# OBJ2100 – Kino Prosjektrapport

Campus Ringerike

Gruppe 21:

Kandidater:

7086

7106

7075

## Innholdsfortegnelse

## 1. Prosjektorganisering og gjennomføring

Gruppen besto av tre medlemmer som fordelte oppgavene basert på hovedkravene i eksamensoppgaven. Alle var involvert i koding og testing, og oppgaver ble løst i samarbeid både via fysiske møter og digitale plattformer.

- **7086 og 7075** jobbet hovedsakelig med kundedelen og billettsystemet.
- **7106** fokuserte på administrasjon og brukerkontroll.
  Utviklingen fulgte en iterativ tilnærming, hvor koden ble testet og forbedret fortløpende.

---

## 2. Systemstruktur og objektorientert design

Systemet er delt inn i flere pakker og klasser med tydelig ansvarsdeling:

- `model`: Domeneobjekter som Film, Visning, Plass, Login, Billett.
- `dao`: Databasetilgang, en klasse per tabell (f.eks. FilmDAO, LoginDAO).
- `service`: Forretningslogikk og tjenestelag.
- `util`: Hjelpeklasser for hash, database og billettkode.
- `app`: Inneholder menyer og applikasjonslogikk i konsollgrensesnittet.

---

## 3. Funksjonalitet og kravoppfyllelse

- **Krav 1**: Planlegger kan registrere filmer, opprette visninger og generere statistikk. Tilgangskontroll er basert på rolle.
- **Krav 2**: Betjent kan selge billetter direkte, ta imot betaling og slette ubetalte billetter – med logging.
- **Krav 3**: Kunde kan reservere plass, velge sete og motta en unik billettkode. Visninger under 30 minutter i fremtiden vises ikke.

- **Krav 4**: Brukeradministrasjon (opprett/slett/endring av PIN) er inkludert. Logging og rollestyring er implementert.

---

## 4. Datasikkerhet og personvern

- PIN-koder lagres trygt ved hjelp av **bcrypt** hashing.
- Kun rollebaserte brukere får tilgang til systemet.
- Kundedata er begrenset til billettkoder (dataminimering).
- Slettinger logges i filen `slettinger.dat`.

  Systemet følger prinsipper for personvern og sikker lagring.

---

## 5. Teknologier og kjøring

- **Språk**: Java
- **Database**: PostgreSQL
- **Eksterne biblioteker**: `bcrypt`, `PostgreSQL JDBC`
- **DAO-mønster** brukes for å strukturere datatilgangen.

**Oppsett:**

1. Kjør `Main.java` for å opprette brukere.
2. Bruk `HovedMeny.java` for å teste funksjonalitet.
3. Inkluder `.jar`-filene for `jdbc` og `bcrypt` i classpath.

---

## 6. Erfaringer og refleksjon

Vi møtte flere tekniske utfordringer med SQL, JDBC og innlogging. Disse ble løst ved hjelp av dokumentasjon, forelesningsmateriell og ChatGPT.
Det ble valgt å bruke konsollgrensesnitt for enkelhet. GUI ble vurdert, men valgt bort på grunn av tidsbegrensninger.

Prosjektet ga innsikt i sikkerhet, bruk av DAO og strukturering av Java-prosjekter for vedlikeholdbarhet.

---

## 7. Vedlegg: Mapper og struktur

- `model`: Film, Visning, Billett, Plass, Login
- `dao`: FilmDAO, LoginDAO, VisningDAO, m.fl.
- `service`: BestillingService, FilmService, StatistikkService
- `util`: HashUtil, Database, BillettKodeGenerator
- `app`: Main, HovedMeny, PlanleggerMeny, BetjentMeny

---

## 8. Tekniske valg og vurderinger

Systemet er bygget med et tydelig skille mellom modell, datatilgang og presentasjonslogikk. Dette gir god modularitet og gjør det enklere å feilsøke og videreutvikle applikasjonen. DAO-mønsteret gjør det lett å endre databasen uten å påvirke resten av systemet. PostgreSQL ble valgt som database for stabilitet og avanserte SQL-funksjoner.
Tabellene er normalisert for fremtidig utvidelse og støtte for flere brukertyper.

---

## 9. Mulige forbedringer og videreutvikling

Videre utvikling kan inkludere:

- Grafisk brukergrensesnitt (JavaFX)
- E-postbekreftelse ved reservasjoner
- Integrasjon av betalingsløsning
- Bedre validering og feilhåndtering
- Logging med dato og tidspunkt
- Innlogging for kunder med historikk
- Automatisk rapportgenerering

- API-lag for tredjeparts integrasjon