Implementation Plan: Nye Hædda Barneskole PM Simulator

Development Sprint: December 9-15, 2025

Document Version: 1.0 **Date:** December 9, 2025 **Target Completion:** December 15, 2025 (23:59) **Realistic Target:** December 14, 2025 (EOD) with December 16 for polish **Team:** SG-Gruppe-14-d2

Executive Summary

This document provides a day-by-day implementation plan to complete the Nye Hædda Barneskole Project Management Simulator from the current state to a functional MVP. Based on analysis of your existing codebase and comprehensive documentation (PRD, Epics, UX Design), this plan focuses on delivering the **Must-Have** features within a 7-day sprint.

Current State Assessment (December 9)

✅ **Completed:**

- ✅ Phase 0: Discovery & Analysis (100%)
- ✅ Phase 1: Planning (PRD, Epics, UX Design) (100%)
- ✅ Phase 2: Solutioning (Architecture, Test Design) (100%)
- ✅ Basic React frontend structure (routing, components)
- ✅ Basic FastAPI backend (CORS, PDF parsing, simple negotiation)

❌ **Not Started (Critical Path):**

- ❌ Supabase Authentication
- ❌ Gemini AI Integration
- ❌ Dashboard UI (budget tracking, constraints)
- ❌ localStorage session management
- ❌ Commitment/renegotiation flow
- ❌ Plan validation logic
- ❌ Visualization features (Gantt, Precedence, History)
- ❌ Static data files (wbs.json, suppliers.json)

Estimated Work Remaining: 116 story points (originally 4-5 weeks) → **Compressed to 7 days**

Scope Adjustment Recommendation

CRITICAL DECISION REQUIRED: To realistically complete by December 15, we recommend the following scope adjustments:

✅ **MUST HAVE (Core MVP):**

1. Supabase Authentication (E1)
2. Dashboard with budget tracking (E2)
3. WBS list and selection (E3)
4. AI Negotiation with Gemini (E4) - **SIMPLIFIED PROMPTS**
5. Commitment flow (E5)
6. Basic plan validation (E6)
7. Export session (E7)
8. Infrastructure (E9)

⚠️ **DEFER TO POST-MVP (December 16+):**

- Gantt Chart (E10.1) - 8 story points
- Precedence Diagram (E10.2) - 8 story points
- History/Timeline (E10.3) - 8 story points
- Advanced help/onboarding (E8.2)

Justification: Visualization features (Epic 10) are valuable but not core to the learning objectives. Students can complete the simulation without them. Focus on delivering a **working negotiation experience first**.

Day-by-Day Implementation Plan

Monday, December 9 (Day 1): Foundation & Setup

Goal: Infrastructure ready, authentication working, static data prepared

Morning (4 hours): Account Creation & Setup

Task 1.1: Supabase Account & Project Setup (1 hour)

Step-by-Step Instructions:

1. **Create Supabase Account**
    - Navigate to: https://supabase.com
    - Click "Start your project"
    - Sign up with GitHub account (recommended for team collaboration)
    - Verify email
2. **Create New Project**
    - Click "New Project"
    - Organization: Create new organization "SG-Gruppe-14"
    - Project Name: `nye-haedda-pm-simulator`
    - Database Password: **SAVE THIS SECURELY** (use a password manager)
    - Region: Europe (Frankfurt or Oslo)
    - Pricing Plan: Free tier (sufficient for 50 concurrent users)
    - Click "Create new project"
    - Wait 2-3 minutes for provisioning
3. **Configure Authentication**
    - In Supabase Dashboard, navigate to: Authentication → Providers
    - Email Provider: **ENABLED** (should be on by default)
    - Confirm Email: **DISABLED** (for MVP speed - enable later for production)
    - Site URL: `http://localhost:3000` (development)
    - Redirect URLs: Add `http://localhost:3000/dashboard`

4. **Get API Credentials**
    - Navigate to: Settings → API
    - Copy the following values:

```
VITE_SUPABASE_URL=https://[your-project-id].supabase.co
VITE_SUPABASE_ANON_KEY=[your-anon-key]
```

    - Create `frontend/.env` file:

```
VITE_SUPABASE_URL=https://[your-project-id].supabase.co
VITE_SUPABASE_ANON_KEY=[your-anon-key]
```

5. **Test Connection**
    - Install Supabase client: `npm install @supabase/supabase-js` (in frontend/)
    - Create test file `frontend/src/supabaseClient.js`:

```
import { createClient } from '@supabase/supabase-js'

const supabaseUrl = import.meta.env.VITE_SUPABASE_URL
const supabaseAnonKey = import.meta.env.VITE_SUPABASE_ANON_KEY

export const supabase = createClient(supabaseUrl, supabaseAnonKey)
```

    - Test in browser console: `import { supabase } from './supabaseClient'; console.log(supabase)`

⚠️ **SECURITY NOTE:** The ANON_KEY is safe to expose in frontend code (it's a public API key with Row Level Security). Never commit the SERVICE_ROLE_KEY to Git.

Task 1.2: Gemini API Account & Setup (1 hour)

Step-by-Step Instructions:

1. **Create Google AI Studio Account**
    - Navigate to: https://aistudio.google.com

- Sign in with Google account
- Accept Terms of Service

2. **Generate API Key**

   - Click "Get API Key" in top-right
   - Click "Create API Key"
   - Select "Create API key in new project" (or use existing Google Cloud project)
   - Project Name: `nye-haedda-ai`
   - Click "Create"
   - **COPY AND SAVE API KEY IMMEDIATELY** (only shown once)
   - Example format: `AIzaSyD...` (40 characters)

3. **Test API Key**

   - Open browser console and test with curl:

```
curl -X POST \
  'https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-
flash-exp:generateContent?key=YOUR_API_KEY' \
  -H 'Content-Type: application/json' \
  -d '{
    "contents": [{
      "parts": [{
        "text": "Hei, jeg heter Bjørn Eriksen. Jeg er totalentreprenør."
      }]
    }]
  }'
```

   - Expected response: JSON with `candidates[0].content.parts[0].text`

4. **Configure Backend Environment**

   - Create `backend/.env` file:

```
GEMINI_API_KEY=AIzaSyD...your-key-here...
SUPABASE_URL=https://[your-project-id].supabase.co
SUPABASE_ANON_KEY=[your-anon-key]
```

   - **IMPORTANT:** Add `.env` to `.gitignore` (should already be there)

5. **Install Gemini SDK**

- Backend: `pip install google-generativeai` (add to requirements.txt)
- Test in Python:

```
import google.generativeai as genai
genai.configure(api_key="YOUR_API_KEY")
model = genai.GenerativeModel('gemini-2.0-flash-exp')
response = model.generate_content("Hei fra Norge!")
print(response.text)
```

Rate Limits (Free Tier):

- 15 requests per minute
- 1,500 requests per day
- **Sufficient for MVP testing** (10-20 students × 100 negotiations = ~2,000 requests)

Task 1.3: Prepare Static Data Files (1 hour)

