

2023

# Rapport Testing av Programvare



Hiwa Abdolahi

S364547

3/15/2023

## Identifikasjon

Denne sluttrapporten gjelder for tre forskjellige typer tester: enhetstest, integrasjonstest og systemtest. Rapporten inneholder en oppsummering av testresultatene og testprosessen for hver av disse testene. Formålet med denne rapporten er å gi en oversikt over testene som er utført, samt presentere de viktigste funnene og resultatene fra hver test. Ved å gi en omfattende rapport om alle testene, kan man ta beslutninger om hva som trenger å forbedres og hva som fungerer bra, slik at de kan lage et bedre og mer pålitelig produkt.

## Sammendrag

Testmiljøet som ble brukt for testingen av produktet besto av flere verktøy som ble brukt til ulike typer testing.

For enhetstesting ble IntelliJ IDEA brukt, som er et integrert utviklingsmiljø (IDE) som gjør det mulig å skrive og kjøre enhetstester for koden. IntelliJ IDEA ble valgt for enhetstesting på grunn av dens funksjonalitet for å teste individuelle komponenter av programvaren.

For integrasjonstesting ble SoapUI brukt. SoapUI er et testverktøy for webtjenester som gjør det mulig å teste integrasjonen mellom forskjellige komponenter av programvaren. SoapUI ble valgt for integrasjonstesting på grunn av dens funksjonalitet for å teste komplekse integrasjonsprosesser. Rapportering av testresultater fra integrasjonstesten ble gjort ved hjelp av Excel, som gjorde det mulig å presentere og analysere testresultatene på en effektiv måte.

For systemtesting ble Selenium IDE for Chrome brukt. Selenium IDE er et verktøy for å automatisere nettlesertesting, og Chrome ble valgt som nettleser på grunn av sin utbredte bruk. Selenium IDE ble valgt for systemtesting på grunn av dens funksjonalitet for å simulere brukerhandlingene og interaksjonene med

programvaren. Rapportering av testresultater fra systemtestingen ble gjort ved hjelp av Microsoft Test Manager (MTM), som gjorde det mulig å samle inn og analysere testresultater fra forskjellige testverktøy og presentere dem på en konsolidert måte.

Samlet sett ble disse testverktøyene brukt for å sikre at produktet ble testet grundig og nøyaktig under forskjellige forhold og situasjoner, og for å kunne rapportere testresultatene på en effektiv måte.

## Summendrag

I enhetstest for AdminKontoController-klassen. Testene sjekker funksjonaliteten til metodene i klassen, som registrerKonto(), endreKonto(), slettKonto() og hentAlleKonti().

Resultatene viser at alle testene var vellykkede, med forventede verdier returnert og ingen feil. Dermed kan vi si at klassen fungerer som den skal og alle testene er bestått.

Testklassen EnhetstestAdminKundeController inneholder enhetstester for AdminKundeController-klassen og tester om funksjonaliteten fungerer som forventet. Testresultatene viser at alle enhetstestene er vellykkede og funksjonaliteten testet i hver testmetode fungerer som forventet.

I enhetstester for klassen BankController, som er ansvarlig for å håndtere HTTP-forespørsler til en bank-API. Testene bruker JUnit og Mockito.

Testene sjekker at BankController fungerer som den skal i forskjellige scenarier, for eksempel når en bruker er logget inn eller ikke, og når det gjelder henting av kundeinformasjon, kontoinformasjon, saldo og transaksjoner. Testene bruker mock-objekter for å simulere at de faktiske avhengighetene fungerer som de skal.

Alle testene ser ut til å bestå, som indikert av at assertEquals og assertNull ikke gir noen feil når de sjekker de faktiske og forventede resultatene.

I enhetstester for sikkerhetsklassen. Testene sjekker om sikkerhetsklassens metoder fungerer som forventet i ulike situasjoner. Testene inkluderer sjekk av gyldige og ugyldige pålogginger, sjekk av

logginn for administratorer, logg ut funksjonalitet, og sjekk av innloggingstilstanden til en bruker. Testene er skrevet ved bruk av Mockito, og kjører alle uten feil.

## Sammendrag av aktiviteter og lærdom:

Som eneste person som jobbet med obligen, ble det viktig for meg å ha en god plan og struktur for testingen. Først startet jeg med å lage en detaljert testplan som inkluderte alle funksjonalitetene som skulle testes og de forventede resultatene.

Underveis i testingen oppstod det noen utfordringer når det gjaldt testspesifikasjoner. Det var noen ganger uklart hva som skulle være det forventede resultatet av en test, og dette måtte diskuteres og avklares med utviklerne.

Jeg utnyttet en rekke testverktøy i løpet av testprosjektet, inkludert JUnit for enhetstesting, SoapUI for API-testing, Excel for rapportering, og Selenium med MTM for systemtesting. Jeg fant ut at disse verktøyene var både brukervennlige og gav god oversikt over testresultatene.

En viktig lærdom fra testprosjektet er at det er viktig å tenke på testingen allerede når man skriver koden. Dette kan hjelpe til med å unngå mange av utfordringene som oppstod under testingen, som uklare testspesifikasjoner og uforutsette feil.

En annen lærdom er at det kan være nyttig å involvere flere personer i testingen, spesielt når man jobber alene med et prosjekt. Å ha en annen persons perspektiv kan avdekke feil og utfordringer som man kanskje ikke ville ha oppdaget selv.

Hvis jeg skulle gjort noe annerledes i testprosjektet, ville jeg ha satt av mer tid til testingen og involvert flere personer i prosessen. Dette ville ha gitt mer tid til å avdekke og løse eventuelle feil og utfordringer, og kunne ha ført til et enda sterkere og pålitelig system.

Coverage: EnhetstestBankController.testHentTransaksjoner

Element	Class, %	Method, %	Line, %
oslomet.testing	70% (7/10)	27% (24/86)	36% (119/322)
API	100% (3/3)	100% (16/16)	100% (71/71)
AdminKontoController	100% (1/1)	100% (4/4)	100% (18/18)
AdminKundeController	100% (1/1)	100% (4/4)	100% (17/17)
BankController	100% (1/1)	100% (8/8)	100% (36/36)
DAL	0% (0/2)	0% (0/17)	0% (0/158)
Models	100% (3/3)	8% (4/48)	36% (25/69)
Sikkerhet	100% (1/1)	100% (4/4)	100% (23/23)
Sikkerhet	100% (1/1)	100% (4/4)	100% (23/23)
Main_Nettbank	0% (0/1)	0% (0/1)	0% (0/1)

```

public void testEndreKontoNotLoggedIn() {
    // Arrange
    Konto konto = new Konto(personnummer: "1234567890", kontonummer: "1"
        saldo: 1000, type: "Brukskonto", valuta: "nok", transaksjoner: null);
    String expected = "Ikke innlogget";
    when(sjekk.loggetInn()).thenReturn(value: null);

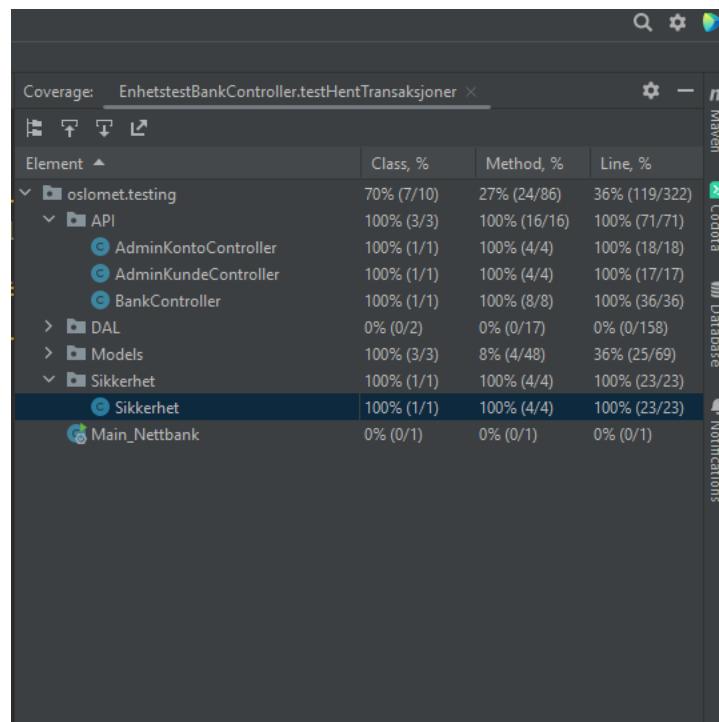
    // Act
    String actual = adminKontoController.endreKonto(konto);

    // Assert
    Assert.assertEquals(expected, actual);
}

no usages
@Test
public void testSlettKonto_OK() {
    // Arrange
    String kontonummer = "12345678901";
    String personnummer = "01010112345";
    when(sjekk.loggetInn()).thenReturn(personnummer);
    when(adminRepository.slettKonto(kontonummer)).thenReturn(value: null);

    // Act
    String result = adminKontoController.slettKonto(kontonummer);
}

