2024-10-30

Projektuppgift

Dt071g – Programmering i C# .NET

Applikation - TimedMath

Markus Vickman

MITTUNIVERSITETET
Avdelningen för informationssystem och -teknologi

Författare: Markus Vickman, mavi2302@student.miun.se
Utbildningsprogram: Webbutveckling, 120 hp
Huvudområde: Datateknik
Termin, år: HT, 2024

Applikation - TimedMath
Markus Vickman 2024-10-30

Sammanfattning

Rapporten beskriver arbetets gång med att skapa en applikation med syfte att lära ut matematik. I applikationen ska matematiska tal lösas och användaren får poäng för dessa. Efter en viss tid tar uppgifterna slut och användaren får se sin totalpoäng som då sparas som ett high-score i en fil. Desto fler lösta uppgifter desto svårare tal måste användaren lösa.

Utvecklingen gjordes i ramverket .NET MAUI med programmeringsspråket C# och styling gjordes i märkspråket XAML. MAUI är ett multiplattforms ramverk och med hjälp av det skapades en applikation som fungerar bra på flera olika plattformar men som specifikt testades på och utvecklades för Android och PC.

Eftersom att XAML märkspråket var nytt för mig avgränsades applikationens design och styling till en enkel men funktionell design.

Resultatet blev en enkel applikation för Android och Windows där huvudräkning av matematik tränas. Applikationen kompilerades till en opackad version för Windows och en intern testversion publicerades även på Google play store.

Nyckelord: C#, XAML, MAUI, .NET.

Innehållsförteckning

1 Introduktion

1.1 Bakgrund och problemmotivering

Matematik är ett viktigt ämne i skolan och alla har nytta av att vara snabba på huvudräkning av enklare tal. För att bli snabb på huvudräkning krävs mycket upprepning.

Därför ska en matematikapplikation skapas. Valt programmeringsspråk är det objektorienterade programmeringsspråket C#[1] och ramverket .NET MAUI användas för att applikationen. .NET MAUI är ett ramverk för att skapa applikationer med ett grafiskt interface och det fungerar på flera olika plattformar[2]. Applikationens fokusplattformar ska vara Android och Windows.

1.2 Avgränsningar

Fokus för uppgiften är vid C# koden och vid MAUI för möjligheten till multiplattforms publicering. Tiden för design och styling ska begränsas eftersom märkspråket XAML är nytt för mig. Design och styling ska vara enkel men funktionell. XAML är ett märkspråk som används av men inte uteslutande av .NET MAUI[3].

Applikationen ska utvecklas för både Android och Windows. Eftersom att tiden är begränsad räcker fungerande prototyper av applikationen och den behöver därför inte publiceras till Google Play Store eller Windows Store.

1.3 Detaljerad problemformulering

För att applikationen ska vara användbar för att öva på matematik kommer först flera problem att behöva lösas.

Tydlig UI

För att fånga användarens uppmärksamhet behöver applikationen vara tydlig och självförklarande.

Matematik frågor

Användaren ska lösa slumpmässiga tal som ska adderas, subtraheras, divideras eller multipliceras.

High-score på tid

För att sporra användaren till att vilja köra applikationen flera gånger ska poäng ges för rätt svar. Totalpoängen ska sparas i ett High-score så att användaren kan tävla mot sig själv eller andra på samma enhet.

Val av räknesätt

Användare ska kunna välja räknesätt om behov finns att specifikt träna ett räknesätt.

2 Teori

C#

C# är ett objektorienterat programmeringsspråk som även innehåller delar av funktionellprogrammering. Det är ett så kallat ”general purpose language” som är tänk för att fungera till flera olika plattformar.[1]

.NET MAUI

.NET MAUI är en del av .NET plattformen. MAUI står för ”Multi-platform App UI” och är till för att binda ihop C#-kod med märkspråket XAML. .NET MAUI fungerar på en mängd olika plattformar.[2].

XAML

XAML är ett märkspråk utvecklat av Microsoft och står för ”Extensible Application Markup Language”. XAML fungerar lite som en blandning av HTML och CSS då både styling och innehåll styrs i XAML-koden.[3]

3 Metod

Applikationen ska programmeras i programmeringsspråket C# och utvecklingen ska göras i Visual Studio med ramverket .NET MAUI. Märkspråket XAML är standard språket för design och innehåll i .NET MAUI och därför ska XAML användas för styling och innehåll.

