MEDIEINSTITUTET

Yh

Meal Plan Helper

Helping you create weekly meal plans

Felicia Gartz Levin,
Degree Project,
FED22M,
Examinator: Martin Haagen,
2024-01-19

Table of Contents

Summary

Meal Plan Helper is a web app developed during a degree project in the Higher Vocational Education Programme Front End Developer at the school Medieinstitutet. The purpose of the app is to attempt to simplify the planning of a weekly meal menu for people who strive to have a more structured routine for such. By being able to upload meals and automatically generate a weekly menu that can also be randomized based on the ingredients that the user has at home, the app should be able to help people with this intention. Focus is set on customizability and easiness to use.

The second purpose is to be able to demonstrate several of my acquired practical skills from the Front-End Developer programme.

The web app is built with React.js and written in TypeScript and has used technologies such as Tanstack Query, React Hook Form, Zod validation, React Router and TailwindCSS as well as Firebase Auth, Database and Storage on the backend. I have learned more about all the tools. The work has been carried out during 162 hours in where I created the possibility to create a user, log in, upload dishes and generate weekly menus both with simple and advanced options. The start page has an overview of the current week and can be paginated backwards and forwards in time. The final result applies to the purpose and partially to the goal. Some features were stripped from the original goal.

Förstudie

Beskrivning

Maträttsroulette kommer att vara en webbapp deployad till Netlify där användare kan skapa en egen sida att logga in på för att spara maträtter som hen gillar för att sedan med ett enkelt knapptryck kunna slumpa fram en hel vecka med maträtter utifrån sitt eget val. Är användaren inte nöjd med någon av maträtterna kommer hen kunna redigera de en i taget. Möjligheten att generera veckomeny med preferenser för specifika dagar kommer också att finnas, då med lite fler knapptryck.
Om användaren vill redigera sin veckomeny ska hen kunna få valet att
slumpa en ny maträtt enbart den dagen
börja skriva in maträtt eller kategori i ett fält och få autocomplete på de som redan finns
kunna klicka på en knapp som leder till ett nytt fönster med https://vadfanskajaglagatillmiddag.nu/ för att enkelt kunna få inspiration
få förslag att addera en helt ny maträtt med “+ Lägg till ny” i autocomplete-listan. Då ska en modal för att lägga till maträtt dyka upp och texten man skrivit dittills ska infogas automatiskt på maträttens namn.
välja att äta ute.

Användaren kommer att efter att ha loggat in se nuvarande veckas veckomatsedel om en sådan finns. Finns den inte ska en knapp med “Generera veckomeny” finnas, liksom en knapp för att generera veckomeny med specifika preferenser. Uppe vid respektive vänster och höger hörn av utrymmet för veckomatsedeln ska det finnas möjlighet att paginera en vecka framåt, respektive en vecka bakåt i tiden.

Användaren ska kunna gå in på en sida med en paginerad översikt över alla veckor som har veckomatsedel. Hen ska kunna söka bland dem och resultat ska filtreras fram alltefter som färre saker matchar det man söker på. Klickar man på en vecka ska man komma till förstasidan och se den vecka som klickats på och kunna paginera framåt och bakåt (som man ju kan på förstasidan). Man ska överallt i appen kunna navigera bakåt och framåt bland sina historiska klick med bakåt- och framåtknapparna i webbläsaren.

Användaren ska kunna gå in på en sida med en paginerad översikt över alla maträtter. Här ska också gå att söka som ovan. Översikten ska kunna sorteras på namn åt båda håll. Man ska kunna söka bland de och få upp förslag med autocomplete. När man klickar på en maträtt i översikten ska man få upp en modal för att kunna se detaljer, redigera den eller ta bort den ur sin databas.

Två eller fler användare ska kunna ansluta till samma content. Den eller de användare som har tillgång till samma content ska kunna se varandras content och få upp denna i autocomplete.
Användare ska kunna välja om de vill ha en eller två maträtter per dag, om de har speciella matpreferenser (dvs de kategorier som recepten ska slumpas ifrån) och ev om hur lång intervall de vill ha tills en maträtt dyker upp igen.

Användaren ska hela tiden kunna komma åt sin profil och menyn uppe i ena hörnet av sin skärm. Inne på sin profil ska hen också kunna redigera profilen och ansluta andra användare som behöver godkänna detta från sitt håll.

Jag kommer att använda mig av Firebase: Firestore Database för datan och Storage för bilder att sedan länka till i datan samt Authentication.
Front-enden ska jag bygga komponentbaserat med React.js. Jag kommer att ta in de npm paket som jag anser behövs längs vägen. Bland annat kommer jag använda react-icons och säkert en del fler. Jag kommer att använda kontext där jag tillhandahåller aktuellt veckonummer, inloggad användares data och preferenser och annan data som behövs i många delar av appen. Hämtningar ska göras med hjälp av React Query.
För att rendera översikter och veckomenyer är planen att använda React Tables. Men om det blir för svårt och tidskrävande kommer jag få göra på andra sätt.
Stylingen kommer att göras med TailwindCSS.
För att lägga till nya maträtter samt skapa nya användare kommer jag använda React Forms och Zod vid behov för verifieringen.
Jag kommer att planera mitt arbete med hjälp av Trello.
Vänligen se bifogad skiss med lo-fi wireframes för att få en vidare uppfattning om user-flow.

Syfte

Alltför ofta denna höst har jag varit på väg hem efter en lång dag på LIAn och undrat vad vi ska äta. Det mesta i matväg ligger i frysen och tar lång tid att tina upp, samtidigt som jag inte vet om vi har allt vi behöver hemma till att laga någon specifik rätt. Jag saknar struktur för måltidsplanering och förberedelser och då blir det ofta att man istället köper snabbmat som är dyrare och mindre näringsrik än hemlagat. När jag och min partner har försökt göra en veckomeny med tillhörande inköpslista har det tagit så lång tid att det inte har känts värt det och inköpen har blivit dyra.
Maträttsroulette är ett försök att förenkla planeringen av veckans måltider för folk som strävar efter att ha en mer strukturerad rutin för dessa, inklusive mig själv och mitt hushåll. Genom att kunna lägga in enkla rätter och automatiskt generera en veckomeny som man också kan anpassa till vad man har hemma ämnar jag råda bot på problemet ovan.

Syftet är även att kunna visa upp flera av mina förvärvade praktiska kunskaper från Front End Developer-utbildningen.

Målbild

Användaren ska på ett enkelt sätt kunna lägga in maträtter och recept som hen tycker om, i en databas och när databasen har minimum antal maträtter ska användaren slumpmässigt få fram en veckomeny genom en knapptryckning.
Användaren ska även kunna välja att generera en veckomeny utefter specifika preferenser.
Användaren ska efter att ha fått en slumpad veckomeny kunna redigera maträtt på specifika dagar under aktuell vecka samt framtida veckor.
När användaren redigerar en veckodag för att skriva in en annan maträtt ska listan autokompletteras med sökningar mot databasen på alla sparade maträtter.
Sidan ska fungera med paginering så att man ska kunna se tidigare och kommande veckomatsedlar.
Användaren ska kunna se en paginerad lista av de veckor som hen har genererat en veckomeny för. När en vecka klickas på ska den visas på helsida.
Användaren ska kunna se en lista över alla maträtter, som går att filtrera och sortera.
Två eller flera användare ska kunna koppla ihop sina konton och därmed se samma veckomatsedlar och samma content (veckomenyer och välja att generera veckomenyer bara från maträtter som någon av de äger. De kommer även att kunna kommentera på recept, t ex i veckomenyn och se varandras kommentarer.
Användare kommer kunna redigera och arkivera recept som de själva skapat.
Appen kommer att vara responsiv med brytpunkter för mobile, tablet, laptop med mobile först.