Goal: Create `wbs.json` and `suppliers.json` from PRD specifications

1. **Create `frontend/public/data/wbs.json`**

   Extract 15 WBS items from the project proposal PDF. Based on PRD Section 8, create:

```
[
  {
    "code": "1.1",
    "name": "Prosjektering",
    "description": "Arkitekttjenester, teknisk prosjektering,
detaljplanlegging",
    "baseline_cost": 50,
    "baseline_duration": 2,
    "dependencies": [],
    "requirements": ["F-001", "F-002"],
    "category": "Planlegging"
  },
  {
    "code": "1.3.1",
    "name": "Grunnarbeid",
    "description": "Masseutskifting, grunnmur, drenering, fundamenter",
    "baseline_cost": 100,
    "baseline_duration": 2,
    "dependencies": ["1.1"],
    "requirements": ["F-003", "K-023"],
    "category": "Konstruksjon"
```

```
  },
  {
    "code": "2.1",
    "name": "Råbygg",
    "description": "Betongarbeider, muring, takkonstruksjon",
    "baseline_cost": 200,
    "baseline_duration": 4,
    "dependencies": ["1.3.1"],
    "requirements": ["F-004", "K-024"],
    "category": "Konstruksjon"
  },
  {
    "code": "2.2",
    "name": "Vinduer og Dører",
    "description": "Montering av vinduer, dører, glass",
    "baseline_cost": 60,
    "baseline_duration": 1.5,
    "dependencies": ["2.1"],
    "requirements": ["F-005"],
    "category": "Innredning"
  },
  {
    "code": "3.1",
    "name": "VVS-installasjon",
    "description": "Rørleggerarbeider, sanitæranlegg, varmeanlegg",
    "baseline_cost": 80,
    "baseline_duration": 2.5,
    "dependencies": ["2.1"],
    "requirements": ["F-006", "K-025"],
    "category": "Teknisk"
  },
  {
    "code": "3.2",
    "name": "Elektrisk installasjon",
    "description": "Elektriske anlegg, belysning, styringssystem",
    "baseline_cost": 70,
    "baseline_duration": 2,
    "dependencies": ["2.1"],
    "requirements": ["F-007", "K-026"],
    "category": "Teknisk"
  },
  {
    "code": "3.3",
    "name": "Ventilasjon",
    "description": "Ventilasjonsanlegg, luftbehandling",
    "baseline_cost": 50,
    "baseline_duration": 1.5,
    "dependencies": ["2.1"],
    "requirements": ["F-008"],
    "category": "Teknisk"
  },
  {
    "code": "3.4",
    "name": "Maling og Overflatebehandling",
    "description": "Malearbeider innvendig og utvendig",
    "baseline_cost": 40,
    "baseline_duration": 2,
```

```
    "dependencies": ["3.1", "3.2"],
    "requirements": ["F-009"],
    "category": "Innredning"
  },
  {
    "code": "4.1",
    "name": "Gulvarbeider",
    "description": "Gulvbelegg, fliser, parkett",
    "baseline_cost": 45,
    "baseline_duration": 1.5,
    "dependencies": ["3.4"],
    "requirements": ["F-010"],
    "category": "Innredning"
  },
  {
    "code": "4.2",
    "name": "Innvendige Vegger",
    "description": "Gipsplater, skillevegger, akustikk",
    "baseline_cost": 55,
    "baseline_duration": 2,
    "dependencies": ["3.4"],
    "requirements": ["F-011"],
    "category": "Innredning"
  },
  {
    "code": "4.3",
    "name": "Fast Inventar",
    "description": "Kjøkkeninnredning, garderober, hyller",
    "baseline_cost": 35,
    "baseline_duration": 1,
    "dependencies": ["4.2"],
    "requirements": ["F-012"],
    "category": "Innredning"
  },
  {
    "code": "5.1",
    "name": "Uteområder",
    "description": "Asfalt, lekeplasser, grøntanlegg",
    "baseline_cost": 50,
    "baseline_duration": 2,
    "dependencies": ["2.1"],
    "requirements": ["F-013"],
    "category": "Utomhus"
  },
  {
    "code": "5.2",
    "name": "Adkomst og Parkering",
    "description": "Vei, parkering, HC-tilgjengelighet",
    "baseline_cost": 40,
    "baseline_duration": 1.5,
    "dependencies": ["5.1"],
    "requirements": ["F-014"],
    "category": "Utomhus"
  },
  {
    "code": "6.1",
    "name": "Brannalarm og Sikkerhet",
```

```
    "description": "Brannalarmanlegg, sikkerhetsinstallasjoner",
    "baseline_cost": 30,
    "baseline_duration": 1,
    "dependencies": ["3.2"],
    "requirements": ["K-027"],
    "category": "Teknisk"
  },
  {
    "code": "6.2",
    "name": "Sluttbefaring og Overtakelse",
    "description": "Dokumentasjon, FDV, innflytting",
    "baseline_cost": 20,
    "baseline_duration": 0.5,
    "dependencies": ["4.3", "5.2", "6.1"],
    "requirements": ["F-015"],
    "category": "Avslutning"
  }
]
```

Total Baseline: 925 MNOK (allows negotiation down to ~680-700 MNOK)

2. **Create** `frontend/public/data/suppliers.json`

```
[
  {
    "id": "bjorn",
    "name": "Bjørn Eriksen",
    "company": "Eriksen Bygg AS",
    "role": "Totalentreprenør",
    "personality": "Pragmatisk og erfaren, fokusert på kvalitet og sikkerhet.
Skeptisk til for stramme budsjetter.",
    "specialties": ["Grunnarbeid", "Råbygg", "Prosjektledelse"],
    "initial_margin": 1.20,
    "concession_rate": 0.05,
    "patience": 3
  },
  {
    "id": "kari",
    "name": "Kari Andersen",
    "company": "Andersen Rør og Varme",
    "role": "Rørlegger",
    "personality": "Detaljorientert og teknisk dyktig. Setter pris på
presisjon i spesifikasjoner.",
    "specialties": ["VVS-installasjon", "Sanitæranlegg"],
    "initial_margin": 1.18,
    "concession_rate": 0.03,
    "patience": 4
  },
  {
    "id": "per",
    "name": "Per Johansen",
    "company": "Johansen Elektro",
    "role": "Elektriker",
    "personality": "Direkte og effektiv. Liker å jobbe raskt og forventer
```

```
klare krav.",
    "specialties": ["Elektrisk installasjon", "Brannalarm"],
    "initial_margin": 1.25,
    "concession_rate": 0.07,
    "patience": 2
  },
  {
    "id": "silje",
    "name": "Silje Henriksen",
    "company": "Henriksen Arkitekter",
    "role": "Arkitekt",
    "personality": "Kreativ og visjonær. Forsvarer designintegritet, men åpen
for konstruktiv dialog.",
    "specialties": ["Prosjektering", "Estetikk"],
    "initial_margin": 1.15,
    "concession_rate": 0.04,
    "patience": 5
  },
  {
    "id": "tor",
    "name": "Tor Kristoffersen",
    "company": "Kristoffersen Malermester",
    "role": "Maler",
    "personality": "Vennlig og fleksibel. Tilpasningsdyktig, men krever
respekt for fagkompetanse.",
    "specialties": ["Maling", "Overflatebehandling"],
    "initial_margin": 1.22,
    "concession_rate": 0.06,
    "patience": 3
  }
]
```

3. **Validation**
   - Total baseline costs: Sum all `baseline_cost` values = 925 MNOK ✓
   - After negotiation (avg -15%): ~786 MNOK (still over 700 - requires strategic choices)
   - Dependencies: Ensure no circular dependencies (use topological sort later)

Task 1.4: Implement Supabase Authentication (1 hour)

Files to Create/Modify:

1. `frontend/src/pages/Login.jsx` (new file)

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { supabase } from '../supabaseClient';
```

```
function Login() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [isRegistering, setIsRegistering] = useState(false);
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  const handleAuth = async (e) => {
    e.preventDefault();
    setLoading(true);
    setError('');

    try {
      if (isRegistering) {
        const { data, error } = await supabase.auth.signUp({
          email,
          password,
        });
        if (error) throw error;
        alert('Konto opprettet! Logger inn...');
      }

      const { data, error } = await supabase.auth.signInWithPassword({
        email,
        password,
      });

      if (error) throw error;
      navigate('/dashboard');
    } catch (error) {
      setError(error.message || 'Feil e-post eller passord');
    } finally {
      setLoading(false);
    }
  };

  return (
    <div className="login-container">
      <div className="login-card">
        <h1>Nye Hædda Barneskole</h1>
        <p>Prosjektledelsessimulator</p>

        <form onSubmit={handleAuth}>
          <label>E-post</label>
          <input
            type="email"
            value={email}
            onChange={(e) => setEmail(e.target.value)}
            required
          />

          <label>Passord</label>
          <input
            type="password"
            value={password}
            onChange={(e) => setPassword(e.target.value)}
```

```
            required
            minLength={6}
          />

          {error && <p className="error">{error}</p>}

          <button type="submit" disabled={loading}>
            {loading ? 'Behandler...' : (isRegistering ? 'Registrer' : 'Logg
Inn')}
          </button>
        </form>

        <p>
          {isRegistering ? 'Har du allerede konto?' : 'Har du ikke konto?'}
          <button onClick={() => setIsRegistering(!isRegistering)}>
            {isRegistering ? 'Logg Inn' : 'Registrer deg'}
          </button>
        </p>
      </div>
    </div>
  );
}

export default Login;
```

2. **Update** `frontend/src/App.js` - Add protected routes:

```
import { useEffect, useState } from 'react';
import { Routes, Route, Navigate } from 'react-router-dom';
import { supabase } from './supabaseClient';
import Login from './pages/Login';
import Dashboard from './pages/Dashboard';

function App() {
  const [session, setSession] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    supabase.auth.getSession().then(({ data: { session } }) => {
      setSession(session);
      setLoading(false);
    });

    const { data: { subscription } } =
supabase.auth.onAuthStateChange((_event, session) => {
      setSession(session);
    });

    return () => subscription.unsubscribe();
  }, []);

  if (loading) return <div>Laster...</div>;

  return (
```

```
      <Routes>
        <Route path="/" element={session ? <Navigate to="/dashboard" /> : <Login
  />} />
        <Route
          path="/dashboard"
          element={session ? <Dashboard session={session} /> : <Navigate to="/"
  />}
        />
      </Routes>
    );
  }

  export default App;
```

Test: After implementation, test registration and login flow. Verify JWT stored in browser DevTools → Application → Local Storage.

Afternoon (4 hours): Dashboard Foundation

Task 1.5: Implement localStorage Session Management (2 hours)

Create `frontend/src/utils/sessionManager.js`:

```
// Session schema based on PRD Section 8
export const initializeSession = (userId, wbsItems, suppliers) => {
  const sessionId = `session_${userId}_${Date.now()}`;
  const session = {
    game_id: sessionId,
    user_id: userId,
    created_at: new Date().toISOString(),
    status: 'in_progress',
    wbs_items: wbsItems,
    suppliers: suppliers,
    current_plan: {},
    plan_history: [],
    chat_logs: {},
    metrics: {
      total_budget_used: 0,
      projected_end_date: null,
      negotiation_count: 0,
      renegotiation_count: 0,
    },
  };

  localStorage.setItem(`nye_haedda_session_${userId}`, JSON.stringify(session));
  return session;
};

export const loadSession = (userId) => {
  const saved = localStorage.getItem(`nye_haedda_session_${userId}`);
  return saved ? JSON.parse(saved) : null;
```