Metod för att lösa problem enligt problemdefinitionen:

Tydlig UI

En grafisk design för UI ska ritas upp i Figma. Designen måste vara responsiv för att minska plattformsspecifikt anpassad kod. Layouten ska vara enkel för att inte förvirra användaren. Stylingen ska kodas i XAML.

Matematik frågor

En metod behövs för att slumpa tal för olika räknesätt och dessa tal ska bli större desto fler poäng användaren har. När talen division ska slumpas ska en uträkning ske med modulus för att kontrollera att svaren inte bli decimaltal. Ett fält behövs för att läsa in användarens svar från skärmen.

High-score på tid

Metod för att ladda in high-score från fil och skriva ut på skärmen behöver skapas. Även en metod för att spara high-score behöver skapas och då ska det också sorteras efter resultat och endast de 10 bästa resultaten ska sparas. Ett fält för att skriva in namnen på användaren som sedan sparas i high-score behöver skapas.

Val av räknesätt

För att välja räknesätt behöver valknappar finnas synliga för användaren. Dessa ska kodas i XAML. En metod som läser in valda räknesätt från skärmen ska också skapas i C#.

3.1 Sammanställd metodlista

1. Wireframe och grafisk design i Figma
2. Applikationens flödesschema
3. Förverkliga design i XAML
4. Applikationens funktionalitet i C#
5. Publicering
6. Användartester

4 Konstruktion

4.1 Wireframe och grafisk design i Figma

En wireframe skapades i Figma vilket går att se i bilaga 1. Layouten gjordes centrerad för att den skulle passa in på flera olika enheter och plattformar. Den hölls också enkel för att ge tydlighet för användaren och för att inte för mycket tid skulle läggas på XAML-koden.

En ikon skapades i webbverktyget Adobe Express Logo Maker[4] genom att välja minimal stil, ange applikationsnamnet ”TimedMath” och den beskrivande texten ”Math speedtest”. Därefter valdes en logotyp och slutgiltig logotyp redigerades i den inbyggda redigeraren.

Utifrån wireframen skapades en grafisk design i Figma som går att se i bilaga 2. Färger valdes där bakgrundsfärgen sattes till ”FDFDFD” även kallat ”whitesmoke”. För knappar valdes .NET MAUIs standard färg för knappar som är en blålila färg ”6A00FF”. Knapparna fick rundade hörn och startknappen breddades för att ge ett effektfullt intryck.

4.2 Applikationens flödesschema

Eftersom applikationen nu hade en tydlig idé och design där stor del av funktionaliteten framgick kunde programmets flödesschema utgå ifrån dessa. Flödesschemats utgick från programmets start och flödesförloppen som sker när start/stoppknappen aktiveras vilket går att se i bilaga 3.

4.3 Förverkliga design i XAML

Först skapades ett nytt .NET MAUI projekt i Visual Studio genom att helt enkelt välja nytt ”.NET MAUI app” och sedan namnge det till ”TimedMath”.

Ikonen lades till i mappen ”\TimedMath\Resources\Images” med namnet timed_math.png.

Utifrån den grafiska designen skapades ett label-element för varje statisktextrad i filen ”MainPage.xaml”. **För varje knapp, inmatningsfällt och valknapp skapades ytterligare element på följande sätt:**

<Image>
Source valdes till ”timed_math.png”.

<Button>
För att knappen skulle kunna initiera metoden för start och stopp lades attributet Clicked=”OnStartClicked” till. Den fick även texten ”Start” och x:Name=”StartBtn”. Namnet ”StartBtn” behövdes för att texten på knappen skulle kunna ändras för att visa timer och ändra texten till ”Stop” direkt ifrån C#-koden.

<VerticalStackLayout>
En layout skapades med attributet x:name=”choices” för att kunna dölja och

visa valknappar beroende på om applikationen är i ”startläge” eller ”stoppläge”.

I denna layout placerades två Switch-element. En för val om tidtagning ska vara aktivt och en för om användaren endast vill träna multiplikationstabellen. Det skapades även fyra CheckBox-element som valknappar för räknesätt. Alla dessa Switch och CheckBox-element namngavs i x:Name=”” för identifiering från C#-koden.

<Label>
Detta label-element används för att skriva ut matematiska frågor till skärmen och därför fick det identifierbara attributet x:Name=”question”. Attributet IsVisible=”False” lades också till för att labeln endast ska visas när applikationen är i ”startläge”.

