

<b>TESTPLAN FOR NETTBANK APPLIKASJON .....</b>	<b>1</b>
1. IDENTIFIKASJON .....	1
2. INNLEDNING .....	1
3. TESTOBJEKTER .....	2
3.1 <i>Enhetstesting Objekter:</i> .....	2
3.2 <i>Integrasjonstesting Objekter:</i> .....	2
3.3 <i>Systemtesting Objekter:</i> .....	2
4. FREMGANGSMÅTE.....	3
4.1 <i>Overordnet Teststrategi for:</i> .....	3
4.2 <i>Planlegging, Utførelse og Kriterier for testene</i> .....	3
5. GODKJENNINGSKRITERIER .....	4
6. TESTMILJØ OG VERKTØY .....	5
<b>SLUTTRAPPORT.....</b>	<b>6</b>
1. IDENTIFIKASJON .....	6
2. SAMMENDRAG .....	6
3. AVVIK I FORHOLD TIL TESTPLAN.....	6
4. VURDERING AV TESTENS GRUNDIGHET .....	6
5. GJENSTÅENDE AKTIVITETER/OPPGAVER .....	8
6. AVVIK I PRODUKTET .....	8
7. VURDERING AV PRODUKTET SOM TESTES .....	9
8. SAMMENDRAG AV AKTIVITETER OG LÆRDOM .....	9
<b>VEDLEGG:.....</b>	<b>10</b>

# Testplan for Nettbank Applikasjon

## 1. Identifikasjon

Laget når: 10.03.2024

Forfattere: S374918 og S321812

Hva testes her: Nettbank Applikasjon

## 2. Innledning

I dette prosjektet så skal det utføres testing av Nettbank-applikasjonen for å sikre funksjonalitet. Ved bruk av enhets-, integrasjons- og systemtesting har vi evaluert hver komponent og funksjonalitet i applikasjonen for å oppdage eventuelle feil eller mangler.

I denne rapporten vil vi presentere våre funn og erfaringer fra testprosessen, samt den metodikken og verktøyene som ble brukt for å oppnå våre mål.

Det første som gjøres er enhetstesting. Her blir API'ene til prosjektet testet grundig for å oppnå 100% 'Code-coverage'. Hver metode i hver Controller blir testet for å se at den

fungerer som den skal. I tillegg så er det utført enhetstesting for Sikkerhets filen, som også fungerer som en Controller.

Videre vil vi diskutere integrasjonstesting, hvor vi har brukt SoapUI til å teste API-er for funksjonalitet og korrekt integrasjon. Her er resultatene for hver test dokumentert i et Excel-dokument.

Til slutt vil vi presentere våre funn fra systemtestingen, hvor vi har benyttet Selenium for Chrome for å replisere brukerhistorier og evaluere nettbankens funksjonalitet fra et brukerperspektiv. Vi har dokumentert våre observasjoner og resultater ved hjelp av Microsoft Test Manager (MTM).

### 3. Testobjekter

#### 3.1 Enhetstesting Objekter:

**BankController:** Denne inkluderer alle metoder som brukes primært av Kunde interaksjon med nettsiden. Disse metodene er avhengig av BankRepository filen.

**AdminKundeController:** Denne har metoder som lar Admin endre/redigere på kunde detaljer som navn, adresse, osv. Denne og neste Controller er avhengig av AdminRepository filen.

**AdminKontoController:** Denne lar Admin endre på Konto detaljer, sånn som Saldo, Kontotype osv.

**Sikkerhet:** Denne filen er også en Controller som sørger for at alt av logg inn/ut blir gjennomført korrekt og for å sørge for at det blir sjekket at man har riktig tilgang til de forskjellige metodene i de andre controllerene.

#### 3.2 Integrasjonstesting Objekter:

**Forhold mellom Controller og Repository:** Her blir det sjekket at Controllerene og Reppositorier fungerer sammen på riktig måte for å hente/endre og lagre data på riktig måte.

**Databasen:** Det blir også sjekket at informasjonen som er endret/lagret eller slettet blir riktig i henhold til databasen.

#### 3.3 Systemtesting Objekter:

**UI:** Her testes brukergrensesnittet til web applikasjonen for å sikre at det er brukervennlig og responsivt.

**Brukere:**

**Kunde:**

Her er det viktig at alle brukerhistoriene til kunder blir testet og sjekker at all funksjonalitet som en kunde ønsker er til stede og funker.

**Admin:**

Her testes det at Administratorer får tilgang til de metoden som de trenger, sånn som redigering av kontoer eller kunder og andre administrativ funksjonalitet.

## 4. Fremgangsmåte

For å effektivt teste Nettbanken så må det planlegges hvordan testingen skal foregå på de forskjellige nivåene. Rekkefølgen blir Enhetstesting, Integrasjonstesting og til slutt Systemtesting.

### 4.1 Overordnet Teststrategi for:

#### *Enhetstesting*

For å få til effektiv enhetstesting så er det viktig å sørge for at alle metodene, spesielt de som returnerer noe, blir testet. Dette er primært for de forskjellige Controller filene; **BankController**, **AdminKontoController**, **AdminKundeController** og **Sikkerhet**.

Enhetstesting lar oss teste hver metode hver for seg, for å sikre at funksjonaliteten de skal ha er der.

#### *Integrasjonstesting*

Integrasjonstesting vil sette søkelys på å teste samspillet mellom forskjellige komponenter, inkludert Controller og Repository for å sikre at data blir håndtert på riktig måte.

#### *Systemtesting*

Systemtestene vil bli brukt for å simulere brukerinteraksjoner med Nettbank applikasjonen for å verifisere at den oppfyller kravene til brukerhistoriene til både Kunde og Admin.

### 4.2 Planlegging, Utførelse og Kriterier for testene

#### *Enhetstest*

##### **Spesifikasjon og Utførelse:**

Utvikle test scenarioer som dekker hver individuell metode i Controller filene. For å utføre enhetstestene bruker vi IntelliJ's innebygde testrammeverk JUnit for å utføre enhetstester og få resultater på dem, samtidig som vi bruker Mockito for å simulere eksterne avhengigheter i testene.

##### **Kriterier:**

For at testene skal kjøres så må testmiljøet bli satt opp med nødvendig ressurser, sånn som Mock data eller stubber for eksterne avhengigheter. Disse avhengighetene er for eksempel inputs eller parameterer som må inn i en metode for at den skal kunne kjøres. Når dette er i orden, kan testene kjøres. For at BankController skal kunne testes, for eksempel, så må de Mock-es BankRepository og Sikkerhet for å kunne teste alle metodene. Dette må settes opp for de andre kontrollerne også.

For at testene skal bli godkjent så må de kjøres og få dokumentert bestått når de kjøres uten feil. Her blir det også dokumentert 'Code-Coverage' for å vise graden av dekning av kodelinjer og betingelser. Her burde det være 100% 'Code-coverage' for alle kontrollerne.

### *Integrasjonstest*

#### **Spesifikasjon og Utførelse:**

Her utvikler vi testtilfeller som tester samspillet mellom forskjellige komponenter i Nettbank applikasjonen. Blant annet, integrasjon mellom kontrollerne, repository og databasen. Her bruker vi initDB metoden for å initialere databasen for hver test kjøring. Her bruker vi programmet SoapUI for å teste oppførselen til applikasjonen. Her tester vi scenarioer, for å vise at data blir hentet eller lagret riktig.

#### **Kriterier:**

Her er det viktig at alle interne grensesnitt blir testet med integrasjonstesting. Det vil si at alt av samspill mellom kontrollerne og repositorier blir gjennomført riktig. For at disse skal bli godkjennes så er det viktig at alt av samspill gjennomføres som forventet. Her tester vi blant annet GET og POST HTML kall for å se at disse henter og skriver riktig data i applikasjonens interne grensesnitt.

### *Systemtest*

#### **Spesifikasjon og Utførelse:**

Her lager vi 'user-stories' eller bruker historier som inneholder funksjonaliteten som vi ønsker at skal være tilstede på applikasjonen. Disse brukerhistoriene blir testet ved bruk av Selenium IDE for Chrome for å kunne teste at interaksjon mellom bruker og nettside funker som forventet. Her blir det inkludert brukerhistorier til en ordinær bruker, og administrator bruker. Dokumentering av testen som gjennomføres via Selenium blir gjort på Microsoft Test Manager (MTM).

#### **Kriterier:**

For at testingen skal anses som fullført så bør alt av funksjonalitet bli testet og alle brukerhistorier bør få bestått for at systemtesten skal få godkjent. Kravene til brukerene er som følger:

**Kundene** vil måtte kunne logg inn, og logge ut. De må også kunne se oversikt over og administrere, kontoene sine, se saldo, se transaksjoner innen en viss dato, og endre på kunde info hvis de ønsker. De skal også kunne registrere nye betalinger, og utføre betalinger som venter.

**Administrator** skal ha tilgang til å se oversikt over kunder og kontoer, muligheten til å endre på kunde og konto informasjon, og mulighet til å endre og slette kunder og/eller kontoer.

## **5. Godkjenningskriterier**

*(Hentet kriterier fra Testplan mal fra Rapporterings modulen i Canvas)*

Testen kan anses som godkjent hvis 95% av de planlagte testene er utført og det ikke finnes noe åpne ‘A’ og ‘B’ feil. I tillegg så bør ikke applikasjonen bli satt i drift hvis det er mange Kategori C eller D feil, siden dette vil påføre at det ikke er en særlig brukbar applikasjon.

Kategori A: Feil i en eller flere funksjoner som forårsaker stopp. For eksempel maskin/system/funksjon.

Kategori B: Feil har alvorlige konsekvenser i systemet eller tilstøtende systemer, for eksempel feil i databaser, alvorlige feil i rapporter slik at disse tolkes feil etc.