```
};

export const saveSession = (session) => {
  localStorage.setItem(`nye_haedda_session_${session.user_id}`,
JSON.stringify(session));
};

export const commitQuote = (session, wbsCode, supplierId, cost, duration) => {
  // Calculate start_date based on dependencies
  const wbsItem = session.wbs_items.find((item) => item.code === wbsCode);
  let startDate = new Date('2025-01-15'); // Default project start

  if (wbsItem.dependencies.length > 0) {
    // Find latest end_date from dependencies
    const depEndDates = wbsItem.dependencies.map((depCode) => {
      const depPlan = session.current_plan[depCode];
      return depPlan ? new Date(depPlan.end_date) : startDate;
    });
    startDate = new Date(Math.max(...depEndDates));
    startDate.setDate(startDate.getDate() + 1); // Start next day
  }

  const endDate = new Date(startDate);
  endDate.setMonth(endDate.getMonth() + duration);

  session.current_plan[wbsCode] = {
    supplier_id: supplierId,
    cost,
    duration,
    start_date: startDate.toISOString().split('T')[0],
    end_date: endDate.toISOString().split('T')[0],
  };

  session.plan_history.push({
    timestamp: new Date().toISOString(),
    action: 'commit',
    wbs_code: wbsCode,
    supplier_id: supplierId,
    cost,
    duration,
  });

  session.metrics.total_budget_used = Object.values(session.current_plan).reduce(
    (sum, item) => sum + item.cost,
    0
  );

  // Calculate projected end date (max end_date from all committed items)
  const allEndDates = Object.values(session.current_plan).map((item) => new
Date(item.end_date));
  session.metrics.projected_end_date = new Date(Math.max(...allEndDates))
    .toISOString()
    .split('T')[0];

  saveSession(session);
  return session;
};
```

```
export const removeQuote = (session, wbsCode) => {
  const removedItem = session.current_plan[wbsCode];
  delete session.current_plan[wbsCode];

  session.plan_history.push({
    timestamp: new Date().toISOString(),
    action: 'remove',
    wbs_code: wbsCode,
    reason: 'renegotiation',
  });

  session.metrics.total_budget_used = Object.values(session.current_plan).reduce(
    (sum, item) => sum + item.cost,
    0
  );
  session.metrics.renegotiation_count += 1;

  saveSession(session);
  return session;
};

export const validatePlan = (session) => {
  const totalCost = session.metrics.total_budget_used;
  const budgetLimit = 700; // MNOK
  const deadline = new Date('2026-05-15');
  const projectedDate = new Date(session.metrics.projected_end_date);

  const errors = [];

  if (totalCost > budgetLimit) {
    errors.push({
      type: 'budget',
      message: `Budsjett overskredet med ${totalCost - budgetLimit} MNOK (Total:
${totalCost}, Grense: ${budgetLimit})`,
      overage: totalCost - budgetLimit,
    });
  }

  if (projectedDate > deadline) {
    const daysLate = Math.floor((projectedDate - deadline) / (1000 * 60 * 60 *
24));
    errors.push({
      type: 'timeline',
      message: `Prosjektet forsinket til ${projectedDate.toLocaleDateString('no-
NO')} (Frist: ${deadline.toLocaleDateString('no-NO')})`,
      daysLate,
    });
  }

  const completedCount = Object.keys(session.current_plan).length;
  if (completedCount < 15) {
    errors.push({
      type: 'incomplete',
      message: `Kun ${completedCount} / 15 oppgaver fullført`,
    });
  }
```

```
  return {
    isValid: errors.length === 0,
    errors,
  };
};
```

Task 1.6: Build Dashboard UI (2 hours)

Create `frontend/src/pages/Dashboard.jsx`:

```
import React, { useEffect, useState } from 'react';
import { initializeSession, loadSession, saveSession } from
'../utils/sessionManager';
import { supabase } from '../supabaseClient';
import ConstraintPanel from '../components/ConstraintPanel';
import WBSList from '../components/WBSList';

function Dashboard({ session: authSession }) {
  const [gameSession, setGameSession] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const loadGameSession = async () => {
      const userId = authSession.user.id;
      let session = loadSession(userId);

      if (!session) {
        // Initialize new session
        const wbsResponse = await fetch('/data/wbs.json');
        const wbsItems = await wbsResponse.json();

        const suppliersResponse = await fetch('/data/suppliers.json');
        const suppliers = await suppliersResponse.json();

        session = initializeSession(userId, wbsItems, suppliers);
      }

      setGameSession(session);
      setLoading(false);
    };

    loadGameSession();
  }, [authSession]);

  const handleLogout = async () => {
    await supabase.auth.signOut();
  };

  if (loading) return <div>Laster økt...</div>;

  return (
```

```
    <div className="dashboard">
      <header className="dashboard-header">
        <h1>Prosjektledelsessimulator</h1>
        <button onClick={handleLogout}>Logg Ut</button>
      </header>

      <ConstraintPanel
        budgetUsed={gameSession.metrics.total_budget_used}
        budgetLimit={700}
        projectedEndDate={gameSession.metrics.projected_end_date}
        deadline="2026-05-15"
      />

      <div className="quick-stats">
        Fremdrift: {Object.keys(gameSession.current_plan).length} / 15 WBS-oppgaver
fullført |{' '}
        {gameSession.metrics.negotiation_count} forhandlinger
      </div>

      <WBSList
        wbsItems={gameSession.wbs_items}
        currentPlan={gameSession.current_plan}
        onSelectWBS={(wbsCode) => {
          // Navigate to supplier selection
          console.log('Selected WBS:', wbsCode);
        }}
      />
    </div>
  );
}

export default Dashboard;
```

Create `frontend/src/components/ConstraintPanel.jsx`:

```
import React from 'react';

function ConstraintPanel({ budgetUsed, budgetLimit, projectedEndDate, deadline }) {
  const percentage = Math.round((budgetUsed / budgetLimit) * 100);
  const progressColor = percentage > 100 ? 'red' : percentage > 97 ? 'yellow' :
'green';

  const projDate = projectedEndDate ? new Date(projectedEndDate) : null;
  const deadlineDate = new Date(deadline);
  const onTime = projDate && projDate <= deadlineDate;

  return (
    <div className="constraint-panel">
      <div className="budget-section">
        <p>Budsjett: {budgetUsed} / {budgetLimit} MNOK ({percentage}%)</p>
        <div className="progress-bar">
          <div
            className={`progress-fill ${progressColor}`}
            style={{ width: `${Math.min(percentage, 100)}%` }}
```

```
          />
        </div>
      </div>

      <div className="timeline-section">
        <p>Frist: {deadlineDate.toLocaleDateString('no-NO')}</p>
        {projDate && (
          <p>
            Forventet: {projDate.toLocaleDateString('no-NO')} {onTime ? '✓' : '✗'}
          </p>
        )}
      </div>
    </div>
  );
}

export default ConstraintPanel;
```

Create `frontend/src/components/WBSList.jsx`:

```
import React from 'react';

function WBSList({ wbsItems, currentPlan, onSelectWBS }) {
  return (
    <div className="wbs-list">
      <h2>Arbeidsnedbrytningsstruktur (WBS)</h2>
      {wbsItems.map((item) => {
        const isCompleted = currentPlan[item.code];
        return (
          <div key={item.code} className="wbs-item">
            <span className="status-icon">{isCompleted ? '🟢' : '🟣'}</span>
            <div className="wbs-content">
              <h4>{item.code} - {item.name}</h4>
              {isCompleted ? (
                <div className="wbs-details">
                  <p>{currentPlan[item.code].cost} MNOK,
{currentPlan[item.code].duration} måneder</p>
                  <button onClick={() =>
onSelectWBS(item.code)}>Reforhandle</button>
                </div>
              ) : (
                <div>
                  <p>Grunnlag: {item.baseline_cost} MNOK, {item.baseline_duration}
måneder</p>
                  <button onClick={() => onSelectWBS(item.code)}>Kontakt
Leverandør</button>
                </div>
              )}
            </div>
          </div>
        );
      })}
    </div>
  );
```

```
}

export default WBSList;
```

End of Day 1 Deliverables:

- ✅ Supabase authentication working
- ✅ Gemini API account created and tested
- ✅ Static data files (wbs.json, suppliers.json)
- ✅ localStorage session management
- ✅ Basic Dashboard UI with budget tracking
- ✅ WBS list display

Testing Checklist:

- ☐ Register new user → Redirects to Dashboard
- ☐ Dashboard loads with 0/700 MNOK budget
- ☐ WBS list shows 15 items with 🟣 pending status
- ☐ localStorage contains session object

Tuesday, December 10 (Day 2): AI Integration & Chat

Goal: Gemini AI chat working with supplier personas

Task 2.1: Implement Gemini AI Backend (3 hours)

Update `backend/app/main.py`:

```
from fastapi import FastAPI, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
import google.generativeai as genai
import os
from dotenv import load_dotenv
import json

load_dotenv()

genai.configure(api_key=os.getenv("GEMINI_API_KEY"))
```

```
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Load supplier personas
with open('../frontend/public/data/suppliers.json', 'r', encoding='utf-8') as f:
    SUPPLIERS = {s['id']: s for s in json.load(f)}

# Load WBS data
with open('../frontend/public/data/wbs.json', 'r', encoding='utf-8') as f:
    WBS_ITEMS = {w['code']: w for w in json.load(f)}

class ChatRequest(BaseModel):
    wbs_code: str
    supplier_id: str
    message: str
    chat_history: list = []

@app.get("/")
async def root():
    return {"message": "Nye Hædda PM Simulator API"}

@app.post("/api/chat")
async def chat(request: ChatRequest):
    try:
        supplier = SUPPLIERS.get(request.supplier_id)
        wbs_item = WBS_ITEMS.get(request.wbs_code)

        if not supplier or not wbs_item:
            raise HTTPException(status_code=400, detail="Invalid supplier or WBS
code")

        # Build system prompt
        system_prompt = f"""
Du er {supplier['name']}, {supplier['role']} fra {supplier['company']}.

