Team Reflection - Week 5

Group 3

11 maj 2020

1 Kundvärde och omfång

1.1 Omfång av applikationen

Vårt mål är att skapa en MVP vars funktionalitet täcker de mest grundläggande behov som digitalt scorekort. Detta innebär att det skall vara lätt att navigera fram och tillbaka mellan menyer, information och hål samt att de funktioner som räknar och byter hål är intuitiva. Efter personen i fråga har gått sina bestämda antal hål skall ett traditionellt scorekort kunna visas upp där varje spelare ska kunna se hur många slag som dem har haft på varje hål. Denna produkt har vi redan skapat och använder denna som bas för lägga till fler funktioner som slutanvändaren önskar. I den sprinten som var jobbade vi med att försöka göra appen så simpel som möjligt. Dock, efter att slutanvändaren har använt appen framgick det att vi behöver göra gränssnittet ännu tydligare.

En funktion som slutanvändaren ville att vi skulle utöka är möjligheten att se historik för den individuella spelaren. Så att personen i frågan kan följa deras utveckling över flera golfrundor.

1.2 User stories / Job stories

Vi valde tidigt att använda *job stories* istället för *user stories*. Vi anser att det var ett bra beslut då den precision som job stories ger har hjälpt oss i arbetet. Under sprint planeringen börjar vi med att utvärdera den *job story* som ligger längst upp i product backlog. Den bryts ner i tasks vars effort estimeras med vårt poäng system. Under de senaste veckorna har våra estimat blivit betydligt bättre och felar inte lika mycket med verkligheten. Efter att efforten för varje task i *job storyn* har estimerats summeras poängen och läggs till *job storyn*. Har vi poäng över för sprinten görs processen om för den *job storyn* som nu ligger överst i product backlog. Detta arbetssätt anser gruppen har fungerat väldigt bra.

1.3 KPI

För att säkerställa att ingen gruppmedlem hamnar på en icke hållbar stressnivå har gruppen tagit fram nya KPI:er. Utöver att efter varje sprint mäta skrivna rader kod och slutförda tasks, mäter vi även hur nöjda gruppmedlemmarna varit med *kommunikationen, arbetsbördan, stressnivån, arbetsprocessen* samt *tillfört värde*. Kanske hade ett tidigare införande av dessa KPI:er motverkat att den student som lämnade gruppen verkligen hade gjort det. Det är så klart svårt att säga, men en lärdom vi alla drog var att det är minst lika viktigt att se efter hur laget mår, och inte bara se till hur det presterar.

1.4 Acceptans tester

För närvarande har vi krav om att innan merge till master måste varje branch gå igenom flera steg för att utföra kodtester och manuella tester av GUI:t.

Dessa steg är utformade på följande vis:

1. Kör alla enhetstester för projektet.
2. Testa knappen Börja ny golfrunda”
3. Välj en bana
4. Lägg till nya spelare
5. Öka och minska antalet slag för en spelare
6. Återgå till huvudmenyn, undersök ifall rundan syns bland oavslutade rundor

Genom att följa dessa steg kan varje person säkerställa att funktionalitet inte brutits av förändringar som gjorts i utvecklarens feature branch. Allt eftersom features läggs till förändras denna process för att täcka de test-behov som applikationen har, och ifall scopet skulle utöka.

Då vi använder Android Studio som utvecklingsmiljö får dessutom utvecklaren en notifikation vid commit. Denna notifikation informerar om det finns varningar och felmeddelanden i koden vilket skall hanteras innan merge till master branch.

2 Socialt kontrakt och insats

2.1 Spenderad tid

Att behöva göra helgarbete är naturligtvis något som ökat gruppens stressnivåer något men vi har hanterat det väl och fortsatt leverera det viktigaste varje sprint. Detta är dock något som förväntas bli bättre då de största deadlinesen i övriga kurser passerar efter nästa sprint.

3 Designbeslut och projektstruktur

Designbeslut har skett demokratiskt under respektive morgonmöte (standups) och detta har hittils fungerat väl. Samtliga beslut är nedskriva i olika dokument som finns i vår Teams-grupp. Dokumenten i fråga är: Kodkvalitet, Designbeslut kod, Grafisk design och UX samt Testning innan merge med master.

3.1 Kodkvalitet och kodstandarder

Vi har aktivt valt att inte använda oss av peer-reviews för att spara tid och effektivisera arbetet. Detta har hittills fungerat väldigt bra eftersom vi är ett relativt litet team (5 personer). Deltagarna ber dock om råd och tips om denne känner att det är användbart. Som kompensation för att inte använda en formell peer-review så har vi upprättat en formell checklista för testning som måste passera (Dels unit tester och dels manuella UI-tester).

Nedan beskrivs ytterligare metodik och dokument som vi använder för att säkerställa god kodkvalitet.

Kodkonventioner
Vi följer de standarder som rekommenderas av Google i Google Java Style Guide [1]. Praktiskt innebär detta att vi försöker förhålla oss till dessa standarder för att få en enhetlig och mer lättläst kod.

Testning
Alla modellklasser skall vara 100% testade. Då det är svårt att testa GUI, eller åtminstone inom den tidsram och med de kunskaper som teamet besitter, har vi valt att ersätta detta med manuella tester av UIt för att försäkra oss att all funktionalitet fungerar korrekt. Vi har därför upprättat ett dokument med en checklista för testning, varav samtliga delar i listan skall genomföras och uppfyllas innan en branch får mergas med master.

Generella riktlinjer
Koden skall i största möjliga mån nyttja arkitekturen och verktygen som finns i Google Jetpack [2]. Detta innebär praktiskt att androidx biblioteket nyttjas, dvs att vi använder de senaste stödbiblioteket från Google för Androidutveckling.

Design mönster
Vi använder i huvudsak MVC vilket i Android har XML filer som View, activity som controller och vår modell är separerad från bägge utan cirkulära beroenden. Utöver detta så tillämpar vi även MVVM pattern där det är användbart, samt repository och DAO patterns för datalagring.

3.2 Vilken dokumentation vi använder och varför

Av extern dokumentation så använder vi främst Androids officiella developer sidor [3], samt deras föreslagna stack Jetpack [2] och den tillhörande arkitektur som finns där.

För intern dokumentation så har vi skapat JavaDocs samt kommentarer för all kod som är relevant, exempelvis är generiska getters och setters exkluderade. Vi har även skapat ett flödesdiagram som beskriver hur navigeringen i appen skall ske mellan olika activities/sidor och vilken knapp som leder var, samt olika UML-klassdiagram som beskriver hela modellen både övergripligt och i detalj, och ett ER-diagram som beskriver databasen.

3.3 Hur vi använder och uppdaterar dokumentationen

Dokumentationen vi har som syfte att agera projektbokföring och etablering av standarder. Då vi använder oss av Microsoft teams som ett av våra kommunikationsmedel medföljer fildelning och filstrukturering. Rent praktiskt innebär detta att vi har olika mappar för relevant dokumentation. Dokumentationen uppdateras löpande vid behov, för att bidra både till kunskapsdelning och etablering av standarder inom projektet. Detta kan ta formen av designdokumentation, mötesprotokoll, projektbeskrivning eller GUI skisser.

Referenser

[1] Google, *Google Java Style Guide*, 2020. URL: `https://google.github.io/styleguide/javaguide.html` (hämtad 2020-05-07).

[2] ——, *Android Jetpack — Android Developers*, 2020. URL: `https://developer.android.com/jetpack` (hämtad 2020-04-24).

[3] ——, *Android Developers*, 2020. URL: `https://developer.android.com/` (hämtad 2020-05-07).