Avgränsning

Jag beslutade ganska tidigt inte att inte sätta upp en egen databas, server och API även om det vore lärorikt, utan att istället använda Firebase för back-enden.
Jag kommer inte lägga någon tid på UX-undersökningar utan bara utgå ifrån mig själv och råd från vänner och nära om hur vi tycker att appen ska fungera.
Jag kommer inte att ha dark/light mode som jag först tänkte.
Förutom wireframes kommer styling och design få komma senare och göras i ett senare skede och i mån av tid ifall arbetet med funktionerna skulle gå över tiden.
Efter samtal med Martin tyckte han att det såg ut att vara för lite jobb för 160 timmar, så jag lade på att ha en sida för News med t ex latest och highest rated dishes. Men efter att ha tidsestimerat så ligger jag redan 11h över tidsbegränsningen och har därför inte utrymme att lägga till en sådan sida. Dock har jag sedan vårt samtal adderat möjlighet till kommentarer samt möjlighet att betygssätta recept.

Planering & tidsestimering

Att testa så att saker funkar är inkluderat i tiden som är beräknad för varje task.

Övergripande funktioner = 46h

Hantera error och avvikelser + styling för dessa - 6h
Bestäm databasstruktur: planering och research – 8h
Bestäm databasstruktur: integrering i Firebase – 8h
Planering och setup av React-app och routes - 1h
Bestäm och bygg mapp- och filstruktur i VSCode, inkl. tomma exporterbara komponenter med rätt namn som kan fyllas med innehåll och funktioner efter hand – 2h
Logik för att kunna koppla ihop användare att se samma content: planering och research - 6h
Logik för att kunna koppla ihop användare: content, funktioner och styling för det - 8h
Planera och skapa kontext - 3h
Hur skicka med långa processer t ex byta maträtt på en dag genom att skapa en ny - hur ska den komma ihåg att det är det den ska göra medan processen att byta till en nyskapad maträtt pågår? Research – 1h
Autentisering - rätt content till rätt personer på alla ställen - 3h

Sign up & sign in pages = 6h

Kunna skapa användare, m.h.a. front end, inkl. styling – 1h
Research och integration av verifiering med Zod - 1h
Kunna skapa användare, back end tar emot, inkl. research - 2h
Kunna logga in, mha front end, inkl styling och integration av verifiering med Zod – 1h
Kunna logga in, backend tar emot, inkl. research – 1h

Användarfrågor vid signup = 5h

Planera logik för frågor, inkl. skapa denna i kontexten - 2h
Logik för funktion att köra endast efter sign-up som sätter användarpreferenser i kontexten, samt navigation efter frågorna - 2h
Front-end styling – 1h

Inställningar och preferenser för användare = 4h

Skapa account settings, såsom preferences settings, i kontexten – 1h
Skapa två subpages i user settings att bläddra emellan, ev med animation: Preferences, Account settings och möjligheten för användaren att redigera de, inkl styling - 3h
Skapa eller redigera maträtt = 15h
Research på create och update i Firebase – 2h
Bygg möjlighet att create och update i min app - 3h

Formulär att fylla i, inkl. styling och verifiering med Zod – 1h
Research på Firebase Storage för att ladda upp bild och referera till i Firebase Database - 1h
Skapa möjlighet att ladda upp bild och få en förhandsgranskning, inkl. styling - 2h
Se om det går att generalisera till en komponent eller om två behövs och bygg det jag kommer fram till, inkl. styling - 2h
Knapp för att arkivera maträtt + planera vad ska hända då. Bl a ska maträtten inte genereras i nya veckomenyer. Koppla till Firebase - 2h
Logik i kommentar-komponenten för att lägga till kommentar om maträtten skapas samtidigt och inte har ett id än – 2h

Generera veckomeny = 19h

Skapa funktion för att slumpa maträtter ur databasen med hänsyn till användarens preferenser och spara genererad veckomeny i databasen - 8h

…med avancerade val

Skicka in delete-knapp till table komponent och skapa tillhörande funktion för att inte generera den dagen i veckomenyn - 2h
Skicka in edit-knapp till table komponent och skapa en modal som dyker upp när den klickas, inkl. styling för den – 1h
Funktionalitet för edit-modalen när veckomeny ska skapas. Spara specifika preferenser på specifika dagar och ta hänsyn till dessa när veckomenyn genereras – 8h

Landing page som inloggad = 10h

Hämta veckomeny från Firebase db m.h.a. axios(?) och React Query, inkl setup och repetition/research av React Query och axios. (Fetchservice) – 6h
Möjlighet att redigera hel vecka ✎ : kunna ta bort en hel dag eller hela veckan. Front end + delete i db - 3h
CSS Styling, inkl. hantera om en vecka i landing page inte har en meny: knappar för att skapa veckomeny med enkelt eller avancerade val – 1h

Redigera veckomeny = 3h

Funktion och styling för när man trycker på redigera veckomeny (lägg till knappar för att redigera etc.) – 1h
Logik för att uppdatera om hel eller del av dag tas bort – uppdatera i db - 2h

Byt maträtt modal = 2h

Ta emot och skicka upp props om vilken slot som ska uppdateras och med vad – 0.5h
Styling – 0.5h

Funktioner för de olika knapparna - 1h

Maträtt detaljsida eller modal = 3h

Hämta info för specifik maträtt (med React Query?) mha id ifrån klickad ruta i veckomenyn eller maträttsöversikten - 1h
Rendera info och styla sida – 2h

Maträtter översiktsida = 6h

Styling - 1h
Drop-downs – 2h
Sortering som funkar med (stannar kvar efter) paginering – 3h

Veckor med menyer översiktssida = 2h

Kopiera från maträtter översiktssida och anpassa – 2h

Generiska eller stora komponenter, inklusive styling för dessa = 50h

Generisk Modal med stängningsknapp och som stänger sig när man klickar utanför den - 3h
Generisk knapp. Kunna ändra färg, vad som händer på onclick, om den ska kunna länka etc - 2h
Generisk pill button vid tillfällen för att välja flera alternativ. Kunna ändra färg, se annorlunda ut om den är active, vad som händer på onclick etc. - 1h
React Table: Research - 2h
Veckomeny. Rendera tabell mha React Tables - context för user preferenses ska tas i beaktning (1 eller 2 dagar). Den ska ta emot props från föräldern om vilken vecka det gäller och vilket innehåll den ska ha. Ska återanvändas både när man skriver in preferenser för att redigera en vecka samt när en veckomeny renderas. Responsiv med kortare veckodagnamn för mobile. Möjlighet att lägga in knappar för redigering av vecka. - 8h
Responsiv meny inkl. drop down och planera struktur - 4h
Kommentarer, deras funktion samt plats i databasen. Möjlighet att kommentera (update i firebase) inkl research för detta - 8h
Paginering inkl research + koppla ihop så att den funkar med React Query - 3h
Autocomplete: research – 1h
Autocomplete när man söker på maträtt (och kategori?), inkl. testning – 8h
Styla och testa autocomplete-komponenten - 2h
Filtrering av översiktssidor - 8h

Total tid 171 timmar

Tidsrapport

Tid anges i timmar. Vissa moment har smält samman eller överlappat varandra och därför har tidsberäkningen för de gjort lika så. Moment är sammanslagna i tabellen med ett “+”-tecken. När jag avrundade tiden och inte tog med några mycket korta moment så blev det sammantaget 158,25 timmar enligt nedan tabell, medan den faktiska tiden som är klockad i Toggl är 162 timmar.