Kategori C: Feil med mindre alvorlige konsekvenser, for eksempel mindre grad av ustabilitet, mindre funksjons-/egenskapsmangler etc.

Kategori D: Feil uten alvorlige konsekvenser, for eksempel layout-/trykkfeil i skjermbilder, rapporter eller dokumentasjon som ikke har konsekvenser for forståelsen av disse.

## 6. Testmiljø og verktøy

### Enhetstest

Her bruker vi innebygd verktøy i IntelliJ IDEA som miljøet for å gjennomføre enhetstesting. Dette inkluderer JUnit, som lar også kjøre tester direkte i IntelliJ og få tilbake bestått/ikke bestått og får muligheten til å se ‘Code-coverage’ av filene vi tester.

For å simulere avhengigheter til hver enkelt metode bruker vi Mockito biblioteket for å kunne ‘hardcode’ parametere eller avhengigheter fra repositorier som en metode trenger for å kunne kjøre. Dette lar oss teste hver metode hver for seg.

### Integrasjonstest

For å utføre integrasjonstestene bruker vi en test verktøy som heter SoapUI. Dette programmet lar oss simulere et miljø som er så nært det reelle produksjonsmiljøet som mulig. Her bruker vi en metode, ‘initDB’, for å initialisere databasen for hver gjennomføring av testene for å kunne se at samspillet mellom applikasjonen og databasen fungerer som forventet.

### Systemtest

Systemtestene utføres i et testmiljø som ligner så mye som mulig på det faktiske produksjonsmiljøet. Her bruker vi også ‘initDB’ metoden som ble brukt i Integrasjonstesting for å også kunne teste database oppførsel når applikasjonen er i bruk. Vi benytter verktøy som ‘Selenium IDE for Chrome’ som lar oss gjenskape brukerhistorier i nettleseren og teste at det forventete utfallet skjer. Selenium blir brukt for å faktisk teste og få en bestått eller ikke bestått for hver brukerhistorie, og vi bruker MTM for testplanlegging og dokumentasjon av testene. Her rapporterer vi feil eller Bugs som fører til at applikasjonen ikke funker.

# Sluttrapport

## 1. Identifikasjon

Tester som gjelder: Enhetstester, Integrasjonstester og Systemtester

Referanse til testplan, test resultater osv. blir inkludert som vedlegg

Laget når: 15.03.2024

Forfattere: S374918 og S321812

## 2. Sammendrag

Testen av systemet resulterte i følgende observasjoner:

**Enhetstesting:** Alle enhetstester ble bestått, og 100% kode dekning ble oppnådd, med unntak av initDB metoden som ikke ble testet siden det var en metode som ble laget av oss for å gjennomføre testing og påvirker ikke funksjonalitet av selve Nettbanken. Dette viser at de individuelle komponentene i systemet funker som forventet.

**Integrasjonstesting:** Alle integrasjonstester ble bestått, noe som indikerer at samspillet mellom forskjellige komponenter i systemet fungerer som forventet.

**Systemtesting:** Her ble det avdekket flere feil, spesielt knyttet til registrering av kunder, opprettelse av kunder og registrering av betalinger. Av de 15 brukerhistoriene som ble laget, så var det 5 som ikke besto testene. Dette peker til at det er noe feil med lagring av ny data siden det bare var metoder som registrerte eller endret på data som feilet. Sletting og henting av data fungerte som forventet.

Overordnet så ser vi at applikasjonen består mesten av testen, men de testene som feiler i systemtesting er svært viktig for at applikasjonen skal fungere som forventet. For at applikasjonen skal være satt i drift så bør disse problemene utarbeides.

## 3. Avvik i forhold til testplan

Ingen avvik mellom den utførte testingen og testplanen. Alle testaktiviteter ble gjennomført i henhold til planen som ble laget og ingen endringer i tester eller verktøy var nødvendig underveis.

## 4. Vurdering av testens grundighet

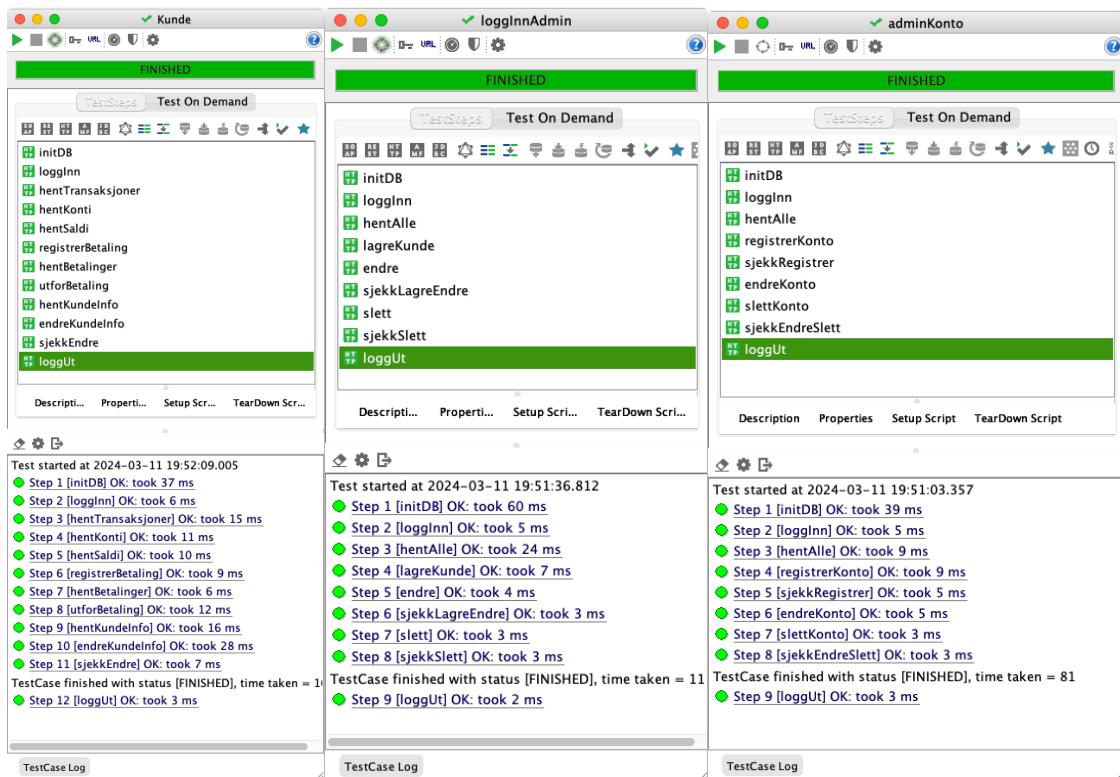
Målet med grundighet var at det skulle være 100% ‘Code-coverage’, at alt av samspill mellom controller og repository ble testet, og at alle brukerhistorier, som var nødvendig for å dekke funksjonaliteten av websiden, var testet. Dette ble gjennomført.

Her ser vi Code-coverage fra enhets testingen:

Element ^	Class, %	Method, %	Line, %	Branch, %
all	100% (3/3)	94% (16/17)	98% (71/72)	97% (33/34)
oslomet.testing.API	100% (3/3)	94% (16/17)	98% (71/72)	97% (33/34)
AdminKontoController	100% (1/1)	100% (4/4)	100% (18/18)	100% (8/8)
AdminKundeController	100% (1/1)	100% (4/4)	100% (17/17)	100% (8/8)
BankController	100% (1/1)	88% (8/9)	97% (36/37)	94% (17/18)
oslomet.testing.enhetstester	100% (4/4)	100% (43/43)	100% (219/219)	100% (0/0)
EnhetsTestAdminKonto	100% (1/1)	100% (13/13)	100% (62/62)	100% (0/0)
EnhetsTestAdminKunde	100% (1/1)	100% (8/8)	100% (38/38)	100% (0/0)
EnhetstestBankController	100% (1/1)	100% (17/17)	100% (93/93)	100% (0/0)
EnhetstestSikkerhet	100% (1/1)	100% (5/5)	100% (26/26)	100% (0/0)
oslomet.testing.Sikkerhet	100% (1/1)	100% (5/5)	83% (20/24)	58% (7/12)
Sikkerhet	100% (1/1)	100% (5/5)	83% (20/24)	58% (7/12)

Figur 1. Code Coverage

Vi ser at alle metodene bortsett fra BankController har 100% coverage av metodene. Den siste metoden som ikke er dekket i BankController var initDB metoden, som var implementert av oss for å kunne gjennomføre integrasjons og systemtesting. Det vil si at alle metodene som var tilstede fra før av i applikasjonen har blitt testet og dermed så har vi 100% code coverage og har oppnådd grundighets målet for enhetstesting.



Figur 2, 3 og 4. Integrasjonstester

Her ser vi integrasjonstestene som var gjennomført på SoapUI. XML filen blir lagt som vedlegg. Vi ser at alle metodene og intern funksjonaliteten var testet, og da møter vi også målet vi satte for integrasjonstestene.

Til slutt så har vi Systemtestene. Her ble det skrevet 15 brukerhistorier som vi følte oppnådde alt av funksjonelle krav en bruker, eller administrator, ville hatt med denne websiden. Vi testet all 15 så vi har dekket det vi ønsket.

Basert på dette så kan vi anslå at testene er tilstrekkelig grundig for å kunne basere vurderinger av applikasjonen på basis av testene.

## 5. Gjenstående aktiviteter/oppgaver

Selv om testene har vært grundig så er det aldri nok tester og alltid flere feil å finne. Det kunne bli satt mer tid på å lage enda flere tester, spesielt med tanke på område som skapte feiler. Registrering av kunder, kontører og transaksjoner og endring av informasjon kan alle bli testet enda mer både for å peile inn nærmere på årsaken, men også for å sørge for at etter disse feilene blir rettet så oppstår det ikke noen flere.

