2026

MAMACITA ORDERPLATTFORM

SLUTDOKUMENTATION OCH ANALYS

GRUPP 3

Alex Kinnander, Clara Westberg, Emma Tufvesson,
Frida Johansson, Jonas Rosén och Oriana Lama

Innehåll

1. Inledning

Detta dokument sammanfattar teamets arbete med Mamacita orderplattform, en demobar MVP för ett digitalt beställningsflöde i snabbmatsmiljö. Projektet har genomförts enligt Scrum i sex sprintar, där fokus har legat på att planera, prioritera, leverera inkrement och förbättra arbetssättet över tid. Dokumentationen visar hur produktmålet har styrt prioriteringar, hur backloggen har förändrats, hur teamet hanterat förändrade förutsättningar samt hur budget och velocity har använts som beslutsunderlag.

2. Produktmål och prioriteringar genom projektet

Projektet har utgått från ett gemensamt produktmål: att möjliggöra expansion till fler restauranger genom ett sammanhängande och anpassningsbart orderflöde som minskar manuella steg och fel.

Detta mål har fungerat som en riktning genom hela projektet och påverkat hur vi har prioriterat i varje sprint.

I början av projektet (sprint 1–3) låg fokus på att få till ett fungerande grundflöde. Vi prioriterade att kunden skulle kunna lägga en beställning, att ordern skulle tas emot och att köket skulle kunna se och hantera den. Här handlade det främst om att snabbt få fram en MVP, snarare än att bygga något avancerat.

I sprint 4 fick vi en förändring där vi behövde ta bort viss funktionalitet och i stället förenkla beställningsflödet. Vi valde då att anpassa valmöjligheterna efter tid på dygnet. Det gjorde flödet enklare och mer standardiserat, vilket vi kopplade till produktmålet eftersom det minskar risken för fel och gör systemet lättare att använda i flera restauranger.

I sprint 5 valde vi att fokusera på design och kundupplevelse i stället för att bygga ny funktionalitet. Anledningen var att vi ansåg att vår MVP redan fungerade tillräckligt bra. Genom att förbättra designen kunde vi öka värdet utan att göra systemet mer komplext. Samtidigt valde vi bort vissa tekniska förbättringar, vilket var en medveten prioritering utifrån tid och värde.

I sista sprinten ligger fokus på att färdigställa leveransen. Det innebär att vi ska kunna visa ett fungerande system, dokumentera vårt arbete och visa att lösningen går att anpassa till olika restaurangkoncept.

Sammanfattningsvis har våra prioriteringar förändrats under projektet. I början handlade det om att få något att fungera, medan vi senare fokuserade mer på att förenkla, förbättra och till slut visa värdet av det vi byggt. Produktmålet har varit en viktig hjälp i dessa beslut, särskilt när vi behövt välja mellan olika alternativ och anpassa oss efter förändringar.

3. Backloggens förändringar genom projektet

I början av projektet arbetade vi med att ta fram en övergripande roadmap och en preliminär backlogg för alla sprintar. Den första versionen blev ganska bred och ofullständig eftersom kravbilden fortfarande var oklar och teamet ännu inte hade full förståelse för hur stort varje backlog item faktiskt var. Vi valde därför att inte låsa alla user stories och acceptanskriterier från start, utan att planera mer detaljerat sprint för sprint utifrån produktmålet och det värde Product Owner bedömde var viktigast just då.

Det gjorde också att backloggen i början upplevdes som otydlig. Teamet saknade till viss del transparens kring vad som väntade i kommande sprintar och vilka delar som var beroende av varandra. För att skapa mer struktur började vi därför arbeta med tydligare epics och features, där exempelvis kundvy, köksvy, orderflöde, API och DevOps kunde kopplas till produktmålet. På så sätt gick backloggen från att vara en ganska bred plan till att bli ett tydligare stöd för planering och prioritering.

Ett konkret exempel på hur backloggen förändrades var sprint 3, när vi insåg att orderflödet inte gick att testa ordentligt eftersom orderdata försvann vid byte av vy eller uppdatering av sidan. Då prioriterades localStorage och inläsning från JSON-fil upp, även om det inte var funktioner som kunden såg direkt. Beslutet togs eftersom de delarna minskade risken i projektet och gjorde det möjligt att visa ett sammanhängande orderflöde från kundvy till köksvy.

Den största förändringen skedde i sprint 4, när nya förutsättningar gjorde att teamet behövde anpassa lösningen till rusningstid och faktisk användning under stress. Backlog items som handlade om vidareutveckling och design prioriterades då ned, medan förenkling av menyutbud och regler för rusningsläge prioriterades upp. Fokus i backloggen förändrades alltså från att bygga mer funktionalitet till att få lösningen att fungera bättre i ett verkligt restaurangflöde. Teamet accepterade samtidigt risken att vissa kundval och förbättringar fick vänta, eftersom snabbare flöde och färre fel bedömdes skapa mer värde.

Allt eftersom projektet fortsatte blev teamet bättre på att skriva user stories och acceptanskriterier på ett konkret sätt. En viktig lärdom kom från att flera i teamet hade arbetat både som utvecklare och som Product Owner. När man själv suttit i utvecklarrollen blev det tydligt hur viktigt det är att backlog items är begripliga, avgränsade och möjliga att testa mot Definition of Done. Det gjorde att senare sprintar kunde planeras mer träffsäkert och att backloggen blev mer transparent för hela teamet.

Sammanfattningsvis förändrades backloggen från en övergripande plan till ett mer levande stöd i projektet. Den användes inte bara för att lista vad som skulle byggas, utan också för att prioritera utifrån produktmål, risk, värde och teamets kapacitet. Den viktigaste lärdomen var att en backlog inte behöver vara perfekt från början, men att den måste utvecklas när teamet lär sig mer och när förutsättningarna förändras.

4. Analyserat prioriteringsbeslut

Under sprint 5 prioriterade vi mellan att vidareutveckla ny funktionalitet eller att i stället förbättra designen av kund vyn, där vi valde att förbättra designen av kund vyn.

Funktionalitet kopplad till flaskhalsar och datainsamling har ett tydligt tekniskt och operativt värde, då det kan bidra till bättre prestanda, insikter och långsiktig optimering av systemet. Samtidigt hade teamet redan implementerat ett fungerande end-to-end-flöde där beställningar kunde genomföras och hanteras i systemet.

Designförbättringar av kund vyn bedömdes däremot ha ett högre direkt användarvärde i detta skede, eftersom de påverkar hur kunden upplever systemet.

Förbättrad användbarhet, tydligare interaktioner och visuella justeringar bidrar till en mer intuitiv beställningsprocess, vilket kan minska fel och öka konvertering.

Vi valde därför att prioritera designförbättringar framför ytterligare funktionell utveckling. Beslutet baserades på att MVP:n ansågs vara tillräckligt stabil, och att ytterligare tekniska förbättringar skulle ge mindre värde jämfört med att förbättra användarupplevelsen inför slutleveransen.

Detta innebar en medveten risk då vi valde bort vissa tekniska förbättringar. Vi bedömde dock att denna risk var acceptabel, eftersom systemets kärnflöde redan fungerade och fokus i stället kunde läggas på att stärka helhetsupplevelsen för användaren.