<Entry>
Sedan skapades ett inmatningsfält för att svara på frågorna och även den fick attribut för synlighet. Elementet fick även attributet för att kalla på en metod med TextChanged=”OnEntryTextChanged” för att svaret skulle kunna kontrolleras vid varje ny inmatad siffra.

<Button>
En knapp skapades med texten skip och attributet Clicked=”OnSkipClicked” för att initiera skip-metoden.

<Entry>
Fältet för att skriva in namn till high-score fick identifieraren x:Name=”enterName”. Den tilldelades även Completed=”SubmitAnswerClicked” för att enter knappen ska initiera spara till high-score metoden.

<Button>
Knappen för att spara till high-score hänvisades till samma metod som inmatningsfältet ovan Clicked=”SubmitAnswerClicked”. Den fick identifieraren x:Name=”EnterNameBtn”.

<Label>
Som anslutning av XAML-dokumentet skapades ett label-element för att skriva ut high-score till skärmen. Elementet fick identifieraren x:Name=highScore.

4.4 Applikationens funktionalitet i C#

Applicationen utgår ifrån filen MainPage.xaml.cs. Eftersom applikationen endast är en sida ligger hela programmet i klassen ”public partial class MainPage : ContentPage”. Den partiella klassen skapades automatiskt när .NET MAUI-projektet skapades. C#-koden delades upp över fyra sidor där alla sidorna började likadant för att tillhöra samma kodbas:

```
”namespace TimedMath {
   public partial class MainPage : ContentPage {”
```

4.4.1 MainPage.xaml.cs

I denna fil startar applikationen genom constructorn till MainPage-klassen. I constructorn körs först metoden InitializeComponent som initieras komponenterna som definieras i XAML-koden och kopplas samman interfacen med en bakomliggande C#-koden. Constructorn initierar även metoden LoadHighScore som läser in och skriver ut high-score på skärmen. Förutom constructorn kodades även följande variabler, klasser och metoder:

Private string answer
Variabel som lagrar svaret som användaren matat in på skärmen. Variabeln går att nå från hela applikationen.

Private int totalPoints
En integer skapades för att lagra de totala poäng användaren har.

Public class Player
En klass skapades för spelar-objekt. Dessa objekt innehåller en integer för poäng och en sträng för spelarens namn.

private void OnEntryTextChanged(object sender, TextChangedEventArgs e)
En metod skapades som var kopplad till svarsfältet för frågorna. Denna metod jämför om inmatat sträng är samma som strängen answer. Om det är rätt svar får integern totalPoints plus 1 i värde och metoden för att framställa mattefrågor initieras ”ChangeLabel”.

private void OnSkipClicked(object sender, EventArgs e)
Det skapades sen en metod kopplad till skip-knappen som vid klick endast initierar metoden ”ChangeLabel”.

private void MultiplicationTableActive(object sender, EventArgs e)
Sista metoden i filen MainPage.xaml.cs är skriven så att den ändrar IsVisible till false för räknesättsknapparna om de redan är aktiva. Är knapparna redan dolda ändras IsVisible till true istället.

4.4.2 startandstopp.xaml.cs

private bool checkIfPressed = false;
Filen startandstopp.xaml.cs inleds med en bool variabel för att avgöra om startknappen är intryckt eller ej.

private CancellationTokenSource cts;
Sedan skapades en CancellationTokenSource med namnet cts för att kunna avbryta fördröjningen i metoden OnStartClicked. En cancellation token används för att avbryta en pågående uppgift[5].

private async void OnStartClicked(object sender, EventArgs e)
En metod skapades som initieras av start/stopp-knappen. Metodens logik byggdes så att den med if-satser kontrollerar om frågorna redan har startat med hjälp av bool checkIfPressed.

Om ”checkIfPressed = false” döljs och visas element för ”startläget”. Här kontrolleras också hur brytaren för multiplikationstabellen är ställd. Om den ej är aktiverad instansieras en cancellationToken. checkIfPressed ändras till true och metoden ”ChangeLabel” initieras följt av en metod för att visa timer i startknappen ”StartButtonTimer”. Efter det i en try/catch skrevs await Task.Delay(120000, ct) för att starta en asynkron delay av metoden. I catch-frasen fångas TaskCanceledExceptions. När delayen är slut initieras metoden StopRunningQuestions(true).

Om däremot startknappen redan blivit aktiverad körs koden cts?.Cancel() för att stoppa fördröjningen. Metoden StopRunningQuestions(true) initieras även här.