```



Admincontroller :

```

10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```

4 usages  
`@Autowired`  
`private Sikkerhet sjekk;`

2 usages  
`@GetMapping("hektAlle")`  
`public List<Konto> hektAlleKonti() {`  
 `String personnummer = sjekk.loggetInn();`  
 `if (personnummer!=null) {`  
 `return repository.hektAlleKonti();`  
 `}`  
 `return null;`

2 usages  
`@PostMapping("register")`  
`public String registerKonto(@RequestBody Konto konto) {`  
 `String personnummer = sjekk.loggetInn();`  
 `if (personnummer!=null) {`  
 `String retur = repository.registerKonto(konto);`  
 `return retur;`  
 `}`  
 `return "Ikke innlogget";`

2 usages  
`@PostMapping("endre")`  
`public String endreKonto(@RequestBody Konto konto) {`  
 `String personnummer = sjekk.loggetInn();`  
 `if (personnummer!=null) {`  
 `return repository.endreKonto(konto);`  
 `}`  
 `return "Ikke innlogget";`

2 usages  
`@GetMapping("slett")`  
`public String slettKonto(String kontonummer) {`  
 `String personnummer = sjekk.loggetInn();`  
 `if (personnummer!=null) {`  
 `return repository.slettKonto(kontonummer);`  
 `}`  
 `return "Ikke innlogget";`

adminKunde controller :

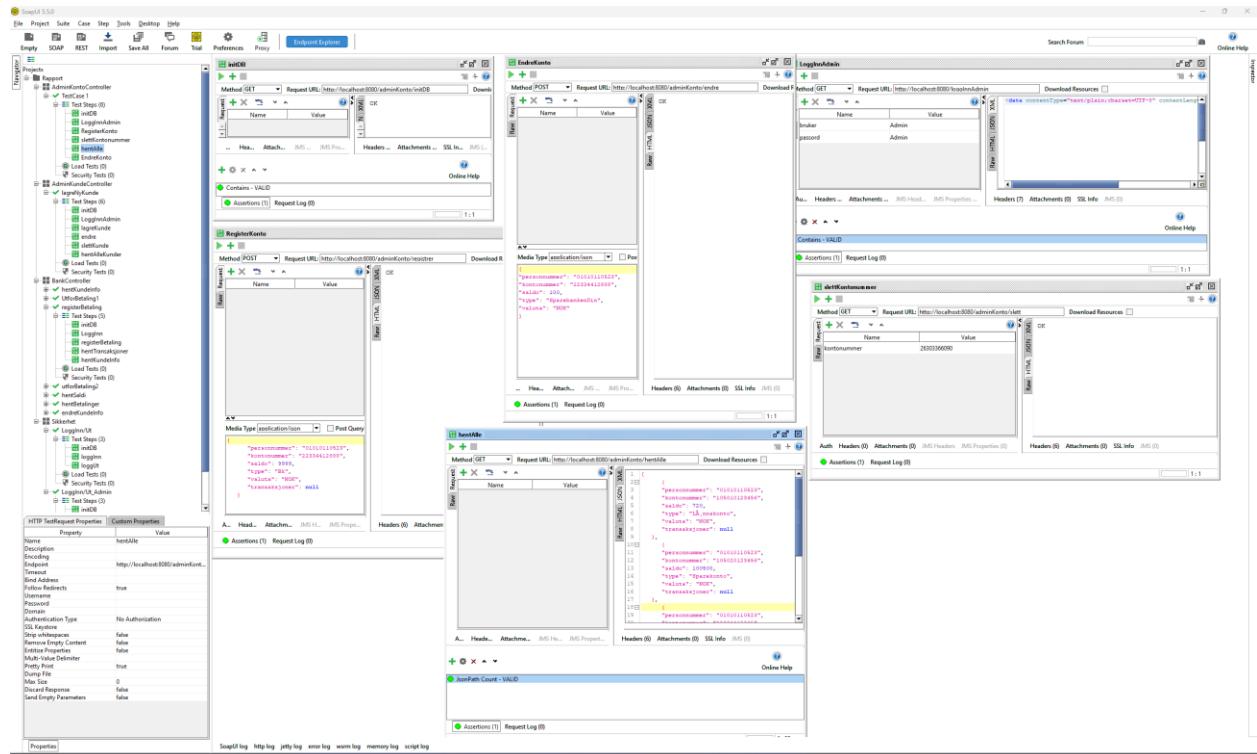
```
2 usages
@GetMapping("/hentAlle")
public List<Kunde> hentAlle() {
    String personnummer = sjekk.loggetInn();
    if (personnummer!=null) {
        return repository.hentAlleKunder();
    }
    return null;
}

2 usages
@PostMapping("/lagre")
public String lagreKunde(@RequestBody Kunde innKunde) {
    String personnummer = sjekk.loggetInn();
    if (personnummer!=null) {
        return repository.registrerKunde(innKunde);
    }
    return "Ikke logget inn";
}

@PostMapping("/endre")
public String endre(@RequestBody Kunde innKunde) {
    String personnummer = sjekk.loggetInn();
    if (personnummer!=null) {
        return repository.endreKundeInfo(innKunde);
    }
    return "Ikke logget inn";
}