5. Förändringen i sprint 4

I starten av sprint 4 kom en uppdatering kring projektets förutsättningar från ledningsgruppen. De tog beslutet att göra en fokusförändring där det inte längre räckte att utveckla en korrekt, logisk och komplett lösning. Lösningen behövde även kunna fungera vid begränsad uppmärksamhet, stress och tidspress för kökspersonalen. Fokuset för denna sprint övergick därför från det planerade sprint målet "säkerställa flöde och design" till att anpassa flödet enligt fokusförändringen. Därför uppstod behovet av göra en stor förändring i sprintens planering och de beslut som behövdes tas i den.

Eftersom förändringen krävde en tydlig förändring/bortprioritering och inte tillägg av funktionalitet behövdes flera av de kort som redan lagts i sprintbackloggen angående ny funktionalitet tas bort för att följa dessa krav. De kort som kvarstod var därför vår lösning på förändringen och två kort gällande design för kund vy och köks vy. När utvecklarna väl började med förändringslösningen så tog den mer tid att göra vad, vilket ledde till att även kund vy design och köks vy design behövde prioriteras bort på grund av tidsbrist i sprinten. Detta innebar att den ursprungliga sprintplaneringen och dess kort flyttades helt tillbaka till backloggen och blev ersatta med den nya förändringslösningen. Detta gav en tydlig påverkan på planeringen då den behövde bytas ut helt för att kunna genomföra lösningen inom sprintens tidsgränser.

Den förändringslösning som implementerades vid sprint 4 var att begränsa urvalet av produkter inom specifika timmar, vid lunchrush och middagsrush. Syftet med detta var att minska den kognitiva belastning som kökspersonalen beskrivs uppleva under rusningstider. Valet att minska antalet olika burgare, drycker och tillbehör, samt ta bort möjlighet till tillägg leder till mindre valmöjligheter för kunden, men även mindre variation i den information som köket behöver ta emot. Detta skulle i sin tur minska risken för fel och därmed förbättra både kundnöjdhet, då de får rätt beställning, och minska personalstress och fel.

Denna förändringslösning gav tydliga förändringar i både planering och beslut och gav sprinten en helt ny riktning. Trots att det skedde stora förändringar under sprintens gång som krävde ny planering och effektivt beslutsfattande så kunde förändringslösningen implementeras korrekt och inom sprintens ramar och krav.

6. Förändringen i sprint 5

Inför sprint 5 fick teamet en uppdatering ifrån styrelsen på Mamacita ett annorlunda jämfört med de tidigare uppdateringarna. De tidigare uppdateringar har handlat om vad som ska prioriteras bort eller hur den befintliga lösningen behöver justeras. Denna sprints uppdatering handlade om en kommande tid med bristfällig information, detta menades med att fullständig information inte kommer att finnas innan teamen ska ta beslut. Tydligt från styrelsen var att dem förväntar sig att teamet tar ansvar när instruktioner saknas, vi förväntades under denna sprint ta ett eget initiativ.

Detta initiativ skulle inte finnas med i uppgiften utan det skulle vara ett initiativ som teamet ansåg behöver tas i projektets nuvarande läge. Initiativet skulle tas för att förstärka värdet på projektet.

Så jämfört med tidigare förändringar där det funnits tydliga instruktioner på vad som förväntas göras är detta något som teamet helt själva skulle komma fram till. Med detta så blir det ett aktivt vägval för teamet som kommer med risker och osäkerheter om vad som är rätt beslut eller inte.

Det som vi i teamet kom fram till att göra under denna sprint var att läggas fokus på design i kund vyn. Detta initiativ grundar sig i att den kund vy som var befintlig i vår

lösning var väldigt grundläggande och detta för att vi som team har vart tydliga med att vi har gått efter en MVP med fokus på funktionalitet och därför under tidigare sprintar prioriterat ned designen. I uppdateringen fanns det mycket som tryckte på att ta ett initiativ som höjde projektets värde. En designad kund vy gör höjer snabbt kundvärdet inte bara utseendemässigt utan även tillgänglighet på knappar och texter. Designen skulle vara ganska enkel men den skulle följa WCAG 2.1 vilket gör det enklare för kunderna att följa sin order från start till slut. Vi ansåg även att den nuvarande lösning utifrån en MVP är bra tillräcklig gjorde de enklare för oss att ta detta initiativ och även kunna motivera vårt beslut om att göra ett mindre initiativ såsom detta.

Med denna uppdatering och initiativ inför sprinten behövdes några kort som låg i backloggen och som var prioriterade för denna sprint prioriteras ned. Detta gjorde att planeringen för denna sprint förändrade sig, sprintens mål var innan uppdateringen kom ”säkerställa att flödet fungerar korrekt” men som efter uppdateringen ändrade sig. Denna sprint skulle alltså egentligen handla om att testa flöde, korrigerar buggar och även uppdatera befintliga funktioner.

Ur ett budgetperspektiv så blev denna sprint billig då timmarna på utvecklarna hamnade under de budgeterade timmarna för dem, men det blev en ökning i timmar på teamets Product Owner. Totalt sätt så hamnade vi strax över 5000kr under budget för sprint 5.

Med denna uppdatering kom risker och osäkerheter, den risk vi tog var framför allt att ta ett så litet initiativ när det fanns fria tyglar till att göra väldigt mycket mer. Vi stod fast vi initiativet och motiverade detta på ett bra sätt. Osäkerheterna var hela uppdateringen, vi visste aldrig om detta var ett bra initiativ eller inte vilket gjorde att det alltid fanns en osäkerhet kring initiativet, men vi kom snabbt till att vi kunde motivera detta och framföra ett bra resonemang till varför vi har gjorde som vi gjorde.

7. Projektets ekonomi

Projektets ekonomi har baserats på ett resursupplägg där Product Owner och Scrum Master arbetat 25 % vardera, medan utvecklingsresursen arbetat 100 %. Det innebär att den största kostnaden har varit kopplad till utvecklingstid, medan styrning och

prioritering hållits på en lägre kostnadsnivå. I planeringen har sprintarna beräknats som fulla arbetsveckor om 40 timmar, utan hänsyn till röda dagar eller faktisk tillgänglighet. Den fullständiga budgetkalkylen med planerad kostnad, faktisk kostnad och avvikelse per sprint redovisas i kapitel 13.

Sett över projektet låg flera sprintar nära eller under budget, vilket visar att den övergripande planeringsmodellen fungerade som ekonomisk baslinje. Samtidigt blev det tydligt att timmar och kostnad inte ensamma visar projektets effektivitet. För att förstå utfallet behöver kostnaden sättas i relation till levererat värde, exempelvis färdigställda uppgifter, minskad risk eller beslut som stärker lösningens långsiktiga användbarhet. Detta syns särskilt i jämförelsen mellan faktisk kostnad per sprint och kostnad per levererad story, som visualiseras i kapitel 14.

Den högsta totala kostnaden uppstod i sprint 3. Det berodde på att teamet lade mycket utvecklingstid på att få ett sammanhängande orderflöde att fungera från kund vy till köks vy och upphämtning. Även om sprinten blev kostsam skapade den ett viktigt värde eftersom den gav projektet en fungerande grund att bygga vidare på. Sprint 3 visar därför att en hög total kostnad kan vara motiverad när sprinten levererar ett centralt inkrement som minskar risk och stärker projektets fortsatta genomförande.

Den tydligaste avvikelsen i relation till leverans finns däremot i sprint 4, där kostnaden per levererad story blev högst. Förändringen kom efter att sprinten redan var planerad och innebar att teamet inte skulle bygga ny funktionalitet, utan i stället förenkla det befintliga flödet för att fungera bättre under stress och tidspress. Planerat arbete flyttades därför tillbaka till backloggen och ersattes av arbete med konfigurerbar JSON-struktur och regler för hur systemet tolkar denna. Dessa uppgifter var tidskrävande och svåra att bryta ned i mindre delar, vilket gav låg velocity och högre kostnad per levererad uppgift.