PERSONLIGHET:
{supplier['personality']}

DIN OPPGAVE:
Du forhandler om WBS-oppgave {wbs_item['code']} - {wbs_item['name']}.
Beskrivelse: {wbs_item['description']}
Grunnlagskostnad: {wbs_item['baseline_cost']} MNOK
Grunnlagsvarighet: {wbs_item['baseline_duration']} måneder

DINE PARAMETRE (HEMMELIG - ikke fortell brukeren):
- initial_margin: {supplier['initial_margin']} (multipliser grunnlagskostnad)
- concession_rate: {supplier['concession_rate']} (hvor mye du kan gå ned per runde)
- patience: {supplier['patience']} (antall forhandlingsrunder før du går bort)

FORHANDLINGSREGLER:
```

```
1. Første tilbud: {wbs_item['baseline_cost']} * {supplier['initial_margin']} =
{wbs_item['baseline_cost'] * supplier['initial_margin']} MNOK
2. Hvis bruker forhandler: Reduser prisen med {supplier['concession_rate'] * 100}%
per runde
3. Hvis bruker er urimelig: Advar etter {supplier['patience']} runder, deretter gå
bort
4. Alltid svar på norsk, bruk konstruksjonsterminologi
5. Referer til tekniske krav (F-koder, K-koder) når relevant

FORMAT FOR TILBUD:
"Basert på [begrunnelse], kan jeg tilby [X] MNOK og [Y] måneder."

Svar kort og naturlig. Maksimum 3 setninger.
"""

        # Build chat history for Gemini
        messages = [{"role": "user", "parts": [system_prompt]}]

        for msg in request.chat_history[-10:]:  # Last 10 messages for context
            role = "user" if msg['sender'] == 'user' else "model"
            messages.append({"role": role, "parts": [msg['message']]})

        messages.append({"role": "user", "parts": [request.message]})

        # Generate response
        model = genai.GenerativeModel('gemini-2.0-flash-exp')
        response = model.generate_content(messages)

        ai_message = response.text

        # Extract offer from response (simple regex)
        import re
        offer_match = re.search(r'(\d+(?:\.\d+)?)\s*MNOK.*?(\d+
(?:\.\d+)?)\s*måned', ai_message, re.IGNORECASE)

        offer = None
        if offer_match:
            offer = {
                "cost": float(offer_match.group(1)),
                "duration": float(offer_match.group(2))
            }

        return {
            "message": ai_message,
            "offer": offer,
            "sender": "ai"
        }

    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Test Backend:

```
cd backend
python -m app.main  # or uvicorn app.main:app --reload
```

Test with curl:

```
curl -X POST http://localhost:8000/api/chat \
  -H "Content-Type: application/json" \
  -d '{
    "wbs_code": "1.3.1",
    "supplier_id": "bjorn",
    "message": "Hei, jeg trenger et pristilbud for Grunnarbeid",
    "chat_history": []
  }'
```

Expected response:

```
{
  "message": "Hei! For Grunnarbeid (WBS 1.3.1) inkludert masseutskifting og
fundamenter, kan jeg tilby 120 MNOK og 2 måneder. Dette sikrer kvalitet og
fremdrift.",
  "offer": {
    "cost": 120.0,
    "duration": 2.0
  },
  "sender": "ai"
}
```

Task 2.2: Build Chat Frontend (3 hours)

Create `frontend/src/pages/Chat.jsx`:

```
import React, { useState, useEffect, useRef } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { loadSession, saveSession } from '../utils/sessionManager';

function Chat({ session }) {
  const { wbsCode, supplierId } = useParams();
  const navigate = useNavigate();
  const [messages, setMessages] = useState([]);
  const [inputMessage, setInputMessage] = useState('');
  const [loading, setLoading] = useState(false);
  const messagesEndRef = useRef(null);

  const gameSession = loadSession(session.user.id);
```

```
const wbsItem = gameSession.wbs_items.find((w) => w.code === wbsCode);
const supplier = gameSession.suppliers.find((s) => s.id === supplierId);

useEffect(() => {
  // Load chat history from localStorage
  const chatKey = `${wbsCode}_${supplierId}`;
  const savedChat = gameSession.chat_logs[chatKey] || [];
  setMessages(savedChat);
}, [wbsCode, supplierId]);

useEffect(() => {
  // Auto-scroll to bottom
  messagesEndRef.current?.scrollIntoView({ behavior: 'smooth' });
}, [messages]);

const sendMessage = async () => {
  if (!inputMessage.trim()) return;

  const userMessage = {
    message: inputMessage,
    sender: 'user',
    timestamp: new Date().toISOString(),
  };

  setMessages([...messages, userMessage]);
  setInputMessage('');
  setLoading(true);

  try {
    const response = await fetch('http://localhost:8000/api/chat', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        wbs_code: wbsCode,
        supplier_id: supplierId,
        message: inputMessage,
        chat_history: messages,
      }),
    });

    const data = await response.json();

    const aiMessage = {
      message: data.message,
      sender: 'ai',
      offer: data.offer,
      timestamp: new Date().toISOString(),
    };

    const updatedMessages = [...messages, userMessage, aiMessage];
    setMessages(updatedMessages);

    // Save to localStorage
    const chatKey = `${wbsCode}_${supplierId}`;
    gameSession.chat_logs[chatKey] = updatedMessages;
    gameSession.metrics.negotiation_count += 1;
    saveSession(gameSession);
```

```
    } catch (error) {
      console.error('Error sending message:', error);
      alert('Feil ved sending av melding. Prøv igjen.');
    } finally {
      setLoading(false);
    }
  };

  const acceptOffer = (offer) => {
    // This will be implemented tomorrow (commitment flow)
    console.log('Accept offer:', offer);
  };

  return (
    <div className="chat-page">
      <header className="chat-header">
        <button onClick={() => navigate('/dashboard')}>← Tilbake til
Oversikt</button>
        <div>
          <h2>{supplier.name} - {supplier.role}</h2>
          <p>WBS {wbsItem.code} - {wbsItem.name}</p>
        </div>
      </header>

      <div className="chat-container">
        <div className="chat-window">
          {messages.map((msg, idx) => (
            <div key={idx} className={`message ${msg.sender}`}>
              <div className="message-bubble">{msg.message}</div>
              {msg.offer && (
                <button
                  className="accept-offer-btn"
                  onClick={() => acceptOffer(msg.offer)}
                >
                  Godta: {msg.offer.cost} MNOK, {msg.offer.duration} måneder
                </button>
              )}
            </div>
          ))}
          {loading && (
            <div className="message ai">
              <div className="message-bubble typing">
                {supplier.name} ser gjennom spesifikasjonene...
              </div>
            </div>
          )}
          <div ref={messagesEndRef} />
        </div>

        <div className="message-input">
          <textarea
            value={inputMessage}
            onChange={(e) => setInputMessage(e.target.value)}
            placeholder="Skriv melding..."
            onKeyDown={(e) => {
              if (e.key === 'Enter' && !e.shiftKey) {
```

```
                e.preventDefault();
                sendMessage();
              }
            }}
          />
          <button onClick={sendMessage} disabled={loading}>
            Send →
          </button>
        </div>
      </div>
    </div>
  );
}

export default Chat;
```

Update routing in `App.js`:

```
<Route
  path="/chat/:wbsCode/:supplierId"
  element={session ? <Chat session={session} /> : <Navigate to="/" />}
/>
```

Task 2.3: Add Supplier Selection Modal (2 hours)

