# Inlämningsuppgift 2

## Projekt: Bankapplikation

### Inledning

**Inledning**

Detta projekt syftar till att utveckla en säker och användarvänlig bankapplikation som gör det möjligt för kunder att hantera sina konton, genomföra överföringar, betala räkningar och granska sin transaktionshistorik. Applikationen ska vara tillgänglig både via webben och mobilplattformar.

**Val av metodologi**

För detta projekt har vi valt att tillämpa den agila modellen eftersom den ger flexibilitet, möjliggör snabb anpassning till förändrade krav och främjar en iterativ process med kontinuerlig feedback från intressenter. Den agila processmodellen består av flera iterativa cykler, kallade "sprintar", där varje sprint resulterar i ett potentiellt levererbart produktinkrement. Nedan beskrivs de relevanta faserna och stegen i den agila processen som är tillämpliga för detta projekt.

### Planeringsfas

**Planeringsfas**

Planeringsfasen är den första och en av de mest kritiska faserna i ett systemutvecklingsprojekt. Här läggs grunden för hela projektets framgång genom att klargöra mål, förväntningar och arbetsprocesser. Ett välplanerat projekt säkerställer att alla involverade parter har en gemensam förståelse av projektets omfattning och att nödvändiga resurser identifieras och fördelas effektivt.

1. **Identifiering av intressenter**
   * **Kunder**: De personer som kommer att använda bankapplikationen. Deras behov och användarupplevelse är centrala delar att undersöka.
   * **Utvecklingsteamet**: Inkluderar utvecklare, testare och designers som har huvudansvaret för att bygga, testa och säkerställa bankapplikationen.
   * **Produktägare**: Kunden som beställer produkten. Det är viktigt att produktägaren är delaktig så att produkten utvecklas enligt deras behov.
   * **Intresseanalys**: Efter identifieringen analyseras varje intressents påverkan och intresse i projektet. Detta görs för att säkerställa att rätt mängd uppmärksamhet och resurser riktas mot de mest kritiska intressenterna. En intressentkarta eller matris kan användas för att visualisera detta.
2. **Kravinsamling**
   * **Workshops**: Genomföra workshops och intervjuer med intressenter för att samla in och prioritera funktionella och icke-funktionella krav. Detta handlar om att samla information, identifiera problem och olika krav samt prioritera dem.
   * **Funktionella krav**: Specifika funktioner som systemet måste ha, som användarregistrering, varukorgsfunktionalitet och betalningshantering.
   * **Icke-funktionella krav**: Krav på systemets prestanda, säkerhet, användarvänlighet och skalbarhet. Exempelvis att webbplatsen måste ladda inom två sekunder eller vara tillgänglig dygnet runt.
3. **Skapa Product Backlog**  
   Efter kravinsamlingen organiseras datan i en **Product Backlog**. Kraven bryts ner till **user stories**, som beskriver funktioner från ett användarperspektiv. Exempel: "Som kund vill jag kunna lägga till produkter i min varukorg så att jag kan köpa dem senare."
   * **Prioritera Backlog**: Backloggen prioriteras så att de mest värdefulla user stories hamnar högst upp, vilket gör att teamet kan fokusera på de viktigaste delarna av projektet.
   * Backloggen är ett levande dokument som uppdateras regelbundet baserat på nya krav eller feedback från intressenter.

### Sprinter

En sprint är en tidsbestämd och iterativ arbetscykel inom agil utveckling, vanligtvis 1–4 veckor lång. Under en sprint fokuserar teamet på att utveckla och leverera ett specifikt produktinkrement baserat på uppgifter från backloggen. Varje sprint inleds med ett planeringsmöte och avslutas med en granskning och retrospektiv.

### Sprint 1: Användarhantering och inloggning

* **Mål:** Utveckla och testa användarregistrering och inloggningsfunktioner med grundläggande säkerhet.
* **Viktiga uppgifter:**
  + Implementera användarregistrering och inloggning: Här handlar det om att skapa ett formulär där användaren kan ange sina uppgifter, såsom namn, e-postadress och lösenord. Formuläret ska vara enkelt och användarvänligt. När användaren skickar informationen, ska den överföras till servern och lagras i databasen. Ett API ska utvecklas för att hantera skapandet av nya användare och säkerställa att deras uppgifter sparas på ett säkert sätt i databasen. Lösenord ska aldrig sparas i klartext, utan krypteras.
  + Implementera säkerhetsfunktioner som lösenordshantering och kryptering: Det är viktigt att använda en krypteringsalgoritm för att skydda lösenordet innan det sparas i databasen. All känslig användardata bör lagras säkert, vilket inte bara gäller lösenord utan även e-postadresser och telefonnummer. För att öka säkerheten kan tvåfaktorsautentisering implementeras. Detta innebär att användaren, förutom lösenordet, även måste ange en kod som skickas till deras e-post eller telefon.
  + Grundläggande användargränssnitt för registrering och inloggning: Gränssnittet ska vara enkelt att navigera och responsivt för både mobil och webb. Det är viktigt att tydliga instruktioner ges, som exempelvis "Registrera dig" och "Logga in". Gränssnittet bör även vara tillgängligt för alla användare, inklusive de med funktionsnedsättningar. Detta kan uppnås genom att göra gränssnittet kompatibelt med skärmläsare för att ytterligare förbättra användarvänligheten.
  + Därefter genomför man enkla tester för att säkerställa att funktionerna fungerar som de ska.
* **Leveransmål:**
  + Fungerande system för användarregistrering och inloggning med säkerhetsfunktioner

