , om hur kan vi leverare en färdig produkt med möjligt att leverera extra funktioner, utan att påverka vårt huvud mål, så ni har lagt till en singelplayer , ett fönster i början att välja vilken mode singel eller multiplayer. Då har uppdaterat vi våran Kod med dem ändringar, singelplayer, (1). png , singelplayer, (2). png ,singelplayer, (3).png.

Virtuallisering:

En översikt över de terminalsteg som krävs för att skapa och navigera genom mappstrukturen i en Ubuntu miljö, ladda ner projektet från GitHub, ger en djupare förståelse för arbetsprocessen, här steg för steg va har gjort :

Navigera till mappen "ideaprojects" /mnt/c/Users/cezar/ideaprojects
Skapa mappen "hangman": mkdir hangman
Navigera in i "hangman": cd hangman
Klona GitHub-repo till den nuvarande mappen: git clone
<https://github.com/SturessonAdam/HangMan.git>
Sen vi har skapat branscher i denna repo DEV , Adam , kaisar och Guakhar . mm
Har skapat kaisar branch:
git branch kaisar
git checkout kaisar

Avslutning:

I gruppmötena använde vi oss av Kanban-metoden för att hantera och organisera vårt arbete.

1-Hantering av Backlog:
Vi använde Trello för att hantera vår Kanban-baserade backlog. Det tillät oss att enkelt lägga till, prioritera och följa upp uppgifter. Backlog-enheten användes som en dynamisk lista där vi kontinuerligt lade till och flyttade uppgifter baserat på deras prioritet och status.

2-Arbetsfördelning:
Arbetsfördelningen bestämdes under våra dagliga möten. Vi diskuterade vilka uppgifter som behövde göras och vem som kunde ta på sig dem. Genom att använda Kanban kunde vi också visualisera arbetsbelastningen och se till att ingen överbelastades eller hade för få uppgifter.

3-Evaluering och Återblick:
Vi höll regelbundna retrospektivmöten för att reflektera över vårt arbete. Vi identifierade framgångar och utmaningar. Vid dessa möten diskuterade vi vad som gick bra och vad som kunde förbättras.

4-Code Reviews:
Vi använde oss av regelbundna code reviews för att säkerställa kvalitet och gemensam kodstandard. Kritik som gavs var konstruktiv och fokuserad på förbättring. Det skapade en miljö där alla kände sig engagerade och bidrog till att höja kvaliteten på koden.

5-Påverkan på Gruppen:
En tydlig påverkan av vårt agila tillvägagångssätt och Kanban-metoden var ökad samarbete. Det skapade en miljö där alla kunde se arbetsflödet och bidra till diskussioner om förbättringar och effektivitet.

6-Svårhanterade Problem och Ändringar:
Vi identifierade att vissa uppgifter hade längre kötider än förväntat. För att hantera detta beslutade vi att prioritera och omfördela resurser. Vi genomförde också mindre justeringar i arbetsprocessen för att öka effektiviteten baserat på våra retrospektivdiskussioner. Det här är ett exempel av ett problem som vi löste:

Vi hade lite krångligt med att starta multiplayer spel, så länge vi starta ett spel på den nästa skärmen, förgående skärm , sluta att fungera. Vi har diskuterat mkt kring detta , till slut vi har en lösning efter vi har definierat problemet. Problemet var I första handelMultiplayerClick uppstod problemet med att flera instanser av Main delade på samma variabler när de hanterade multi spelarscener. Detta ledde till oönskat beteende, där ett spel som startades senare påverkade tidigare instanser på ett sätt som inte var avsett.

att förhindra att flera spelinstanser delar på samma variabler och grafiska element. I vårt fall gjordes detta genom att skapa en ny instans av Main klassen för varje multiplayer-spel.
Multiplayer-spelet, användes new main() för att skapa en ny instans av Main i handelMultiplayerClick
En ny metod initMultiplayerGame använder vi för att initiera inställningar specifika för varje multiplayer-skärm och för att koppla scenen med den nya instansen av main. Bilder är länkat med user story 2.
Närsomhelst då det krävdes så mötas vi på Discord utöver våra dagliga digitala standups och delade ändringar som gjordes.

God jul och Gott nytt år!