Create `frontend/src/components/SupplierModal.jsx`:

```
import React from 'react';

function SupplierModal({ wbsItem, suppliers, onSelect, onClose }) {
  // Filter suppliers by relevant specialties
  const relevantSuppliers = suppliers.filter((s) =>
    s.specialties.some((spec) =>
      wbsItem.category.toLowerCase().includes(spec.toLowerCase()) ||
      wbsItem.name.toLowerCase().includes(spec.toLowerCase())
    )
  );

  // If no matches, show all suppliers
  const displaySuppliers = relevantSuppliers.length > 0 ? relevantSuppliers :
suppliers;

  return (
    <div className="modal-overlay" onClick={onClose}>
      <div className="modal-content" onClick={(e) => e.stopPropagation()}>
        <button className="modal-close" onClick={onClose}>×</button>

        <h2>Velg Leverandør</h2>
        <p>WBS {wbsItem.code} - {wbsItem.name}</p>
```

```
        <div className="supplier-grid">
          {displaySuppliers.map((supplier) => (
            <div key={supplier.id} className="supplier-card" onClick={() =>
onSelect(supplier.id)}>
              <h3>{supplier.name}</h3>
              <p className="supplier-role">{supplier.role}</p>
              <p className="supplier-company">{supplier.company}</p>
              <p className="supplier-personality">{supplier.personality}</p>
            </div>
          ))}
        </div>
      </div>
    </div>
  );
}

export default SupplierModal;
```

Update `WBSList.jsx` to show modal:

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import SupplierModal from './SupplierModal';

function WBSList({ wbsItems, currentPlan, suppliers }) {
  const [selectedWBS, setSelectedWBS] = useState(null);
  const navigate = useNavigate();

  const handleSupplierSelect = (supplierId) => {
    navigate(`/chat/${selectedWBS.code}/${supplierId}`);
  };

  return (
    <div className="wbs-list">
      <h2>Arbeidsnedbrytningsstruktur (WBS)</h2>
      {wbsItems.map((item) => {
        const isCompleted = currentPlan[item.code];
        return (
          <div key={item.code} className="wbs-item">
            {/* ... existing code ... */}
            <button onClick={() => setSelectedWBS(item)}>
              {isCompleted ? 'Reforhandle' : 'Kontakt Leverandør'}
            </button>
          </div>
        );
      })}

      {selectedWBS && (
        <SupplierModal
          wbsItem={selectedWBS}
          suppliers={suppliers}
          onSelect={handleSupplierSelect}
          onClose={() => setSelectedWBS(null)}
```

```
        />
      )}
    </div>
  );
}

export default WBSList;
```

End of Day 2 Deliverables:

- ✅ Gemini AI backend working
- ✅ Chat UI with real-time AI responses
- ✅ Supplier selection modal
- ✅ Chat history persistence in localStorage

Testing Checklist:

- ☐ Click "Kontakt Leverandør" → Supplier modal opens
- ☐ Select supplier → Chat page opens
- ☐ Send message → AI responds within 2-3 seconds
- ☐ AI response includes offer (cost + duration)
- ☐ Refresh page → Chat history persists

Wednesday, December 11 (Day 3): Commitment Flow & Validation

Goal: Complete core simulation loop

Task 3.1: Implement Commitment Flow (3 hours)

Update `Chat.jsx` - Accept Offer Function:

```
import { commitQuote } from '../utils/sessionManager';

const acceptOffer = (offer) => {
  const confirmed = window.confirm(
    `Godta tilbud?\n\nWBS: ${wbsItem.code} - ${wbsItem.name}\nLeverandør:
${supplier.name}\nKostnad: ${offer.cost} MNOK\nVarighe: ${offer.duration}
måneder\n\nDette vil oppdatere prosjektplanen din.`
  );
```

```
  if (confirmed) {
    try {
      const updatedSession = commitQuote(
        gameSession,
        wbsCode,
        supplierId,
        offer.cost,
        offer.duration
      );

      // Add system message to chat
      const systemMessage = {
        message: `✅ Tilbud godtatt og forpliktet til plan`,
        sender: 'system',
        timestamp: new Date().toISOString(),
      };

      const updatedMessages = [...messages, systemMessage];
      setMessages(updatedMessages);

      const chatKey = `${wbsCode}_${supplierId}`;
      updatedSession.chat_logs[chatKey] = updatedMessages;
      saveSession(updatedSession);

      // Show toast notification
      alert(`${wbsItem.code} ${wbsItem.name} lagt til i plan!`);

      // Navigate back to dashboard
      setTimeout(() => navigate('/dashboard'), 1000);
    } catch (error) {
      alert('Feil ved forpliktelse: ' + error.message);
    }
  }
};
```

Task 3.2: Implement Dependency Validation (1 hour)

Update `sessionManager.js` - Add dependency check to `commitQuote`:

```
export const commitQuote = (session, wbsCode, supplierId, cost, duration) => {
  const wbsItem = session.wbs_items.find((item) => item.code === wbsCode);

  // Check dependencies
  for (const depCode of wbsItem.dependencies) {
    if (!session.current_plan[depCode]) {
      throw new Error(
        `Kan ikke forplikte ${wbsItem.code} ${wbsItem.name} før ${depCode} er
fullført.`
      );
    }
  }
```

```
  // ... rest of existing code ...
};
```

Task 3.3: Implement Renegotiation Flow (2 hours)

Update `WBSList.jsx` - Renegotiate button:

```
const handleRenegotiate = (wbsCode) => {
  const confirmed = window.confirm(
    'Dette vil fjerne oppgaven fra planen og åpne chat igjen. Fortsette?'
  );

  if (confirmed) {
    const updatedSession = removeQuote(gameSession, wbsCode);
    saveSession(updatedSession);

    // Force re-render
    window.location.reload();
  }
};

// In render:
{isCompleted && (
  <button onClick={() => handleRenegotiate(item.code)}>Reforhandle</button>
)}
```

Task 3.4: Implement Plan Validation & Submission (2 hours)

Create `frontend/src/components/ValidationModal.jsx`:

```
import React from 'react';

function ValidationModal({ validation, onClose, onExport }) {
  if (validation.isValid) {
    return (
      <div className="modal-overlay" onClick={onClose}>
        <div className="modal-content success" onClick={(e) =>
e.stopPropagation()}>
          <button className="modal-close" onClick={onClose}>✕</button>

          <h1>🎉 Plan Godkjent!</h1>
          <p>Gratulerer! Du har lykkes med å fullføre planleggingsfasen.</p>

          <div className="stats-table">
            <div className="stat-row">
              <span>Total Kostnad:</span>
              <strong>{validation.totalCost} MNOK</strong>
```

```
            </div>
            <div className="stat-row">
              <span>Fullføringsdato:</span>
              <strong>{validation.endDate}</strong>
            </div>
            <div className="stat-row">
              <span>Tid Brukt:</span>
              <strong>{validation.timeSpent} minutter</strong>
            </div>
            <div className="stat-row">
              <span>Forhandlinger:</span>
              <strong>{validation.negotiationCount}</strong>
            </div>
          </div>

          <button className="primary-btn" onClick={onExport}>Eksporter Økt</button>
          <button className="secondary-btn" onClick={onClose}>Start Nytt
Spill</button>
        </div>
      </div>
    );
  }

  // Error modal
  return (
    <div className="modal-overlay" onClick={onClose}>
      <div className="modal-content error" onClick={(e) => e.stopPropagation()}>
        <button className="modal-close" onClick={onClose}>×</button>

        <h1>❌ Planvalidering Mislyktes</h1>
        <p>Feil funnet:</p>

        <ul className="error-list">
          {validation.errors.map((error, idx) => (
            <li key={idx}>{error.message}</li>
          ))}
        </ul>

        {validation.suggestedItems && (
          <div className="suggestions">
            <p><strong>Forslag:</strong> Vurder å reforhandle disse oppgavene:</p>
            <ul>
              {validation.suggestedItems.map((item, idx) => (
                <li key={idx}>{item.code} - {item.name} ({item.cost} MNOK)</li>
              ))}
            </ul>
          </div>
        )}

        <button className="primary-btn" onClick={onClose}>Tilbake til
Planlegging</button>
      </div>
    </div>
  );
}

export default ValidationModal;
```

Update `Dashboard.jsx` - Add Submit button:

```
import { validatePlan } from '../utils/sessionManager';

const [showValidation, setShowValidation] = useState(false);
const [validationResult, setValidationResult] = useState(null);

const handleSubmitPlan = () => {
  const completedCount = Object.keys(gameSession.current_plan).length;

  if (completedCount < 15) {
    alert(`Du har kun fullført ${completedCount} / 15 oppgaver. Fullfør alle før
innsending.`);
    return;
  }

  const validation = validatePlan(gameSession);

  // If errors, suggest most expensive items to renegotiate
  if (!validation.isValid && validation.errors.some(e => e.type === 'budget')) {
    const sortedItems = Object.entries(gameSession.current_plan)
      .map(([code, plan]) => ({
        code,
        ...gameSession.wbs_items.find(w => w.code === code),
        cost: plan.cost
      }))
      .sort((a, b) => b.cost - a.cost)
      .slice(0, 3);

    validation.suggestedItems = sortedItems;
  }

  // Add stats for success modal
  if (validation.isValid) {
    const timeSpent = Math.round((Date.now() - new Date(gameSession.created_at)) /
60000);
    validation.totalCost = gameSession.metrics.total_budget_used;
    validation.endDate = gameSession.metrics.projected_end_date;
    validation.timeSpent = timeSpent;
    validation.negotiationCount = gameSession.metrics.negotiation_count;
  }

  setValidationResult(validation);
  setShowValidation(true);
};

const handleExport = () => {
  const dataStr = JSON.stringify(gameSession, null, 2);
  const dataBlob = new Blob([dataStr], { type: 'application/json' });
  const url = URL.createObjectURL(dataBlob);

  const link = document.createElement('a');
  link.href = url;
  link.download = `nye_haedda_session_${Date.now()}.json`;
```