Budgetöverskridandet i sprint 4 var ett medvetet beslut. Teamet bedömde att förändringen skapade ett större långsiktigt värde än vad kostnadsavvikelsen indikerade. Genom att styra produktutbud via JSON i stället för hårdkodad logik blev lösningen mer flexibel och lättare att anpassa. Detta visade sig senare i sprint 6, där samma grund kunde användas för att demonstrera ett annat restaurangkoncept genom konfiguration snarare än genom en separat ny lösning.

I sprint 5 och 6 valde teamet däremot medvetet att inte fortsätta utveckla ny funktionalitet i samma takt. Efter sprint 4 uppfyllde prototypen de ställda kraven för en demobar MVP, vilket gjorde att ytterligare funktionalitet hade inneburit ökad kostnad och risk utan att nödvändigtvis skapa motsvarande värde. Funktioner som fortsatt flaskhalslogik, mer avancerad datainsamling och ytterligare optimering av köksflödet prioriterades därför ned.

Detta visar hur budgeten påverkade prioriteringen direkt. I stället för att använda kvarvarande kapacitet till ej beställd funktionalitet valde teamet att hålla nere kostnaderna, stabilisera lösningen och fokusera på slutleverans, dokumentation och demo. Beslutet innebar att projektet inte försökte öka värdet genom mer scope, utan genom att maximera värdet av den MVP som redan fanns.

Sammantaget visar ekonomin att projektet inte bara har styrts utifrån kostnad, utan utifrån relationen mellan kostnad, risk och värde. Sprint 3 visar att en hög total kostnad kan vara motiverad när den skapar ett centralt inkrement och möjliggör fortsatt framdrift. Sprint 4 visar att även hög kostnad per levererad story kan vara motiverad när beslutet stärker lösningens långsiktiga användbarhet. Sprint 5 och 6 visar motsatsen: att det ibland är mer ekonomiskt rationellt att avgränsa scope och undvika nyutveckling när produkten redan uppfyller sitt syfte. På så sätt har budgeten använts som ett aktivt beslutsunderlag, inte bara som en efterhandsredovisning. Detaljerna bakom utfallet redovisas i budgetkalkylen i kapitel 13 och visualiseras i kapitel 14.

8. Velocity

I projektet har velocity använts som ett mått på leverans per sprint, baserat på antal färdigställda uppgifter. En uppgift har betraktats som klar först när den uppfyllt Definition of Done, vilket säkerställer att endast färdig och användbar funktionalitet inkluderas i mätningen. Detta följer uppgiftens definition där velocity mäts som antal färdiga backlogg uppgifter per sprint. Resultatet sammanställs i diagrammet i avsnitt 14.1, där planerade stories jämförs med färdigställda stories per sprint.

Sett över projektets sprintar går det inte att se en enkel trend där velocity successivt ökar eller stabiliseras över tid. I stället visar diagrammet i avsnitt 14.1 att teamet redan

i början av projektet hade en relativt hög leveransförmåga, särskilt i sprint 1–3. Förutom sprint 4 stämde planerade stories dessutom ganska väl överens med färdigställda stories, vilket tyder på att teamets sprintplanering i huvudsak varit realistisk. Den största skillnaden mellan plan och utfall uppstod i sprint 4, där förändrade förutsättningar gjorde att planerade backlogg items behövde pausas eller omprioriteras.

Den mest framträdande avvikelsen återfinns i sprint 4, där velocity är markant lägre än i övriga sprintar. Endast ett fåtal uppgifter färdigställdes, samtidigt som flera planerade uppgifter pausades. Vid en första analys kan detta tolkas som låg effektivitet, men en djupare genomgång visar att detta snarare är ett resultat av ett medvetet beslut. Under sprinten genomfördes en förändring i projektets inriktning som innebar att fokus flyttades från att leverera ny funktionalitet till att hantera hur systemet används under stress. De uppgifter som genomfördes var tidskrävande och svåra att bryta ned i mindre delar, vilket innebär att ett velocity-mått baserat på antal uppgifter blir missvisande.

Detta belyser en central begränsning i velocity som mätetal. Velocity mäter mängden levererad output, men fångar inte alltid komplexitet eller det långsiktiga värdet av det som byggs. I sprint 4 levererades få uppgifter, men dessa utgjorde en grundläggande förändring av systemets arkitektur genom införandet av en konfigurerbar JSON-struktur och tillhörande regelhantering. Samtidigt visar diagrammet i avsnitt 14.3 att kostnaden per levererad story blev hög just i denna sprint, vilket ytterligare visar att velocity behöver analyseras tillsammans med kostnad, risk och värde.

I de senare sprintarna syns inte främst en ökad velocity, utan snarare ett skifte i typ av arbete. Efter sprint 4 hade teamet redan byggt stora delar av det funktionella flödet, vilket gjorde att sprint 5 och 6 kunde fokusera mer på förbättringar, design, dokumentation, anpassning till nytt restaurangkoncept och slutleverans. Att velocityn är lägre i dessa sprintar betyder därför inte nödvändigtvis att teamet levererade mindre värde, utan att arbetet bestod av färre men mer avgränsade uppgifter. Detta visar att velocity behöver tolkas tillsammans med sprintmål och innehåll, inte enbart som antal färdiga stories.

En ytterligare faktor som påverkar tolkningen av velocity i detta projekt är användningen av AI som utvecklingsresurs. Till skillnad från traditionell utveckling är kopplingen mellan tid och leverans inte linjär, då vissa uppgifter kan lösas mycket snabbt medan andra kräver flera iterationer, manuell testning och förfining av prompts. Detta syns i diagrammet i avsnitt 14.2, där förbrukade timmar jämförs med antal färdiga stories. Diagrammet visar att högre tidsåtgång inte automatiskt innebär fler färdigställda stories, vilket gör att velocity i detta projekt främst bör ses som ett relativt mått för jämförelse mellan sprintar, snarare än ett absolut mått på produktivitet.

Sammantaget visar analysen att velocity är ett användbart verktyg för att identifiera mönster och avvikelser i projektets genomförande, särskilt i kombination med kontextuell förståelse. Sprint 4 utgör ett tydligt exempel där låg velocity inte motsvarar lågt värdeskapande, medan de senare sprintarna visar hur en stabil och högre velocity kan uppnås när osäkerheten minskar. Detta understryker vikten av att komplettera velocity med kvalitativa analyser för att få en rättvisande bild av projektets effektivitet och värdeleverans.

9. DevOps strategi

9.1 Build

Build-steget ska säkerställa att systemet byggs på ett enhetligt, automatiserat och kvalitetssäkrat sätt inför varje uppdatering, vilket minskar risken för fel i orderflödet och bidrar till en stabilare leverans till verksamheten.

All kod integreras och byggs automatiskt via en CI-process, exempelvis i GitHub Actions, där varje förändring automatiskt valideras innan den går vidare i leveransflödet. Vid varje förändring installeras beroenden, applikationen byggs och grundläggande kontroller genomförs för att säkerställa att systemet fungerar som förväntat innan det påverkar andra delar av systemet eller når användaren.

Eftersom systemet i stor utsträckning styrs av konfigurationsdata, såsom menystruktur och stationslogik i JSON-format, inkluderas även validering av denna data i build-processen. Detta innebär att felaktiga eller ofullständiga konfigurationer upptäcks tidigt, innan de påverkar funktionaliteten i köks- och kundvyer.