2 usages
@GetMapping("/slett")
public String slett(String personnummer) {
    String p = sjekk.loggetInn();
    if (p !=null) {
        return repository.slettKunde(personnummer);
    }
    return "Ikke logget inn";
}
```

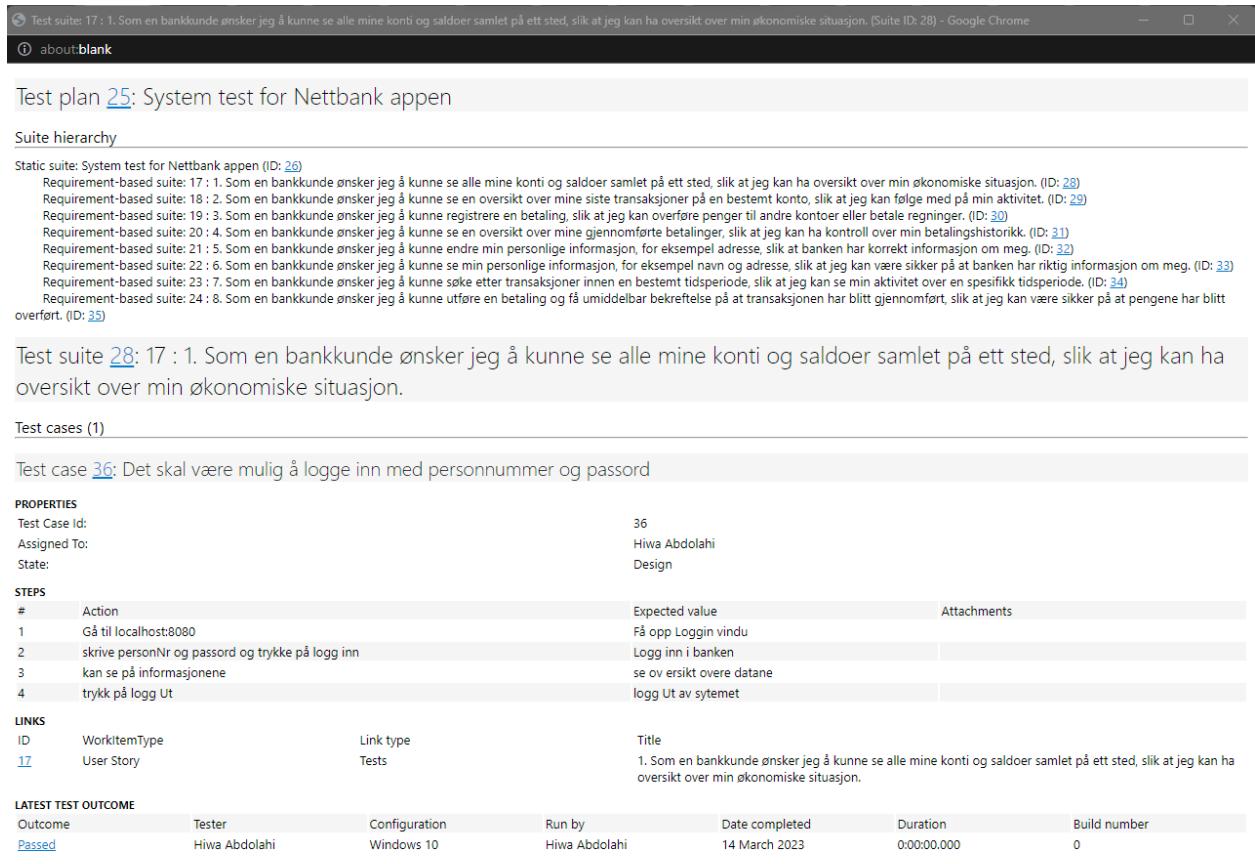
## Integrasjon Test:



## System test:

### Bruker history:

1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon.



Test plan [25](#): System test for Nettbank appen

Suite hierarchy

Static suite: System test for Nettbank appen (ID: [26](#))

- Requirement-based suite: 17 : 1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon. (ID: [28](#))
  - Requirement-based suite: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet. (ID: [29](#))
  - Requirement-based suite: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger. (ID: [30](#))
  - Requirement-based suite: 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk. (ID: [31](#))
  - Requirement-based suite: 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel navn og adresse, slik at banken har korrekt informasjon om meg. (ID: [32](#))
  - Requirement-based suite: 22 : 6. Som en bankkunde ønsker jeg å kunne se min personlige informasjon, for eksempel navn og adresse, slik at jeg kan være sikker på at banken har riktig informasjon om meg. (ID: [33](#))
  - Requirement-based suite: 23 : 7. Som en bankkunde ønsker jeg å kunne søke etter transaksjoner innen en bestemt tidsperiode, slik at jeg kan se min aktivitet over en spesifikk tidsperiode. (ID: [34](#))
  - Requirement-based suite: 24 : 8. Som en bankkunde ønsker jeg å kunne utføre en betaling og få umiddelbar bekrefelse på at transaksjonen har blitt gjennomført, slik at jeg kan være sikker på at pengene har blitt overført. (ID: [35](#))

Test suite [28](#): 17 : 1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon.

Test cases (1)

Test case [36](#): Det skal være mulig å logge inn med personnummer og passord

**PROPERTIES**

Test Case Id:	36
Assigned To:	Hiwa Abdolah
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	Gå til localhost:8080	Få opp Loggin vindu	
2	skrive personNr og passord og trykke på logg inn	Logg inn i banken	
3	kan se på informasjonene	se ov ertsikt over datane	
4	trykk på logg Ut	logg Ut av systemet	

**LINKS**

ID	WorkItem Type	Link type	Title
<a href="#">17</a>	User Story	Tests	1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon.

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Passed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet.

Dette var riktige, og jeg brukte kontoNr: **105010123456**

Fra Dato 2000-01-01 til 2100-01-01

Test suite: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet. (Suite ID: 29) - Google Chrome

about:blank

Test plan 25: System test for Nettbank appen

Suite hierarchy

Static suite: System test for Nettbank appen (ID: 26)

Requirement-based suite: 17 : 1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon. (ID: 28)

Requirement-based suite: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet. (ID: 29)

Requirement-based suite: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger. (ID: 30)

Requirement-based suite: 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk. (ID: 31)

Requirement-based suite: 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel navn og adresse, slik at banken har korrekt informasjon om meg. (ID: 32)

Requirement-based suite: 22 : 6. Som en bankkunde ønsker jeg å kunne se min personlige informasjon, for eksempel navn og adresse, slik at jeg kan være sikker på at banken har riktig informasjon om meg. (ID: 33)

Requirement-based suite: 23 : 7. Som en bankkunde ønsker jeg å kunne søke etter transaksjoner innen en bestemt tidsperiode, slik at jeg kan se min aktivitet over en spesifikk tidsperiode. (ID: 34)

Requirement-based suite: 24 : 8. Som en bankkunde ønsker jeg å kunne utføre en betaling og få umiddelbar bekrefteelse på at transaksjonen har blitt gjennomført, slik at jeg kan være sikker på at pengene har blitt overført. (ID: 35)

Test suite 29: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet.

Test cases (1)

Test case 37: Det skal være mulig å hente alle transaksjoner for en gitt konto

**PROPERTIES**

Test Case Id:	37
Assigned To:	Hiwa Abdolah
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	logg in med personnr 01010110523 med passord HEIHEI	logginn	
2	kan velge transaksjoner	går til transaksjon vindu	
3	kan velge alle tillengelig konto for gitt personNr	får opp liste med alle kontoer og kan bestemme fra/til dato for transaksjoner	
4	valgte konto : 20102012345. fra 2000 til 2100 trykk på hent alle transaksjoner	henter alle transaksjoner	

**LINKS**

ID	WorkItemType	Link type	Title
38	Bug	Tests	Det skal være mulig å hente alle transaksjoner for en gitt konto Failed (siden finnes info i DB om gitt konto, programme feiler å hente ikke de transaksjonene)
18	User Story	Tests	2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet.

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Passed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger.

Test suite: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger. (Suite ID: 30) - Google Chrome  
about:blank

Test plan 25: System test for Nettbank appen

Suite hierarchy

Static suite: System test for Nettbank appen (ID: 26)

- Requirement-based suite: 17 : 1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon. (ID: 28)
- Requirement-based suite: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet. (ID: 29)
- Requirement-based suite: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger. (ID: 30)
- Requirement-based suite: 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk. (ID: 31)
- Requirement-based suite: 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel navn og adresse, slik at banken har korrekt informasjon om meg. (ID: 32)
- Requirement-based suite: 22 : 6. Som en bankkunde ønsker jeg å kunne se min personlige informasjon, for eksempel navn og adresse, slik at jeg kan være sikker på at banken har riktig informasjon om meg. (ID: 33)
- Requirement-based suite: 23 : 7. Som en bankkunde ønsker jeg å kunne søke etter transaksjoner innen en bestemt tidsperiode, slik at jeg kan se min aktivitet over en spesifikk tidsperiode. (ID: 34)
- Requirement-based suite: 24 : 8. Som en bankkunde ønsker jeg å kunne utføre en betaling og få umiddelbar bekrefteelse på at transaksjonen har blitt gjennomført, slik at jeg kan være sikker på at pengene har blitt overført. (ID: 35)

Test suite 30: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger.

Test cases (1)

Test case 39: kan registrere betalinger

**PROPERTIES**

Test Case Id:	39
Assigned To:	Hiwa Abdolah
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	etter logg inn : trykk på register betaling	få opp register betaling vindu	
2	füll informasjonene	register betalinger fra info som oppgi	
3	trykk på register betaling	gir en bekrefteelse på betalingen	

**LINKS**

ID	WorkItem Type	Link type	Title
19	User Story	Tests	3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger.
40	Bug	Tests	kan registrere betalinger Failed (systemet feiler og etter klick på register betaling registerer ikke noe betalinger)

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Failed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine saldoer, slik at jeg kan ha oversikt over pengene mine.

Test suite: 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk. (Suite ID: 31) - Google Chrome

Test plan [25](#): System test for Nettbank appen

Suite hierarchy

Static suite: System test for Nettbank appen (ID: [26](#))

- Requirement-based suite: 17 : 1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon. (ID: [28](#))
- Requirement-based suite: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet. (ID: [29](#))
- Requirement-based suite: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontoer eller betale regninger. (ID: [30](#))
- Requirement-based suite: 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk. (ID: [31](#))
- Requirement-based suite: 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel adresse, slik at banken har korrekt informasjon om meg. (ID: [32](#))
- Requirement-based suite: 22 : 6. Som en bankkunde ønsker jeg å kunne se min personlige informasjon, for eksempel navn og adresse, slik at jeg kan være sikker på at banken har riktig informasjon om meg. (ID: [33](#))
- Requirement-based suite: 23 : 7. Som en bankkunde ønsker jeg å kunne søke etter transaksjoner innen en bestemt tidsperiode, slik at jeg kan se min aktivitet over en spesifikk tidsperiode. (ID: [34](#))
- Requirement-based suite: 24 : 8. Som en bankkunde ønsker jeg å kunne utføre en betaling og få umiddelbar bekrefteelse på at transaksjonen har blitt gjennomført, slik at jeg kan være sikker på at pengene har blitt overført. (ID: [35](#))

Test suite [31](#): 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk.

Test cases (1)

Test case [41](#): saldo på alle aktive kontoer

**PROPERTIES**

Test Case Id:	41
Assigned To:	Hiwa Abdolah
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	gå til localhost:8080 og trykk på logg inn	få opp logg in vindu	
2	logg inn med pNr og passord	logg inn i banken	
3	trykke på saldo	viser saldo på aktive kontoer	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel adresse, passord osv. slik at banken har korrekt informasjon om meg.

Test suite 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel adresse, slik at banken har korrekt informasjon om meg. [Suite ID: 32] - Google Chrome

about:blank

Test plan [25](#): System test for Nettbank appen

Suite hierarchy

Static suite: System test for Nettbank appen (ID: [26](#))

- Requirement-based suite: 17 : 1. Som en bankkunde ønsker jeg å kunne se alle mine konti og saldoer samlet på ett sted, slik at jeg kan ha oversikt over min økonomiske situasjon. (ID: [28](#))
- Requirement-based suite: 18 : 2. Som en bankkunde ønsker jeg å kunne se en oversikt over mine siste transaksjoner på en bestemt konto, slik at jeg kan følge med på min aktivitet. (ID: [29](#))
- Requirement-based suite: 19 : 3. Som en bankkunde ønsker jeg å kunne registrere en betaling, slik at jeg kan overføre penger til andre kontører eller betale regninger. (ID: [30](#))
- Requirement-based suite: 20 : 4. Som en bankkunde ønsker jeg å kunne se en oversikt over mine gjennomførte betalinger, slik at jeg kan ha kontroll over min betalingshistorikk. (ID: [31](#))
- Requirement-based suite: 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel adresse, slik at banken har korrekt informasjon om meg. (ID: [32](#))
- Requirement-based suite: 22 : 6. Som en bankkunde ønsker jeg å kunne se min personlige informasjon, for eksempel navn og adresse, slik at jeg kan være sikker på at banken har riktig informasjon om meg. (ID: [33](#))
- Requirement-based suite: 23 : 7. Som en bankkunde ønsker jeg å kunne søke etter transaksjoner innen en bestemt tidsperiode, slik at jeg kan se min aktivitet over en spesifikk tidsperiode. (ID: [34](#))
- Requirement-based suite: 24 : 8. Som en bankkunde ønsker jeg å kunne utføre en betaling og få umiddelbar bekrefteelse på at transaksjonen har blitt gjennomført, slik at jeg kan være sikker på at pengene har blitt overført. (ID: [35](#))

Test suite [32](#): 21 : 5. Som en bankkunde ønsker jeg å kunne endre min personlige informasjon, for eksempel adresse, slik at banken har korrekt informasjon om meg.

Test cases (1)

Test case [42](#): kan endre informasjon om brukere

PROPERTIES	
Test Case Id:	42
Assigned To:	Hiwa Abdolah
State:	Design

STEPS			
#	Action	Expected value	Attachments
1	gå til localhost:8080 trykk pålogg inn	vise logginn side til brukeren	
2	fyll inn passord og personNr og trykk pålogg inn	logg inn brukeren	
3	trykk på endre.info	få opp endre info vindu	
4	endre min adresse fra Askerveien 22 til Askerveien 1 trykk Endret info	endre brukerens info som de ønsker	

LATEST TEST OUTCOME						
Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Passed</a>	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

## Brukerhistorier for "Admin"

1. Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon.

Test suite: 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon. (Suite ID: 50) - Google Chrome  
about:blank

Test plan [43](#): Admin Nettbank

Suite hierarchy

Static suite: Admin Nettbank (ID: [44](#))

- Requirement-based suite: 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon. (ID: [50](#))
- Requirement-based suite: 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring. (ID: [51](#))
- Requirement-based suite: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få en konto hos banken. (ID: [52](#))
- Requirement-based suite: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken. (ID: [53](#))
- Requirement-based suite: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har alle tilgang til systemet. (ID: [54](#))

Test suite [50](#): 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon.

Test cases (1)

Test case [55](#): admin kan se alle kunder i banken

**PROPERTIES**

Test Case Id:	55
Assigned To:	Hiwa Abdolah
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	gå til localhost:8080 å trykk pålogg inn Admin	Åpne en vindu for admin loggen	
2	fyll inn brukernavn og passord (Admin,Admin)	flytte meg til nytt vindu der jeg logget inn som admin i systemet	
3	trykk pålogg inn	kan se på alle registrerte kunder i databasen	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Passed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring.

Test suite: 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring. (Suite ID: 51) - Google Chrome

about:blank

### Test plan [43](#): Admin Nettbank

#### Suite hierarchy

Static suite: Admin Nettbank (ID: [44](#))

- Requirement-based suite: 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon. (ID: [50](#))
- Requirement-based suite: 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring. (ID: [51](#))
- Requirement-based suite: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få et konto hos banken. (ID: [52](#))
- Requirement-based suite: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken. (ID: [53](#))
- Requirement-based suite: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har hele tilgang til systemet. (ID: [54](#))

Test suite [51](#): 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring.

#### Test cases (1)

Test case [56](#): oppdatere info om registrerte kunder

PROPERTIES			
Test Case Id:	56	Assigned To:	Hiwa Abdolah
State:	Design		
STEPS			
#	Action	Expected value	Attachments
1	gå til localhost:8080	vise nettsiden til banken	
2	trykk på logg inn Admin	vise logginn vindu	
3	fyll skjema for admin logg inn med bruke navn og passord (Admin, Admin)	logge inn brukeren som admin i systemet	
4	bytte navn fra Lene til Lene1 og trykk på Endre knappen	register nytt info i systemet/databasen	

LINKS						
ID	WorkItemType	Link type	Title			
<a href="#">57</a>	Bug	Tests	oppdatere info om registrerte kunder Failed (knappen fungerer ikke/har ikke noe response)			
<a href="#">46</a>	User Story	Tests	2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring.			

LATEST TEST OUTCOME						
Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
<a href="#">Failed</a>	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få et konto hos banken.

Test suite: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få en konto hos banken. (Suite ID: 52) - Google Chrome  
about:blank

### Test plan 43: Admin Nettbank

#### Suite hierarchy

Static suite: Admin Nettbank (ID: 44)

- Requirement-based suite: 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon. (ID: 50)
- Requirement-based suite: 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring. (ID: 51)
- Requirement-based suite: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få en konto hos banken. (ID: 52)
- Requirement-based suite: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken. (ID: 53)
- Requirement-based suite: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har alle tilgang til systemet. (ID: 54)

Test suite 52: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få en konto hos banken.

#### Test cases (1)

Test case 58: være mulig å registrere ny kunde i systemet

**PROPERTIES**

Test Case Id:	58
Assigned To:	Hiwa Abdolah
Status:	Design

**STEPS**

#	Action	Expected value	Attachments
1	gå til localhost:8080	vise banknetsiden	
2	trykk på logg inn som admin	vise logg inn vindu	
3	füll : bruker og passord (Admin, Admin)	logg inn brukeren som admin	
4	trykk på register ny kunde	få opp registering vindu	
5	füll ut skjemaet for ny kunde å trykk på register ny kunde	register ny kunde i databasen	

**LINKS**

ID	WorkitemType	Link type	Title
47	User Story	Tests	3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få en konto hos banken.
59	Bug	Tests	være mulig å registrere ny kunde i systemet Failed (Registerer ikke noe !)

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Failed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken.

Test suite: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken. (Suite ID: 53) - Google Chrome  
about:blank

Test plan 43: Admin Nettbank

Suite hierarchy

Static suite: Admin Nettbank (ID: 44)

Requirement-based suite: 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon. (ID: 50)

Requirement-based suite: 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring. (ID: 51)

Requirement-based suite: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få et konto hos banken. (ID: 52)

Requirement-based suite: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken. (ID: 53)

Requirement-based suite: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har tilgang til systemet. (ID: 54)

Test suite 53: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken.

Test cases (1)

Test case 60: være mulig å slette kunde fra systemet

**PROPERTIES**

Test Case Id:	60
Assigned To:	Hiwa Abdolah
State:	Design

**STEPS**

#	Action	Expected value	Attachments
1	gå til localhost:8080	vise bankens nettside	
2	trykk på logg inn som admin	vise logg inn skjema	
3	fyll skjemaet med brukernavn og passord Admin Admin trykk på logg inn	logge inn brukeren som admin	
4	trykk på endre kunde	vise Endre Kunde vindu	
5	trykk på slett på den kunden som jeg vil slette	vise spørre skjema om du er sikker om du skal slette denne kunden med OK eller Cancel knapper	
6	trykk på OK	slett Kunden fra systemet	

**ATTACHMENTS**

Name	Size	Date Attached	Comments
<a href="#">download.png</a>	11kb	Tue Mar 14 2023	

**LATEST TEST OUTCOME**

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Passed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

5. Som en administrator vil jeg kunne slette et type konto for en kunde fra Systemet, slik at kunden ikke lenger har den type kontoen i banken.

Test suite: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har alle tilgang til systemet. (Suite ID: 54) - Google Chrome  
about:blank

### Test plan 43: Admin Nettbank

#### Suite hierarchy

Static suite: Admin Nettbank (ID: 44)

- Requirement-based suite: 45 : Som en administrator vil jeg kunne hente alle kunder fra databasen, slik at jeg kan se alle kunder og deres informasjon. (ID: 50)
- Requirement-based suite: 46 : 2. Som en administrator vil jeg kunne endre informasjonen om en kunde i databasen, slik at jeg kan oppdatere informasjonen i tilfelle en endring. (ID: 51)
- Requirement-based suite: 47 : 3. Som en administrator vil jeg kunne registrere en ny kunde i databasen, slik at kunden kan få et konto hos banken. (ID: 52)
- Requirement-based suite: 48 : 4. Som en administrator vil jeg kunne slette en kunde fra databasen, slik at kunden ikke lenger er registrert i banken. (ID: 53)
- Requirement-based suite: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har alle tilgang til systemet. (ID: 54)

Test suite 54: 49 : 5. Som en administrator vil jeg kunne logge inn som «admin» slik at administratoren har alle tilgang til systemet.

#### Test cases (1)

Test case 61: Å ha rett til å slette kontoType for en KontoNr

PROPERTIES			
Test Case Id:	61	Assigned To:	Hiwa Abdolah
State:	Design		
<b>STEPS</b>	Action	Expected value	Attachments
1	gå til localhost:8080	vise nettsiden	
2	logginn som admin ved å trykke på logg inn admin skrive bruker navn og passord (Admin, Admin)	logg in brukeren som admin i systemet	
3	trykke på endre konto	få opp endre konto vindu	
4	bestemte å slette Lønnskonto for kontoNr: 105010123456 ved å trykke på slett knappen	spør om brukeren er sikkert å slette	
5	trykk på OK	slett kontoTypen fra Systemet	

LINKS			
ID	WorkItemType	Link type	Title
49	User Story	Tests	5. Som en administrator vil jeg kunne slette en type konto for en kunde fra Systemet, slik at kunden ikke lenger har den type kontoen i banken.

#### LATEST TEST OUTCOME

Outcome	Tester	Configuration	Run by	Date completed	Duration	Build number
Passed	Hiwa Abdolah	Windows 10	Hiwa Abdolah	14 March 2023	0:00:00.000	0

s364547 / Nettbank / Test Plans

Test Plans

Mine All + New Test Plan

Filter by title

Title	State	Area Path	Iteration Path	Assigned To	Test Plan ID
System test for Nettbank appen	Active	Nettbank	Nettbank	Hiwa Abdolah	25
Admin Nettbank	Active	Nettbank	Nettbank	Hiwa Abdolah	43

Hiwa Abdolah

s364547