```
    link.click();

    URL.revokeObjectURL(url);
  };

  // In render:
  <button
    className="submit-plan-btn"
    onClick={handleSubmitPlan}
    disabled={Object.keys(gameSession.current_plan).length < 15}
  >
    Send Inn Plan
  </button>

  {showValidation && (
    <ValidationModal
      validation={validationResult}
      onClose={() => setShowValidation(false)}
      onExport={handleExport}
    />
  )}
```

End of Day 3 Deliverables:

- ✅ Commitment flow (accept offer → update plan)
- ✅ Dependency validation
- ✅ Renegotiation flow (remove from plan)
- ✅ Plan validation (budget, timeline, completeness)
- ✅ Success/error modals
- ✅ Export session JSON

Testing Checklist:

- ☐ Accept offer → Dashboard shows updated budget
- ☐ Try to commit WBS 2.1 before 1.3.1 → Error message
- ☐ Renegotiate item → Removed from plan, chat reopens
- ☐ Submit plan with budget overrun → Error modal with suggestions
- ☐ Submit plan successfully → Success modal
- ☐ Export session → JSON file downloads

Thursday, December 12 (Day 4): UI/UX Polish & Testing

Goal: Production-ready core MVP

Task 4.1: Add CSS Styling (4 hours)

Create `frontend/src/App.css`:

```
/* Variables */
:root {
  --blue-600: #3B82F6;
  --blue-700: #2563EB;
  --green-500: #10B981;
  --yellow-500: #F59E0B;
  --red-500: #EF4444;
  --gray-50: #F9FAFB;
  --gray-100: #F3F4F6;
  --gray-300: #D1D5DB;
  --gray-600: #4B5563;
  --gray-900: #111827;
}

* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: 'Inter', -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
  background-color: var(--gray-50);
  color: var(--gray-900);
}

/* Login Page */
.login-container {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}

.login-card {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  max-width: 400px;
  width: 100%;
}

.login-card h1 {
  font-size: 1.5rem;
  margin-bottom: 0.5rem;
}
```

```css
.login-card form {
  margin-top: 1.5rem;
}

.login-card label {
  display: block;
  font-weight: 500;
  font-size: 0.875rem;
  margin-bottom: 0.25rem;
  color: var(--gray-600);
}

.login-card input {
  width: 100%;
  padding: 0.5rem;
  margin-bottom: 1rem;
  border: 1px solid var(--gray-300);
  border-radius: 4px;
  font-size: 0.875rem;
}

.login-card input:focus {
  outline: none;
  border-color: var(--blue-600);
  box-shadow: 0 0 0 2px rgba(59, 130, 246, 0.2);
}

.login-card button[type="submit"] {
  width: 100%;
  padding: 0.75rem;
  background-color: var(--blue-600);
  color: white;
  border: none;
  border-radius: 4px;
  font-weight: 600;
  cursor: pointer;
}

.login-card button[type="submit"]:hover {
  background-color: var(--blue-700);
}

.login-card button[type="submit"]:disabled {
  background-color: var(--gray-300);
  cursor: not-allowed;
}

.login-card .error {
  color: var(--red-500);
  font-size: 0.875rem;
  margin-bottom: 1rem;
}

/* Dashboard */
.dashboard {
  max-width: 1200px;
  margin: 0 auto;
```

```
  padding: 2rem;
}

.dashboard-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 2rem;
  padding-bottom: 1rem;
  border-bottom: 1px solid var(--gray-300);
}

.dashboard-header h1 {
  font-size: 1.5rem;
}

.dashboard-header button {
  padding: 0.5rem 1rem;
  background-color: white;
  border: 1px solid var(--gray-300);
  border-radius: 4px;
  cursor: pointer;
}

/* Constraint Panel */
.constraint-panel {
  background: white;
  padding: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
  margin-bottom: 1.5rem;
}

.budget-section, .timeline-section {
  margin-bottom: 1rem;
}

.budget-section p {
  font-size: 0.875rem;
  font-weight: 500;
  margin-bottom: 0.5rem;
}

.progress-bar {
  height: 24px;
  background-color: var(--gray-300);
  border-radius: 12px;
  overflow: hidden;
}

.progress-fill {
  height: 100%;
  transition: width 500ms ease-in-out;
}

.progress-fill.green {
  background-color: var(--green-500);
```

```
}

.progress-fill.yellow {
  background-color: var(--yellow-500);
}

.progress-fill.red {
  background-color: var(--red-500);
}

/* WBS List */
.wbs-list {
  background: white;
  padding: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
  max-height: 600px;
  overflow-y: auto;
}

.wbs-item {
  display: flex;
  align-items: flex-start;
  padding: 1rem 0;
  border-bottom: 1px solid var(--gray-100);
}

.wbs-item:last-child {
  border-bottom: none;
}

.status-icon {
  font-size: 1.25rem;
  margin-right: 0.75rem;
}

.wbs-content {
  flex: 1;
}

.wbs-content h4 {
  font-size: 0.875rem;
  font-weight: 500;
  margin-bottom: 0.5rem;
}

.wbs-content p {
  font-size: 0.75rem;
  color: var(--gray-600);
  margin-bottom: 0.5rem;
}

.wbs-content button {
  padding: 0.25rem 0.75rem;
  font-size: 0.75rem;
  background-color: var(--blue-600);
  color: white;
```

```
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.wbs-content button:hover {
  background-color: var(--blue-700);
}

/* Chat Page */
.chat-page {
  max-width: 900px;
  margin: 0 auto;
}

.chat-header {
  background: white;
  padding: 1rem 1.5rem;
  border-bottom: 1px solid var(--gray-300);
  display: flex;
  align-items: center;
  gap: 1rem;
}

.chat-header button {
  padding: 0.5rem 1rem;
  background: none;
  border: none;
  color: var(--blue-600);
  cursor: pointer;
  font-size: 0.875rem;
}

.chat-container {
  display: flex;
  flex-direction: column;
  height: calc(100vh - 120px);
}

.chat-window {
  flex: 1;
  padding: 1.5rem;
  overflow-y: auto;
  background-color: var(--gray-50);
}

.message {
  display: flex;
  margin-bottom: 1rem;
}

.message.user {
  justify-content: flex-end;
}

.message.ai, .message.system {
  justify-content: flex-start;
```

```
}

.message-bubble {
  max-width: 70%;
  padding: 0.75rem 1rem;
  border-radius: 8px;
  font-size: 0.875rem;
}

.message.user .message-bubble {
  background-color: var(--blue-600);
  color: white;
  border-bottom-right-radius: 0;
}

.message.ai .message-bubble {
  background-color: white;
  color: var(--gray-900);
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.1);
  border-bottom-left-radius: 0;
}

.message.system .message-bubble {
  background-color: var(--gray-100);
  color: var(--gray-600);
  font-size: 0.75rem;
  font-style: italic;
  text-align: center;
  max-width: 100%;
}

.typing {
  animation: pulse 1.5s infinite;
}

@keyframes pulse {
  0%, 100% { opacity: 1; }
  50% { opacity: 0.5; }
}

.accept-offer-btn {
  margin-top: 0.5rem;
  padding: 0.5rem 0.75rem;
  background-color: var(--green-500);
  color: white;
  border: none;
  border-radius: 4px;
  font-size: 0.75rem;
  cursor: pointer;
}

.accept-offer-btn:hover {
  background-color: #059669;
}

.message-input {
  background: white;
```

```
  padding: 1rem;
  border-top: 1px solid var(--gray-300);
  display: flex;
  gap: 0.5rem;
}

.message-input textarea {
  flex: 1;
  padding: 0.75rem;
  border: 1px solid var(--gray-300);
  border-radius: 4px;
  resize: none;
  font-family: inherit;
  font-size: 0.875rem;
}

.message-input button {
  padding: 0.75rem 1.5rem;
  background-color: var(--blue-600);
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  font-weight: 600;
}

.message-input button:disabled {
  background-color: var(--gray-300);
  cursor: not-allowed;
}

/* Modal */
.modal-overlay {
  position: fixed;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: rgba(0, 0, 0, 0.5);
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 1000;
}

.modal-content {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.2);
  max-width: 600px;
  width: 90%;
  max-height: 80vh;
  overflow-y: auto;
  position: relative;
}
```

```css
.modal-close {
  position: absolute;
  top: 1rem;
  right: 1rem;
  background: none;
  border: none;
  font-size: 1.5rem;
  cursor: pointer;
  color: var(--gray-600);
}

.modal-content h1 {
  font-size: 1.5rem;
  margin-bottom: 1rem;
}

.modal-content.success h1 {
  color: var(--green-500);
}

.modal-content.error h1 {
  color: var(--red-500);
}

.stats-table {
  background-color: var(--gray-50);
  padding: 1rem;
  border-radius: 4px;
  margin: 1.5rem 0;
}