Moment	Estimat	Faktisk tid
Övergripande funktioner	**46**	**16,5**
Hantera error och avvikelser + styling för dessa	6	1
Bestäm databasstruktur: planering och research	8	0,2
Bestäm databasstruktur: integrering i Firebase	8	3
Planering och setup av React-app och routes + Bestäm och bygg mapp- och filstruktur i VSCode, inkl. tomma exporterbara komponenter med rätt namn som kan fyllas med innehåll och funktioner efter hand	1 + 2	2,25
Logik för att kunna koppla ihop användare att se samma content: planering och research + Logik för att kunna koppla ihop användare: content, funktioner och styling för det	6 + 8	0
Planera och skapa kontext	3	3,3
Hur skicka med långa processer t ex byta maträtt på en dag genom att skapa en ny - hur ska den komma ihåg att det är det den ska göra medan processen att byta till en nyskapad maträtt pågår? Research	1	inkluderat i övriga moment
Autentisering - rätt content till rätt personer på alla ställen	3	6,5
Sign up & sign in pages	**6**	**3,5**
Kunna skapa användare, m.h.a. front end, inkl. styling	1	1,5
Research och integration av verifiering med Zod	1	1,6
Kunna logga in, mha front end, inkl styling och integration av verifiering med Zod	1	0,3
Kunna logga in, backend tar emot, inkl. research	1	?
Användarfrågor vid signup	**5**	**12,5**
Planera logik för frågor, inkl. skapa denna i kontexten	2	1

Logik för funktion att köra endast efter sign-up som sätter användarpreferenser i kontexten, samt navigation efter frågorna + Front-end styling + Generisk pill button vid tillfällen för att välja flera alternativ. Kunna ändra färg, se annorlunda ut om den är active, vad som händer på onclick etc.	2 + 1 + 1	11,5
Inställningar och preferenser för användare	**4**	**9,5**
Skapa account settings, såsom preferences settings, i kontexten	1	2
Skapa två subpages i user settings att bläddra emellan, ev med animation: Preferences, Account settings och möjligheten för användaren att redigera de, inkl styling	3	7,5
Skapa eller redigera maträtt	**15**	**19,25**
Research på create och update i Firebase + Bygg möjlighet att create och update i min app + Formulär att fylla i, inkl. styling och verifiering med Zod	2 + 3 + 1	10,25
Research på Firebase Storage för att ladda upp bild och referera till i Firebase Database	1	0,5
Skapa möjlighet att ladda upp bild och få en förhandsgranskning, inkl. styling	2	7,5
Se om det går att generalisera till en komponent eller om två behövs och bygg det jag kommer fram till, inkl. styling	2	1
Knapp för att arkivera maträtt + planera vad ska hända då. Bl a ska maträtten inte genereras i nya veckomenyer. Koppla till Firebase	2	0
Logik i kommentar-komponenten för att lägga till kommentar om maträtten skapas samtidigt och inte har ett id än	2	0
Generera veckomeny inkl med avancerade val + **Byt maträtt modal**	**19 + 2**	**33**
Skapa funktion för att slumpa maträtter ur databasen med hänsyn till användarens preferenser och spara genererad veckomeny i databasen	8	9
Skicka in delete-knapp till table komponent och skapa tillhörande funktion för att inte generera den dagen i veckomenyn + Skicka in edit-knapp till table komponent och skapa en modal som dyker upp när den klickas, inkl. styling för den	2 + 1 + 8 + 0,5 + 0,5 + 1 = 13	24

+ Funktionalitet för edit-modalen när veckomeny ska skapas. Spara specifika preferenser på specifika dagar och ta hänsyn till dessa när veckomenyn genereras + **Byt maträtt modal = 2h:** Ta emot och skicka upp props om vilken slot som ska uppdateras och med vad + Styling + Funktioner för de olika knapparna		
Landing page som inloggad	**10**	**26**
Hämta veckomeny från Firebase db m.h.a. axios(?) och React Query, inkl setup och repetition/research av React Query och axios. (Fetchservice)	6	16
Möjlighet att redigera hel vecka : kunna ta bort en hel dag eller hela veckan. Front end + delete i db	3	6,6
CSS Styling, inkl. hantera om en vecka i landing page inte har en meny: knappar för att skapa veckomeny med enkelt eller avancerade val	1	3,5
Redigera veckomeny	**3**	**-**
Funktion och styling för när man trycker på redigera veckomeny (lägg till knappar för att redigera etc.) – 1h + Logik för att uppdatera om hel eller del av dag tas bort – uppdatera i db	1 +2	inkluderat i övriga moment
Byt maträtt modal	**-**	**-**
Se tid för Generera veckomeny		
Maträtt detaljsida eller modal	**3**	**2**
Hämta info för specifik maträtt (med React Query?) mha id ifrån klickad ruta i veckomenyn eller maträttsöversikten	1	0,5
Rendera info och styla sida	2	1,3
Maträtter översiktsida	**6**	**0**
Styling	1	0
Drop-downs	2	0
Sortering som funkar med (stannar kvar efter) paginering	3	0
Veckor med menyer översiktssida	**2**	**0**
Kopiera från maträtter översiktssida och anpassa	2	0
Generiska eller stora komponenter, inklusive styling för dessa	**50**	**22,5**

Generisk Modal med stängningsknapp och som stänger sig när man klickar utanför den	3	2,75
Generisk pill button vid tillfällen för att välja flera alternativ. Kunna ändra färg, se annorlunda ut om den är active, vad som händer på onclick etc.	1	inkluderat i Generera veckomeny
Generisk knapp. Kunna ändra färg, vad som händer på onclick, om den ska kunna länka etc	2	1,25
React Table: Research	2	0
Veckomeny inkl generisk veckomeny	8	9,5
Responsiv meny inkl. drop down och planera struktur	4	4
Kommentarer, deras funktion samt plats i databasen. Möjlighet att kommentera (update i firebase) inkl research för detta	8	0
Paginering inkl research + koppla ihop så att den funkar med React Query	3	5
Autocomplete: research + Autocomplete när man söker på maträtt (och kategori?), inkl. Testning + Styla och testa autocomplete-komponenten	1 + 8 + 2	Delvis inkluderat i Generera veckomeny + 0
Filtrering av översiktssidor	8	0
Extra tillagt	**-**	**13,5**
Deploy och genomgång av app (mindre buggfixar) samt snygga till		2
Loggbok och planering	-	10
Styla bakgrund	-	1
Trello board		0,6
Total tid	**171**	**162 i Toggl (158,25 med ovan avrundning)**