Build-steget innefattar även automatiserade tester och kodkontroller, såsom enklare tester av orderflödet samt kontroll av kodstandard.

Den färdiga versionen av systemet paketeras därefter som en enhetlig artefakt, exempelvis via Docker, vilket möjliggör att samma version kan användas i alla miljöer (test, demo och produktion), och därmed minskar risken för miljörelaterade fel vid leverans.

En automatiserad build-process möjliggör kortare feedbackloopar, vilket innebär att fel upptäcks tidigt och kan åtgärdas innan de påverkar flera delar av systemet. Detta minskar risken i projektet och skapar bättre underlag för prioritering och beslut under projektets gång. För teamet innebär detta en högre leveransförmåga eftersom osäkerhet minskar och arbetet blir mer förutsägbart.

Build-processen kan liknas vid ett restaurangkök där varje rätt måste tillagas enligt ett standardiserat recept innan den serveras.

9.2 Test

Vi säkerställer kvaliteten genom att kontinuerligt integrera testning och kvalitetskontroller under hela projektets utvecklingsprocess. Testerna innefattar exempelvis enhetstester, integrationstester och säkerhetstester, vilka sker automatiskt som en inbyggd del av utvecklarflödet.

Att dessa tester sker kontinuerligt i stället för att vara ett sista steg i slutet gör att fel upptäcks tidigare och hinner åtgärdas innan de förvärras. Detta innebär snabbare feedback för utvecklarna kring kodkvalitet, minskning av antalet fel i produktionen och mer effektiv leverans, då de automatiserade testerna möjliggör mindre manuellt arbete.

Faktumet att de automatiserade testerna är integrerade i CI/CD-pipelinen och körs vid varje kodändring ger trygghet och transparens i koden som levereras, eftersom buggar identifieras direkt när koden skrivs och säkerställer att ny kod inte förstör befintlig funktionalitet.

Detta ger bättre samarbete och ansvarsfördelning mellan utvecklare och testare, förbättrad stabilitet, snabbare feedback och ett mer kostnadseffektivt arbetssätt.

Det är även tack vare vårt agila arbetssätt som vi säkerställer kvaliteten, då våra Scrum ceremonier som exempelvis stand up, retrospektiv och sprint planning utgör en kontinuerlig utvärdering av den kod som produceras.

9.3 Release

Syftet med en release i detta projekt är att så tidigt som möjligt kunna testa flödet från en order i kund vyn till hur en beställning ser ut i köks vyn. Detta för att kunna se hur flödet är sammanhängande och fungerar tillsammans.

Varje release som görs är inte ett slutsteg utan används för att minska risken, få feedback och för att senare kunna prioritera i backlogg inför kommande sprint.

Tanken är att release genomförs i slutet av varje sprint efter sprint review där vi demar det vi har åstadkommit och kopplar det till sprint målet. Resultatet visas oavsett om funktionalitet är begränsad.

Ett inkrement anses releasebart när det uppfyller teamets Definition of Done:

- Kraven är uppfyllda enligt user story och acceptanskriterier
- Koden är skriven, granskad och godkänd
- Inga kritiska buggar finns kvar
- Dokumentationen är uppdaterad
- Produktägare har godkänt

Med ett sprintupplägg på 1 vecka reduceras risken då fel upptäcks tidigt och snabbt. Varje release bidrar också till prioritering i backloggen inför kommande sprintar.

9.4 Upptäckt av fel

Teamet behöver snabbt kunna upptäcka fel i en release, särskilt i vår beställningslösning där problem direkt påverkar kundernas möjlighet att beställa och därmed verksamhetens intäkter.

Genom kontinuerlig övervakning följer vi både systemets tekniska funktion och användarnas beteenden för att tidigt se avvikelser. Vi kombinerar data från användarflöden, tester och faktisk användning för att hitta både tekniska fel och problem som påverkar upplevelsen.

Tidigt upptäckta fel gör att vi kan minska påverkan på kunder, prioritera rätt åtgärder och fatta beslut baserat på verkliga data.

När ett fel uppstår ska projektledaren snabbt samla teamet, informera berörda och se till att felet hanteras strukturerat med fokus på att stabilisera systemet och förstå orsaken.

När situationen är löst går teamet igenom vad som hänt, varför felet inte upptäcktes tidigare och hur arbetssätten kan förbättras.

9.5 Rollback

En fungerande rollbackstrategi hjälper oss projektledare att minimera och hantera risker med teamets releaser, men också att hålla användarna nöjda. Detta gör vi genom att fokusera på många små och täta releaser i stället för stora uppdateringar.

Om vi släpper en ändring där något går fel är det enklare att hitta felet och snabbt åtgärda det.

I vår rollback-strategi använder vi tekniker såsom feature flags. Detta gör att vi kan rulla ut ny kod men hålla funktionen avstängd tills vi ser att allt fungerar som det ska. Om problem uppstår kan vi i stället stänga av funktionen utan att påverka hela systemet.

En viktig del med en bra rollback-strategi är att den skapar trygghet i teamet. När utvecklarna vet att det finns ett säkert sätt att backa om något går fel vågar de leverera oftare.

Utöver detta ger rollback även värdefulla data inför framtiden, exempelvis hur ofta vi behöver backa en release och hur snabbt problem löses. Detta gör att vi kan förbättra processen över tid.

10. API specifikation

Metod	Endpoint	Syfte	Kommentar
GET	/api/menu	Hämtar aktuella menydata, inklusive produkter, kategorier, priser, tillval, allergener, stationer, tillagningstider m.fl.	I prototypen hanteras detta via JSON-fil, men endpointen visar hur menydata skulle kunna hämtas i ett riktigt API.
POST	/api/orders	Skapar en ny order från kund vyn och skickar ordern vidare till orderflödet.	Ordern får ordernummer och innehåller valda produkter, anpassningar, tillägg, tillval, pris och stationsinformation.
GET	/api/orders	Hämtar aktiva ordrar som ska visas i orderkö och köksvyer.	Används av köksvyerna för att visa vilka beställningar som ska hanteras.
PATCH	/api/orders/{orderId}/grill-done	Markerar att grillstationens del av ordern är färdig.	Uppdaterar orderstatus så att servering/tillbehör kan se att grilldelen är klar.
PATCH	/api/orders/{orderId}/sides-done	Markerar att tillbehörsstationens del av ordern är färdig.	Uppdaterar orderstatus så att servering/grill kan se att tillbehörsdelen av ordern är klar.
PATCH	/api/orders/{orderId}/serving-done	Markerar att serveringsstationen har plockat ihop ordern.	Visar att ordern är redo att lämnas ut till kund.
PATCH	/api/orders/{orderId}/picked-up	Markerar att kunden har hämtat sin order.	Tar bort ordern från aktiva vyer och avslutar orderflödet.
POST	/api/checkout	Simulerar betalning innan ordern skickas vidare.	I MVP:n är betalningen mockad, men endpointen visar hur betalningssteget skulle kunna kopplas in.

API-specifikationen är framtagen på konceptuell nivå. I prototypen hanteras flödet förenklat med React Context, localStorage och JSON-data, men endpointsen visar hur systemets delar skulle kunna kommunicera i en mer produktionsnära lösning. Syftet är

att visa hur kundvy, orderkö, köksvyer och betalningsflöde hänger ihop i ett sammanhängande orderflöde.