### Sprint 2: Kontohantering och överföringar

1. **Mål:**
   * Utveckla funktioner för att visa och hantera kontosaldon samt genomföra banköverföringar.
2. **Viktiga uppgifter:**
   * kontosaldo och transaktioner: kontosaldo från databasen. Detta säkerställer att hämtningen av kontosaldot sker i realtid, så att användaren alltid har tillgång till den senaste informationen. På användargränssnittet (UI) kan kontosaldot visas direkt på användarens dashboard. Det är viktigt att användarens upplevelse är sömlös och att informationen uppdateras utan att användaren behöver uppdatera sidan manuellt. Det är dessutom viktigt att kunna hantera flera olika kontotyper, såsom sparkonto, lönekonto, och eventuellt kreditkonton, för att ge användaren en fullständig översikt över sina finanser. Vidare bör systemet vara tillräckligt flexibelt för att stödja framtida kontotyper eller finansiella produkter.
   * Överföringsfunktion mellan konton: Här behöver vi skapa en API-endpoint för överföringar mellan användarens egna konton, exempelvis från ett lånekonto till ett sparkonto. När överföringen genomförs ska kontosaldot uppdateras i realtid för att säkerställa att användaren alltid har korrekt information. För att förbättra användarupplevelsen bör det även finnas en bekräftelsefunktion som visar en sammanfattning av överföringen innan den slutförs. Det kan också vara fördelaktigt att införa notifikationer via e-post eller SMS som bekräftar att överföringen har genomförts framgångsrikt. Dessutom bör säkerhetsåtgärder, som tvåfaktorsautentisering, inkluderas för att skydda känsliga transaktioner.
   * Säkerhetsaspekter (tvåfaktorsautentisering vid överföringar): När användaren vill göra överföringar, särskilt till externa konton, är det viktigt att implementera tvåfaktorsautentisering för att öka säkerheten. Genom att införa denna säkerhetsåtgärd kan vi försäkra användarna om att ingen annan kan förflytta pengar utan deras tillåtelse. Koden som skickas som en del av autentiseringen bör också ha en tidsbegränsning, till exempel 5 minuter. Om koden inte används inom denna tidsram, ska en ny kod genereras automatiskt för att säkerställa att transaktionen förblir säker och att processen förblir smidig för användaren. Dessutom kan det vara bra att meddela användaren om varje autentiseringsförsök för att öka medvetenheten om eventuell obehörig aktivitet.
   * Efter detta är det viktigt att testa Integrationerna för att säkerställa korrekt funktionalitet och ta bort buggar.
3. **Leveransmål:**
   * Fungerande funktioner för kontohantering och överföringar.
   * Grundläggande säkerhetsfunktioner implementerade.

### Sprint 3: Betalningar, transaktionshistorik och systemoptimering

1. **Mål:**
   * Implementera betalningsfunktioner och visa transaktionshistorik. Avsluta med fullständiga systemtester och optimeringar.
2. **Viktiga uppgifter:**
   * Implementera funktioner för att betala räkningar: I denna del behöver vi skapa en API-endpoint som möjliggör att användaren kan initiera betalningar till externa konton. Vid betalningen ska systemet validera att användaren har tillräckligt saldo på sitt konto och att alla nödvändiga uppgifter, såsom kontonummer, är korrekta. För att förenkla processen för användaren kan betalningsmottagare sparas i en databas, vilket gör framtida betalningar smidigare och snabbare. Det är också viktigt att designa ett användarvänligt formulär för att enkelt fylla i betalningsinformation. Formuläret bör erbjuda möjligheten att spara mottagare och visa en sammanfattning innan betalningen genomförs, så att användaren kan bekräfta all information innan transaktionen sker.

**Leveransmål:**

* + Slutförda och testade funktioner för betalningar och transaktionshistorik.
  + Fullständig systemoptimering och feedbackhantering inför vidare utveckling eller release.

Agil-modellen är en passande modell för utveckling av en bankapplikation eftersom den möjliggör kontinuerlig feedback, snabb anpassning till förändrade krav och en iterativ process. Bankapplikationer kräver ofta flexibla lösningar på grund av höga krav på säkerhet, prestanda och användarvänlighet. Genom att arbeta i sprintar kan teamet leverera mindre, användbara produktinkrement i ett tidigt skede, vilket gör det möjligt att snabbt reagera på nya regleringar, kundfeedback eller säkerhetskrav.

I detta projekt har vi implementerat centrala funktioner såsom användarregistrering, inloggning, kontohantering, överföringar och betalningar genom att bryta ner utvecklingen i tre sprintar. Varje sprint fokuserar på specifika funktioner som testas och optimeras iterativt. Med Agil kan vi säkerställa att applikationen lever upp till både funktionella och icke-funktionella krav, samtidigt som vi minimerar riskerna genom ständiga iterationer och testning.