.stat-row {
  display: flex;
  justify-content: space-between;
  margin-bottom: 0.5rem;
  font-size: 0.875rem;
}

.error-list {
  margin: 1rem 0;
  padding-left: 1.5rem;
  color: var(--red-500);
}

.suggestions {
  margin-top: 1.5rem;
  padding: 1rem;
  background-color: var(--gray-50);
  border-radius: 4px;
}

.primary-btn, .secondary-btn {
  width: 100%;
  padding: 0.75rem;
  margin-top: 0.5rem;
  border: none;
  border-radius: 4px;
```

```
  font-weight: 600;
  cursor: pointer;
}

.primary-btn {
  background-color: var(--blue-600);
  color: white;
}

.primary-btn:hover {
  background-color: var(--blue-700);
}

.secondary-btn {
  background-color: white;
  border: 1px solid var(--gray-300);
  color: var(--gray-900);
}

/* Supplier Modal */
.supplier-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 1rem;
  margin-top: 1.5rem;
}

.supplier-card {
  padding: 1rem;
  border: 1px solid var(--gray-300);
  border-radius: 8px;
  cursor: pointer;
  transition: all 200ms;
}

.supplier-card:hover {
  border-color: var(--blue-600);
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.supplier-card h3 {
  font-size: 1rem;
  margin-bottom: 0.5rem;
}

.supplier-role {
  font-weight: 600;
  color: var(--blue-600);
  font-size: 0.875rem;
}

.supplier-company {
  font-size: 0.75rem;
  color: var(--gray-600);
  margin-bottom: 0.75rem;
}
```

```css
.supplier-personality {
  font-size: 0.75rem;
  color: var(--gray-600);
  font-style: italic;
}
```

Task 4.2: Manual Testing (2 hours)

Test Scenarios:

1. **Complete Happy Path (30 min)**
    - Register → Login → Dashboard loads
    - Negotiate with 5 suppliers → Accept offers
    - Check budget updates in real-time
    - Complete all 15 WBS items (simulate quickly by accepting first offers)
    - Submit plan → Success modal
    - Export JSON
2. **Budget Overrun Scenario (15 min)**
    - Accept all first offers (high costs)
    - Submit plan → Error modal
    - Follow suggestions → Renegotiate expensive items
    - Submit again → Success
3. **Dependency Validation (10 min)**
    - Try to commit WBS 2.1 before 1.3.1 → Error
    - Commit 1.3.1 → Now 2.1 allowed
4. **Chat Persistence (10 min)**
    - Chat with supplier → Close chat
    - Reopen chat → History preserved
5. **Cross-Browser Testing (20 min)**
    - Test in Chrome, Firefox, Edge
    - Check responsive layout (tablet: 768px)
6. **AI Quality Testing (30 min)**

- Test 10 negotiations with different prompts:
  - "Trenger pristilbud"
  - "For høyt, kan du gå ned?"
  - "Hva med kvalitet?"
  - "Referanse til F-003 krav"
- Verify AI uses Norwegian terminology
- Check concession behavior (price goes down gradually)

Task 4.3: Bug Fixes & Edge Cases (2 hours)

Common Issues to Address:

1. **LocalStorage Quota**
   - Add storage check: If >4MB, warn user to export and clear old sessions
2. **API Error Handling**
   - If Gemini API fails, show: "AI-tjenesten er midlertidig utilgjengelig. Prøv igjen."
   - Add retry button
3. **Date Calculation Edge Case**
   - Ensure dependencies start *after* prerequisite ends (not same day)
4. **Empty Chat Message**
   - Disable send button if input is empty or whitespace-only
5. **Refresh During Chat**
   - Preserve unsent message in sessionStorage

End of Day 4 Deliverables:

- ✅ Production-ready UI with CSS
- ✅ Tested complete simulation flow
- ✅ Bug fixes for edge cases
- ✅ Cross-browser compatibility

Testing Checklist:

- ☐ Complete simulation from register to export (works smoothly)
- ☐ AI responds naturally in Norwegian
- ☐ Budget tracking accurate (manual calculation matches)
- ☐ All 15 WBS items committable
- ☐ Validation logic correct (budget/timeline/dependencies)

Friday, December 13 (Day 5): Deployment & Production Setup

Goal: Live on Vercel with production configuration

Task 5.1: Vercel Frontend Deployment (2 hours)

Step-by-Step Instructions:

1. **Create Vercel Account**
    - Navigate to: https://vercel.com
    - Sign up with GitHub account
    - Connect GitHub repository: `SG-Gruppe-14-d2`
2. **Configure Frontend Deployment**
    - Click "New Project"
    - Import `SG-Gruppe-14-d2` repository
    - Framework Preset: Vite (auto-detected)
    - Root Directory: `frontend`
    - Build Command: `npm run build`
    - Output Directory: `dist`
    - Environment Variables:

```
VITE_SUPABASE_URL=https://[your-project-id].supabase.co
VITE_SUPABASE_ANON_KEY=[your-anon-key]
VITE_BACKEND_URL=https://[will-be-updated-after-backend-deploy]
```

    - Click "Deploy"

3. **Wait for Build (2-3 minutes)**
    - Vercel builds and deploys automatically
    - Production URL: `https://nye-haedda.vercel.app` (or custom)
4. **Update Supabase Redirect URLs**
    - In Supabase Dashboard → Authentication → URL Configuration
    - Site URL: `https://nye-haedda.vercel.app`
    - Redirect URLs: Add `https://nye-haedda.vercel.app/dashboard`

Task 5.2: Vercel Backend Deployment (2 hours)

Prepare FastAPI for Vercel Serverless:

1. **Create `backend/vercel.json`:**

```
{
  "builds": [
    {
      "src": "app/main.py",
      "use": "@vercel/python"
    }
  ],
  "routes": [
    {
      "src": "/(.*)",
      "dest": "app/main.py"
    }
  ]
}
```

2. **Create `backend/requirements.txt`:**

```
fastapi
uvicorn
google-generativeai
python-dotenv
pydantic
```

3. **Update `backend/app/main.py` - Fix file paths for production:**

```
import os
import json
from pathlib import Path

# Get base directory
BASE_DIR = Path(__file__).resolve().parent.parent

# Load suppliers and WBS from JSON (use relative paths)
def load_json_file(filename):
    file_path = BASE_DIR / 'data' / filename
    if not file_path.exists():
        raise FileNotFoundError(f"{filename} not found")
    with open(file_path, 'r', encoding='utf-8') as f:
        return json.load(f)

SUPPLIERS = {s['id']: s for s in load_json_file('suppliers.json')}
WBS_ITEMS = {w['code']: w for w in load_json_file('wbs.json')}
```

4. **Copy data files to backend:**

```
mkdir backend/data
cp frontend/public/data/wbs.json backend/data/
cp frontend/public/data/suppliers.json backend/data/
```

5. **Deploy to Vercel:**
   - Create new Vercel project for backend
   - Import same GitHub repo
   - Root Directory: `backend`
   - Environment Variables:

     ```
     GEMINI_API_KEY=[your-key]
     ```

   - Deploy
6. **Update Frontend Environment Variable:**
   - In Vercel Frontend project → Settings → Environment Variables
   - Update `VITE_BACKEND_URL` to backend URL: `https://nye-haedda-api.vercel.app`
   - Redeploy frontend

Task 5.3: Production Testing (2 hours)

Full Production Test:

1. Open production URL: `https://nye-haedda.vercel.app`
2. Register new user → Login
3. Complete full simulation (negotiate 15 WBS items)
4. Submit plan → Export JSON
5. Check all features work identically to localhost

Performance Testing:

- Run Lighthouse audit:
  - Performance: Target >85
  - Accessibility: Target >90
  - Best Practices: Target >90

Security Checklist:

- ☐ `.env` files NOT committed to Git
- ☐ Gemini API key NOT exposed in frontend
- ☐ HTTPS enforced (Vercel default)
- ☐ CORS only allows production URLs

Task 5.4: Documentation & Handoff (2 hours)

Create `README.md` in root:

```
# Nye Hædda Barneskole - Prosjektledelsessimulator

## Live Demo
🚀 **Production:** https://nye-haedda.vercel.app

## Features
- ✅ Supabase Authentication
- ✅ AI-powered negotiation with Gemini 2.0 Flash
- ✅ Real-time budget tracking
- ✅ 15 WBS items from real construction project
- ✅ Dependency validation
- ✅ Plan validation & export

## Local Development

### Prerequisites
- Node.js 18+
```

```
- Python 3.9+
- Supabase account
- Google AI Studio API key

### Frontend Setup
```bash
cd frontend
npm install
cp .env.example .env  # Add your Supabase credentials
npm run dev  # http://localhost:3000
```

Backend Setup

```
cd backend
pip install -r requirements.txt
cp .env.example .env  # Add your Gemini API key
uvicorn app.main:app --reload  # http://localhost:8000
```

Deployment

Vercel (Production)

- Frontend: Auto-deploy from `main` branch
- Backend: Auto-deploy from `main` branch
- Environment variables set in Vercel dashboard