Det har ikke blitt gjort noe testing på server kapasitet, eller om det er flere brukere om gangen og det kunne bli gjort mer av. Etter feilretting så må testene oppdateres og med alt av ny funksjonalitet som kan bli introdusert inn i applikasjonen så må nye tester introduseres.

## 6. Avvik i produktet

### *Sammendrag av testresultatene:*

Enhetstestene og integrasjonstestene oppdaget ikke noen feil, men systemtestene avdekket flere feil i systemet under testing.

### *Identifiserte Avvik:*

Her ble det primært funnet feil i tilknytning av lagring eller endring av data.

Brukerhistoriene dette gjelder var som følger:

Bruk – Endring av informasjon og registrering av betaling

Admin – Endring av konto eller kunde informasjon, og registrering av ny kunde eller konto.

Her fikk alle samme feilmelding som var

«org.springframework.web.HttpMediaTypeNotSupportedException: Content type 'application/x-www-form-urlencoded;charset=UTF-8' not supported»

Dette er ofte forsaket av at en Controller metode forventer en annen type data enn det som blir sendt inn som parameter. Her gjelder det alle POST metodene som har @RequestBody i parameterne.

### *Løsning:*

Her kan det være noen forskjellige løsninger. En løsning kan være å endre @RequestBody til @RequestParam, men da må man spesifisere hver del av, for eksempel Konto objektet, sånn som '@RequestParam kontonr, @RequestParam personnr' osv.

En annen løsning kunne vært å endre på client siden for at det blir sendt et JSON objekt, som kan leses av @RequestBody, ved å sette 'Content-Type' til 'application/json'. Da burde @RequestBody funke som forventet.

## 7. Vurdering av produktet som testes

Vurderingen av det produktet, eller Nettbank applikasjonen, baseres på testresultatene og de forhåndsdefinerte godkjenningskriteriene, som var beskrevet i testplanen.

Applikasjonen har bestått enhetstestene og integrasjonstestene kriteriene. Dette indikerer en tilfredsstillende funksjonalitet og samspill mellom de ulike komponentene i systemet. Enhetstestene og integrasjonstestene bestått, og gir dermed en viss grad av tillit til systemets pålitelighet og stabilitet og indikererer at det meste av koden funker som den skal.

På den andre siden, så avdekket systemtestene flere alvorlige feil knyttet til registrering og lagring av data. Den største feilen var relatert til manglende støtte for innholdstypen 'application/x-www-form-urlencoded; charset=UTF-8'. Disse feilene har potensial til å ha alvorlige konsekvenser for applikasjonens funksjonalitet, spesielt med tanke på hvor viktig rollen til dataregistrering og lagring spiller i en bankapplikasjon. Feil knyttet til kritiske funksjonaliteter som registrering av nye kunder, opprettelse av kontører og gjennomføring av betalinger utgjør en betydelig risiko hvis de ikke blir løst før produktet settes i drift.

## 8. Sammendrag av aktiviteter og lærdom

Prosjektet har vært en meget lærerik opplevelse og har lært oss nyttige verktøy som kan brukes i fremtidige prosjekter. Viktigheten av testing var noe som vi var kjent med fra før, men å se de i praksis var veldig innsiktsfull.

Utfordringer i prosjektet var det å planlegge å sikre at testene var omfattende nok og var utført riktig. I tillegg så var det ikke bare tester man skulle lære, men to-tre nye programmer som man måtte lære for å utføre disse testene. Det var noe utfordringer med å være både grundig, men også effektiv i testingen med tanke på tid og resursbegrensninger.

Prosjektet fikk oss til å prøve ut og eksperimentere med forskjellig verktøy som kan brukes for testing. JUnit var vi kjent med fra før, men bruk av Mockito i IntelliJ var nytt, i

tillegg til SoapUI, Selenium og MTM. Disse verktøyene har vært ekstremt fint å få lært siden det er veldig lett å se overføringsverdien til arbeidslivet i fremtiden.

En ting som kunne blitt gjort annerledes ville kanskje vært at vi hadde satt mer tid på planlegging før man begynte å utføre testing. Dette kunne spart tid og gjort at man kunne være mer fokusert med testingen. I fremtiden vil det kanskje også være nyttig å prøve ut annet verktøy for testing for å utvide kunnskapsbasen vår innen testing.

## Vedlegg:

Excel-ark, Kode, MTM filer osv.

Test Case	Test Step	Step Description	Expected	Test Data	Actual	Status	Tester	Date	Comments
Kunde Bruker	<b>loggInnKunde</b>	Initialize the database	OK		PASS	Alex		07.03.24	
	<b>initDB</b>		OK		PASS	Alex		07.03.24	
	<b>loggInnKunde</b>	Logg inn i nettbanken som kunde	OK		PASS	Alex		07.03.24	
	<b>hentTransaksjoner</b>	Henter alle transaksjoner for bruker							
	> antall Transaksjoner		3	\$[transaksjoner[*]	3	PASS	Alex	07.03.24	del av hentTransaksjoner
	> personnummer		12345678910	\$[personnummer	12345678910	PASS	Alex	07.03.24	del av hentTransaksjoner
	<b>hentKonto</b>	Henter konto							
	> antall kontoer		3	\$[*]	3	PASS	Alex	07.03.24	del av hentKonto
	> kontonummer		105010123456	\$[1].kontonummer	105010123456	PASS	Alex	07.03.24	del av hentKonto
	<b>hentSaldi</b>	Henter saldo							
	> antall kontoer		3	\$[*]	3	PASS	Alex	07.03.24	del av hentSaldi
	<b>loggInnKunde</b>		12345678910	personnummer	12345678910	PASS	Alex	07.03.24	del av hentSaldi
	> personnummer		1005000	\$[1].saldo	100500.0	PASS	Alex	07.03.24	del av hentSaldi
	<b>saldo matcher</b>		OK	Transaksjon betaling	OK	PASS	Alex	07.03.24	
	<b>registrerBetalning</b>	Registrer en betaling							
	<b>hentBetalinger</b>	Hent alle betalingar							
	> antall betalingar		3	\$[1.*]	3	PASS	Alex	07.03.24	del av hentBetalinger
	<b>loggInnKunde</b>		[Fjordkraft, Skagen, Hustleie]	\$[1.*].melding	[Fjordkraft, Skagen, Hustleie]	PASS	Alex	07.03.24	del av hentBetalinger
	> meldingen matcher		2	\$[1.*]	2	PASS	Alex	07.03.24	del av hentBetalinger
	<b>uttorBetalning</b>	Uttør en av betalingene							
	<b>loggInnKunde</b>	Burde være en mindre enn før							
	> antall betalingar		[Skagen, Hustleie]	\$[1.*].melding	[Skagen, Hustleie]	PASS	Alex	07.03.24	del av uttorBetalning
	<b>loggInnKunde</b>								
	<b>hentKundelinfo</b>	Hent all kunde info							
	> antall kunde		8	\$[*]	8	PASS	Alex	07.03.24	del av hentKundelinfo
	<b>loggInnKunde</b>		12345678910	\$[1].personnummer	12345678910	PASS	Alex	07.03.24	del av hentKundelinfo
	> meldingen matcher		Askerveien 22	\$[1].adresse	Askerveien 22	PASS	Alex	07.03.24	del av hentKundelinfo
	<b>loggInnKunde</b>								
	<b>uttorKundelinfo</b>	Hent kunde info							
	> antall kunde		8	\$[1.*]	8	PASS	Alex	07.03.24	del av uttorKundelinfo
	<b>loggInnKunde</b>		12345678910	\$[1].personnummer	12345678910	PASS	Alex	07.03.24	del av hentKundelinfo
	> meldingen matcher		Askerveien 22	\$[1].adresse	Askerveien 22	PASS	Alex	07.03.24	del av hentKundelinfo
	<b>loggInnKunde</b>								
	<b>endreKundelinfo</b>	Endre på kunde detaljer (feks. Navn)							
	<b>slettEtEndre</b>	Sjekk at endringen gikk gjennom							
	<b>loggInnKunde</b>								
	> adresse		Testveien	\$[1].adresse	Testveien	PASS	Alex	07.03.24	del av sjekkEndre
	<b>loggInnKunde</b>								
	> fornavn		Test	\$[1].fornavn	Test	PASS	Alex	07.03.24	del av sjekkEndre
	<b>loggInnKunde</b>								
	> antall feil		8	\$[*]	8	PASS	Alex	07.03.24	del av sjekkEndre
	<b>loggUt</b>			assert messageExchange.hasResponse() == false;	PASS	PASS	Alex	07.03.24	Skal være blank
<b>ADMIN BRUKER</b>									
adminInnKonto	<b>initDB</b>	Initialize the DB	OK		OK	PASS	Alex	11.03.24	
	<b>loggInnAdmin</b>	Logg inn i nettbank som admin	Logget inn	bruker, passord	Logget inn	PASS	Alex	11.03.24	
	<b>hentAlle</b>	Henter alle kontoer							
	<b>adminKonto</b>	> antall kontoer	3	\$[*]	3	PASS	Alex	11.03.24	del av hentAlle (admin)
	<b>adminKonto</b>	> kontonr matcher	105020123456	\$[1].kontonummer	105020123456	PASS	Alex	11.03.24	del av hentAlle (admin)
	<b>adminKonto</b>	> personnr matcher	12345678910	\$[1].personnummer	12345678910	PASS	Alex	11.03.24	del av hentAlle (admin)
	<b>adminKonto</b>	Registrerer ny/konto	OK	Konto konto	OK	PASS	Alex	11.03.24	
	<b>slettKontorRegister</b>	Sjekke at registrering av/konto funker							
	<b>adminKonto</b>	> antall kontoer	4	\$[*]	4	PASS	Alex	11.03.24	del av sjekkRegistert
	<b>adminKonto</b>	> saldo matcher	10000.0	\$[1].saldo	10000.0	PASS	Alex	11.03.24	del av sjekkRegistert
	<b>adminKonto</b>	> kontonr matcher	444444444444	\$[1].kontonummer	444444444444	PASS	Alex	11.03.24	del av sjekkRegistert
	<b>endreKonto</b>	Endrer på konto detaljer	OK	Konto konto	OK	PASS	Alex	11.03.24	
	<b>slettKonto</b>	Slette en konto	OK	kontonummer	OK	PASS	Alex	11.03.24	
	<b>slettEndreKonto</b>	Sjekke at slettning av konto funker							
	<b>adminKonto</b>	> saldo matcher	100000.0	\$[1].saldo	100000.0	PASS	Alex	11.03.24	del av sjekkEndreSlett