11. Definition of Done

Kriterium	**Vad det innebär i projektet**
Kraven är uppfyllda enligt user story och acceptanskriterier	Backlog itemet levererar det värde och den funktion som var planerad.
Koden är skriven, granskad och godkänd	Lösningen är genomgången av teamet och bedöms fungera enligt syftet.
Inga kritiska buggar finns kvar	Funktionen kan användas i demo utan att orderflödet bryts eller ger felaktigt resultat.
Dokumentationen är uppdaterad	Relevant dokumentation, exempelvis promptlogg, AI-bilaga, sprintanteckningar eller backlogg, är uppdaterad.
Product Owner har godkänt	PO har kontrollerat att leveransen motsvarar prioritering, sprintmål och acceptanskriterier.

Teamets Definition of Done användes som en gemensam kvalitetsgräns för när ett backlogg item kunde räknas som färdigt. För att ett item skulle räknas som klart behövde kraven vara uppfyllda enligt user story och acceptanskriterier, koden vara skriven, granskad och godkänd, inga kritiska buggar finnas kvar, dokumentationen vara uppdaterad och Product Owner ha godkänt leveransen.

DoD var viktig eftersom den avgjorde vad som fick räknas in i sprintens leverans och velocity. Ett backlogg item som var påbörjat, delvis fungerande eller gick att visa i begränsad form räknades inte som klart om det inte uppfyllde samtliga kriterier. På så sätt fungerade DoD som ett stöd för kvalitet och gjorde att teamet kunde skilja på faktiskt levererat arbete och arbete som behövde flyttas vidare till kommande sprint.

I projektet blev detta särskilt tydligt när buggar i orderflödet påverkade om en funktion kunde betraktas som färdig. Exempelvis kunde en funktion inte räknas som helt klar om en produkt inte hamnade i rätt köksvy eller om flödet inte kunde demonstreras utan att brytas. DoD bidrog därför till mer realistisk uppföljning av både kvalitet, risk och velocity.

12. Sprintsammanfattning

Sprint	Scrum Master	Product Owner	Övrigt ansvar
Sprint 1	Frida	Emma	Alex ansvarade för ekonomi och utvecklade tillsammans med Jonas, Clara och Oriana.
Sprint 2	Oriana	Frida	Emma ansvarade för ekonomi och utvecklade tillsammans med Alex, Jonas och Clara.
Sprint 3	Clara	Oriana	Frida ansvarade för ekonomi och utvecklade tillsammans med Emma, Alex och Jonas.
Sprint 4	Jonas	Clara	Oriana ansvarade för ekonomi och utvecklade tillsammans med Frida, Emma och Alex.
Sprint 5	Alex	Jonas	Clara ansvarade för ekonomi och utvecklade tillsammans med Oriana, Frida och Emma.
Sprint 6	Emma	Alex	Jonas ansvarade för ekonomi och utvecklade tillsammans med Clara, Oriana och Frida.

12.1 Sprint 1

Sprintens mål var att skapa det första fungerande kundflödet för beställningar, ett enkelt och tydligt orderflöde där kunden kan välja produkter, göra anpassningar, se

priset och skicka sin beställning. Fokus låg helt på kund vyn och att få ett klickbart, sammanhängande flöde på plats, utan koppling till backend eller API.

Teamet levererade en fungerande MVP där kunden kan bygga en beställning, se totalpris i realtid, ändra val och få en bekräftelse när ordern skickas. Designen hölls medvetet enkel för att prioritera funktion och tydlighet. Detta knöt direkt an till produktmålet genom att skapa ett strukturerat och felminimerande orderflöde som kan skalas till fler restauranger.

Betalningsflödet valdes bort eftersom det inte var nödvändigt för sprint målet och skulle byggas i fel ordning. Sprinten gav också viktiga lärdomar kring arbetssätt, framför allt att Jira-strukturen behövde förenklas och att teamet behövde bättre förberedelse inför kommande sprintplaneringar.

Sammanfattningsvis uppfyllde sprinten sitt mål och lade grunden för ett tydligt, användbart och skalbart orderflöde.

Från retro:
Det blev tydligt att lo-fi wireframes behövs för att alla ska förstå flöden och acceptanskriterier bättre. Gruppen vill också undvika att fastna i långa diskussioner och i stället gå vidare snabbare. Kunskapen kring epics, user stories och acceptanskriterier behöver stärkas, liksom hur man promptar effektivt tillsammans. Fokus ska fortsatt ligga på funktion framför design. Dessutom behöver backloggen i Jira struktureras om och teamet enas om hur och när kort ska flyttas för att skapa ett tydligare och mer effektivt arbetsflöde.

12.2 Sprint 2

12.3 Sprint 3

Sprintmål: "att bygga ett orderflöde där kunden kan skapa en beställning som kan visas i kundkorgen. Därefter tas ordern emot och prioriteras så att kök och personal kan hantera den effektivt." Ordern skulle kunna följas från start till slut.

Leverans:

Som resultat av många uppgifter och problem med exempelvis testmiljön så gick leveransen över budget med 5725 kr. Beslutet att gå över budget motiverades av vikten i att kunna säkerställa en så komplett sprintleverans som möjligt under de rådande förutsättningarna. Att sprinten drog över budget kunde sedan balanseras av de andra sprintarna som i stället låg under budget, vilket gör att projektet ändå har en hållbar ekonomisk budget.

Sprint 3 inleddes med ett långt sprintplaneringsmöte som drog över tid till stor del på grund utav minimal förberedelse av sprintbacklog och sprint mål innan sprintplaneringen. Detta ledde till viss frustration och irritation i gruppen hos utvecklarna, då de la ner mer tid än planerat på planering, vilket PO och Scrum master kunde ha förberett bättre innan. Därför lade Scrum master till ett extra retromöte på måndagen för att samla upp gruppen och hantera konflikten rakt på i stället för att ignorera den. Under mötet gjordes retrospektivet mad, sad, glad, där alla gruppmedlemmar fick bidra med åsikter och tankar kring hur sprinten gått. Där lyftes problemet kring avsaknad av proaktivitet och tydlighet i PO och Scrum masters rollansvar. Diskussionen gjorde att rollansvar kunde cementeras och att en tryggare grund inför framtida sprintar kunde byggas för gruppen.

Denna markanta förbättring gav en tydlig effekt till följande sprint då rollerna och ansvarsfördelningen blev mer konkretiserade och en bättre grund i produktbackloggen gjorde det lättare för PO att förbereda material inför sprintplaneringen. Tack vare detta flöt planeringen på tydligt bättre än sprinten innan vilket upplevdes positivt av samtliga av gruppmedlemmarna.

12.4 Sprint 4

Sprint mål:

”Förenkla beställningsflödet genom att anpassa beställarens valmöjligheter baserat på klockslag”.

Sprint 4 var en sprint som blev påverkad av en stor uppdatering som handlade om att förändra flödet i kund vyn under rusningstid. Styrelsen såg att nuvarande lösning gjorde att kökspersonalen inte följer varje steg i beställningen. Vi fick inte lägga till

funktioner, vyer, mer information eller logik utan det var ett bortval som skulle göras för att fixa detta problem.

Detta gjorde att sprinten behövdes planeras om, prioriteras om och nya tankar med lösningar behövde komma fram. Alla i teamet började fundera på vad som skulle kunna vara en lösning på problemet. Till slut kom idén om att dra ned på valmöjligheterna för kunderna under rusningstid. Detta ledde till sprint målet ”Förenkla beställningsflödet genom att anpassa beställarens valmöjligheter baserat på klockslag”. Den här lösningen hjälpte köket på de sättet att antalet till val blev mindre för kunderna och det blev färre olika beställningar för köket att hantera.