```
1 package oslomet.testing;
2
3
4 import org.junit.Assert;
5 import org.junit.Test;
6 import org.junit.runner.RunWith;
7 import org.mockito.InjectMocks;
8 import org.mockito.Mock;
9 import org.mockito.junit.MockitoJUnitRunner;
10 import org.mockito.junit.jupiter.MockitoSettings;
11 import org.mockito.quality.Strictness;
12 import oslomet.testing.API.AdminKontoController;
13 import oslomet.testing.DAL.AdminRepository;
14 import oslomet.testing.Models.Konto;
15 import oslomet.testing.Sikkerhet.Sikkerhet;
16 import java.util.ArrayList;
17 import java.util.List;
18 import static org.junit.jupiter.api.Assertions.assertEquals;
19 import static org.mockito.Mockito.when;
20
21
22 @RunWith(MockitoJUnitRunner.class)
23 @MockitoSettings(strictness = Strictness.LENIENT)
24 public class EnhetstestAdminKontoController {
25
26     @InjectMocks
27     //denne skal testes
28     private AdminKontoController adminKontoController;
29
30     @Mock
31     //denne skal Mock-es
32     private AdminRepository adminRepository;
33
34     @Mock
35     //denne skal Mocke-s
36     private Sikkerhet sjekk;
37
38
39     @Test
40     public void testHentAlleKonto_OK() {
41
42
43         // Arrange
44         String personnummer = "12345678901"; // example personnummer
45         List<Konto> expected = new ArrayList<>();
46         expected.add(new Konto("123456789", "12345678901", 1000, "Brukskonto", "nok", null));
47         expected.add(new Konto("987654321", "986575231", 2000, "Brukskonto", "nok", null));
48         when(sjekk.loggetInn()).thenReturn(personnummer);
49         when(adminRepository.hentAlleKonti()).thenReturn(expected);
50
51         // Act
52         List<Konto> actual = adminKontoController.hentAlleKonti();
53
54         // Assert
55         Assert.assertEquals(expected, actual);
56     }
57
58     @Test
59     public void testHentAlleKontiScenario() {
60
61         // Arrange
62         when(sjekk.loggetInn()).thenReturn(null);
63
64         // Act
65         List<Konto> actual = adminKontoController.hentAlleKonti();
66
67         // Assert
68         Assert.assertNull(actual);
69     }
70 }
```