Koden är skriven så att om brytaren för att spela utan timer är aktiverad så initieras ChangeLabel och checkIfPressed ändras till true. Om start/stopp-knappen aktiverades igen initieras metoden StopRunningQuestions(false).

private void StopRunningQuestions(bool timedRun)
Denna metod innehåller koden som ska köras när start/stopp-knappen aktiveras en andra gång. Metodens kod är skriven så att den återställer texter och element på skärmen till som de var när applikationen startades. Detta görs med hänvisning till x:Name i XAML koden. Exempel för att ändra texten på startknappen tillbaka till start användes StartBtn.Text = ”Start”; eftersom att x:Name på startknappen är ”StartBtn”. De element som visades igen fick IsVisible = true och de som doldes fick IsVisible = false. För att stoppa mattefrågorna och för att startknappen skulle visas som inte intryckt i logiken så fick checkIfPressed värdet false.

private async void StartButtonTimer()
Den här metoden skapades för att skriva ut tidnedräkningen i start/stopp-knappen. Koden är strukturerad så att när metoden initieras lagras aktuellt klockslag i en DateTime. Efter det startar en while-loop som loopar så länge som checkIfPressed = true. Vilket betyder att loopen avslutas när mattefrågorna har avslutats. Loopen startar med en 1 sekunders delay. I varje loop subtraheras aktuell tid med den tid som lagrades när metoden startade. På så sätt har förfluten tid räknats ut. Eftersom att mattefrågorna körs i 120 sekunder så räknades kvarvarande tid ut genom att subtrahera 120 med förfluten tid. Varje loop avslutas med att skriva ut text till startknappen på följande sätt ”StartBtn.Text = $"Stop - {secondsRemaining}"”.

4.4.3 math.xaml.cs

private string[] CheckMathMethod()
Denna metod är skapad så att den läser in läget från interfacens checkbox-rutor för de fyra räknesätten. De räknesätt som är i kryssade lagras i en sträng array. Metoden avslutar med att returnera en slumpad sträng från sträng arrayen.

private int MaxRandomNumber()
Den här metoden användes för att returnera högsta värdet för att slumpa tal till de matematiska frågorna. Högsta värdet bestäms i if/else-satser genom att höja returnera ett högre värde desto fler poäng användaren har fått.

private void ChangeLabel()
Den här metoden står för att bestämma och räkna ut mattefrågor. Först hämtas räknesätt från metoden CheckMathMetod. Sedan hämtas högsta tillåtna tal att slumpa från metoden MaxRandomNumber. En if-sats används för att avgöra om multiplikation är valt eller inte. Om inte så kontrolleras räknesätt från den hämtade räknesätts-strängen. Utifrån valt räknesätt räknas talen ut och sparas i answer-variabeln samt att frågan skrivs ut till skärmen.

För subtraktion presenteras alltid största numret först. För division så adderas ett till nämnaren innan uträkning om den är noll. För att talet alltid ska bli ett heltal kontrolleras talen i en while-loop och nya slumpas tills de inte längre ger rest vid delning med modulus på detta sätt while ((num1 % num2) != 0).

4.4.4 highscoretofile.xaml.cs

private static string FileLocation()
Metodens innehåll är endast en sträng med high-score-filens sökväg som också returnerades vid initiering.

private void LoadHighScore()
Metoden användes för att läsa in high-score från fil. En Try/Catch används för att säkerställa att filen fungera korrekt. Vid fel ersätts filen med en ny. Varje rad i high score-filen deserialiseras och läggs till i en stäng med en StringBuilder. Varje sträng som läggs in innehåller också en radbrytning för formatering. Sedan skrivs hela high-score ut till ett element på skärmen.

private void SortHighScore()
För att high score-resultaten skulle vara sorterade skapades en metod för detta. Metoden inleds med en array av Player objekt. För varje rad i high-score-filen skapas ett nytt Player objekt som lagras i arrayen. Arrayen sorteras på följade sätt: Array.Sort(players, (x, y) => x.Score.CompareTo(y.Score)). De tio bästa resultaten sparades in i en ny array som sedan JSON serialiserades och sparades rad för rad till high-score-filen. Till sist initierades metoden LoadHighScore.

private void SubmitAnswerClicked(object sender, EventArgs e)
Den här metoden initierades av submit-knappen eller av inmatningsfältet för namn till high-score. Metoden kräver att inmatningsfältet inte är tomt för att gå vidare. När metoden initieras skapas ett nytt Player objekt som Json serialiserades till high-score-filen. Sedan initieras metoden SortHighScore.