Testing

Run Unit Tests (Not implemented yet)

```
cd frontend
npm run test
```

Manual Testing
```

See `docs/test-design.md` for comprehensive test cases.

Project Structure

```
SG-Gruppe-14-d2/
├── frontend/           # React + Vite frontend
│   ├── src/
│   │   ├── pages/     # Login, Dashboard, Chat
│   │   ├── components/
│   │   └── utils/     # sessionManager.js
│   └── public/data/   # wbs.json, suppliers.json
├── backend/           # FastAPI backend
│   ├── app/
│   │   └── main.py   # Gemini AI integration
│   └── data/         # Static JSON files
└── docs/             # PRD, Epics, UX Design, etc.
```

Contributors

- SG-Gruppe-14-d2
- Course: IBE160 Programmering med KI

```
---

**End of Day 5 Deliverables:**
- ✅ Production deployment on Vercel
- ✅ All environment variables configured
- ✅ Production testing complete
- ✅ README documentation
- ✅ Live demo accessible

**Testing Checklist:**
- [ ] Production URL works
- [ ] Registration and login functional
- [ ] AI chat responds (Gemini API connected)
- [ ] Budget tracking accurate
- [ ] Export JSON downloads correctly
- [ ] Lighthouse score >85 (Performance)

---

### Saturday-Sunday, December 14-15 (Days 6-7): Buffer & Polish

**Goal:** Final polish, user testing, bug fixes
```

Task 6.1: User Acceptance Testing (3 hours)

Recruit 3-5 Test Users:
- Classmates or friends
- Give them task: "Complete the simulation without instructions"
- Observe where they struggle

Feedback Collection:
1. Was the UI intuitive?
2. Were AI suppliers realistic?
3. Did you understand budget constraints?
4. Any bugs or confusing moments?

Fix Top 3 Issues Immediately

Task 6.2: Final UI Polish (2 hours)

Improvements:
1. Add loading skeleton screens (instead of "Laster...")
2. Improve toast notifications (use library like `react-hot-toast`)
3. Add animations:
   - Modal fade-in (200ms)
   - Budget bar slide (500ms)
   - Page transitions
4. Add favicon and meta tags (SEO)

Task 6.3: Norwegian Language Review (1 hour)

Checklist:
- [ ] All button labels in Norwegian
- [ ] Error messages in Norwegian
- [ ] AI responses use Norwegian terminology
- [ ] Date formatting: "15. mai 2026" (not "May 15, 2026")
- [ ] Number formatting: "700 MNOK" (space, not comma)

Task 6.4: Documentation Video (Optional, 1 hour)

Create 3-minute Demo Video:
1. Screen recording walkthrough
2. Show: Register → Negotiate → Submit → Export
3. Narrate key features in Norwegian
4. Upload to YouTube (unlisted)
5. Add link to README

Task 6.5: Final Testing & Bug Bash (3 hours)

Comprehensive Test:
- [ ] Complete simulation 3 times with different strategies
- [ ] Test all error scenarios

```
- [ ] Verify all 15 WBS items negotiable
- [ ] Check localStorage doesn't exceed 5MB
- [ ] Test on mobile (basic check - full responsive not MVP)

**Known Issues to Document:**
- Visualization features (Gantt, Precedence) not implemented (post-MVP)
- Mobile optimization limited (tablet minimum 768px)

---

**End of Weekend Deliverables:**
- ✅ User testing feedback incorporated
- ✅ UI polished and professional
- ✅ Norwegian language verified
- ✅ Final bugs fixed
- ✅ Ready for submission

---

## Risk Mitigation Strategies

### High-Risk Items

| Risk | Mitigation | Owner |
|------|-----------|-------|
| **Gemini API Rate Limits** | Use caching for common prompts, implement
exponential backoff | Backend dev |
| **LocalStorage Quota Exceeded** | Add storage monitoring, prompt export at 80% |
Frontend dev |
| **Dependency Calculation Bug** | Write unit tests for `commitQuote` function,
manual validation | All |
| **AI Produces Unrealistic Offers** | Test with 50 scenarios, tune prompts, add
fallback logic | Backend dev |
| **Time Overrun (Can't finish by Dec 15)** | **CRITICAL: Defer Epic 10
(Visualization) to post-MVP** | Team decision |

### Contingency Plans

**If Behind Schedule by Wednesday (Day 3):**
- **Cut:** Gantt Chart, Precedence Diagram, History (Epic 10) → Saves 24 story
points
- **Focus:** Core negotiation loop (Epics 1-7, 9)
- **Deliver:** Working MVP without visualizations

**If Gemini API Fails:**
- **Fallback:** Use hardcoded negotiation logic (like current
`backend/app/main.py`)
- **Impact:** Less realistic, but functional for demonstration

**If Supabase Issues:**
- **Fallback:** Use mock authentication (localStorage only)
- **Impact:** Multi-user not supported, but single-user MVP works

---

## Daily Standup Template
```

```
**Every Morning (15 min):**

1. **Yesterday:**
   - What did you complete?
   - Any blockers?

2. **Today:**
   - What will you work on?
   - Expected time?

3. **Blockers:**
   - Any issues blocking progress?
   - Need help from team?

**Use Slack/Discord for async updates if team distributed**

---

## Success Metrics

**Minimum Viable Product (MVP) Criteria:**

✅ **Must Have (Non-Negotiable):**
1. User can register and log in (Supabase)
2. User can negotiate with AI suppliers (Gemini)
3. User can commit quotes to plan
4. Budget and timeline tracked in real-time
5. Plan validation (budget ≤700 MNOK, date ≤May 15, 2026)
6. Export session as JSON
7. Norwegian language throughout
8. Deployed to production (Vercel)

⚠️ **Should Have (Defer if Necessary):**
- Gantt Chart visualization
- Precedence Diagram
- History/Timeline view
- Advanced help/onboarding

**Stretch Goals (December 16+):**
- Implement deferred Epic 10 features
- Add mobile responsiveness (<768px)
- Email confirmation for registration
- Real-time collaboration (multiplayer)

---

## Key Contacts & Resources

**Critical Resources:**

1. **Supabase Dashboard:** https://supabase.com/dashboard
2. **Google AI Studio:** https://aistudio.google.com
3. **Vercel Dashboard:** https://vercel.com/dashboard
4. **Project Documentation:** `/docs` folder
5. **GitHub Repo:** https://github.com/[your-org]/SG-Gruppe-14-d2

**Support Channels:**
```

```
- **Supabase Support:** https://supabase.com/support
- **Gemini API Docs:** https://ai.google.dev/docs
- **Vercel Support:** https://vercel.com/help

---

## Appendix A: Environment Variables Checklist

**Frontend (`.env`):**
```bash
VITE_SUPABASE_URL=https://[project-id].supabase.co
VITE_SUPABASE_ANON_KEY=[anon-key]
VITE_BACKEND_URL=http://localhost:8000  # Dev
VITE_BACKEND_URL=https://[backend].vercel.app  # Prod
```

Backend (`.env`):

```
GEMINI_API_KEY=AIzaSy...
SUPABASE_URL=https://[project-id].supabase.co  # Optional
SUPABASE_ANON_KEY=[anon-key]  # Optional
```

Vercel Environment Variables:

- Set in Vercel Dashboard → Project → Settings → Environment Variables
- Separate configs for Production, Preview, Development

Appendix B: Quick Command Reference

Frontend:

```
npm install                    # Install dependencies
npm run dev                    # Start dev server (localhost:3000)
npm run build                  # Build for production
npm run preview                # Preview production build
```

Backend:

```
pip install -r requirements.txt   # Install dependencies
uvicorn app.main:app --reload     # Start dev server (localhost:8000)
python -m pytest                  # Run tests (if implemented)
```
```

Git Workflow:

```
git status                      # Check changes
git add .                       # Stage all changes
git commit -m "feat: add chat UI"  # Commit with message
git push origin main            # Push to GitHub
```

Appendix C: Troubleshooting Guide

Common Issues

1. "Module not found: @supabase/supabase-js"

- Solution: `npm install @supabase/supabase-js`

2. "CORS error when calling backend"

- Solution: Check `CORS middleware` in `backend/app/main.py`
- Ensure frontend URL in `allow_origins`

3. "Gemini API returns 403 Forbidden"

- Solution: Check API key in backend `.env`
- Verify API key is valid in Google AI Studio

4. "LocalStorage quota exceeded"

- Solution: Prompt user to export and clear old sessions
- Implement auto-pruning of chat_logs >50 messages

5. "Build fails on Vercel"

- Check build logs in Vercel dashboard
- Common: Missing environment variables
- Solution: Add all required env vars in Vercel settings

End of Implementation Plan

Next Steps:

1. Review this plan with team
2. Assign tasks to team members
3. Start Day 1 (Monday, December 9) with Supabase setup
4. Daily standups to track progress
5. Adjust scope if falling behind (defer Epic 10)

Realistic Assessment: With focused effort (6-8 hours/day), this plan is **achievable** for a core MVP (Epics 1-7, 9) by December 15. Visualization features (Epic 10) should be considered **stretch goals** for December 16+.

Good luck! Lykke til! 🚀