```
64    }
65
66    @Test
67    public void testRegistrerKonto() {
68        // Arrange
69        Konto konto = new Konto("123456789", "12345678901", 1000, "Brukskonto", "nok", null);
70        String expected = "Konto registrert";
71        when(sjekk.loggetInn()).thenReturn("12345678901");
72        when(adminRepository.registrerKonto(konto)).thenReturn(expected);
73
74        // Act
75        String actual = adminKontoController.registrerKonto(konto);
76
77        // Assert
78        Assert.assertEquals(expected, actual);
79    }
80
81    @Test
82    public void testRegistrerKontoNotLoggedIn() {
83        // Arrange
84        Konto konto = new Konto("123456789", "12345678901", 1000, "Brukskonto", "nok", null);
85        String expected = "Ikke innlogget";
86        when(sjekk.loggetInn()).thenReturn(null);
87
88        // Act
89        String actual = adminKontoController.registrerKonto(konto);
90
91        // Assert
92        Assert.assertEquals(expected, actual);
93    }
94
95
96    @Test
97    public void testEndreKontoLoggedIn() {
98        // Arrange
99        Konto konto = new Konto("123456789", "12345678901",
100            1000, "Brukskonto", "nok", null);
101        String expected = "Konto endret";
102        when(sjekk.loggetInn()).thenReturn("12345678901");
103        when(adminRepository.endreKonto(konto)).thenReturn(expected);
104
105        // Act
106        String actual = adminKontoController.endreKonto(konto);
107
108        // Assert
109        Assert.assertEquals(expected, actual);
110    }
111
112    @Test
113    public void testEndreKontoNotLoggedIn() {
114        // Arrange
115        Konto konto = new Konto("123456789", "12345678901",
116            1000, "Brukskonto", "nok", null);
117        String expected = "Ikke innlogget";
118        when(sjekk.loggetInn()).thenReturn(null);
119
120        // Act
121        String actual = adminKontoController.endreKonto(konto);
122
123        // Assert
124        Assert.assertEquals(expected, actual);
125    }
126
```