Risken vi accepterade med detta var att kunder som vill ha alla valmöjligheter och som äter under dessa tider försvinner, medan vi prioriterade att kunna få ut de beställningar som görs under den här tiden i tid. Detta gör att kunder som äter under rusningstid kommer att få sin mat i tid och det i sig blir ett höjt kundvärde.

I slutet av sprinten hölls ett retrospective som hade fokus på stop, start och continue. Från denna retrospective tog vi med oss att vi som grupp ska fortsätta ha högt i tak, fortsätta med tydliga agendor för mötena. Proaktivitet kring mötena och arbetet tog vi med oss också, vilket hör ihop med tydliga agendor för mötet. Något som vi skulle sluta med var något som har hängt med oss under projektet vilket är att tvivla på oss själva. Vi skulle våga lita på de beslut vi tar och stå fast vid det.

12.5 Sprint 5

Sprint mål:
Komplettera ett fungerande flöde med design för att höja kundupplevelsen.

Leverans:
Under sprinten valde teamet att fokusera på att förbättra designen i kund vyn i stället för att utveckla ny funktionalitet. Detta inkluderade bland annat visuella förbättringar som bilder, tydligare knappar och bättre struktur, samt ökad tillgänglighet i gränssnittet.

Utöver detta löstes även en tidigare bugg i orderflödet, vilket säkerställde att beställningar hanteras korrekt genom hela systemet.

En viktig händelse under sprinten var att teamet migrerade kodbasen från Lovable till GitHub, vilket ökade förståelsen för systemets uppbyggnad och bidrog till arbetet med DevOps-strategin.

Valet att fokusera på design baserades på att teamet ansåg att den befintliga MVP:n redan hade ett tillräckligt fungerande flöde, och att förbättrad användarupplevelse därför skapade mer värde i detta skede.

Förbättring från retrospektiv:
En viktig lärdom från sprinten var vikten av att lita på tidigare beslut och att inte överanalysera nya initiativ. Teamet upplevde att tydligare riktning och snabbare beslutsfattande bidrog till en mer effektiv sprint.

Inför nästa sprint beslutade teamet att fortsätta arbeta med tydliga acceptanskriterier och att behålla ett strukturerat arbetssätt, särskilt med tanke på den ökade arbetsbelastningen kopplad till slutleveransen.

12.6 Sprint 6

Sprint mål:
Leverera en komplett och demobar slutprodukt samt dokumentation som visar systemets värde, skalbarhet och teamets Scrum-arbete.

Leverans:
I sprint 6 låg fokus på att färdigställa slutleveransen, förbereda presentationen och säkerställa att systemet kunde demonstreras som en sammanhängande MVP. Teamet arbetade med att visa ett komplett orderflöde från kund vy till köks vy och upphämtning, samt att färdigställa den dokumentation som visar hur projektet har styrts genom Scrum, prioriteringar, budget, velocity och hantering av förändringar.

En viktig del av sprinten var att visa att lösningen är anpassningsbar till olika restaurangkoncept. Eftersom vår lösning bygger på att menydata hämtas från en JSON-fil innebär anpassningen att ett nytt restaurangkoncept kan demonstreras genom att byta eller justera produkter, kategorier, stationer, tillval och tillagningstider i konfigurationen. På så sätt kan samma grundflöde återanvändas utan att en separat lösning behöver byggas för varje restaurangtyp.

Detta stärker kopplingen till produktmålet, eftersom lösningen inte bara visar ett fungerande orderflöde för ett koncept, utan även visar hur plattformen kan skalas och anpassas till fler restauranger. Sprinten bidrog därför framför allt till att visa systemets värde som en återanvändbar och konfigurerbar lösning, snarare än en engångslösning för ett specifikt koncept.

Förbättring från retrospektiv:
Retrospektivet i sprint 6 hinner inte genomföras innan uppgiftens deadline, men är inplanerat efter slutpresentationen. Syftet är att samla lärdomar från hela projektet på individ-, team- och projektnivå. Fokus för retrospektivet kommer att vara vad teamet tar med sig från projektets genomförande, vilka arbetssätt som fungerat bäst och vilka förbättringar som skulle behöva göras i ett framtida projekt.

Eftersom retrospektivet sker efter deadline redovisas det inte som genomfört i slutdokumentationen. Däremot är det en planerad del av teamets avslutande förbättringsarbete och kopplar till Scrum genom att teamet även efter sista sprinten reflekterar över arbetssätt, samarbete och lärande.

13. Budgetkalkyl

LÖNER

Roll	Månadslön	Timmar/mån	Timkostn	Timmar/
PO	55000	168	327	10
Scrum Master	55000	168	327	10
Utvecklare	48000	168	285	40

BUDGETPLANERING

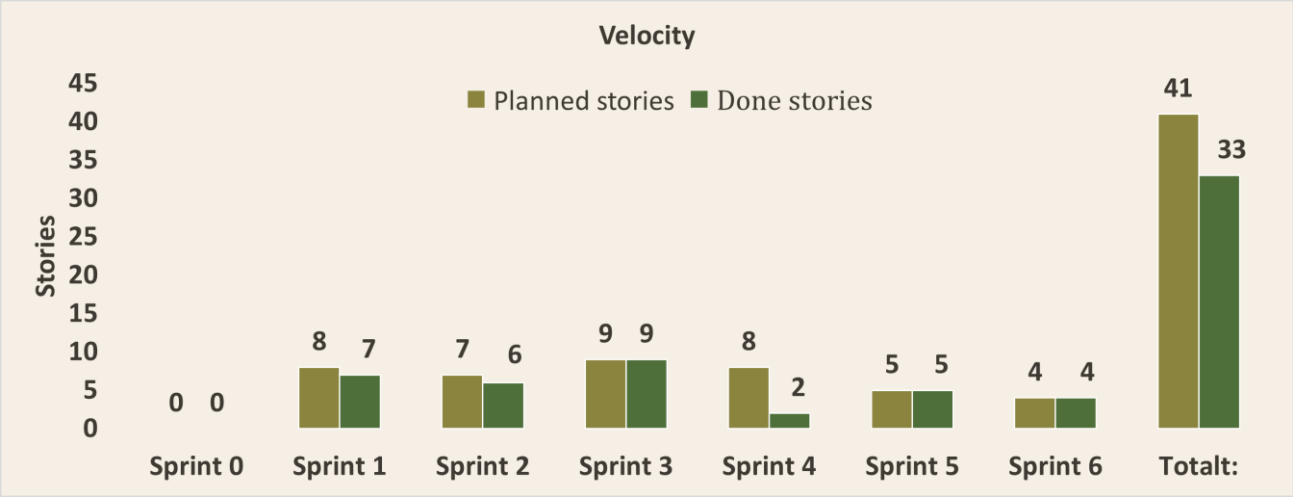
Sprint	PO timmar	SM timmar	Dev timm	PO kostn	SM kostn	Dev kostn	Total kostn
Sprint 0	10	10	0	3270	3270	0	6540
Sprint 1	10	10	40	3270	3270	11400	17940
Sprint 2	10	10	40	3270	3270	11400	17940
Sprint 3	10	10	40	3270	3270	11400	17940
Sprint 4	10	10	40	3270	3270	11400	17940
Sprint 5	10	10	40	3270	3270	11400	17940
Sprint 6	10	10	40	3270	3270	11400	17940
Totalt:	**70**	**70**	**240**	**22890**	**22890**	**68400**	**114180**