adminKonto	> kontonr matcher	444444444444	\$[2] kontonummer	PASS	Alex	11.03.24	del av sjekkEndreSlett
adminKonto	> antall kontoer	3	\$[*]	PASS	Alex	11.03.24	del av sjekkEndreSlett
<b>logUt</b>			assert messageExchange.hasResponse() = false;	PASS	Alex	11.03.24	
<b>ADMIN BRUKER</b>							
loggetInnAdmin	<b>initDB</b>	Initialize the DB	OK	OK	PASS	Alex	11.03.24
loggetInnAdmin	<b>loggInnAdmin</b>	Logg inn i netbank som admin	Logget inn	Logget inn	PASS	Alex	11.03.24
loggetInnAdmin	<b>hentAlle</b>	Henter alle kunder					
loggetInnAdmin	> antall kunder	2	\$[*]	PASS	Alex	11.03.24	del av hentAlle
loggetInnAdmin	> personnr matcher	12345678910	\$[1].personnummer	PASS	Alex	11.03.24	del av hentAlle
lagreKunde	<b>lagreKunde</b>	Lagrer ny kunde	OK	Kunde innKunde	PASS	Alex	11.03.24
endre	<b>endre</b>	Endre kunde detaljer	OK	Kunde innKunde	PASS	Alex	11.03.24
sjekkLagreEndre	<b>sjekkAt lagring/ending funket</b>						
loggetInnAdmin	> antall + 1	3	\$[*]	PASS	Alex	11.03.24	del av sjekkLagreEndre
loggetInnAdmin	> endret navn	Successfullt changed	\$[0].etternavn	PASS	Alex	11.03.24	del av sjekkLagreEndre
loggetInnAdmin	> endret adresse	Successfullt changed	\$[0].adresse	PASS	Alex	11.03.24	del av sjekkLagreEndre
sjett	<b>sjett</b>	Stettet kunde	OK	personnummer	PASS	Alex	11.03.24
sjekkSlett	<b>sjekkSlett</b>	Sjekket slettning funket					
loggetInnAdmin	> antall kunder	2	\$[*]	PASS	Alex	11.03.24	del av sjekkSlett
loggetInnAdmin	> personnr matcher	12345678910	\$[1].personnummer	PASS	Alex	11.03.24	del av sjekkSlett
<b>logUt</b>		assert messageExchange.hasResponse() = false;		PASS	Alex	11.03.24	Skal være blank

```

1 package oslomet.testing.enhetstester;
2
3 import org.junit.Test;
4 import org.junit.runner.RunWith;
5 import org.mockito.InjectMocks;
6 import org.mockito.Mock;
7 import org.mockito.junit.MockitoJUnitRunner;
8 import org.springframework.mock.web.MockHttpSession;
9 import oslomet.testing.DAL.BankRepository;
10 import oslomet.testing.Sikkerhet.Sikkerhet;
11
12 import static org.junit.jupiter.api.Assertions.assertEquals;
13 import static org.junit.jupiter.api.Assertions.assertNull; // Adds the assertNull method for
14 import static org.mockito.ArgumentMatchers.anyString;
15 import static org.mockito.Mockito.*;
16
17 @RunWith(MockitoJUnitRunner.class)
18 public class EnhetstestSikkerhet {
19
20     @InjectMocks
21     // denne skal testes
22     private Sikkerhet sikkerhetsController;
23
24     @Mock
25     // denne skal Mock'es
26     private BankRepository repository;
27
28     @Mock
29     // denne skal Mock'es
30     private MockHttpSession session;
31
32     // Checks that log in method works
33     // Done by Professor
34     @Test
35     public void test_sjekkLoggInn() {
36         // arrange
37         when(repository.sjekkLoggInn(anyString(),anyString())).thenReturn("OK");
38
39         // setningen under setter ikke attributten, dvs. at det ikke er mulig å sette en attributt i
40         // dette oppsettet
41         session.setAttribute("Innlogget", "12345678901");
42
43         // act
44         String resultat = sikkerhetsController.sjekkLoggInn("12345678901","HeiHeiHei");
45         // assert
46         assertEquals("OK", resultat);
47     }
48
49     // Test to see log out method works correctly and sets the session attribute to null
50     @Test
51     public void test_loggUt() {
52         // arrange
53         session.setAttribute("Innlogget", "12345678901"); // simulates a user logging in
54
55         // act
56         sikkerhetsController.loggUt(); // if log out works, the session attribute is set to null
57
58         // assert
59         assertNull(session.getAttribute("Innlogget")); // if the session attribute is null, the user
60         // is logged out
61     }

```

```

62
63     // Test for logging in as admin with correct/incorrect credentials
64     @Test
65     public void test_loggInnAdminOK() {
66         // arrange
67         String mockBruker = "Admin"; // mock username and password
68         String mockPassord = "Admin";
69
70         MockHttpSession session = new MockHttpSession(); // creates a mock session for the user
71         sikkerhetsController.setSession(session);
72
73         // act
74         String resultat = sikkerhetsController.loggInnAdmin(mockBruker, mockPassord);
75         // if the username and password are correct, the user is logged in and shoudl return "Logget inn"
76
77         // assert
78
79         assertEquals("Admin", session.getAttribute("Innlogget"));
80         assertEquals("Logget inn", resultat);
81         // checks that user is logged in and attribute is correctly set to "Admin"
82     }
83
84
85     @Test
86     public void test_loggInnAdminFeil() {
87         // arrange
88         String mockFeilBruker = "jeg er ikke Admin"; // mock username and password
89         String mockFeilPassord = "jeg er ikke Admin";
90
91         // act
92         String result = sikkerhetsController.loggInnAdmin(mockFeilBruker, mockFeilPassord);
93         // if user and password is wrong, should return "Ikke logget inn"
94
95         // assert
96         assertEquals("Ikke logget inn", result);
97         assertNull(session.getAttribute("Innlogget"));
98         // checks that user is not logged in and attribute is correctly set to null
99     }
100
101
102     // Test to see if the loggetInn method returns the correct username
103     @Test
104     public void test_loggetInn() {
105         // arrame
106         String mockBruker = "12345678901"; // mock username
107         MockHttpSession mockSesh = new MockHttpSession(); // creates a mock session for the user
108         sikkerhetsController.setSession(mockSesh); // sets the mock session
109         mockSesh.setAttribute("Innlogget", mockBruker); // in the mock session, we simulate logging
110         in
111
112         // act
113         String result = sikkerhetsController.loggetInn(); // if user is logged in, should return the
114         // username
115
116         // assert
117         assertEquals(mockBruker, result); // checks that the username is returned
118     }
119 }
```

```

1 package oslomet.testing.enhetstester;
2
3 import org.junit.Test;
4 import org.junit.runner.RunWith;
5 import org.mockito.InjectMocks;
6 import org.mockito.Mock;
7 import org.mockito.junit.MockitoJUnitRunner;
8
9 import oslomet.testing.Sikkerhet.Sikkerhet;
10 import oslomet.testing.Models.Konto;
11 import oslomet.testing.DAL.AdminRepository;
12 import oslomet.testing.API.AdminKontoController;
13
14 import java.util.ArrayList;
15 import java.util.List;
16
17 import static org.junit.jupiter.api.Assertions.assertEquals;
18 import static org.junit.jupiter.api.Assertions.assertNull; // Adds the assertNull method for
19 import static org.mockito.Mockito.*;
20
21 @RunWith(MockitoJUnitRunner.class)
22 public class EnhetsTestAdminKonto {
23     // TEST ALL OF THE METHODS IN THE AdminKontoController CLASS
24     @InjectMocks
25     private AdminKontoController controller;
26
27     @Mock
28     private AdminRepository repository;
29
30     @Mock
31     private Sikkerhet sjekk;
32
33     // Tests for hentAlle method both if logged in and not logged in
34     @Test
35     public void test_hentAlleKontiLoggetInn() {
36         // arrange
37         String mockPNr = "12345678901"; // mock personnummer
38         List<Konto> mockKonti = new ArrayList<>();
39         when(sjekk.loggetInn()).thenReturn(mockPNr); // mock log in
40         when(repository.hentAlleKonti()).thenReturn(mockKonti); // mocks returning accounts
41
42         // act
43         List<Konto> result = controller.hentAlleKonti(); // actual return of accounts
44
45         // assert
46         assertEquals(mockKonti, result); // Should be the same if it worked
47     }
48
49     @Test
50     public void test_hentAlleIkkeLoggetInn() {
51         // arrange
52         when(sjekk.loggetInn()).thenReturn(null); // Returns null because null means not logged in
53
54         // act
55         List<Konto> result = controller.hentAlleKonti(); // tries to get all accounts when not logged in
56
57         // assert
58         assertNull(result); // Should be null since user is not logged in
59     }
60
61     // Tests for registrerKonto methods for the different cases: logged in, not logged in, and error/
62     // missing PNr
63     @Test