```
127
128     @Test
129     public void testSlettKonto_OK() {
130         // Arrange
131         String kontonummer = "12345678901";
132         String personnummer = "01010112345";
133         when(sjekk.loggetInn()).thenReturn(personnummer);
134         when(adminRepository.slettKonto(kontonummer)).thenReturn("Konto slettet");
135
136         // Act
137         String result = adminKontoController.slettKonto(kontonummer);
138
139         // Assert
140         assertEquals("Konto slettet", result);
141     }
142
143
144     @Test
145     public void testSlettKonto_IKKEinnlogget() {
146         // Arrange
147         String kontonummer = "12345678901";
148         when(sjekk.loggetInn()).thenReturn(null);
149
150         // Act
151         String result = adminKontoController.slettKonto(kontonummer);
152
153         // Assert
154         assertEquals("Ikke innlogget", result);
155     }
156
157 }
158
```

```
1 package oslomet.testing;
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.API.AdminKundeController;
9 import oslomet.testing.DAL.AdminRepository;
10 import oslomet.testing.Models.Kunde;
11 import oslomet.testing.Sikkerhet.Sikkerhet;
12 import java.util.ArrayList;
13 import java.util.List;
14 import static org.junit.jupiter.api.Assertions.assertEquals;
15 import static org.junit.jupiter.api.Assertions.assertNull;
16 import static org.mockito.Mockito.*;
17
18
19 @RunWith(MockitoJUnitRunner.class)
20 public class EnhetstestAdminKundeController {
21
22     @InjectMocks
23     // denne skal testes
24     private AdminKundeController adminKundeController;
25
26     @Mock
27     // denne skal Mock'es
28     private AdminRepository adminRepository;
29
30     @Mock
31     // denne skal Mock'es
32     private Sikkerhet sjekk;
33
34
35     @Test
36     public void hentAlle_loggetInn() {
37         // Arrange
38         List<Kunde> kunder = new ArrayList<>();
39         kunder.add(new Kunde("1234523543", "Ole", "Navn342", "Adresse1", "2020", "12345678", "", "Admin"));
40         kunder.add(new Kunde("1231354325", "Per", "Navn23", "Adresse1", "1111", "12345678", "", "Admin"));
41         when(sjekk.loggetInn()).thenReturn("Admin1");
42         when(adminRepository.hentAlleKunder()).thenReturn(kunder);
43
44         // Act
45         List<Kunde> result = adminKundeController.hentAlle();
46
47         // Assert
48         assertEquals(kunder, result);
49         verify(sjekk, times(1)).loggetInn();
50         verify(adminRepository, times(1)).hentAlleKunder();
51     }
52
53     @Test
54     public void hentAlle_ikkeLoggetInn() {
55         // Arrange
56         when(sjekk.loggetInn()).thenReturn(null);
57
58         // Act
59         List<Kunde> result = adminKundeController.hentAlle();
60
61         // Assert
```

```
62     assertNull(result);
63     verify(sjekk, times(1)).loggetInn();
64     verify(adminRepository, times(0)).hentAlleKunder();
65 }
66
67
68 @Test
69 public void lagreKunde_LoggetInn() {
70     // Arrange
71     Kunde kunde = new Kunde("01010112345", "Ola", "Nordmann", "Osloveien 82",
72         "0270", "Oslo", "12345678", "hemmelig");
73     when(sjekk.loggetInn()).thenReturn("01010112345");
74     when(adminRepository.registerKunde(kunde)).thenReturn("Kunde lagret");
75
76     // Act
77     String resultat = adminKundeController.lagreKunde(kunde);
78
79     // Assert
80     assertEquals("Kunde lagret", resultat);
81     verify(sjekk, times(1)).loggetInn();
82     verify(adminRepository, times(1)).registerKunde(kunde);
83 }
84
85 @Test
86 public void lagreKunde_IkkeLoggetInn() {
87     // Arrange
88     Kunde kunde = new Kunde("01010112345", "Ola", "Nordmann", "Osloveien 82",
89         "0270", "Oslo", "12345678", "hemmelig");
90     when(sjekk.loggetInn()).thenReturn(null);
91
92     // Act
93     String resultat = adminKundeController.lagreKunde(kunde);
94
95     // Assert
96     assertEquals("Ikke logget inn", resultat);
97     verify(sjekk, times(1)).loggetInn();
98     verify(adminRepository, never()).registerKunde(any(Kunde.class));
99 }
100
101
102 @Test
103 public void endre_loggetInn_returnererRiktigMelding() {
104     // Arrange
105     Kunde eksisterendeKunde = new Kunde("12345678901", "Ola", "Nordmann", "Oslo", "0123",
106         "Oslo", "12345678", "passord123");
107     Kunde oppdatertKunde = new Kunde("12345678901", "Per", "Hansen", "Bergen", "4567",
108         "Bergen", "87654321", "passord123");
109     when(sjekk.loggetInn()).thenReturn("Admin");
110     when(adminRepository.endreKundeInfo(oppdatertKunde)).thenReturn("Endring vellykket");
111
112     // Act
113     String resultat = adminKundeController.endre(oppdatertKunde);
114
115     // Assert
116     verify(sjekk).loggetInn();
117     verify(adminRepository).endreKundeInfo(oppdatertKunde);
118     assertEquals("Endring vellykket", resultat);
119 }
120
121 @Test
122 public void endre_ikkeLoggetInn_returnererRiktigMelding() {
123     // Arrange
124     Kunde oppdatertKunde = new Kunde("12345678901", "Per", "Hansen", "Bergen", "4567",
125         "Bergen", "87654321", "passord123");
126
127     // Act
128     String resultat = adminKundeController.endre(oppdatertKunde);
129
130     // Assert
131     verify(sjekk).loggetInn();
132     verify(adminRepository).endreKundeInfo(oppdatertKunde);
133     assertEquals("Endring vellykket", resultat);
134 }
```

```
122 "Bergen", "87654321", "password123");
123     when(sjekk.loggetInn()).thenReturn(null);
124
125     // Act
126     String resultat = adminKundeController.endre(oppdatertKunde);
127
128     // Assert
129     verify(sjekk).loggetInn();
130     verifyNoInteractions(adminRepository);
131     assertEquals("Ikke logget inn", resultat);
132 }
133
134 @Test
135 public void slettKunde_loggetInn_returnererSlettetMelding() {
136     // Arrange
137     String loggetInnPersonnummer = "01010112345";
138     String personnummer = "02020212345";
139     when(sjekk.loggetInn()).thenReturn(loggetInnPersonnummer);
140     when(adminRepository.slettKunde(personnummer)).thenReturn("Kunde med personnummer " +
personnummer + " er slettet");
141
142     // Act
143     String resultat = adminKundeController.slett(personnummer);
144
145     // Assert
146     assertEquals("Kunde med personnummer " + personnummer + " er slettet", resultat);
147     verify(sjekk).loggetInn();
148     verify(adminRepository).slettKunde(personnummer);
149 }
150
151 @Test
152 public void slettKunde_ikkeLoggetInn_returnererIkkeLoggetInnMelding() {
153     // Arrange
154     String personnummer = "02020212345";
155     when(sjekk.loggetInn()).thenReturn(null);
156
157     // Act
158     String resultat = adminKundeController.slett(personnummer);
159
160     // Assert
161     assertEquals("Ikke logget inn", resultat);
162     verify(sjekk).loggetInn();
163     verify(adminRepository, never()).slettKunde(personnummer);
164 }
165
166
167 }
168
```