FAKTISK KOSTNAD

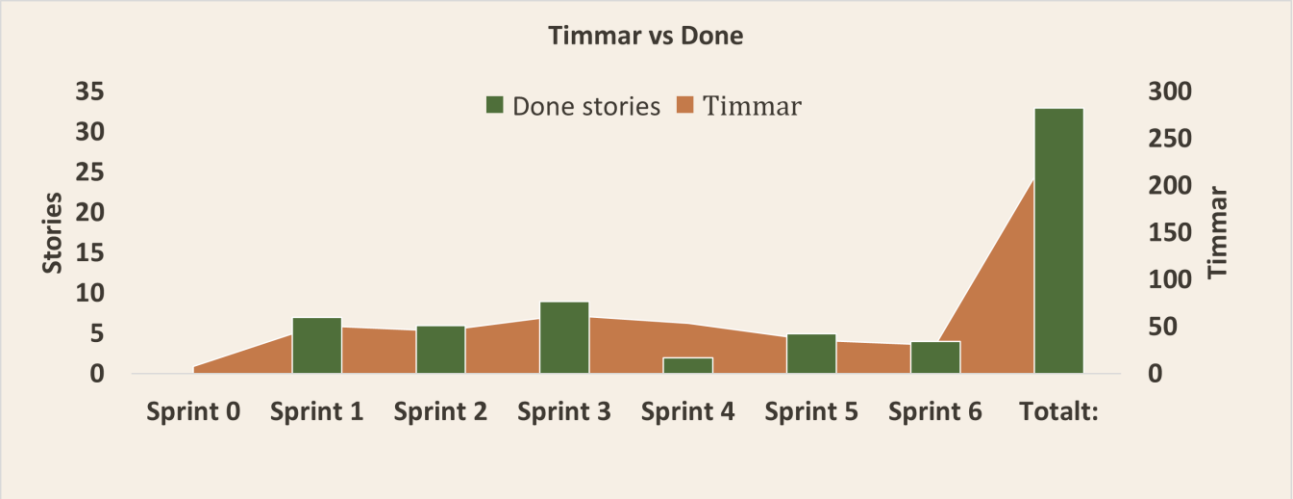
Sprint	PO timmar	SM timmar	Dev timm	PO kostn	SM kostn	Dev kostn	Total kostn	Utfall	Effekt
Sprint 0	5,75	5,75	3,75	1880	1880	1069	4829	-1711	Under budget
Sprint 1	8,5	7,5	44,5	2780	2453	12683	17915	-26	Enligt budget
Sprint 2	9,5	8,5	38,5	3107	2780	10973	16859	-1082	Under budget
Sprint 3	12,5	11,5	55,5	4088	3761	15818	23666	5726	Över budget
Sprint 4	10,5	9	45,5	3434	2943	12968	19344	1404	Över budget
Sprint 5	11,25	6,25	24,75	3679	2044	7054	12776	-5164	Under budget
Sprint 6	10,5	5,5	19,5	3434	1799	5558	10790	-7151	Under budget
Totalt:	**68,5**	**54**	**232**	**22400**	**17658**	**66120**	**106178**	**-8003**	**Under budget**

14. Diagram

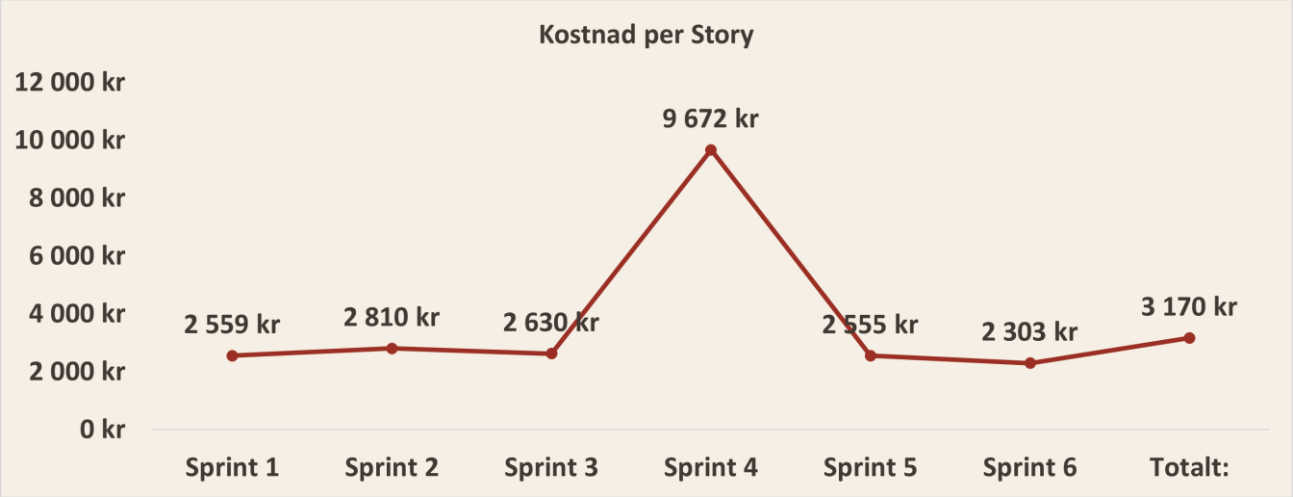
14.1 Velocity per sprint



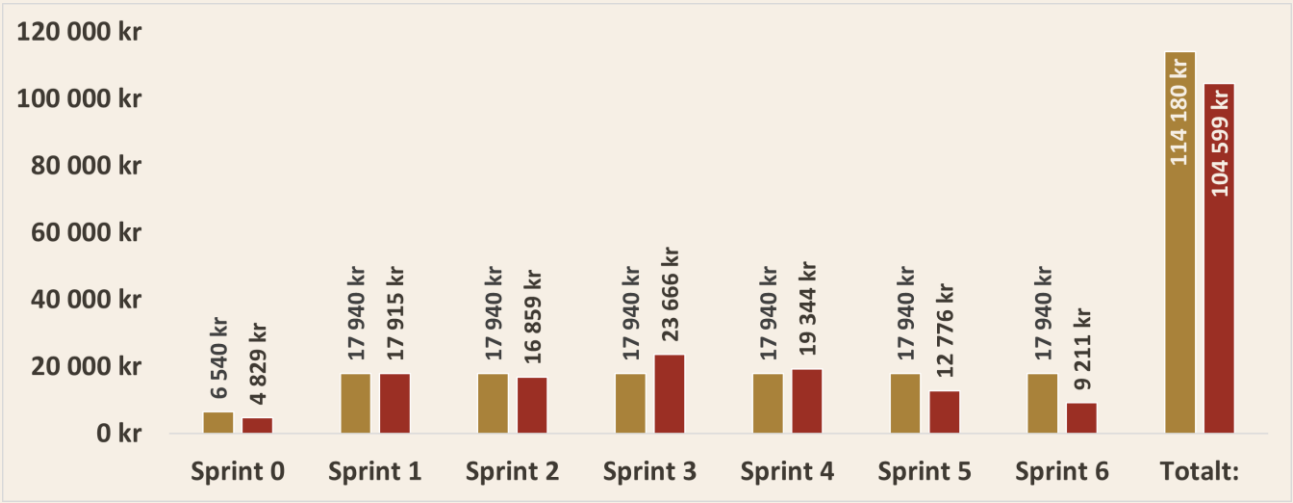
14.2 Förbrukade timmar och antal färdiga stories per sprint