4.5 Publicering

För att publicera applikationen skapades plattforms specifika release-inställningar i filen TimedMath.csproj.

4.5.1 Android

För att signera applikationen behövdes en key i keystore. Den används för att kunna verifiera applikationen autenticitet[6]. Kommando som skrevs i terminalen går att se i illustration 1.

```
PS C:\Users\marku\Desktop\webbutveckling-ht2024\c#\timed_math\TimedMath> keytool -genkey -v
-keystore timedmath.keystore -alias timedmathAlias -keyalg RSA -keysize 2048 -validity 10000
```

Illustration 2. Skapa key till keystore.

Användarnamn, lösenord och sökväg för nyckel lades till i androidinställningarna i TimedMath.csprof. Det är också här applikationsversionen sätts och den behöver vara unik för varje ny version till Google play store.

För att publicera applikationen användes Visual Studios meny. I andra menyraden valdes Release, Any CPU och framework .net8.0 – Android. Sedan build och till sist publish selection. Applikationen skickades in till mitt verifierade utvecklarkonto hos Google för ”internal testing”. Applikationen fungerade nu på Android men användare behöver läggas till i testlistan.

4.5.2 Windows

För att få applikationen godkänd hos Microsoft store krävdes ett privacy statment på applikationens webbplats. Eftersom att detta inte fanns och inte heller tid för att skapa det så publicerades Applikationen som uppackad och startas från en exe-fil. Ikoner och namn för applikationen lades till i package.appxmanifest. Illustration 2 visar kod för publicering.

```
PS C:\Users\marku\Desktop\webbutveckling-ht2024\c#\timed_math\TimedMath> dotnet publ
ish -f net8.0-windows10.0.19041.0 -c Release -p:RuntimeIdentifierOverride=win10-x64
-p:WindowsPackageType=None -p:WindowsAppSDKSelfContained=true
```

Illustration 2. Publicering till Windows.

4.6 Användartester

Applikationen beta-testades av fem personer på sina Android telefoner. Den testades också av tre personer på Windows. Användartesterna resulterade bland annat i följande förbättringar och buggfixar:

1. Textstorlek justerades på Android och namn för räknesätt ändrades.
2. Färgtema ändrades på Android då färgerna skiljde sig från dem i emulatorn.
3. Problem med att tidtagning inte stoppade så CancellationToken skrevs om.
4. En metod för att skriva ut tid kvar i start/stopp-knappen infördes.
5. Poäng uppdateras direkt på skärmen vid rätt svar.

5 Resultat

Resultatet blev en matteapplikation för att träna huvudräkning som fungerar bra på både Android och Windows. I bilaga 4 visas inställningar och utseende för den färdiga applikationen. Applikationen innehåller valmöjligheter för användaren samt ger feedback om vad som händer. Men möter applikationen problemdefinitionen?

Tydlig UI

Applikationen har en tydlig, enkel och självförklarande UI. Tydliga namn på knappar och kryssrutor tar bort behovet av beskrivande texter. Designen lever upp till kraven då den gör det lätt att göra rätt.

Matematik frågor

Användaren presenteras med slumpmässiga tal från alla fyra räknesätt. Dessa tal ökar också i storlek desto fler poäng användaren har. Feedback ges vid rätt svar då ett poäng läggs till på skärmen och nästa fråga visas. Med detta sammantaget och att det även går att skippa frågor möter applikationen problemdefinitionen och går även ett steg längre än så med användarupplevelsen.

High-score på tid

Applikationen ger ett poäng för varje rätt svar som visas direkt på skärmen. Den lever upp till kraven då de totala poängen går att spara i ett high-score som är sorterat med högst poäng först.

Val av räknesätt

Eftersom att användaren presenteras med kryssrutor där räknesätt kan väljas möter applikationen kravet. Den innehåller även ett val för att träna huvudräkning utan tidtagning och även då visas poäng med de går inte att spara till high-score.

6 Diskussion

Resultatet för matteapplikationen nådde problemdefinitionen och jag är nöjd med utfallet. Tidsbegränsningen var inget problem för denna uppgift eftersom att tiden räckte för att skapa en fungerande applikation med .NET MAUI för både Android och Windows.

Det finns flera förbättringsområden för applikationen. Det skulle varit användbart om användaren kunde välja årskurs och på så sätt få frågor baserade på aktuell kursplan. Då hade applikationen kunnat varit ett väldigt användbar som komplement till vanlig skolundervisning.

I Förlängningen med omfattande utveckling hade applikationen kunnat utvecklats för att innehålla alla skolans ämnen med anpassat innehåll efter årskurs. Då skulle applikationen vara en stor resurs för både skola och elever.

Det var väldigt roligt att arbeta med C# .Net eftersom att det gav mig möjligheten att enkelt skapa program för desktop. Möjligheterna med C# .NET är stora då de möjliggör multiplattforms utveckling där i det här fallet .NET MAUI användes.

Källförteckning

[1] Microsoft, ”A tour of the C# language”, https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview Ändrad 2024-08-05. Hämtad 2024-10-28.

[2] Microsoft, ”What if .NET MAUI?”, https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-8.0 Ändrad 2024-06-21. Hämtad 2024-10-28.

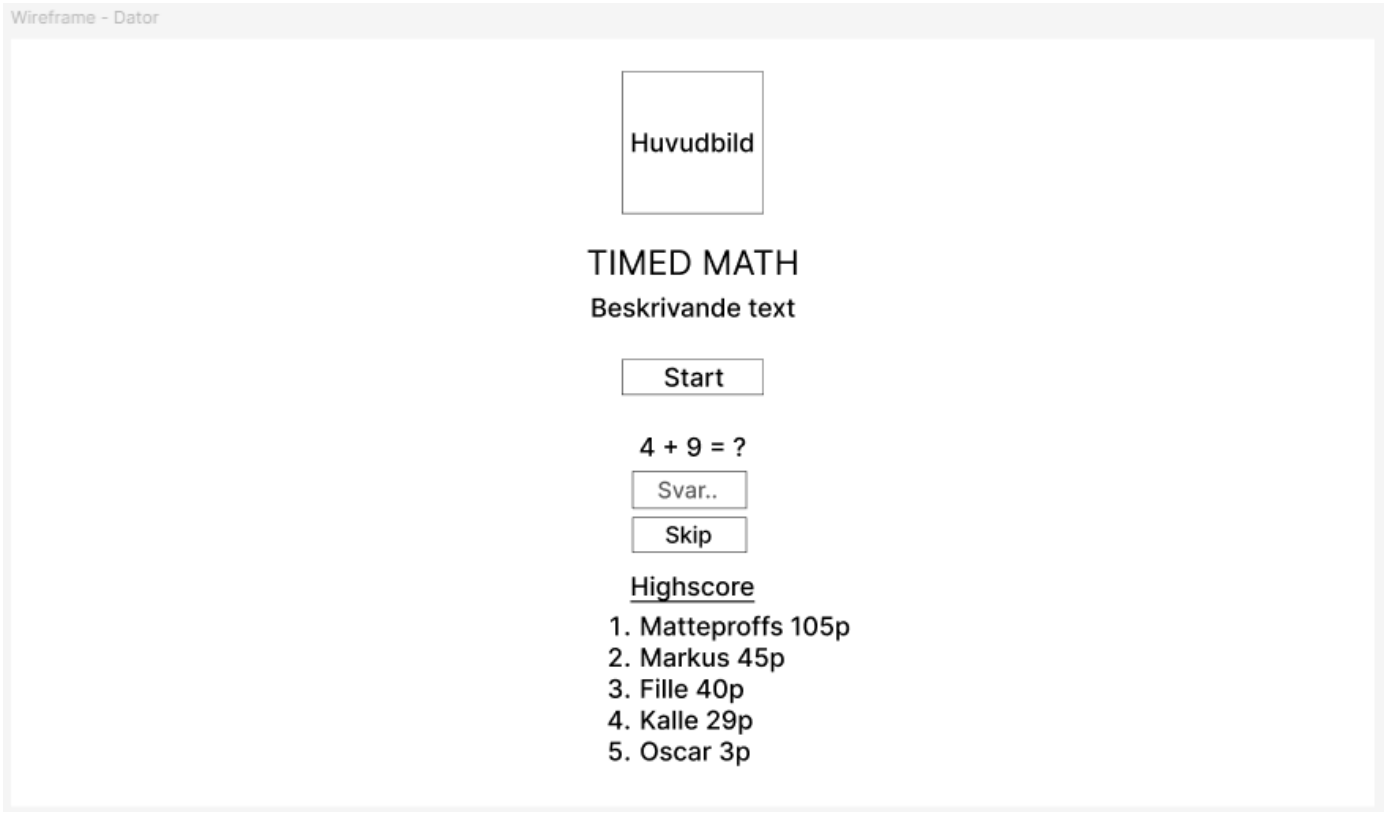
[3] Microsoft, ”Get started with XAML”, https://learn.microsoft.com/en-us/dotnet/maui/xaml/fundamentals/get-started?view=net-maui-8.0 Ändrad 2024-08-30. Hämtad 2024-10-28.