Beskrivning av arbetsprocessen

Arbetet började med att jag skapade en ny React-app med Vite och TypeScript samt några npm-paket som jag visste att jag skulle använda, såsom React Router. Därefter gjorde jag skelettet av den filstruktur som jag tänkt mig, där jag också skapade tomma exporterbara komponenter samt några gäst-routes såsom sidor för inloggning och att skapa användare. När dessa var på plats så stylade jag sign-in- och sign-up-sidorna med TailwindCSS inklusive formulären. Därefter påbörjade jag sign-up- och sign-in-funktionaliteten med hjälp av Firebase Auth. För att få det att funka skapade jag en fil med en Firebase-service som

kopplar ihop min app med Firebase Auth, Database och Storage. Jag skapade och stylade formulär som använde Zod och React Hook Forms för verifiering. Stylingen gjordes med TailwindCSS och i Zod-schema filen definierade jag verifieringsreglerna, vilka sedan gjordes till en egen TypeScript-typ som används i verifieringen. Vid denna tidpunkt påbörjade jag även en generisk knapp-komponent som jag senare skulle återkomma till och vidareutveckla och göra mer anpassningsbar. I samband med sign-up och sign-in-sidorna satte jag även upp TailwindCSS konfigureringen och började fylla den med mina egenvalda färger. Här skapade jag också en generisk Alert-komponent. Både Button- och Alert-komponenterna skulle jag komma att använda mycket framöver så jag ville ha de på plats. I TailwindCSS-konfigurationen lade jag även en del tid på att skapa funktioner för att få ljusare och mörkare toner av färger så att om man vill ändra grundnyans i en Button- eller Alert-färg så ändras dess mörkare och ljusare varianter automatiskt. Jag ville använda en single source of truth till detta och lade därför en del tid på det.
Det föll sig så att sign-up- och sign-in-formulären behövde mer tid så jag fortsatte på dessa med styling tills jag blev nöjd, samt implementerade felhantering via en custom hook och lade till villkorat visade (conditionally rendered) felmeddelanden.
När ovanstående var klart skapade jag en generisk loading spinner-komponent samt skyddade routes så att man blir omdirigerad till sign-in-sidan ifall man försöker gå in på en skyddad route utan att vara inloggad. Här korrigerade jag även layouten på hela sidan i App.tsx, så att skärmen aldrig scrollar om den inte behöver det på grund av innehållet, för en bättre användarupplevelse. Jag gjorde även det möjligt att ändra lösenord, samt möjlighet att lägga till användarnamnet direkt vid sign-up.

Härefter var det dags att skapa användar-frågor som skulle dyka upp direkt efter att man skapat en ny användare. Därför skapade jag default-inställningar direkt när användaren skapas samt ger användaren direkt efter sign-up möjligheten att modifiera dessa. För att åstadkomma detta skapade jag användar-frågor och en generisk Pill-komponent för att visa vilka inställningar som var satta just då, samt ge användaren möjlighet att ändra de. På några Pill-grupper kan man bara välja ett val, medan på andra kan man välja flera. Jag spenderade en hel del tid, del att få Pills:en att visa rätt val, dels att skapa dessa alternativ i databasen, kopplade till ett user document i Firebase database. Slutresultatet blev att de ändras i databasen så fort man markerar en Pill bland valen. Först skapade jag en ny kontext för att kunna komma åt user preferences överallt, men sen insåg jag att detta inte behövdes och tog bort det. Här hade jag även en bugg där de gamla värdena skrev över de nya så jag fick lägga en del tid på att lösa detta. I detta skede skapade jag även en custom hook att hantera de flesta Firebase-uppdateringar i. Min kunskap på routes var ringrostig samt hade jag inte använt nested routes förut. Så först använde jag mig av <Outlet/> men ville kunna skicka in props, så då gjorde jag om routingen, vilket tog lite tid att lära mig och att få rätt

på. När jag var nöjd hade jag optimerade återanvändbara funktioner samt en enda route som omfamnade samtliga fråge-routes så att användaren för bättre UX kan navigera med webbläsarens bakåt- och framåtpilar. Detta kom att bli ett av de mest tidskrävande momenten i hela arbetet med 11,5 timmars arbete, istället för de estimerade 5 timmarna. Detta beror troligen dels på att det var i början av projektet, så det tog tid att sätta sig in i kodandet och bestämma strukturen jag ville ha, dels som sagt att jag inte använt nested routes förut.

I detta skede skapade jag en bakgrund via tailwind-konfigurationen för ett trevligare utseende i appen, samt skapade en generisk Divider med en symbol som jag skulle komma att utveckla vidare efter hand och vid behov. Här gjorde jag också en sida dit användaren kan komma för att när som helst ändra sina inställningar, och då använde jag också nested routes.

Jag skapade här en navigationsmeny för sidan för att enklare kunna navigera när jag utvecklar och för att den ändå kommer behövas senare.

När de initiala inställningarna och funktionerna var klara var det dags att fortsätta till det faktiska utseendet och de faktiska funktionerna som inloggade användare ska kunna använda. För att ens kunna generera en veckomatsedel behövdes måltider, därför bestämde jag mig för att nu i turordningen bygga möjligheten för användaren att skapa måltider, samt att hantera fall där det finns för få måltider med hänsyn till användarens preferenser och det faktiska innehållet i databasen. Jag skapade även här StarRating-komponenten för att kunna betygsätta sina måltider, påbörjade en generisk modal-komponent samt en generisk paginerings-komponent, allt som skulle användas på förstasidan som syns för inloggade användare. När sakerna på förstasidan var på plats så slutförde jag sidan för att lägga till en maträtt, där jag lade en del tid för att få rätt på valideringen, vilket känns viktigt för att få in rätt data i databasen att arbeta med, med tanke på att användare kan skicka in vad som helst om man inte validerar ordentligt. På denna sida lägger jag också till en dropzone för bild där en förhandsgranskning syns och som på submit av formuläret laddar upp bilden till Firebase Storage, får tillbaks en url som länkar till bilden och som jag sedan lägger till på imageUrl propertyn för måltiden som läggs till. För användaren att ange kategorier använde jag ett npm paket som heter React-Select (1) som verkade väldigt smidigt. Utmaningen blev att kunna använda det tillsammans med React Hook Form, som jag använde för valideringen.

Jag såg en bugg i Dividern och fixade den samt såg lite fler småfel och buggar i stylingen och formuläret som jag fixade. Dessutom förbättrade jag koden och användarupplevelsen genom att bl. a. scrolla upp sidan när man skickar in formuläret och lägga till Alerts med felmeddelanden och bekräftelse på lyckad uppladdning.