```
1 package oslomet.testing;
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.API.BankController;
9 import oslomet.testing.DAL.BankRepository;
10 import oslomet.testing.Models.Konto;
11 import oslomet.testing.Models.Kunde;
12 import oslomet.testing.Models.Transaksjon;
13 import oslomet.testing.Sikkerhet.Sikkerhet;
14 import java.util.*;
15 import static org.junit.jupiter.api.Assertions.assertEquals;
16 import static org.junit.jupiter.api.Assertions.assertNull;
17 import static org.mockito.ArgumentMatchers.any;
18 import static org.mockito.ArgumentMatchers.anyString;
19 import static org.mockito.Mockito.*;
20
21
22 @RunWith(MockitoJUnitRunner.class)
23 public class EnhetstestBankController {
24
25     @InjectMocks
26     // denne skal testes
27     private BankController bankController;
28
29     @Mock
30     // denne skal Mock'es
31     private BankRepository repository;
32
33     @Mock
34     // denne skal Mock'es
35     private Sikkerhet sjekk;
36
37     @Test
38     public void hentKundeInfo_loggetInn() {
39
40         // arrange
41         Kunde enKunde = new Kunde("01010110523",
42             "Lene", "Jensen", "Askerveien 22", "3270",
43             "Asker", "22224444", "HeiHei");
44
45         when(sjekk.loggetInn()).thenReturn("01010110523");
46
47         when(repository.hentKundeInfo(anyString())).thenReturn(enKunde);
48
49         // act
50         Kunde resultat = bankController.hentKundeInfo();
51
52         // assert
53         assertEquals(enKunde, resultat);
54     }
55
56     @Test
57     public void hentKundeInfo_IkkeloggetInn() {
58
59         // arrange
60         when(sjekk.loggetInn()).thenReturn(null);
61
62         //act
63         Kunde resultat = bankController.hentKundeInfo();
```

```
64
65     // assert
66     assertNull(resultat);
67 }
68
69 @Test
70 public void hentKonti_LoggetInn() {
71     // arrange
72     List<Konto> konti = new ArrayList<>();
73     Konto konto1 = new Konto("105010123456", "01010110523",
74         720, "Lønnskonto", "NOK", null);
75     Konto konto2 = new Konto("105010123456", "12345678901",
76         1000, "Lønnskonto", "NOK", null);
77     konti.add(konto1);
78     konti.add(konto2);
79
80     when(sjekk.loggetInn()).thenReturn("01010110523");
81
82     when(repository.hentKonti(anyString())).thenReturn(konti);
83
84     // act
85     List<Konto> resultat = bankController.hentKonti();
86
87     // assert
88     assertEquals(konti, resultat);
89 }
90
91 @Test
92 public void hentKonti_IkkeLoggetInn() {
93     // arrange
94
95     when(sjekk.loggetInn()).thenReturn(null);
96
97     // act
98     List<Konto> resultat = bankController.hentKonti();
99
100    // assert
101    assertNull(resultat);
102 }
103
104
105 @Test
106 public void testHentTransaksjoner() {
107     // arrange
108     String kontoNr = "12345678901";
109     String fraDato = "2022-01-01";
110     String tilDato = "2022-01-31";
111     String personnummer = "01010110523";
112     Konto expectedKonto = new Konto(kontoNr, personnummer, 1000, "Lønnskonto", "NOK", null);
113 }
114
115     when(sjekk.loggetInn()).thenReturn(personnummer);
116     when(repository.hentTransaksjoner(kontoNr, fraDato, tilDato)).thenReturn(expectedKonto);
117
118     // act
119     Konto resultKonto = bankController.hentTransaksjoner(kontoNr, fraDato, tilDato);
120
121     // assert
122     assertEquals(expectedKonto, resultKonto);
123 }
124
125 @Test
```

```
125     public void testHentTransaksjoner_ikke() {
126         // arrange
127         String kontoNr = "12345678901";
128         String fraDato = "2022-01-01";
129         String tilDato = "2022-01-31";
130
131         when(sjekk.loggetInn()).thenReturn(null);
132
133         // act
134         Konto resultKonto = bankController.hentTransaksjoner(kontoNr, fraDato, tilDato);
135
136         // assert
137         assertNull(resultKonto);
138     }
139
140
141
142     @Test
143     public void hentSaldi_OK() {
144
145         // arrange
146         String personnummer = "01010110523";
147         List<Konto> konti = new ArrayList<>();
148         Konto konto1 = new Konto("105010123456", personnummer,
149             720, "Lønnskonto", "NOK", null);
150         Konto konto2 = new Konto("105010123456", personnummer,
151             1000, "Sparekonto", "NOK", null);
152         konti.add(konto1);
153         konti.add(konto2);
154
155         when(sjekk.loggetInn()).thenReturn(personnummer);
156         when(repository.hentSaldi(personnummer)).thenReturn(konti);
157
158         // act
159         List<Konto> resultat = bankController.hentSaldi();
160
161         // assert
162         assertEquals(konti, resultat);
163     }
164
165     @Test
166     public void hentSaldi_ikke() {
167
168         // arrange
169         when(sjekk.loggetInn()).thenReturn(null);
170
171         // act
172         List<Konto> resultat = bankController.hentSaldi();
173
174         // assert
175         assertNull(resultat);
176     }
177
178
179
180     @Test
181     public void registrerBetalning_OK() {
182         // arrange
183         Transaksjon enTransaksjon = new Transaksjon(1, "12345678910", 130.00, "2020-03-04", "Hei
Hei", "Avventer", "01010110523");
184
185         when(sjekk.loggetInn()).thenReturn("01010110523");
186         when(repository.registrerBetalning(any(Transaksjon.class))).thenReturn("Betalning
```

```
186 registrert");
187
188     // act
189     String resultat = bankController.registrerBetaling(enTransaksjon);
190
191     // assert
192     assertEquals("Betalning registrert", resultat);
193 }
194
195 @Test
196 public void registrerBetaling_Ikke() {
197     // arrange
198     Transaksjon enTransaksjon = new Transaksjon(1, "12345678910", 130.00, "2020-03-04", "Hei
199     Hei", "Avventer", "01010110523");
200
201     when(sjekk.loggetInn()).thenReturn(null);
202
203     // act
204     String resultat = bankController.registrerBetaling(enTransaksjon);
205
206     // assert
207     assertNull(resultat);
208 }
209
210
211 @Test
212 public void hentBetalinger_loggetInn() {
213     // arrange
214     String personnummer = "01010110523";
215     List<Transaksjon> betalinger = new ArrayList<>();
216     Transaksjon betaling1 = new Transaksjon(2, "01010110523",
217         500.00, "23-02-2023", "Mat", "avventer", "12345678901");
218     Transaksjon betaling2 = new Transaksjon(3, "01010110523",
219         1000.00, "22-02-2023", "Husleie", "avventer", "12345678901");
220     betalinger.add(betaling1);
221     betalinger.add(betaling2);
222
223     when(sjekk.loggetInn()).thenReturn(personnummer);
224     when(repository.hentBetalinger(personnummer)).thenReturn(betalinger);
225
226     // act
227     List<Transaksjon> resultat = bankController.hentBetalinger();
228
229     // assert
230     assertEquals(betalinger, resultat);
231 }
232
233 @Test
234 public void hentBetalinger_ikkeLoggetInn() {
235     // arrange
236     when(sjekk.loggetInn()).thenReturn(null);
237
238     // act
239     List<Transaksjon> resultat = bankController.hentBetalinger();
240
241     // assert
242     assertNull(resultat);
243 }
244
245
246 @Test
247 public void utførBetaling_loggetInn() {
```