```

```

63 public void test_RegistrerKonto_LoggedIn() {
64     // arrange
65     Konto mockKonto = new Konto("12345678901", "12345", 1000, "Sparing", "NOK", new ArrayList<>());
66     when(sjekk.loggetInn()).thenReturn(mockKonto.getPersonnummer()); // mock log in
67     when(repository.registrerKonto(mockKonto)).thenReturn("OK"); // mock registering account
68
69     // act
70     String result = controller.registrerKonto(mockKonto); // actual attempt to register account
71
72     // assert
73     assertEquals("OK", result); // Should be OK if it worked
74 }
75
76 @Test
77 public void test_RegistrerKonto_Error() {
78     // arrange
79     Konto mockKonto = new Konto("12345678901", "12345", 1000, "Sparing", "NOK", new ArrayList<>());
80     when(sjekk.loggetInn()).thenReturn(mockKonto.getPersonnummer()); // mock log in
81     when(repository.registrerKonto(mockKonto)).thenReturn("Feil"); // mock error :
82     // enten ingen pnr eller hvis det er noe feil med SQL query returnerer begge
83     // "Feil"
84
85     // act
86     String result = controller.registrerKonto(mockKonto);
87
88     // assert
89     assertEquals("Feil", result); // Should be "Feil" since we arranged an error
90 }
91
92 @Test
93 public void test_RegistrerKonto_NotLoggedIn() {
94     // arrange
95     Konto mockKonto = new Konto("12345678901", "12345", 1000, "Sparing", "NOK", new ArrayList<>());
96     when(sjekk.loggetInn()).thenReturn(null); // mock not being logged in
97
98     // act
99     String result = controller.registrerKonto(mockKonto); // should return "Ikke innlogget"
100
101    // assert
102    assertEquals("Ikke innlogget", result); // Should be null since user is not logged in
103 }
104
105
106
107 // Tests endreKonto method for all the different cases: logged in, not logged in, error, missing
108 // PNr and missing KontoNr
109 @Test
110 public void test_EndreKonto_loggetInn() {
111     // arrange
112     Konto mockKonto = new Konto("12345678901", "12345", 1000, "Sparing", "NOK", new ArrayList<>());
113     when(sjekk.loggetInn()).thenReturn(mockKonto.getPersonnummer()); // mock log in
114     when(repository.endreKonto(mockKonto)).thenReturn("OK"); // mocks successful change
115
116     // act
117     String result = controller.endreKonto(mockKonto); // actual attempt to change account
118
119     // assert
120     assertEquals("OK", result); // Should be OK if it worked
121 }
122
123 @Test
124 public void test_EndreKonto_ikkeLoggetInn() {

```

```

125 // arrange
126 Konto mockKonto = new Konto(); // does't matter cause we're not logged in
127 when(sjekk.loggetInn()).thenReturn(null); // mock not being logged in
128
129 // act
130 String result = controller.endreKonto(mockKonto); // should return "Ikke innlogget"
131
132 // assert
133 assertEquals("Ikke innlogget", result);
134 }
135
136
137 @Test
138 public void test_endreKonto_FeilPNr() {
139 // arrange
140 Konto mockKonto = new Konto("", "12345", 1000, "Sparing", "NOK", new ArrayList<>());
141 when(sjekk.loggetInn()).thenReturn(mockKonto.getPersonnummer()); // mock log in
142 when(repository.endreKonto(mockKonto)).thenReturn("Feil i personnummer"); // mocks error
143
144 // act
145 String result = controller.endreKonto(mockKonto); // attempt to change account with missing pnr
146
147 // assert
148 assertEquals("Feil i personnummer", result); // Should be "Feil i personnummer" since we
arranged an error
149 }
150
151 @Test
152 public void test_endreKonto_feilKontoNr() {
153 // arrange
154 Konto mockKonto = new Konto("12345678901", "", 1000, "Sparing", "NOK", new ArrayList<>());
155 // purposely no kontonummer
156 when(sjekk.loggetInn()).thenReturn(mockKonto.getPersonnummer()); // mock log in
157 when(repository.endreKonto(mockKonto)).thenReturn("Feil i kontonummer"); // mocks error
158
159 // act
160 String result = controller.endreKonto(mockKonto); // attempt to change account without kontonr
161
162 // assert
163 assertEquals("Feil i kontonummer", result); // Same as above test but with kontonr
164 }
165
166 @Test
167 public void test_endreKonto_error () {
168 // arrange
169 Konto mockKonto = new Konto("12345678901", "12345", 1000, "Sparing", "NOK", new ArrayList<>());
170 when(sjekk.loggetInn()).thenReturn(mockKonto.getPersonnummer()); // mock log in
171 when(repository.endreKonto(mockKonto)).thenReturn("Feil"); // mocks exception catch return
172
173 // act
174 String result = controller.endreKonto(mockKonto); // attempt to change account
175
176 // assert
177 assertEquals("Feil", result); // Exception e return is "Feil" so it should match
178 }
179
180 @Test
181 public void test_slettKonto_loggetInn() {
182 // arrange
183 String KontoNr = "12345";
184 when(sjekk.loggetInn()).thenReturn("12345678901"); // mock log-in
185 when(repository.slettKonto(KontoNr)).thenReturn("OK"); // mock successful delete
186

```

```

187    // act
188    String result = controller.slettKonto(KontoNr); // attempt to delete account
189
190    // assert
191    assertEquals("OK", result); // Should be OK if it worked
192 }
193
194 @Test
195 public void test_slettKonto_ikkeLoggetInn() {
196     // arrange
197     String kontoNr = "12345";
198     when(sjekk.loggetInn()).thenReturn(null); // mock not being logged in
199
200     // act
201     String result = controller.slettKonto(kontoNr); // attempt to delete account when not logged in
202
203     // assert
204     assertEquals("Ikke innlogget", result); //
205 }
206
207 @Test
208 public void test_slettKonto_error() {
209     // arrange
210     String kontoNr = "12345";
211     when(sjekk.loggetInn()).thenReturn("12345678901"); // mock log-in
212     when(repository.slettKonto(kontoNr)).thenReturn("Feil kononummer"); // mocks error with the
elusive typo
213
214     // act
215     String result = controller.slettKonto(kontoNr); // attempts to delete
216
217     // assert
218     assertEquals("Feil kononummer", result); // We love typos
219 }
220
221
222 }
223

```

```

1 package oslomet.testing.enhetstester;
2
3 import org.junit.Test;
4 import org.junit.runner.RunWith;
5 import org.mockito.InjectMocks;
6 import org.mockito.Mock;
7 import org.mockito.junit.MockitoJUnitRunner;
8 import oslomet.testing.Sikkerhet.Sikkerhet;
9 import oslomet.testing.Models.Kunde;
10 import oslomet.testing.DAL.AdminRepository;
11 import oslomet.testing.API.AdminKundeController;
12
13 import java.util.ArrayList;
14 import java.util.List;
15
16 import static org.junit.jupiter.api.Assertions.assertEquals;
17 import static org.junit.jupiter.api.Assertions.assertNull; // Adds the assertNull method for
18 import static org.mockito.Mockito.*;
19
20 @RunWith(MockitoJUnitRunner.class)
21 public class EnhetsTestAdminKunde {
22
23     @InjectMocks
24     // denne skal testes
25     private AdminKundeController controller;
26
27     @Mock
28     // denne skal Mock'es
29     private AdminRepository repository;
30
31     @Mock
32     // Mock'es
33     private Sikkerhet sjekk;
34
35
36     // Test to see if hentAlle method returns all customers if logged in
37     @Test
38     public void test_hentAlleLoggetInn() {
39         // arrange
40         String mockPNr = "12345678901"; // mock personnummer
41         List<Kunde> mockKunder = new ArrayList<>();
42         when(sjekk.loggetInn()).thenReturn(mockPNr); //mocks logging in
43         when(repository.hentAlleKunder()).thenReturn(mockKunder); // mocks returning all customers
44
45         // act
46         List<Kunde> result = controller.hentAlle(); // returns all customers
47
48         // assert
49         assertEquals(mockKunder, result); // Should be the same if it worked
50     }
51
52     // Test to see if hentAlle method returns null if not logged in
53     @Test
54     public void test_hentAlleIkkeLoggetInn() {
55         // arrange
56         when(sjekk.loggetInn()).thenReturn(null); // mocks not being logged in (returning null means not
57         // logged in)
58
59         // act
60         List<Kunde> result = controller.hentAlle(); // (tries) to return all customers
61
62         // assert
63         assertNull(result); // Should be null if not logged in

```

```

63 }
64
65
66 // Test to see if lagreKunde method works if logged in
67 @Test
68 public void test_lagreKundeLoggetInn() {
69     // arrange
70     String mockPNr = "12345678901"; // mock personnummer
71     Kunde mockKunde = new Kunde("12345678901",
72         "Lene", "Jensen", "Askerveien 22", "3270",
73         "Asker", "22224444", "HeiHei");
74
75     when(sjekk.loggetInn()).thenReturn(mockPNr); // mocks logging in
76     when(repository.registrerKunde(mockKunde)).thenReturn("OK"); // mocks registering a customer
77
78     // act
79     String result = controller.lagreKunde(mockKunde); // registers a customer
80
81     // assert
82     assertEquals("OK", result); // Should be "OK" if it worked
83 }
84
85 // Test to see if lagreKunde method returns "Ikke logget inn" if not logged in
86 @Test
87 public void test_lagreKundeIkkeLoggetInn() {
88     // arrange
89     when(sjekk.loggetInn()).thenReturn(null); // mocks not being logged in
90
91     // act
92     String result = controller.lagreKunde(new Kunde());
93
94     // assert
95     assertEquals("Ikke logget inn", result); // Should be "Ikke logget inn" if not logged in
96 }
97
98
99 // Test to see if endre method works if logged in
100 @Test
101 public void test_endreLoggetInn() {
102     // arrange
103     String mockPNr = "12345678901"; // mock pnummer
104     Kunde mockKunde = new Kunde("12345678901",
105         "Lene", "Jensen", "Askerveien 22", "3270",
106         "Asker", "22224444", "HeiHei");
107
108     when(sjekk.loggetInn()).thenReturn(mockPNr); // mock log in
109     when(repository.endreKundeInfo(mockKunde)).thenReturn("OK"); // mocks endring
110
111     // act
112     String result = controller.endre(mockKunde);
113
114     // assert
115     assertEquals("OK", result); // Should be "OK" if it worked
116
117 }
118
119 // Test to see if endre method returns "Ikke logget inn" if not logged in
120 @Test
121 public void test_endreIkkeLoggetInn() {
122     // arrange
123     Kunde mockKunde = new Kunde("12345678901",
124         "Lene", "Jensen", "Askerveien 22", "3270",
125         "Asker", "22224444", "HeiHei");