Det är svårt för mig att bara låta bli styling och funktioner som inte ser bra ut eller inte fungerar helt som de ska. VSCode gav även fel på hur StarRating-komponenten renderas, även om den fungerade så det felet rättade jag också till, samt snyggade till stjärnornas utseende.
Hela arbetsprocessen för att lägga till maträtt tog 19,25 timmar jämfört med estimerade 15 timmar. Jag har svårt att gå vidare om jag ser saker som inte ser rätt ut eller inte känner mig 100% klar.
Härefter lade jag en hel del tid på att få till hämtningen av måltider med rätt queries, räkna de, samt att stylea en generisk ContentContainer-komponent som skulle bli bakgrunden till alla sidor med innehåll för att ge ett enhetligt utseende i appen. Jag har hela tiden försökt göra koden så återanvändbar som möjligt. Senare skulle jag upptäcka att Firebase har så kallade ”aggregation queries” (2) där man bland annat kan räkna antalet dokument i en kollektion utan att hämta all data i den, och då använde jag det istället för att hämta information om antalet maträtter i databasen, för att undvika onödigt stora hämtningar. Jag lade till meddelanden och en CTA-knapp för fall där det fanns för få meddelanden. Det krävdes en del logik för att kolla av de möjliga fallen och rendera olika innehåll beroende på de olika omständigheter som kunde finnas. Att använda sig av veckonummer blev också svårlöst. Npm-paketet date-fns (3) kunde ge mig vilken vecka det var (4), men när jag skiftade år i pagineringen fick jag problem. Fick lägga mycket tid på att lista ut hur funktionen för date-fns skulle användas och skapa logik för att skapa maträtt på rätt vecka samt att förebygga att fel vecka anges eller visas (5). Detta gick ihop med arbetet med pagineringen lite senare.

Nu kunde jag lägga till funktionen att generera en slumpmässig veckomatsedel om tillräckligt med måltider finns. Det gick ganska fort att lyckas lägga till ett genererat veckoschema med antingen 1 eller 2 måltider per dag i databasen. Att dels rendera det i tabellen samt att styla tabellen kom att ta längre tid. Jag fick göra om WeekPlan-typen några gånger. Det blev också en hel del arbete med att kolla av om WeekPlan[’meals’] -typen var av typ OneMealPerDay eller TwoMealsPerDay så att TypeScript skulle godkänna koden.
När jag lyckats lägga in en vecka med rätt typer så var det dags att rendera tabellen med veckomenyn för antingen en eller två rutor per dag. Jag bestämde mig för att inte rendera veckomenyn med React Tables då det kändes mer avancerat än det behövde vara. Jag skulle behöva lägga tid på att lära mig det, vilket jag inte bedömde att jag hade. Först så hämtade jag måltider från databasen för varje cell där varje cell var en komponent med egen hämtning, men insåg snart att det gav 7 till 14 hämtningar till databasen, i stället för 1 per vecka, därför skrev jag om detta. Det tog totalt sätt mer tid än väntat att styla tabellen, speciellt i redigeringsläge, som jag skulle komma att skapa senare, på mobila skärmar.

När renderingen av veckomenyn var klar så skapade jag en paginering som fungerar både med pilarna i webbläsaren och med pilarna i pagineringskomponenten, med hjälp av sökparametrar. Det var här jag fick lägga mycket tid på att få rätt på pagineringen, speciellt när den bytte mellan år. Det var också nu jag bestämde mig för att implementera Tanstack Query och behövde därför skriva en funktion som manuellt hämtar från Firebase. Denna gjorde jag också generisk och fick tänka om en del från hur jag gjort när jag först strömmat all data från Firebase. Tanstack Query funkar mycket smidigt med paginering eftersom jag kan casha data för varje visad vecka och inte behöver hämta vecka på nytt varje gång användaren bläddrar fram och tillbaka. Här hanterade jag även ev buggar jag kunde komma på – såsom att omdirigera användare som försöker söka på en url med en vecka som inte finns, samt skapade en Not Found-sida för att hantera sådana fall. All hämtning och att rätt data skulle hämtas vid rätt tillfälle men inte i onödan på Landing-page och dess komponenter var det i examensarbetet som tog absolut längst tid då jag fick återkomma till det och modifiera det så många gånger. Jämfört med 6 timmars estimerat arbete tog det totalt sett 16 timmar att få rätt på det.

När jag hade en fungerande paginering på plats fixade jag ytterligare styling på Button-, Logo-, MealDetails- och Modal-komponenterna. Jag tog hand om flera buggar i hur veckomenyn renderas, query till databasen när det finns null-värden (vilket man inte kan ha med i queryn) samt när Tanstack Query behöver hämta.

Nästa stora sjok att avklara var huvudfunktionen i min app, men jag behövde tidigare förberedelser för att underlätta utvecklingen av denna och se om det blir rätt eller fel. Denna funktion var alltså att generera veckomeny med avancerade alternativ.
Här visste jag inte riktigt hur logiken skulle se ut så det var trevande i början och jag provade mig fram. När veckomeny ska genereras avancerat så ser man först en tabell med endast frågetecken där alla värdena är null. För en bättre användarupplevelse valde jag att skapa en förhandsgranskning där jag sparar aktuella värden direkt i databasen så att om man laddar om sidan, eller lämnar förhandsgranskningen och kommer tillbaka igen så ska den ha de värden den hade när man som användare lämnade den. Dock ska den, ifall man bytt inställningar, gamla förhandsgranskningen raderas och en ny ska skapas som passar de nuvarande användarpreferenserna som valts i inställningar, eller om man bytt inställningar så att det inte finns tillräckligt med recept så kommer användaren att se det vanliga som visas. Om användaren använder bakåtpilen från inställningar till redigeringsläget oh det inte finns tillräckligt med maträtter i databasen utgående från användarens inställda preferenser så ska hen omdirigeras till den vanliga översiktssidan som hanterar fall av för få maträtter. Detta var en bugg som jag kom på efteråt och som jag ville lägga tid på att hantera eftersom

det finns en relativt stor chans att användaren efter att den klickat på att generera avancerat, kanske inser att den villa ha annorlunda inställningar.

För att veta vilken dag som klickas på och som ska redigeras så renderade jag allt tabellinnehåll för måltider i en funktion som returnerar JSX, där jag skickar in index till denna funktion samt vilken knapp som klickas på så att rätt åtgärd sker i koden. Det var klurigt men roligt att komma fram till hur jag skulle göra här.
För att välja måltid på respektive ruta i tabellen så valde jag att öppna en popupmeny för klickad ruta och i den hantera val samt uppdatering till förhandsgransknings-dokumentet i databasen. Det var först när man som användare i popupmenyn är nöjd med valet och klickar spara som jag ville att måltiden sparas till databasen på aktuell dag i det preliminära veckoschemat. I popupmenyn ville jag ha olika val och de blev slutligen: välj specifik rätt, slumpa via kategori, välj att äta ute, välj inget mål mat eller välj att bli överraskad. Även här hanterade jag fel som till exempel att det inte finns recept att slumpa från en viss kategori, och gav användaren en förklaring på varför de inte kan slumpa ifrån den kategorin samt instruktioner på vad hen kan göra i stället. Jag använde även här React Select men eftersom jag inte använde React Hook Form var det ett annorlunda tillvägagångssätt för att använda värdena från React Selecten än jag tidigare använt i mina formulär och som var mig svåra att förstå till en början.