14.3 Kostnad per levererad story



14.4 Budget vs faktisk kostnad per sprint



15. AI-bilaga

Sprint	Syfte / Funktion	Tid	Vad fungerade bra	Problem / Risker	Påverkan & Lärdomar
Sprint 1	Bygga första kundvyn för MFFO där kunden kan välja produkter, göra anpassningar och se prisuppdatering i realtid.	Estimat: 5 dagar Faktisk tid: 40 timmar	AI följde backlog och acceptanskriterier utan egna lösningar. Ett fungerande orderflöde skapades.	Funktioner saknades i kravspecen, exempelvis flera rätter i samma order, allergeninformation och tydligare anpassningar.	Gav en stabil grund för projektet. Teamet lärde sig vikten av tydliga Jira-kort och precisa prompts.
Sprint 2	Bygga köksvy och kundstatusvy med orderöversikt per station. Senare kompletterades sprinten med mockdata i alla vyer.	Estimat: 4-5 dagar Faktisk tid: 33 timmar	Samtliga vyer togs fram. Mockdata gjorde lösningen lättare att demonstrera och testa.	Information och ordrar försvann vid refresh eller byte av vy. Backend, persistent lagring och realtidsuppdatering saknades.	Tekniska risker identifierades tidigt. Teamet såg behov av bättre riskhantering och tydligare planering inför sprint reviews.

Sprint	Syfte / Funktion	Tid	Vad fungerade bra	Problem / Risker	Påverkan & Lärdomar
Sprint 3	Koppla ihop kundvy och köksvy till ett komplett orderflöde med lagring mellan vyer.	Estimat: 5 dagar Faktisk tid: 47 timmar (27+20)	Ordrar kunde följas genom hela flödet och sparas via Local Storage och JSON-data.	Fel lagringslösning gjorde att testmiljön inte fungerade initialt. Kategorier och tilläggspriser försvann efter ändringar i arkitekturen.	Skapade stort verksamhetsvärde genom fungerande helhetsflöde. Teamet lärde sig vikten av att säkra information vid tekniska förändringar.
Sprint 4	Anpassa produktutbud under rusningstid för att minska belastning på köket samt förbättra datahantering i orderflödet.	Estimat: 5,5 dagar Faktisk tid: 40 timmar	Tidsstyrning och begränsat produktutbud fungerade enligt plan. Efter förtydliganden förbättrades dataflödet.	Fel orderdata och anpassningar visades på fel stationer. En produkt visades inte alls i köket.	Risk för teknisk skuld och förseningar. Teamet såg behov av bättre förståelse för datalogik och mer precisa prompts.

Sprint	Syfte / Funktion	Tid	Vad fungerade bra	Problem / Risker	Påverkan & Lärdomar
Sprint 5	Fixa buggar och förbättra design samt användarupplevelse i kundvyn.	Estimat: 2,5 dagar Faktisk tid: 21,5 timmar	Buggen med “Crispy Butcher” löstes. Designen förbättrades med bilder, spacing, färgkodning och WCAG-anpassning.	Designpromptar lämnade stort tolkningsutrymme utan tydliga visuella referenser.	Minskad teknisk skuld och förbättrad användarupplevelse. Teamet lärde sig vikten av tydliga designkrav och referensmaterial.
Sprint 6	Implementera stöd för två restaurangkoncept via separata JSON-filer utan att ändra systemets struktur.	Estimat: 1,5 dagar Faktisk tid: 17 timmar	Två fungerande restaurangkoncept kunde visas och systemet blev skalbart.	Vissa produkter hamnade i fel kategorier men kunde korrigeras manuellt.	Viktigt för att uppfylla projektkraven och visa skalbarhet. Tidigare erfarenheter av JSON-struktur gjorde promptarna mer effektiva.

16. Risklogg

Riskloggen visar att projektets största risker har flyttat över tid. I början handlade riskerna främst om bristande struktur och otydliga kopplingar mellan systemets delar. Under sprint 3 blev de mest kritiska riskerna tekniska, särskilt kopplat till endpoints, JSON-data och testmiljö. I senare sprintar skiftade fokus mot leveransrisker, exempelvis buggar, DevOps-strategi, slutdokumentation och demo. Det visar att

teamet successivt gick från osäkerhet i planering och systemförståelse till riskhantering kopplad till färdigställande och slutleverans.

Sprint	Risk	S	K	RV	Åtgärd / hantering	Ansvar
Sprint 1	Ingen riskbedömning gjordes inför sprinten	–	–	–	Teamet tog lärdom och beslutade att arbeta mer aktivt med risklogg framåt	Team
Sprint 2	Många röda dagar kan påverka om sprinten hinns klart	–	–	–	Planera kapacitet mer realistiskt utifrån faktisk tillgänglighet	Team
Sprint 2	Kopplingen mellan kundvy och köksvy kan bli svår	–	–	–	Ha fokus på risklogg och beroenden framåt	Team
Sprint 3	Otydliga endpoints leder till felbygge	5	5	25	Förtydliga förväntade endpoints i prompten	Dev
Sprint 3	JSON-filen saknar nödvändig information	4	5	20	Säkerställa att all information finns med	Dev
Sprint 3	API:et får inte fram rätt data i systemet	3	5	15	Tydligare prompt kring ordernummer och datakoppling	Dev
Sprint 3	Dubbelarbete på grund av okunskap	4	3	12	Lägga tid på gemensam förståelse innan utveckling	PO + Dev
Sprint 3	Det vi bygger skapar inte tillräckligt värde	2	5	10	Använda prioriteringsmodell	PO
Sprint 3	Testmiljön/LocalStorage löser inte problemet	2	5	10	Läsa på, söka information och testa lösningen	Dev

Sprint	Risk	S	K	RV	Åtgärd / hantering	Ansvar
Sprint 3	För lite testning gör att buggar missas	4	2	8	Avsätta tid för testning och fler testare	PO
Sprint 4	Teamet överarbetar MVP med för komplex logik/design	3	3	9	Hålla sig till scope och MVP	Team
Sprint 4	Ändringar i JSON kan påverka andra delar och skapa fel	2	4	8	Gemensam förståelse och tydlig promptning	Dev
Sprint 4	Bristande förståelse för validering och testning	4	2	8	Läsa på om rimlig testning i projektet	Dev
Sprint 4	Nya ändringar/förutsättningar från Arwind missförstås	4	1	4	Läsa och förstå ändringen innan sprintarbetet startar	Team
Sprint 5	Identifierad bugg löses inte	3	5	15	Tydlig frågeställning och promptning till AI-utvecklaren	Dev
Sprint 5	Grillkapaciteten fortsätter vara en utmaning	2	3	6	Få feedback från kund/pilot	PO
Sprint 5	Designen uppfyller inte WCAG-krav	1	4	4	Tydligt acceptanskriterium kring WCAG	PO
Sprint 5	DevOps-strategin drar ut på tiden	2	2	4	Ta höjd för arbetet inför kommande sprint	Team
Sprint 6	JSON-filen läses inte in korrekt och stör flödet	3	4	12	Vara noggranna med JSON-strukturen	Dev

Sprint	Risk	S	K	RV	Åtgärd / hantering	Ansvar
Sprint 6	Slutdokumentationen saknar viktiga delar	2	5	10	Teamet ses och stämmer av vad varje medlem producerat	Scrum
Sprint 6	Demo-miljön fungerar inte för båda restaurangkoncepten	2	4	8	Backup-plan med wireframes som visar konceptlösningen	Dev
Sprint 6	Demo under slutpresentation fallerar	1	4	4	Testa demon innan presentationen	Dev