```

```

126     when(sjekk.loggetInn()).thenReturn(null); // null = not logged in
127
128     // act
129     String result = controller.endre(mockKunde);
130
131     // assert
132     assertEquals("Ikke logget inn", result); // Should be "Ikke logget inn" if not logged in
133 }
134
135
136 // Test to see if slett method works if logged in
137 @Test
138 public void test_slettLoggetInn() {
139     // arramge
140     String mockPNr = "12345678901"; // mock pnummer
141
142     when(sjekk.loggetInn()).thenReturn(mockPNr); // mock log in
143     when(repository.slettKunde(mockPNr)).thenReturn("OK"); // mocks slett
144
145
146     // act
147     String result = controller.slett(mockPNr);
148
149     // assert
150     assertEquals("OK", result); // If it worked, it should return "OK"
151 }
152
153 // Test to see if slett method returns "Ikke logget inn" if not logged in
154 @Test
155 public void test_slettIkkeLoggetInn() {
156     // arrange
157     String mockPNr = "69696969696"; // mock personnummer
158     when(sjekk.loggetInn()).thenReturn(null); // null = not logged in
159
160
161     // act
162     String result = controller.slett(mockPNr);
163
164     //assert
165     assertEquals("Ikke logget inn", result); // If not logged in, it should return "Ikke logget inn
166
167 }

```

```

1 package oslomet.testing.enhetstester;
2
3 import org.junit.Test;
4 import org.junit.runner.RunWith;
5 import org.mockito.InjectMocks;
6 import org.mockito.Mock;
7 import org.mockito.junit.MockitoJUnitRunner;
8
9 // Not used:
10 // import org.springframework.web.bind.annotation.GetMapping;
11 // import org.springframework.web.bind.annotation.PostMapping;
12 // import org.springframework.web.bind.annotation.RequestBody;
13
14 import oslomet.testing.API.BankController;
15 import oslomet.testing.DAL.BankRepository;
16 import oslomet.testing.Models.Konto;
17 import oslomet.testing.Models.Kunde;
18 import oslomet.testing.Models.Transaksjon;
19 import oslomet.testing.Sikkerhet.Sikkerhet;
20
21 import java.util.ArrayList;
22 import java.util.List;
23
24 import static org.junit.jupiter.api.Assertions.assertEquals;
25 import static org.junit.jupiter.api.Assertions.assertNull;
26 import static org.mockito.ArgumentMatchers.anyString;
27 import static org.mockito.ArgumentMatchers.any;
28 import static org.mockito.Mockito.when;
29
30 @RunWith(MockitoJUnitRunner.class)
31 public class EnhetstestBankController {
32
33     @InjectMocks
34     // denne skal testes
35     private BankController bankController;
36
37     @Mock
38     // denne skal Mock'es
39     private BankRepository repository;
40
41     @Mock
42     // denne skal Mock'es
43     private Sikkerhet sjekk;
44
45     @Test
46     public void hentKundeInfo_loggetInn() { // Tests that method returns correct info if the user is
logged in
47         // arrange
48         Kunde enKunde = new Kunde("01010110523",
49             "Lene", "Jensen", "Askerveien 22", "3270",
50             "Asker", "22224444", "HeiHei");
51
52         when(sjekk.loggetInn()).thenReturn("01010110523");
53
54         when(repository.hentKundeInfo(anyString())).thenReturn(enKunde);
55         // act
56         Kunde resultat = bankController.hentKundeInfo();
57         // assert
58         assertEquals(enKunde, resultat);
59     }
60
61     @Test
62     public void hentKundeInfo_IkkeLoggetInn() { // Tests that the method returns null if the user is

```

```

62  not logged in
63      // arrange
64      when(sjekk.loggetInn()).thenReturn(null);
65      // act
66      Kunde resultat = bankController.hentKundeInfo();
67      // assert
68      assertNull(resultat);
69  }
70
71 // Tests that the method returns the konti (accounts) if the user is logged
72 // in/logged out
73 @Test
74 public void hentKonti_LoggetInn() {
75     // arrange
76     List<Konto> konti = new ArrayList<>(); // Creates a list of accounts to test, these
77     Konto konto1 = new Konto("105010123456", "01010110523",
78         720, "Lønnskonto", "NOK", null);
79     Konto konto2 = new Konto("105010123456", "12345678901",
80         1000, "Lønnskonto", "NOK", null);
81     konti.add(konto1);
82     konti.add(konto2);
83
84     when(sjekk.loggetInn()).thenReturn("01010110523");
85
86     when(repository.hentKonti(anyString())).thenReturn(konti);
87
88     // act
89     List<Konto> resultat = bankController.hentKonti();
90
91     // assert
92     assertEquals(konti, resultat);
93 }
94
95 @Test
96 public void hentKonti_IkkeLoggetInn() {
97     // arrange
98
99     when(sjekk.loggetInn()).thenReturn(null);
100
101    // act
102    List<Konto> resultat = bankController.hentKonti();
103
104    // assert
105    assertNull(resultat);
106 }
107
108 // Tests the method that returns transactions if the user is logged in/logged
109 // out
110 @Test
111 public void hentTransaksjoner_LoggetInn() {
112     // arrange : here we are setting up the test
113     String kontoNr = "12345678901", fraDato = "2023-01-01", tilDato = "2023-12-31";
114     // The mock account number and dates to test since these are the parameters for
115     // the method
116
117     Konto mockKonto = new Konto("105010123456", "01010110523",
118         720, "Lønnskonto", "NOK", null);
119
120     when(sjekk.loggetInn()).thenReturn("12345678901"); // Simulates a user being logged in
121     when(repository.hentTransaksjoner(kontoNr, fraDato, tilDato)).thenReturn(mockKonto);
122     // Simulates the account being returned
123
124     // act : here we are calling the method we want to test

```

```

125     Konto result = bankController.hentTransaksjoner(kontoNr, fraDato, tilDato); // Expected
126     result
127     // assert : here we are checking if the method returns the expected result
128     assertEquals(mockKonto, result); // Checks if the expected result is equal to the actual
129 }
130
131 @Test
132 public void hentTransaksjoner_ikkeLoggetInn() {
133     // arrange
134     String kontoNr = "12345678901", fraDato = "2023-01-01", tilDato = "2023-12-31";
135     // Same as loggetInn test
136
137     when(sjekk.loggetInn()).thenReturn(null); // The user is not logged in
138     // act
139     Konto result = bankController.hentTransaksjoner(kontoNr, fraDato, tilDato);
140
141     // assert
142     assertNull(result);
143     // The result should be null if the user is not logged in
144     // as the method should not return any account details if the user is not logged
145     // in
146 }
147
148 // Tests the method that returns saldi (balances) if the user is logged
149 // in/logged out
150 @Test
151 public void hentSaldi_LoggetInn() {
152     // arrange
153     List<Konto> mockKonti = new ArrayList<>(); // Creates a list of konti to test
154     Konto konto1 = new Konto("105010123456", "01010110523",
155         720, "Lønnskonto", "NOK", null);
156     Konto konto2 = new Konto("105010123456", "12345678901",
157         1000, "Lønnskonto", "NOK", null);
158     mockKonti.add(konto1);
159     mockKonti.add(konto2);
160
161     when(sjekk.loggetInn()).thenReturn("12345678901");
162     when(repository.hentSaldi(anyString())).thenReturn(mockKonti); // Simulates the account(s)
being returned
163
164     // act
165     List<Konto> result = bankController.hentSaldi();
166
167     // assert
168     assertEquals(mockKonti, result);
169 }
170
171 @Test
172 public void hentSaldi_IkkeLoggetInn() {
173     // arrange
174     when(sjekk.loggetInn()).thenReturn(null); // Not logged in
175
176     // act
177     List<Konto> result = bankController.hentSaldi();
178
179     // assert
180     assertNull(result); // The result should be null if the user is not logged in
181 }
182
183 // Tests the method that registers a payment if the user is logged in/logged out
184 // Also tests if if betaling(transaksjon) is registered successfully or not