Under arbetet med funktionaliteten för att generera avancerade alternativ försökte jag även styla tabellen för mobila skärmar att scrolla sidledes i stället för att gå utanför sin förälder.
När förhandsgranskningen var på plats och fungerade så gjorde jag en knapp som utlöser en funktion för att först fylla ej ifyllda (null-) värden med slumpade maträtter enligt användarens preferenser, kopiera över denna till kollektionen för ordinarie veckomenyer och när det lyckats, radera förhandsgranskningen samt att navigera till översiktsvyn för den vecka man nyss skapat. Där fick jag problem med att den nya veckomenyn inte renderades på grund av Tanstack Querys cashing, så jag fick komma på sätt att utlösa en återhämtning av datan för aktuell, nyligen uppdaterad, veckomeny.
När de grundläggande funktionerna i målbilden för att generera avancerad veckomeny var på plats så lade jag till möjligheten att välja ”No meal” på rutorna för måltider i tabellen, genom att klicka på en röd knapp med ett kryss. Detta val finns även i popupmenyn där valen i huvudsak görs.
Därnäst skapade jag en generisk tabellkomponent som jag återanvände både i översikten, när veckomeny ska genereras, samt när en veckomeny ska redigeras, vilket var nästa funktion jag byggde, detta för att kunna ha en unison styling som Tailwind rekommenderar (6) samt för att optimera kod. Detta medförde att jag fick skicka in props och villkorligt

rendera vissa saker och funktioner beroende på vad komponenten skulle användas till. Detta är nog den största generiska komponent jag gjort hittills.
Ungefär här när arbetet började närma sig sitt slut så insåg jag att jag behöver användarpreferenser på många ställen och bestämde mig därför för att lägga till de i typen för WeekPlan och därmed i dokumenten för veckomenyer, så att jag alltid hade tillgång till de. Vad jag inte tänkte på var att användarpreferenser ju kan ändras, men insåg sen att det var bra att ha de kvar för att då jag ska redigera en veckomeny ska den ta hänsyn till de preferenser som fanns vid skapandet av den (till exempel antal mål per dag, för att inte tidigare mål i veckomenyn helt ska försvinna om det skulle ändras till ett mål per dag när man försöker redigera en veckomeny).
Det dök upp flertalet buggar och arbetet drog ut på tiden. Dessutom glömde jag i min besatthet att lösa problemet att klocka de enskilda momenten och därför har jag lagt ihop de alla i tabellen för tidsestimering ovan. Allt som allt tog arbetet med att generera veckomeny 24 timmar istället för de estimerade 24 timmarna.

Som tidigare så gjorde jag efter detta lite mindre fix och styling nu efter ett JavaScript-tungt moment. Jag gjorde bland annat en generisk komponent för skiftningsbar (togglable) visning av info som stöd till användaren.
Efter denna lilla andningspaus gick jag vidare för att skapa möjligheten för användaren att kunna redigera en befintlig veckomeny och fick även här fundera över hur jag ville göra med logiken och om jag ville lägga editeringen i en egen route, vilket jag till slut bestämde mig för att göra. När användaren klickar på att redigera en veckomeny så skapar jag igen upp en förhandsgranskning i databas med samma id som sist (men den gamla som eventuellt fanns när veckomenyn skapas raderades ju när den kopierades över till en ny veckomeny).
Den generiska tabellkomponenten används igen men med något olika funktioner. Dock fungerar det mesta väldigt likt som när man genererar veckomeny med avancerade alternativ så jag gjorde en generisk funktion i en custom hook till vilken man skickar en parameter för vad funktionen ska användas till, för återanvändbar optimerad kod. Dock används den bara på två ställen till två olika saker så det kan hända att jag lade tid på att optimera här i onödan. I redigeringsläget lade jag även till en knapp för att radera en hel veckomeny från en vecka, samt en popupmeny för att bekräfta att man är säker på sitt beslut. Även här dök det upp buggar med hämtningen när redering genomförts och man skulle bli omdirigerad till en tom översiktssida, men innan man laddat om sidan så visades veckomenyn. Detta löste jag genom att skicka med en state i navigeringen som avlyssnades av en useEffect som i så fall körde en återhämtning av aktuell veckas veckomeny (vilket då är ingen).
Som sista hand vid examensarbetet innan jag nådde 160 timmar och det var dags för redovisning så dök det upp en del buggar jag inte lagt märke till förut, främst angående hur

veckomenyn renderas och hur data hämtas. Datahämtningen på startsidan gick därför över tiden med 6,6 timmars arbete i stället för de 3 timmar som var estimerade.
Sist så laddade jag upp appen till Netlify så att den finns att använda live.

162 arbetstimmar hade nu använts och det var dags för redovisning nästa dag, därför kunde jag inte göra vidare arbete medan enligt målbilden skulle man som användare även kunna redigera sina maträtter, kommentera på maträtter, samt kunna koppla ihop sitt konto med en annan användare för att se samma innehåll. Det skulle finnas översiktssidor för måltider och veckor som skulle kunna sorteras, filtreras och sökas bland. Inget av det sistnämnda finns med i appen eftersom det tyvärr inte hanns med. Allt annat i målbilden finns med.

Erfarenheter

Tillvägagångssätt

Den initiala planeringen i förstudien där jag definierade syfte, målbild, delade in arbetet i mindre moment och så vidare på ett strukturerat sätt var mycket användbar för det kommande arbetet. Det gav en överblick och satte i gång de planeringsprocesser som krävdes. Jag började med att lägga in allt i Trello men använde senare mest min tidsestimering när jag tittade på vad som behövdes göras för att hålla mig till planen. Jag använde ReactJs, React Router, React Hook Form, Tanstack Query, Zod validation och TailwindCSS för att utveckla min app. Detta var främst för att jag använt dessa metoder innan och kände mig mest bekväm med de till ett arbete som jag inte kände mig bekväm med att utföra. Skulle jag valt andra metoder och verktyg är jag rädd att det skulle ta alltför lång tid att lära mig de och ta värdefull tid ifrån att faktiskt utveckla den produkt jag tänkt mig. Dessutom gick det så snabbt i slutet av vår sista kurs att jag knappt hann med att ta in vad vi lärt oss så jag ville gärna bli bättre på och fördjupa de kunskaper jag redan hade eller hade påbörjat. Repetition är kunskapens moder.
TailwindCSS lärde jag mig under min LIA och vill gärna vidareutveckla min kunskap även i det. Jag lär mig hellre något ordentligt så att jag behärskar det, än att lära mig små bitar av många olika språk, i alla fall så tidigt som nu i mitt liv som utvecklare, även om jag är nyfiekn på andra språk och ramverk. Men jag vill kunna bygga faktiska projekt och inte bara kunna lite av varje.

När jag valde turordning på vilka uppgifter jag skulle ta mig an var mitt resonemang att de bygger på varandra och att jag först behövde skapa en användare och logga in för att gå vidare, vilket var vad jag började med. Därefter behövde jag kunna skapa måltider för att lägga till de i databasen och senare kunna generera de. Jag hade också kunnat göra det