```
248     // arrange
249     String personnummer = "01010110523";
250     int txID = 1;
251
252     when(sjekk.loggetInn()).thenReturn(personnummer);
253     when(repository.utforBetaling(txID)).thenReturn("OK");
254     List<Transaksjon> betalinger = new ArrayList<>();
255     when(repository.hentBetalinger(personnummer)).thenReturn(betalinger);
256
257     // act
258     List<Transaksjon> resultat = bankController.utforBetaling(txID);
259
260     // assert
261     verify(sjekk, times(1)).loggetInn();
262     verify(repository, times(1)).utforBetaling(txID);
263     verify(repository, times(1)).hentBetalinger(personnummer);
264     assertEquals(betalinger, resultat);
265 }
266
267 @Test
268 public void utførBetaling_ikkeLoggetInn() {
269     // arrange
270     int txID = 1;
271
272     when(sjekk.loggetInn()).thenReturn(null);
273
274     // act
275     List<Transaksjon> resultat = bankController.utforBetaling(txID);
276
277     // assert
278     verify(sjekk, times(1)).loggetInn();
279     verify(repository, never()).utforBetaling(anyInt());
280     verify(repository, never()).hentBetalinger(anyString());
281     assertNull(resultat);
282 }
283
284 @Test
285 public void utførBetaling_feilUtføring() {
286     // arrange
287     String personnummer = "01010110523";
288     int txID = 1;
289
290     when(sjekk.loggetInn()).thenReturn(personnummer);
291     when(repository.utforBetaling(txID)).thenReturn("Feil");
292
293     // act
294     List<Transaksjon> resultat = bankController.utforBetaling(txID);
295
296     // assert
297     verify(sjekk, times(1)).loggetInn();
298     verify(repository, times(1)).utforBetaling(txID);
299     verify(repository, never()).hentBetalinger(anyString());
300     assertNull(resultat);
301 }
302
303
304 @Test
305 public void testEndreKundeInfo() {
306     // Mock input data
307     String personnummer = "12345678901";
308     Kunde innKunde = new Kunde(personnummer, "Ola", "Nordmann", "Storgata 1",
309                               "3040", "Drammen", "123456987", "12345625");
310 }
```

```
311     // Mock session data
312     when(sjekk.loggetInn()).thenReturn(personnummer);
313
314     // Mock repository method call
315     when(repository.endreKundeInfo(innKunde)).thenReturn("OK");
316
317     // Call controller method and check output
318     String result = bankController.endre(innKunde);
319     assertEquals("OK", result);
320
321     // Verify that the repository method was called with the expected input
322     verify(repository).endreKundeInfo(innKunde);
323 }
324
325 }
326
327
328
```

```
1 package oslomet.testing;
2
3
4 import static org.junit.Assert.assertEquals;
5 import static org.mockito.Mockito.*;
6 import javax.servlet.http.HttpSession;
7 import org.junit.Test;
8 import org.junit.jupiter.api.BeforeEach;
9 import org.junit.runner.RunWith;
10 import org.mockito.InjectMocks;
11 import org.mockito.Mock;
12 import org.mockito.MockitoAnnotations;
13 import org.mockito.junit.MockitoJUnitRunner;
14 import org.springframework.test.util.ReflectionTestUtils;
15 import oslomet.testing.DAL.BankRepository;
16 import oslomet.testing.Sikkerhet.Sikkerhet;
17
18
19 @RunWith(MockitoJUnitRunner.class)
20 public class EnhetstestSikkerhet {
21
22     @InjectMocks
23     //denne skal testes :
24     private Sikkerhet sjekk;
25
26     @Mock
27     private HttpSession session;
28
29     @Mock
30     BankRepository bankRepository;
31
32
33     @BeforeEach
34     public void setUp() {
35         MockitoAnnotations.initMocks(this);
36         sjekk = new Sikkerhet();
37         ReflectionTestUtils.setField(sjekk, "rep", bankRepository);
38         ReflectionTestUtils.setField(sjekk, "session", session);
39     }
40
41
42
43
44     @Test
45     public void sjekkLoggInn_validInputs_returnOk() {
46         String personnummer = "12345678901";
47         String passord = "passord123";
48         String resultat = "OK";
49         when(bankRepository.sjekkLoggInn(personnummer, passord)).thenReturn(resultat);
50
51         String actual = sjekk.sjekkLoggInn(personnummer, passord);
52
53         assertEquals("OK", actual);
54         verify(session).setAttribute("Innlogget", personnummer);
55     }
56
57     @Test
58     public void sjekkLoggInn_invalidPersonnummer_returnFeilPersonnummer() {
59         String personnummer = "123";
60         String passord = "passord123";
61
62         String faktiske = sjekk.sjekkLoggInn(personnummer, passord);
63     }
}
```

```
64     assertEquals("Feil i personnummer", faktiske);
65     verifyNoInteractions(session);
66     verifyNoInteractions(bankRepository);
67 }
68
69 @Test
70 public void sjekkLoggInn_invalidPassord_returnFeilPassord() {
71     String personnummer = "12345678901";
72     String passord = "pwd";
73
74     String faktiske = sjekk.sjekkLoggInn(personnummer, passord);
75
76     assertEquals("Feil i passord", faktiske);
77     verifyNoInteractions(session);
78     verifyNoInteractions(bankRepository);
79 }
80
81 @Test
82 public void sjekkLoggInn_invalidCredentials_returnFeilPersonnummerEllerPassord() {
83     String personnummer = "12345678901";
84     String passord = "passord123";
85     String resultat = "Feil";
86     when(bankRepository.sjekkLoggInn(personnummer, passord)).thenReturn(resultat);
87
88     String faktiske = sjekk.sjekkLoggInn(personnummer, passord);
89
90     assertEquals("Feil i personnummer eller passord", faktiske);
91     verifyNoInteractions(session);
92     verify(bankRepository).sjekkLoggInn(personnummer, passord);
93 }
94
95
96
97 @Test
98 public void testLoggUt() {
99     HttpSession mockSession = mock(HttpSession.class);
100    Sikkerhet sjekk = new Sikkerhet();
101
102 // Set up session attribute "Innlogget" to simulate user being logged in
103    String personnummer = "12345678901";
104
105
106 // Call loggUt method, which should set "Innlogget" to null
107    sjekk.loggUt(mockSession);
108
109 // Verify that the session attribute "Innlogget" has been set to null
110    verify(mockSession).setAttribute("Innlogget", null);
111 }
112
113
114 @Test
115 public void testLoggInnAdmin_WithCorrectCredentials_ShouldReturnLoggetInn() {
116     // Arrange
117     String bruker = "Admin";
118     String passord = "Admin";
119
120     // Act
121     String actualResult = sjekk.loggInnAdmin(bruker, passord);
122
123     // Assert
124     assertEquals("Logget inn", actualResult);
125     verify(session).setAttribute("Innlogget", "Admin");
126     verify(session, never()).setAttribute("Innlogget", null);
127 }
```

```
127 }
128
129     @Test
130     public void testLoggInnAdmin_WithIncorrectCredentials_ShouldReturnIkkeLoggetInn() {
131         // Arrange
132         String bruker = "test";
133         String passord = "test";
134
135         // Act
136         String actualResult = sjekk.loggInnAdmin(bruker, passord);
137
138         // Assert
139         assertEquals("Ikke logget inn", actualResult);
140         verify(session).setAttribute("Innlogget", null);
141         verify(session, never()).setAttribute("Innlogget", "Admin");
142     }
143
144
145     @Test
146     public void testLoggetInn() {
147         // Arrange
148         HttpSession mockSession = mock(HttpSession.class);
149         when(mockSession.getAttribute("Innlogget")).thenReturn("test-user");
150         Sikkerhet sjekk = new Sikkerhet();
151         sjekk.session = mockSession;
152
153         // Act
154         String actual = sjekk.loggetInn();
155
156         // Assert
157         assertEquals("test-user", actual);
158     }
159
160     @Test
161     public void testIkkeLoggetInn() {
162         // Arrange
163         HttpSession mockSession = mock(HttpSession.class);
164         String expected = null;
165
166         // Set up mock session to simulate user not being logged in
167         //when(mockSession.getAttribute("Innlogget")).thenReturn(null);
168
169         // Act
170         String actual = sjekk.loggetInn();
171
172         // Assert
173         assertEquals(expected, actual);
174     }
175 }
```