```

```

185  @Test
186  public void registrerBetalingOK_LoggetInn() {
187      // arrange
188      Transaksjon mockTransaksjon = new Transaksjon(); // Mock transaction
189      when(sjekk.loggetInn()).thenReturn("12345678901"); // Logged in
190      when(repository.registrerBetaling(any(Transaksjon.class))).thenReturn("OK");
191      // Simulates the transaction being registered
192
193      // act
194      String result = bankController.registrerBetaling(mockTransaksjon); // Simulating a
registration
195
196      // assert
197      assertEquals("OK", result); // The result should be "OK" if the transaction is registered
198  }
199
200  @Test
201  public void registrerBetalingFeil_LoggetInn() {
202      // arrange
203      Transaksjon mockTransaksjon = new Transaksjon(); // Mock transaction
204      when(sjekk.loggetInn()).thenReturn("12345678901"); // Logged in
205      when(repository.registrerBetaling(any(Transaksjon.class))).thenReturn("Feil");
206      // ^ Simulates the transaction not being registered or something going wrong
207
208      // act
209      String result = bankController.registrerBetaling(mockTransaksjon);
210
211      // assert
212      assertEquals("Feil", result); // The result of a failed transaction when logged in should
be "Feil"
213  }
214
215  @Test
216  public void registrerBetaling_ikkeLoggetInn() {
217      // arrange
218      Transaksjon mock = new Transaksjon(); // Mock transaction
219      when(sjekk.loggetInn()).thenReturn(null); // Not logged in
220
221      // act
222      String result = bankController.registrerBetaling(mock);
223
224      // assert
225      assertNull(result); // The result should be null if the user is not logged in
226  }
227
228  // Tests the method that returns list of betalinger(transactions) if the user is
229  // logged in/logged out
230  @Test
231  public void hentBetalinger_loggetInn() {
232      // arrange
233
234      int txID = 1;
235      String fraTilKontonummer = "12345678901";
236      double belop = 1000;
237      String dato = "2023-01-01";
238      String melding = "Test";
239      String avventer = "OK";
240      String kontonummer = "12345678901";
241
242      List<Transaksjon> mock = new ArrayList<>(); // Mock list of transactions
243      Transaksjon mockTransaksjon = new Transaksjon(txID, fraTilKontonummer, belop, dato, melding
, avventer, kontonummer);
244      // int txID, String fraTilKontonummer, double belop, String dato, String

```

```

245     // melding, String avventer,
246     // String kontonummer
247     mock.add(mockTransaksjon);
248
249
250     when(sjekk.loggetInn()).thenReturn("12345678901");
251     when(repository.hentBetalinger(anyString())).thenReturn(mock);
252
253     // act
254     List<Transaksjon> result = bankController.hentBetalinger(); // Should return a list of
transactions
255
256     // assert
257     assertEquals(mock, result);
258 }
259
260 @Test
261 public void hentBetalinger_ikkeLoggetInn() {
262     // arrange
263     when(sjekk.loggetInn()).thenReturn(null);
264
265     // act
266     List<Transaksjon> result = bankController.hentBetalinger();
267
268     // assert
269     assertNull(result);
270 }
271
272 // Tests the method that performs a betaling(transaction) if the user is logged
273 // in/logged out
274 @Test
275 public void utførBetaling_LoggetInn() {
276     // arrange
277     int txID = 1; // Mocks a transaction ID
278
279
280     List<Transaksjon> mock = new ArrayList<>(); // Mock list of transactions
281
282     when(sjekk.loggetInn()).thenReturn("12345678901");
283     when(repository.utførBetaling(txID)).thenReturn("OK"); // Simulates a successful
transaction being performed
284     // Even though theres no test for if theres enough money in the account to
285     // perform the transaction, we assume there is
286
287     // act
288     List<Transaksjon> result = bankController.utførBetaling(txID);
289
290     // assert
291     assertEquals(mock, result); // The result should be a list of transactions
292 }
293
294 @Test
295 public void utførBetaling_ikkeLoggetInn() {
296     // arrange
297     int txID = 1;
298     when(sjekk.loggetInn()).thenReturn(null); // as per usual, not logged in
299
300     // act
301     List<Transaksjon> result = bankController.utførBetaling(txID);
302
303     // assert
304     assertNull(result); // as per, the result should be null if the user is not logged in
305 }

```

```

306
307     // Tests the method that updates the customer's info if the user is logged
308     // in/logged out
309     @Test
310     public void endreKundeInfo_loggetInn() {
311         // arrange
312         Kunde mockKunde = new Kunde("12345678901",
313             "Lene", "Jensen", "Askerveien 22", "3270",
314             "Asker", "22224444", "HeiHei");
315
316         when(sjekk.loggetInn()).thenReturn("12345678901");
317         when(repository.endreKundeInfo(any(Kunde.class))).thenReturn("OK");
318         // Simulates successfully updating their info since OK is only returned if
319         // successful
320
321         // act
322         String result = bankController.endre(mockKunde);
323
324         // assert
325         assertEquals("OK", result);
326         // The result should be "OK" if the user is logged in and the info is
327         // successfully updated
328
329     }
330
331     @Test
332     public void endreKundeInfo_ikkeLoggetInn() {
333         // arrange
334         Kunde mockKunde = new Kunde("12345678901",
335             "Lene", "Jensen", "Askerveien 22", "3270",
336             "Asker", "22224444", "HeiHei");
337         when(sjekk.loggetInn()).thenReturn(null); // Not logged in
338
339         // act
340         String result = bankController.endre(mockKunde);
341
342         // assert
343         assertNull(result);
344     }
345 }
346

```

✓ EnhetsTestAdminKonto (os 224 ms)	✓ EnhetstestBankController (- 421 ms)
✓ test_endreKonto_loggetInn 216 ms	✓ hentKonti_IkkeLoggetInn 407 ms
✓ test_registrerKonto_Error 1 ms	✓ hentBetalinger_IkkeLogget 1 ms
✓ test_endreKonto_FeilPNr 0 ms	✓ hentKundeInfo_loggetInn 3 ms
✓ test_endreKonto_ikkeLogg 1 ms	✓ utførBeting(LoggetInn) 1 ms
✓ test_registrerKonto_NotLogg 1 ms	✓ registrerBetingOK_Logge 1 ms
✓ test_endreKonto_feilKonto 1 ms	✓ hentTransaksjoner_IkkeLogget 0 ms
✓ test_hentAlleIkkeLoggetInn 0 ms	✓ endreKundeInfo_loggetInn 0 ms
✓ test_hentAlleKontiLoggetInn 1 ms	✓ endreKundeInfo_IkkeLogg 1 ms
✓ test_slettKonto_error 1 ms	✓ hentKonti_LoggetInn 1 ms
✓ test_registrerKonto_Logge 0 ms	✓ registrerBeting_IkkeLogg 1 ms
✓ test_slettKonto_loggetInn 0 ms	✓ registrerBetingFeil_Logget 0 ms
✓ test_slettKonto_ikkeLogge 1 ms	✓ hentKundeInfo_Ikkelogget 0 ms
✓ test_endreKonto_error 1 ms	✓ hentSaldi_LoggetInn 1 ms
	✓ hentTransaksjoner_Logget 1 ms
	✓ utførBeting_IkkeLoggetInn 1 ms
	✓ hentBetalinger_loggetInn 1 ms
	✓ hentSaldi_IkkeloggetInn 1 ms

✓ EnhetstestSikkerhet (oslom 298 ms)	✓ EnhetsTestAdminKunde (os 311 ms)
✓ test_loggetInn 292 ms	✓ test_slettLoggetInn 304 ms
✓ test_loggUt 3 ms	✓ test_endreLoggetInn 1 ms
✓ test_logglInnAdminFeil 1 ms	✓ test_hentAlleLoggetInn 1 ms
✓ test_sjekkLoggetInn 2 ms	✓ test_slettIkkeLoggetInn 1 ms
✓ test_logglInnAdminOK 0 ms	✓ test_endrelkkeloggetInn 1 ms
	✓ test_lagreKundeIkkeLogge 1 ms
	✓ test_lagreKundeLoggetInn 1 ms
	✓ test_hentAlleIkkeLoggetInn 1 ms

Element ^	Class, %	Method, %	Line, %	Branch, %
all	100% (3/3)	94% (16/17)	98% (71/72)	97% (33/34)
oslomet.testing.API	100% (3/3)	94% (16/17)	98% (71/72)	97% (33/34)
AdminKontoController	100% (1/1)	100% (4/4)	100% (18/18)	100% (8/8)
AdminKundeController	100% (1/1)	100% (4/4)	100% (17/17)	100% (8/8)
BankController	100% (1/1)	88% (8/9)	97% (36/37)	94% (17/18)
oslomet.testing.enhetstester	100% (4/4)	100% (43/43)	100% (219/219)	100% (0/0)
EnhetsTestAdminKonto	100% (1/1)	100% (13/13)	100% (62/62)	100% (0/0)
EnhetsTestAdminKunde	100% (1/1)	100% (8/8)	100% (38/38)	100% (0/0)
EnhetstestBankController	100% (1/1)	100% (17/17)	100% (93/93)	100% (0/0)
EnhetstestSikkerhet	100% (1/1)	100% (5/5)	100% (26/26)	100% (0/0)
oslomet.testing.Sikkerhet	100% (1/1)	100% (5/5)	83% (20/24)	58% (7/12)
Sikkerhet	100% (1/1)	100% (5/5)	83% (20/24)	58% (7/12)

▼ ✗ Admin

✓ Admin - Log In

✗ Admin - Endre Kunde Info

✗ Admin - Registrer Kunde

✓ Admin - Slett Kunde

✗ Admin - Endre Konto

✗ Admin - Registrer Konto

✓ Admin - Slett Konto

✓ Admin - Log Ut

▼ ✗ Bruker

✓ Bruker - Se Saldo

✓ Bruker - Log in

✗ Bruker - Registrer Betaling

✓ Bruker - Utfør betaling

✓ Bruker - Endre Kunde Info

✓ Bruker - Vis transaksjoner

✓ Bruker - Log ut

## Test plan [6](#): Systemtesting av Nettbank

### Properties

---

Area Path:	Nettbank App
Iteration:	Nettbank App
Owner:	Alexander Cody Mc Corkle
State:	Active
Start date:	Tuesday, March 12, 2024
End date:	Tuesday, March 19, 2024

## Test suite [7](#): Systemtesting av Nettbank

### Test suite [41](#): Admin - Log In

#### Test cases (1)

---

##### Test case [30](#): Admin - Log Inn

###### PROPERTIES

Test Case Id:	30
Assigned To:	Alexander Cody Mc Corkle
State:	Design

###### STEPS

#	Action	Expected value	Attachments
1	Åpner nettbank	viser velkomst siden	
2	Trykker på Logg Inn Admin knappen	Får opp innloggsiden til admin	
3	Taster inn 'Admin' i bruker, og 'Admin' i passord		
4	Trykker på Logg Inn	Logger inn som admin og viser adminKunde siden	