manuellt eller genom att mocka data men eftersom jag ändå skulle göra ett UI för det tyckte jag att det var lika bra att få det gjort samt att använda det UI:t för att själv lägga till maträtter som godkänns av valideringen, så att jag inte skapar några onödiga buggar med fingerade data som inte gått igenom validering.
Jag valde därefter att skapa och styla ett flertal komponenter och lägga rätt så mycket tid på utseendet i appen. Jag vet fortfarande inte om detta var rätt beslut. Jag och många med mig tenderar att lägga för mycket tid på CSS för att få utseendet till hur man vill ha det medan jag samtidigt vill att appen ska se bra ut och kunna återanvända några bas-komponenter på många ställen såsom Alert, Button, Container och Modal.
När jag gjorde pagineringen blev jag påmind om hur mycket jag uppskattar Tanstack Query som är smidigt och lättanvänt. Jag hade dock fler problem med dess cashing än jag hade innan, men skulle fortsätta vilja använda det om jag gjorde appen igen då det passar perfekt till att öka prestandan och farten när man bläddrar igenom sidor med hämtade data.
Jag lärde mig mer om Nested Routes i React Router genom att använda det i de initiala frågorna samt under sidan för inställningar.
React är smidigt att använda för att jag kan det sedan innan, men jag blev också sugen under mitt arbete på att lära mig andra JavaScript-ramverk för omväxlings och lärandes skull. Om jag hade byggt appen igen hade jag gärna försökt mig på Next.js, ett JavaScript-ramverk som ska vara väldigt populärt just nu (8) och som jag gärna hade lärt mig. Jag hade också gärna satt upp min egen back-end, API och autentisering, men eftersom saker tar den tid de gör för mig känner jag att jag skulle vilja kunna mer av dessa innan jag bygger ett projekt under tidspress med dem. Dock har jag hört att det bästa sättet att lära sig och att utvecklas som utvecklare är att bygga egna projekt, så ett hobbyprojekt med dessa delar i hade säkert varit lärorikt, eller ett annat projekt med mindre tidspress på att leverera än detta examensarbete.
Jag blev också positivt överraskad över hur användbart och relativt enkelt det var att använda Zod för valideringen. När jag gick lite djupare i det var det dock flera saker som jag inte blev klok på men inte kände att jag hade tid att sätta mig in i. React Hook Form använde jag bara för att kunna använda Zod eftersom det är det sätt jag gjort validering på förut och vet hur man gör. Skulle jag göra appen igen skulle jag nog använda ett UI-bibliotek för att snabba upp arbetet med styling.
Jag lade mycket tid på att hantera buggar innan jag gick vidare till nästa moment då jag som person oftast vill göra saker ordentligt, samt att jag inte vill bygga vidare på dåligt fungerande kod. Arbetet med att generera avancerade alternativ och dess funktioner tog definitivt mest tid av alla moment i hela arbetet med 33 timmar i stället för estimerade 21 timmar.

Något jag inte är nöjd med är att inte ha lyckats slumpa valfritt antal maträtter ur databasen när man genererar veckomatsedel slumpmässigt utan att jag i stället hämtar och slumpar

alla maträtter, vilket inte är hållbart i längden, men kände att det skulle ta för lång tid att lära sig då jag inte förstod de instruktioner jag hittade (7) och var stressad över att försöka hålla mig till tidsplanen.
Efter att ha genomfört detta arbete känner jag absolut att jag lärt mig mer om React.js, speciellt custom hooks och de facto att hooks inte kan kallas på villkorligt och hur man då använder de på ett smart sätt när man till exempel först behöver hämta data som sedan behövs som en parameter i en hook. Jag har lärt mig och övat mig på att använda nya sätt att typa genom TypeScript som jag börjat lära mig under LIAn.
Jag har till större del insett värdet i olika npm-paket, såsom React Select och date-fns och uppskattar den tidsbesparing och goda funktionalitet de erhållit.
Efter att ha sett mina klasskamraters redovisningar var det många som önskade att de planerade bättre från början.
Jag tycker att jag planerade översikten över arbetet relativt bra i början även om min tidsestimering inte var bra, men jag har svårt att se att om jag hade planerat tiden bättre i början så hade jag hunnit med allt. Många oväntade buggar tog helt enkelt mycket längre tid än beräknat.
Dock inser jag att jag eventuellt troligen hade kunnat planera användarflödet vid genererandet av avancerad veckomeny bättre, liksom säkert är fallet med en del annan funktionalitet. Det tog onödigt lång tid att bestämma flödet när jag väl skulle påbörja det.

Något som jag skulle velat göra annorlunda är att använda mig utav Error Boundaries, i stället för att hålla på med try-catch, loading states, rendera error på många ställen i appen. När jag tittade på att använda Error Boundaries såg jag att det används klasser, vilket jag inte förstår mig på och jag blev åter igen rädd att det skulle konsumera för mycket av min dyrbara tid, så jag lät bli at sätta mig in i det.
Jag lade också en del onödig tid på att skapa upp en filstruktur och ett skelett för mappar. Jag fick ändå döpa om filer, ta bort oanvända filer och mappar och flytta runt filer samt döpa om filer och mappar.
Jag har märkt med mig själv att jag tenderar att överskatta min tid i början av ett projekt och ta saker i långsammare, mer njutbart tempo medan ju närmare deadline desto mer stressigt blir det för mig. Detta är en egenskap jag behöver åtgärda, dels genom att sprida ut planeringen av arbetet mer jämt över tid.
Det var också onödigt att lägga tid på att skapa funktionalitet för att ändra epost-adress och lösenord under inställningar. Det är inget som bidrar till syftet eller målbilden.

Tidsestimat resp. faktisk tid

De flesta moment i min tidsestimering tog lägre tid än estimerat, något jag egentligen hade väntat mig då jag har uppfattningen om mig själv som en ambitiös och klok men just nu

mycket långsam utvecklare. Denna bild av mig själv bekräftades och förstärktes. Det var efter det initiala samtalet med Martin då han tyckte att mitt arbete ”såg lite tunt ut” som jag lade till fler moment för att möta de krav på ett tillräckligt stort arbete jag uppfattade fanns, men jag antog egentligen hela tiden att jag troligen inte skulle hinna med allt. Dock hann jag inte ens med de översiktssidor samt redigering av måltider som jag själv hade med redan från början. Jag förstår självklart också att man inte bör skapa en tidsestimering som man från början själv inte tror kommer hålla.
Tidsestimeringen jag gjorde var enligt ”the good path”, att det mesta skulle gå rätt och inte inräknat buggar. Det var mig extremt svårt att kunna veta hur lång tid det tar att lösa en bugg. En gång satt jag fast i 4 timmar för att jag skrivit ”week” och inte ”weekPlan”, och försökte ändra på en massa andra saker i koden för att få det att fungera och kunde inte förstå varför det inte gjorde det. Hade jag bara skrivit rätt ord från början hade just det momentet tagit 4 timmar kortare. Det är ett exempel på hur svårt det är att estimera tid. Jag tänker att ju mer trygg och van man är i metoderna man använder ju snabbare kommer det gå och desto mer säker kan man bli i sitt tidsestimat. När man är nybörjare och ständigt lär sig nytt är chansen att fastna och behöva spendera timtals på att hitta svar, mycket större.

Mål resp. resultat

Jag har nått en del av men inte alla de mål som jag har satt upp i förstudien. Detta beror på olika faktorer. Eftersom jag försökt vara noggrann med bl. a. buggar innan jag har gått vidare till nästa uppgift har vissa moment tagit mig väldigt långt tid. Jag har även ett flertal gånger uppehållit mig vid CSS-styling, som är benäget att bli tidskrävande. Jag tenderar även att ”pilla med småsaker” emellan större uppgifter och att ägna mig åt styling som är en annan och mer avslappnande belastning på hjärnan för mig än invecklad JavaScript-logik som jag använt i mina större moment. De flesta saker jag hade med i målbilden är uppnådda. De mål som inte är uppnådda är att man som användare ska kunna redigera sina maträtter, kommentera på maträtter, kunna koppla ihop sitt konto med en annan användare för att se samma innehåll samt ha översiktssidor för måltider och veckor som skulle kunna sorteras, filtreras och sökas bland. Inget av dessa hanns med, tyvärr. Tiden tog helt enkelt slut när jag hamnade på 162 timmar dagen före redovisningen. Jag är osäker på vad jag hade kunnat göra annorlunda annat än att låta bli stylingen och gå vidare trots buggar, och då få en app med dåligt utseende och dålig funktionalitet, och det vill jag inte.
Angående stylingen borde jag kanske oftare överväga att skriva upp det jag ser som är fel och fixa det senare för att hålla fokus på min uppgift.