[4] Adobe, ”Free logo maker.”, https://www.adobe.com/express/create/logo Hämtad 2024-10-30.

[5] Microsoft, ”What if .NET MAUI?”, https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-cancellation Ändrad 2022-08-12. Hämtad 2024-10-30.

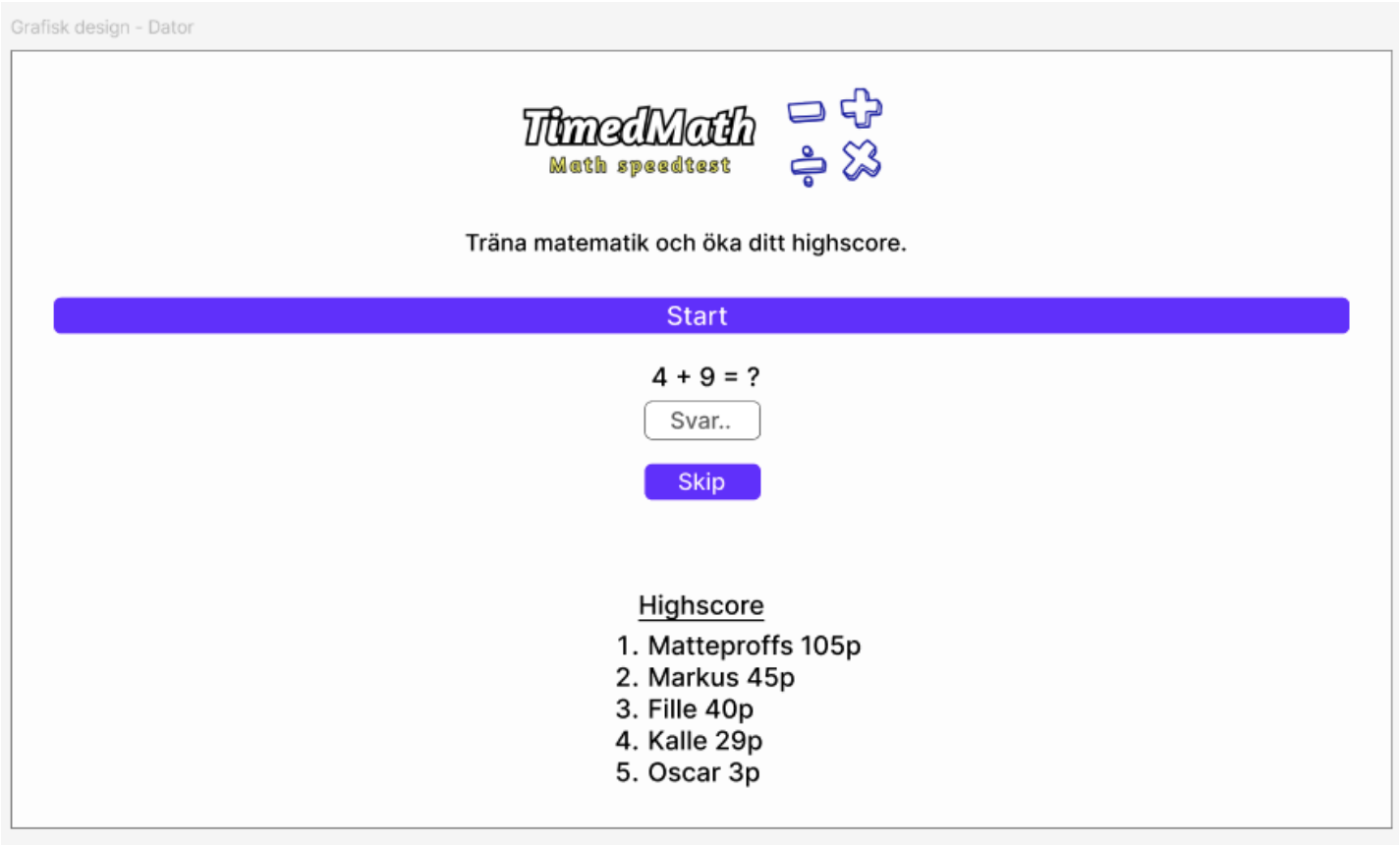
[6] Microsoft, ”Get started with XAML”, https://learn.microsoft.com/sv-se/dotnet/maui/android/deployment/publish-google-play?view=net-maui-8.0 Ändrad 2024-07-18. Hämtad 2024-10-30.

Bilaga 1. Wireframe



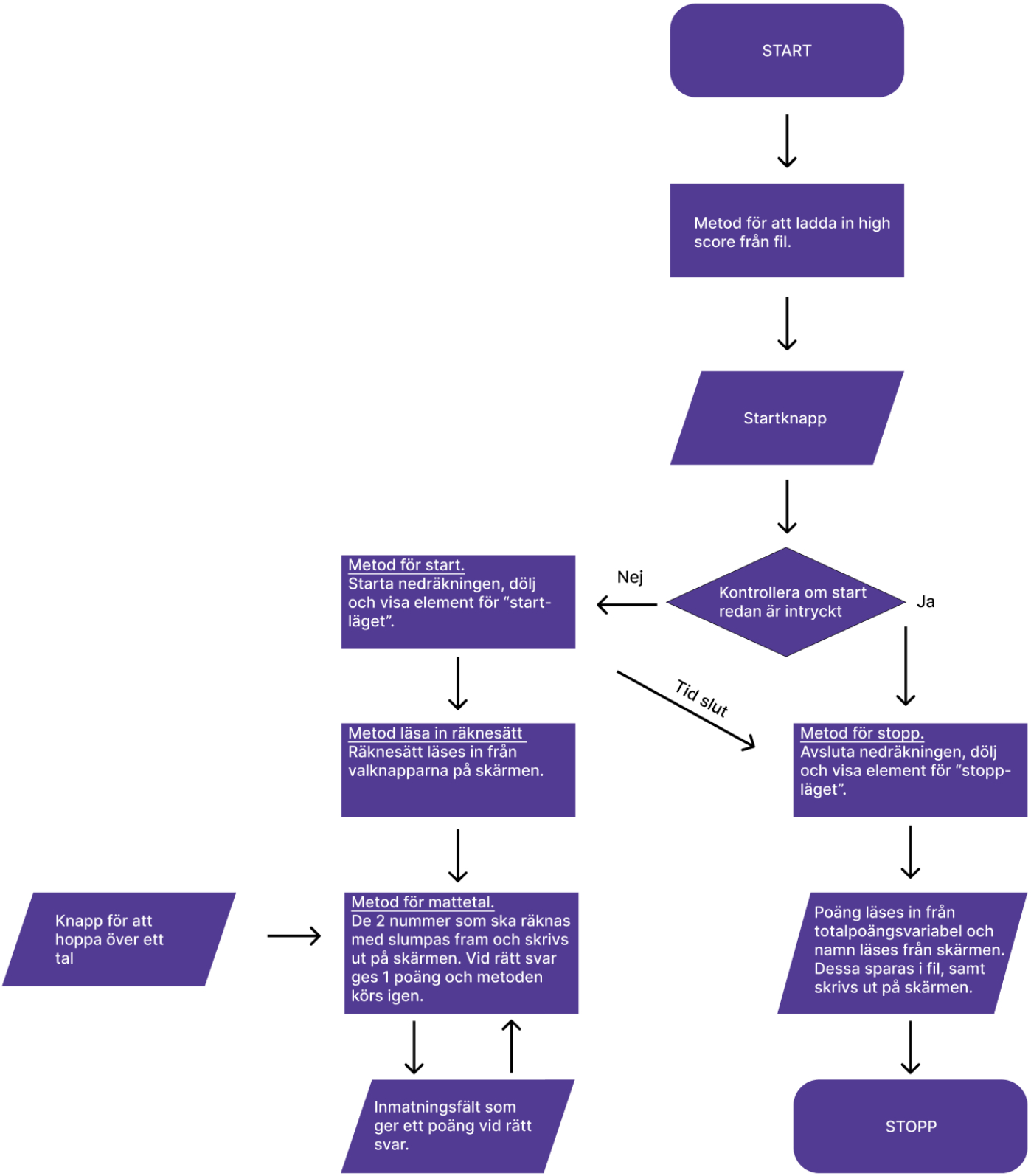
Bilaga 2. Grafisk design

Grafisk design - Dator



Grafisk design - Android

Bilaga 3. Flödesschema



Bilaga 4. Färdig applikation