###### LATEST TEST OUTCOME

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Tuesday, March 12, 2024	0:00:00.000	0

---

## Test suite [59](#): Admin - Registrer Konto

#### Test cases (1)

---

##### Test case [61](#): Admin - Registrer Konto

###### PROPERTIES

Test Case Id:	61
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Åpne localhost:8080	Viser websiden	
2	Trykker på Logg Inn Admin knapp og skriver inn Admin Admin og trykker 'Logg Inn'	Logger inn og viser liste av Kunder	
3	Navigerer til 'Registrer Konto'	Viser skjema for registrering av ny Konto	
4	Skriver inn i Kontonummer: '12341234123', i Type: 'Sparekonto'. i Valute: 'NOK', i Personnummer: '12345678910' og deretter trykker på 'Registrer Konto' knappen	Konto bør bli registrert og helst vise noe form for bekrefteelse	
5	Naviger til Endre Konto for å se om Konto har blitt registrert	Burde se den nye kontoen i listen av kontoer	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Failed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [57](#): Admin - Endre Konto

### Test cases (1)

#### Test case [63](#): Admin - Endre Konto

**PROPERTIES**

Test Case Id:	63
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Logger inn som Admin, med 'Admin' 'Admin'	Burde komme inn i Admin siden	
2	Navigerer til 'Endre Konto' siden	Får opp liste over kontoer registrert til kunder	
3	Endrer øverste konto (105010123456) og oppdaterer Saldo til 1000000.00, og trykker på 'Endre' knappen på Høyre.	Får noe bekrefteelse over endring	
4	For å bekrefte at det gikk gjennom, så refresher vi siden	Saldo bør fortsatt være 1000000.0	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Failed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [58](#): Admin - Slett Konto

### Test cases (1)

#### Test case [65](#): Admin - Slett Konto

**PROPERTIES**

Test Case Id:	65
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Åpne localhost:8080 og logg inn som Admin med bruker 'Admin', passord 'Admin'	Får logger inn som Admin	
2	Navigerer til Endre Konto siden	Får opp liste av Kontoer registrert	
3	Trykker på 'Slett' knappen til høyre på kontonr (105010123456)	Får opp bekreftelse 'Slett kunde med kontonummer 105010123456'	
4	Trykker OK på Bekrefte	Konto er fjernet fra listen	
5	Refresher siden for å bekrefte	Konto er fortsatt borte	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [55](#): Admin - Registrer Kunde

Test cases (1)

### Test case [66](#): Admin - Registrer Kunde

**PROPERTIES**

Test Case Id:	66
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Åpner localhost:8080 og logger inn som Admin med bruker 'Admin', passord 'Admin'.	Kommer inn pa websiden	
2	Navigerer til 'Registrer Kunde' siden	Viser registreringsskjemaet for ny Kunde	
3	Skriver inn Personnummer '12341234123', Fornavn: 'Test', Etternavn: 'Test', Adresse: 'Testveien 22', Postnr: '4321', Poststed: 'Oslo', Telefonnr: '12341234', Passord: 'TestTest' og trykker på Registrer Kunde	Får noe form for bekreftelse	
4	Navigerer til Endre Kunde for å se at den har blitt registrert	Ser ny kunde i liste av Kunder	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Failed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [54](#): Admin - Endre Kunde Info

Test cases (1)

### Test case [68](#): Admin - Endre Kunde Info

**PROPERTIES**

Test Case Id:	68
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Åpne localhost:8080 og logg inn som Admin med bruker 'Admin', passord 'Admin'	Logger inn og ser start siden til Admin	
2	Navigerer til 'Endre Kunde'	Viser liste av Kunder	
3	Endrer på Per Hansen Osloveien 82 til 'Test' 'Test' 'Testveien 82' og trykker på 'Endre'	Noe form for bekrefte	
4	Refresher siden for å se om endringer er registrert	Endret navn bør fortsatt vises	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Failed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [56](#): Admin - Slett Kunde

### Test cases (1)

#### Test case [70](#): Admin - Slett Kunde

**PROPERTIES**

Test Case Id:	70
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Åpner localhost:8080 og logger inn som Admin med bruker 'Admin', passord 'Admin'	Kommer seg inn på websiden som Admin og ser startsiden	
2	Navigerer til 'Endre Kunde' siden	Ser liste av Kunder	
3	Trykker på 'Slett' knappen på høyre siden for å slette kunde Per Hansen	Får opp bekrefte 'Slett kunde med personnummer 12345678910'	
4	Trykker på 'OK'	Per Hansen er ikke lengre i listen	
5	Refresher for å bekrefte	Per Hansen fortsatt ikke i listen	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [60](#): Admin - Log Ut

### Test cases (1)

#### Test case [71](#): Admin - Log Ut

**PROPERTIES**

Test Case Id:	71
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Åpner localhost:8080 og logger inn som Admin med bruker 'Admin', passord 'Admin'	Logger inn i websiden og ser startsiden som Admin	
2	Trykker på Logg Ut på høyre side	Blir tatt tilbake til Logg Inn siden	
3	Navigerer til Saldo og Kunde Info for å se at det ikke er noe informasjon synlig	Blanke sider	
4			

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Friday, March 15, 2024	0:00:00.000	0

## Test suite [22](#): Bruker - Log In

### Test cases (1)

#### Test case [29](#): Bruker - Log Inn

**PROPERTIES**

Test Case Id:	29
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	open browser	Få opp velkomst siden	
2	Trykker på logg inn	Får opp Logg in skjema	
3	Taster inn 12345678910 i personnummer og 'HeiHei' som passord.		
4	Trykk på logg inn	Får opp Saldo oversikt	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Tuesday, March 12, 2024	0:00:00.000	0

## Test suite [23](#): Bruker - Registrer Betaling

### Test cases (1)

#### Test case [31](#): Bruker - Registrer en betaling

**PROPERTIES**

Test Case Id:	31
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Gå til localhost:8080	Får opp logg inn siden	
2	Log inn som bruker	Viser saldo siden	
3	Navigerer til 'Registrer Betaling'	Får opp registrer betalings vinduet	
4	Velger kontonr (105010123456), skriver inn motakker kontoNr (12345123456), fyller inn dato (2020-10-10), skriver beløp (500.00)og skriver melding (Test)		
5	Trykker på Registrer Betaling's knappen	Skal vise noe form for confirmation	
6	Navigerer til utfør betaling for å se om betaling er registrert	Test betalingen bør vises i tabellen	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Failed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Tuesday, March 12, 2024	0:00:00.000	0

## Test suite [35](#): Bruker - Utfør Betaling

### Test cases (1)

#### Test case [33](#): Bruker - Utfør Betaling

**PROPERTIES**

Test Case Id:	33
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Open localhost:8080	Home page is shown	
2	Log in with personnummer (12345678910) and password (HeiHei)	Successfully logged in and see Saldo	
3	Navigate to Utfør Betalinger page	See pending payments	
4	Click on Utfør betaling for Skatten	Payment should disappear from table if successful	
5	Navigate to Saldo page	New balance for Skatten should be 319.60	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Wednesday, March 13, 2024	0:00:00.000	0

## Test suite [36](#): Bruker - Endre Kunde Info

### Test cases (1)

#### Test case [42](#): Edit Kunde Info

**PROPERTIES**

Test Case Id:	42
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Open localhost:8080	Opens up webpage	
2	Logg inn with '12345678910' and 'HeiHei'	Successfully log in	
3	Navigate to 'Kunde Info' page	Displays Kunde Info page with inputs	
4	Change Fornavn and Etternavn to 'Test', Address to 'Testveien 22', Postnr to '1234', poststed to 'Oslo' and telefonnr to '22224445' and click on Endre Info	Successfully changes the info	
5			

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Wednesday, March 13, 2024	0:00:00.000	0

## Test suite [37](#): Bruker - Vis Transaksjoner

### Test cases (1)

#### Test case [43](#): View transaksjoner

**PROPERTIES**

Test Case Id:	43
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Open localhost:8080	Website displays correctly	
2	Log in with user: 12345678910 and password: HeiHei	Logs in successfully	
3	Navigate to 'Transaksjoner'	Shows Transaksjoner page with inputs	
4	Select kontonr '105010123456', type in '2000-01-01' in 'FraDato' and '2020-01-01' in 'TilDato' and click 'Søk' period	Correctly displays all transactions with the selected period	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Wednesday, March 13, 2024	0:00:00.000	0

## Test suite [38](#): Bruker - Se Saldo

### Test cases (1)

#### Test case [34](#): Bruker - Se Saldo

**PROPERTIES**

Test Case Id:	34
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Open localhost:8080	Successfully display website	
2	Type in 12345678910 and HeiHei and click Logg Inn	Successfully logged in	
3	Should be on the Saldo page, if not navigate to Saldo	Should display a table with the saldo for all accounts for the logged in user	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Wednesday, March 13, 2024	0:00:00.000	0

---

## Test suite [39](#): Bruker - Log Ut

### Test cases (1)

#### Test case [44](#): Log out as user

**PROPERTIES**

Test Case Id:	44
Assigned To:	Alexander Cody Mc Corkle
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Go to localhost:8080	See home page of Nettbank	
2	Type in 12345678910 and HeiHei and click Logg Inn	Should display log in page	
3	Click on the Log Ut button in the top right of the website	Should return you to log in page	
4	Navigate to Saldo	Blank page since logged out	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Alexander Cody Mc Corkle	Windows 10	Alexander Cody Mc Corkle	Wednesday, March 13, 2024	0:00:00.000	0

---