Jag förstår också att man som utvecklare ska kunna vara snabb samt hålla tidsplaner. Som tidigare nämnt tror jag att detta är något man lär sig ju mer man övar på (d.v.s. gör i sitt jobb) och detta examensarbete var mitt första tidsestimerade projekt någonsin, så att jag

misslyckades med att tidsestimera betyder inte att jag är en dålig utvecklare, utan att jag behöver öva mig på att göra det mer.

Slutsats

En oväntad erfarenhet som jag fått är att använda mig mer av tidsfunktioner i JavaScript. Det är något jag läge undvikit eftersom jag tyckt att det är besvärligt. Det var besvärligt för mig även här och fick lägga mycket tid på det för att få fram rätt vecka och år i appen, i och med pagineringen fanns det inget datum utan mina vecko- och årsnummer var bara relativa till varandra. Jag kunde med hjälp av getWeek()-funktionen från date-fns översätta ett datum till en vecka men hade inget sätt att översätta en vecka till ett datum för att veta vilken vecka som kom före eller efter.

Jag har lärt mig mer och repeterat grundläggande JavaScript såsom higher order array methods som map, filter och reduce och andra sätt som jag sorterat, filtrerat och jämfört på.

Fler oväntade lärdomar är på det sätt jag tar tiden. I Toggl så underlättade det mycket att redovisa tid när jag lagt rubrikerna till de grupperade momenten före namnet på de specifika momenten. Då blir momenten rätt grupperade i översikten för projektet i Toggl, om man sorterar de i alfabetisk ordning. I stort så lärde jag mig även att använda Toggl på ett bättre sätt och genom att klocka min tid har jag lärt mig att det oftast går kortare tid än vad det känns som, så det är alltid bra att klocka sin tid om man vill veta hur lång tid som går i stället för att uppskatta hur mycket tid man lagt ner.

En sak som skulle hjälpt hade varit att skriva ännu tydligare commit-meddelanden, kanske med momentets rubrik och namn som första mening för att lättare koppla ihop de med de registrerade tiderna.

I övrigt har jag fått bekräftat det jag redan visste om mig själv: att jag är envis och tycker det är kul med problemlösning men också att jag i nuläget är mycket långsam samt osäker på att testa nya saker, vilket är något jag kommer behöva göra i mitt framtida arbetsliv. Jag känner dock fortfarande att programmering är för mig och att jag ser fram emot en framtida karriär inom detta.

Min slutsats blir att syftet är uppfyllt eftersom jag har en app som fyller syftet ”att förenkla planeringen av veckans måltider för folk som strävar efter att ha en mer strukturerad rutin för dessa, inklusive mig själv och mitt hushåll. Genom att kunna lägga in enkla rätter och automatiskt generera en veckomeny som man också kan anpassa till vad man har hemma ämnar jag […hjälpa användare att komma på och planera olika maträtter att lägga in i sin veckomeny]” (min anmärkning).

Ett ytterligare syfte ”är även att kunna visa upp flera av mina förvärvade praktiska kunskaper från Front End Developer-utbildningen”, vilket jag har gjort.

Målbilden från förstudien är dock inte helt men delvis uppfylld, eftersom det var flera moment som jag inte hunnit med.

Koncept och terminologi

Term	Förklaring
Vite	en lokal utvecklar-server som gör att man kan se sin kod live på en ”hemsida”
TypeScript	typsäker Javascript
React	JavaScript-ramverk
npm	Node package manager: ett pakethanteringsverktyg för JavaScript och Node.js som tillhandahåller användbara paket.
React Router	ett verktyg för att navigera runt på en hemsida utan att ladda om sidan
Route	en av sidorna man kan navigera till i ovan beskrivning
Firebase	en molnbaserad plattform från Google som erbjuder tjänster för utvecklare
Firebase Auth	autentisering via Firebase tjänst
Firebase Database	en databas som Firebase tillhandahåller
Firebase Storage	storage – kan lagra filer, t ex bildfiler
collection	en kollektion i databasen (ungefär som en mapp i datorn) som omsluter alla dokument av samma typ
document	ett objekt eller värde i kollektionen som är fyllt med data som
Zod	ett bibliotek för validering i TypeScript
React Hook Forms	ett bibliotek för att hantera formulär i React-applikationer.
custom hook	återanvändbara funktioner i React som bara kan användas i React-komponenter
sign-up	skapandet av en ny användare
sign-in	inloggning
pill	ovalformad knapp som används för att visa vilka val som är valda
wireframe	en skiss över hur en kommande app eller projekt ska se ut
rendering	det som koden får att visas på skärmen
conditional rendering	villkorlig rendering – element visas på skärmen endast om villkoren för att visa de är uppfyllda

queries	förfrågningar till databasen om att returnera den specifika efterfrågade datan
komponent	en byggsten i React med vilken man tillsammans med andra komponenter bygger upp en React-app
generisk komponent	en mycket återanvändbar och flexibel, generell, komponent
properties	egenskaper hos en komponent som skickas in när denna kallas på i en föräldra-komponent
Alert	en ruta med ett meddelande till användaren
CTA	call to action – ofta en knapp som får användaren att agera till att gå vidare i processen
bugg	från engelskans ”bug” – ett fel i koden som får appen att fungera på oönskade sätt
UX	user experience - användarupplevelse
UI	user interface – gränssnittet som användaren använder för att interagera med appen – allt det som syns
commit-meddelande	en beskrivning i fritext som laddas upp tillsammans med koden till den versionshantering med. git som utvecklaren använder
try-catch	kodstruktur för att hantera fel och undantag genom att "försöka" utföra en viss kod och "fånga" eventuella fel som uppstår
state (från useState)	dynamiskt datavärde i React-komponenter för att hantera förändringar och uppdateringar på ett reaktivt sätt

Referenser och källförteckning

1. **React Select.** ***React Select.*** **[Online] https://react-select.com/home.**
2. Summarize data with aggregation queries. ***Firebase documentation.*** **[Online] https://firebase.google.com/docs/firestore/query-data/aggregation-queries#use_the_count_aggregation.**
3. date-fns. ***date-fns.*** **[Online] https://date-fns.org/.**
4. docs/getWeek. ***date-fns.*** **[Online] https://date-fns.org/v3.2.0/docs/getWeek.**
5. ISO_week_date. ***Wikipedia.*** **[Online] https://en.wikipedia.org/wiki/ISO_week_date.**
6. Reusing Styles. ***TailwindCSS.*** **[Online] https://tailwindcss.com/docs/reusing-styles#extracting-components-and-partials.**
7. Instruction by Firebase employee on how to get random documents from a collection. ***Stack overflow.*** **[Online]**

https://stackoverflow.com/questions/46798981/firestore-how-to-get-random-documents-in-a-collection/46801925#46801925.

Deployad app: https://meal-plan-helper.netlify.app/?week=3&year=2024

Tid i Toggl: