Complete Implementation Plan v3.0

Nye Hædda Barneskole - PM Simulator

Based on UX Mockups + Functional Flows + Complete Documentation Analysis

Date: December 14, 2025 **Version:** 3.0 (Final) **Architecture:** Next.js 14+ + Supabase + Gemini AI

Table of Contents

---

1. Executive Summary

Project Vision

Create an immersive PM learning tool where Norwegian university students experience realistic budget negotiations through AI-powered supplier interactions, learning critical project management skills in a safe, gamified environment.

Key Differentiators

- **AI-Powered Negotiation:** 4 distinct AI agents with realistic Norwegian personalities
- **3-Tier Budget Model:** Unique visualization showing locked vs. negotiable budget
- **Explicit Decision Making:** NO automatic acceptance - every decision requires conscious user action
- **Inflexible Timeline:** Owner agent NEVER extends deadline (teaches time management)
- **Norwegian Context:** All UI, errors, and AI responses in Norwegian

POC Scope

- **3 Negotiable WBS Items** (105 + 60 + 180 = 345 MNOK baseline)
- **12 Locked WBS Items** (390 MNOK already committed)
- **4 AI Agents** (1 Owner + 3 Suppliers)
- **Budget Challenge:** 35 MNOK deficit (345 MNOK needed, only 310 MNOK available)
- **Duration:** 45-60 minutes per game session

---

2. Critical Requirements from UX Mockups

2.1 Visual Design System

Color Palette:

- **Primary Blue:** `#3b82f6` (negotiable items, CTAs)

- **Success Green:** `#10b981` (accept buttons, progress)
- **Warning Yellow:** `#f59e0b` (budget alerts, owner agent)
- **Danger Red:** `#ef4444` (errors, over budget)
- **Neutral Gray:** `#f9fafb` (backgrounds), `#111827` (text)

Typography:

- **Font:** Inter, fallback to system sans-serif
- **Title:** 20px, bold (`font-size:20px; font-weight:700`)
- **Heading:** 16px, bold (`font-size:16px; font-weight:700`)
- **Body:** 13px, medium (`font-size:13px; font-weight:500`)
- **Description:** 12px, normal (`font-size:12px`)
- **Small:** 10-11px (`font-size:10px`)

Spacing & Layout:

- **Card Padding:** 24px (`p-6`)
- **Section Spacing:** 24px (`mb-6`)
- **Border Radius:** 6-8px for cards, 4px for buttons
- **Main Content:** 900px width
- **Sidebar:** 420px width

2.2 Three-Tier Budget Display (CRITICAL)

Exact mockup structure:

TIER 1: Tilgjengelig (Available)

- Label: "Tilgjengelig budsjett (3 forhandlbare pakker)"
- Display: `0 / 310 MNOK` (updates as commitments made)
- Progress bar: Green (0-70%), Yellow (70-90%), Red (>90%), overflow Red (>100%)
- Text below bar: `X% brukt - Y MNOK gjenstår`
- Background: `#f9fafb`, Border: `#d1d5db`

TIER 2: Låst (Locked)

- Label: "Låst budsjett (12 kontraktfestede pakker)"
- Display: `390 MNOK` (constant, never changes)
- Subtext: "Dette budsjettet er allerede forpliktet og kan ikke endres"

- Background: `#f3f4f6`, Border: `#d1d5db`

TIER 3: Totalt (Total)

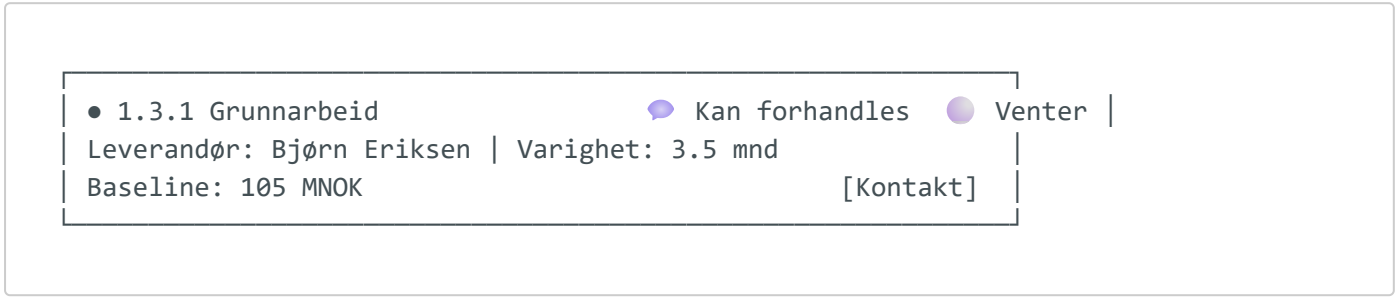
- Label: "Totalt budsjett"
- Display: `390 / 700 MNOK` (Tier 2 + Tier 1 used)
- Subtext: `56% brukt | 310 MNOK gjenstår ✓`
- Background: `#f9fafb`, Border: `#d1d5db`

Warning Banner (Below Tier 3):

- Background: `#fef3c7` (yellow-100)
- Border: `#f59e0b` (yellow-500), 2px
- Icon: ⚠
- Title: "VIKTIG: Budsjettutfordring" (bold, `#92400e`)
- Line 1: "3 forhandlbare pakker estimert til 345 MNOK (105+60+180)"
- Line 2: "Tilgjengelig budsjett: 310 MNOK → Underskudd: 35 MNOK"

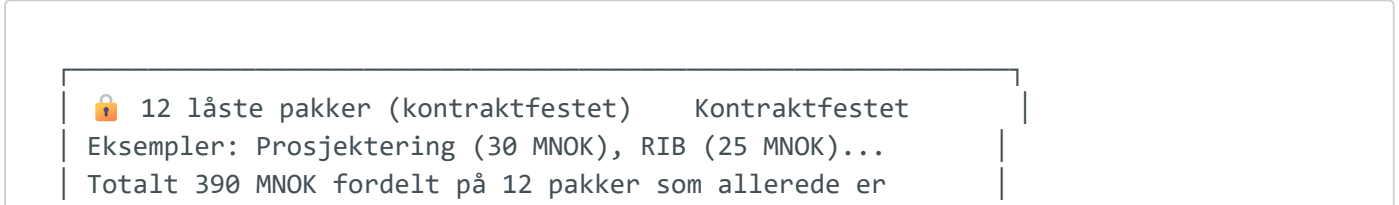
2.3 WBS Items Structure

Negotiable Items (3 total):



- **Background:** `#eff6ff` (blue-50)
- **Border:** `#3b82f6` (blue-500), 2px
- **Left accent:** 4px blue vertical bar
- **Badge:** " Kan forhandles" (blue background, `#dbeafe`)
- **Status:** Venter | Under forhandling | Forpliktet
- **Button:** "Kontakt" (blue, `#3b82f6`, white text)

Locked Items Preview (12 total):



```
| forhandlet                                                    |
└──────────────────────────────────────────────────────────────┘
```

- **Background:** `#f3f4f6` (gray-200), opacity 0.7
- **Border:** `#d1d5db` (gray-400), 1px
- **Left accent:** 2px gray vertical bar
- **Badge:** "Kontraktfestet" (gray background, `#e5e7eb`)
- **Text:** Gray (`#6b7280`)
- **Non-clickable**

2.4 Chat Interface (From Mockup)

Chat Header:

- Avatar: Circle with initials (e.g., "KA" for Kari Andersen)
- Agent name: Bold, 18px
- WBS info: "Leverandør 2: Fundamentering (WBS 1.3.2)" (14px, gray)
- Toggle: "Forhandler med: [Leverandør] / [Eier]"
  - Active: Blue background (`#3b82f6`), white text
  - Inactive: Gray background (`#f3f4f6`), dark text

Message Display:

- **User messages:** Right-aligned, blue background (`#eff6ff`), blue border (`#3b82f6`)
- **AI messages:** Left-aligned, gray background (`#f9fafb`), gray border (`#d1d5db`)
- **Timestamp:** Small text (11px), gray, right-aligned
- **Offer box:** Green background (`#f0fdf4`), green border (`#10b981`, 2px)
  - Title: "✓ TILBUD KLAR" (bold)
  - Details: WBS name, cost, duration, quality

Accept/Reject Buttons (CRITICAL):

```
┌──────────────────────────────────────────┐    ┌──────────────────┐
│  ✓ Godta tilbud: 55 MNOK, 2.5 mnd        │    │ X Avslå og       │
│                                          │    │   reforhandle    │
└──────────────────────────────────────────┘    └──────────────────┘
   Green (#10b981), large (280x50px)            Gray (#f3f4f6), smaller
```

- **Accept:** Green (`#10b981`), white text, 15px, bold, large (280px wide)
- **Reject:** Gray (`#f3f4f6`), dark text, secondary, smaller (180px wide)
- **Spacing:** 20px gap between buttons
- **Note below:** "Ved å godta, oppdateres budsjettet ditt automatisk" (green text)

Right Sidebar (3 sections):

1. **Current WBS Info** (Blue box, `#eff6ff`)
    - WBS name, baseline, supplier, critical path status, current status
2. **Budget Impact Preview** (Yellow box, `#fef3c7`)
    - "Hvis godtatt (55 MNOK):"
    - Tier 1: 55/310 MNOK (18%)
    - Tier 2: 390 MNOK
    - Tier 3: 445/700 MNOK (64%)
    - "Gjenstår: 255 MNOK for 2 pakker"
3. **Negotiation Capabilities** (Gray box, `#f9fafb`)
    - "Kari Andersen kan:"
        - ✓ Redusere pris (mot kvalitet)
        - ✓ Endre varighet (mot kostnad)
        - ✓ Justere materialer/metoder
    - "Kari Andersen kan IKKE:"
        - ✗ Forlenge prosjektdeadline (red text)
        - ✗ Endre scope (kun eier) (red text)
        - ✗ Øke total budsjett (kun eier) (red text)
4. **Documents** (Optional)
    - PDF icons with filenames
    - Specifications, cost estimates, timeline

2.5 Dashboard Sidebar (Right)

Section 1: Budget Challenge (Red box, `#fee2e2`)

- Title: "⚠ BUDSJETTUTFORDRING" (bold, `#991b1b`)
- Total budget: 700 MNOK

- Locked (12): 390 MNOK
- Available (3): 310 MNOK
- Divider line
- Baseline estimates:
  - WBS 1.3.1: 105 MNOK
  - WBS 1.3.2: 60 MNOK
  - WBS 1.4.1: 180 MNOK
  - Total: 345 MNOK

Section 2: Solution (Green box, `#f0fdf4`)

- Title: "✓ LØSNING" (bold, `#065f46`)
- "Du må forhandle ned med 35 MNOK:"
- 1. Forhandle med leverandører → Reduser pris, kvalitet eller scope
- 2. Forhandle med eier (Anne-Lise) → Øk budsjett eller reduser scope → Tid kan IKKE forlenges (red text)

Section 3: AI Agents (4 cards)

---

3. Technical Architecture

3.1 Framework Choice: Next.js 14+ (App Router)

Why Next.js:

- Server-side rendering for better SEO and performance
- API routes eliminate need for separate backend
- Server actions for mutations
- Built-in authentication patterns with Supabase
- TypeScript support out of the box
- Optimized image and font loading

Directory Structure:

```
frontend/
├── app/
│   ├── (auth)/
│   │   ├── login/
│   │   │   └── page.tsx
│   │   ├── register/
│   │   │   └── page.tsx
│   │   └── layout.tsx
│   ├── dashboard/
│   │   └── page.tsx
│   ├── game/
│   │   ├── [sessionId]/
│   │   │   ├── chat/
│   │   │   │   └── page.tsx
│   │   │   ├── visualizations/
│   │   │   │   ├── gantt/
│   │   │   │   │   └── page.tsx
│   │   │   │   ├── precedence/
│   │   │   │   │   └── page.tsx
│   │   │   │   └── history/
│   │   │   │       └── page.tsx
│   │   │   └── page.tsx
│   │   └── new/
│   │       └── page.tsx
│   ├── api/
│   │   ├── sessions/
│   │   │   ├── route.ts
│   │   │   └── [id]/
│   │   │       ├── route.ts
│   │   │       ├── commitments/
│   │   │       │   └── route.ts
```

```
│   │   │       └── agent-status/
│   │   │           └── route.ts
│   │   ├── chat/
│   │   │   └── route.ts
│   │   └── validate/
│   │       └── route.ts
│   ├── layout.tsx
│   └── page.tsx
├── components/
│   ├── ui/
│   │   ├── button.tsx
│   │   ├── card.tsx
│   │   ├── modal.tsx
│   │   └── progress-bar.tsx
│   ├── budget/
│   │   ├── three-tier-budget.tsx
│   │   ├── budget-impact-preview.tsx
│   │   └── budget-challenge-card.tsx
│   ├── wbs/
│   │   ├── wbs-item-card.tsx
│   │   ├── wbs-list.tsx
│   │   └── commitment-modal.tsx
│   ├── chat/
│   │   ├── chat-interface.tsx
│   │   ├── message-bubble.tsx
│   │   ├── offer-box.tsx
│   │   └── accept-reject-buttons.tsx
│   ├── agents/
│   │   ├── agent-card.tsx
│   │   ├── agent-list.tsx
│   │   └── agent-toggle.tsx
│   ├── visualizations/
│   │   ├── gantt-chart.tsx
│   │   ├── precedence-diagram.tsx
│   │   └── history-timeline.tsx
│   └── layout/
│       ├── navbar.tsx
│       ├── sidebar.tsx
│       └── footer.tsx
├── lib/
│   ├── supabase/
│   │   ├── client.ts
│   │   ├── server.ts
│   │   └── middleware.ts
│   ├── gemini/
│   │   ├── client.ts
│   │   ├── prompts.ts
│   │   └── types.ts
│   ├── api/
│   │   ├── sessions.ts
│   │   ├── chat.ts
│   │   ├── commitments.ts
│   │   └── validation.ts
│   ├── design-system.ts
│   ├── types.ts
│   ├── utils.ts
│   └── constants.ts
```

```
├── public/
│   ├── images/
│   └── fonts/
├── middleware.ts
├── tailwind.config.ts
├── tsconfig.json
└── package.json
```

3.2 Database Schema (Supabase PostgreSQL)

4 Main Tables:

1. **game_sessions** - User game state
2. **wbs_commitments** - WBS package commitments
3. **negotiation_history** - Chat messages with AI
4. **agent_timeouts** - Agent lock state for timeout mechanic

Key Features:

- Row-level security (RLS) for data isolation
- Computed columns for budget percentages
- Constraints for budget validation
- Triggers for automatic timestamp updates
- Indexes for performance

3.3 AI Integration (Google Gemini)

Model: `gemini-1.5-pro` or `gemini-1.5-flash`

4 System Prompts (Loaded from `docs/AI_AGENT_SYSTEM_PROMPTS.md`):

1. **Anne-Lise Berg (Owner):**
    - Role: Municipal project owner
    - Can: Approve budget increases (max 15%), approve scope reduction
    - CANNOT: Extend timeline (100% rejection rate)
    - Personality: Professional, cautious, budget-conscious
2. **Bjørn Eriksen (Supplier 1 - Grunnarbeid):**
    - Role: Site preparation contractor

- Negotiation focus: Price vs. quality trade-offs
- Initial margin: 1.20
- Min acceptable: 88% of baseline (92 MNOK)
- Patience: 3 rounds

3. **Kari Andersen (Supplier 2 - Fundamentering):**

   - Role: Foundation contractor
   - Negotiation focus: Time vs. cost trade-offs
   - Initial margin: 1.18
   - Faster delivery = +30% cost
   - Patience: 4 rounds

4. **Per Johansen (Supplier 3 - Råbygg):**

   - Role: Structural shell contractor
   - Negotiation focus: Scope reduction options
   - Initial margin: 1.25
   - Scope reductions: 10-18 MNOK per feature
   - Patience: 3 rounds

API Call Flow:

```
User message → Frontend
  ↓
API Route (/api/chat) → Extract JWT → Validate user
  ↓
Load agent system prompt + conversation history
  ↓
Call Gemini API with full context
  ↓
Parse response for offers, disagreements, timeout triggers
  ↓
Save to negotiation_history table
  ↓
Check disagreement count → Trigger timeout if needed
  ↓
Return response to frontend
```

---

4. Implementation Phases (Detailed)

PHASE 1: Foundation & Database (Day 1, 4 hours)

1.1 Database Setup (30 min)

Action: Run SQL migration in Supabase Dashboard

1. Copy complete SQL schema from Section 3.2
2. Go to Supabase Dashboard → SQL Editor
3. Paste and execute
4. Verify 4 tables created with RLS policies

Test:

```
SELECT table_name FROM information_schema.tables
WHERE table_schema = 'public';
```

Expected: `game_sessions`, `wbs_commitments`, `negotiation_history`, `agent_timeouts`

1.2 Next.js Project Setup (30 min)

```
# Initialize Next.js project
cd frontend
npx create-next-app@latest . --typescript --tailwind --app

# Install dependencies
npm install @supabase/supabase-js @supabase/ssr
npm install @google/generative-ai
npm install @radix-ui/react-dialog @radix-ui/react-toggle
npm install date-fns recharts lucide-react

# Install dev dependencies
npm install -D @types/node
```

Create `.env.local`:

```
NEXT_PUBLIC_SUPABASE_URL=https://cmntglldaqrekloixxoc.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_anon_key
SUPABASE_SERVICE_ROLE_KEY=your_service_role_key
```

```
GEMINI_API_KEY=your_gemini_key
GEMINI_MODEL=gemini-1.5-pro
```

1.3 Design System Setup (60 min)

File: `lib/design-system.ts`

```ts
export const colors = {
  primary: { blue: '#3b82f6', blueBg: '#eff6ff', /* ... */ },
  success: { green: '#10b981', greenBg: '#f0fdf4', /* ... */ },
  warning: { yellow: '#f59e0b', yellowBg: '#fef3c7', /* ... */ },
  danger: { red: '#ef4444', redBg: '#fee2e2', /* ... */ },
  neutral: { gray900: '#111827', /* ... */ },
};

export const typography = {
  title: 'text-xl font-bold text-gray-900',
  heading: 'text-base font-bold text-gray-900',
  text: 'text-[13px] font-medium text-gray-700',
  desc: 'text-xs text-gray-600',
  small: 'text-[10px] text-gray-500',
};
```

File: `tailwind.config.ts`

```ts
import type { Config } from 'tailwindcss';

const config: Config = {
  content: [
    './pages/**/*.{js,ts,jsx,tsx,mdx}',
    './components/**/*.{js,ts,jsx,tsx,mdx}',
    './app/**/*.{js,ts,jsx,tsx,mdx}',
  ],
  theme: {
    extend: {
      colors: {
        // Match mockup colors exactly
        primary: '#3b82f6',
        success: '#10b981',
        warning: '#f59e0b',
        danger: '#ef4444',
      },
      fontFamily: {
        sans: ['Inter', 'system-ui', 'sans-serif'],
      },
    },
  },
  plugins: [],
};
```

```
export default config;
```

1.4 Supabase Integration (60 min)

File: `lib/supabase/client.ts`

```
import { createBrowserClient } from '@supabase/ssr';

export function createClient() {
  return createBrowserClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
  );
}
```

File: `lib/supabase/server.ts`

```
import { createServerClient, type CookieOptions } from '@supabase/ssr';
import { cookies } from 'next/headers';

export async function createClient() {
  const cookieStore = cookies();

  return createServerClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
    {
      cookies: {
        get(name: string) {
          return cookieStore.get(name)?.value;
        },
      },
    }
  );
}
```

File: `middleware.ts`

```
import { createServerClient, type CookieOptions } from '@supabase/ssr';
import { NextResponse, type NextRequest } from 'next/server';

export async function middleware(request: NextRequest) {
  let response = NextResponse.next({
    request: { headers: request.headers },
  });
```

```
  const supabase = createServerClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
    {
      cookies: {
        get(name: string) {
          return request.cookies.get(name)?.value;
        },
        set(name: string, value: string, options: CookieOptions) {
          request.cookies.set({ name, value, ...options });
          response = NextResponse.next({
            request: { headers: request.headers },
          });
          response.cookies.set({ name, value, ...options });
        },
        remove(name: string, options: CookieOptions) {
          request.cookies.set({ name, value: '', ...options });
          response = NextResponse.next({
            request: { headers: request.headers },
          });
          response.cookies.set({ name, value: '', ...options });
        },
      },
    }
  );

  await supabase.auth.getUser();

  return response;
}

export const config = {
  matcher: [
    '/((?!_next/static|_next/image|favicon.ico|.*\\.
(?:svg|png|jpg|jpeg|gif|webp)$).*)',
  ],
};
```

1.5 Gemini Integration (60 min)

File: `lib/gemini/client.ts`

```
import { GoogleGenerativeAI } from '@google/generative-ai';

const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY!);

export async function chatWithAgent(params: {
  agentId: string;
  systemPrompt: string;
  conversationHistory: { role: string; content: string }[];
  userMessage: string;
  gameContext?: Record<string, any>;
}) {
```

```
  const model = genAI.getGenerativeModel({
    model: process.env.GEMINI_MODEL || 'gemini-1.5-pro',
  });

  // Build full prompt
  const fullPrompt = buildPrompt(params);

  const result = await model.generateContent(fullPrompt);
  const response = await result.response;
  return response.text();
}

function buildPrompt(params: any) {
  const parts = [
    '# System Prompt (Din rolle)',
    params.systemPrompt,
    '',
  ];

  if (params.gameContext) {
    parts.push('# Nåværende spillstatus');
    parts.push(`Budsjett brukt: ${params.gameContext.budget_used} /
${params.gameContext.total_budget} NOK`);
    parts.push('');
  }

  if (params.conversationHistory.length > 0) {
    parts.push('# Tidligere samtale');
    params.conversationHistory.slice(-10).forEach((msg: any) => {
      const role = msg.role === 'user' ? 'Bruker' : 'Deg';
      parts.push(`${role}: ${msg.content}`);
    });
    parts.push('');
  }

  parts.push('# Ny melding fra prosjektlederen');
  parts.push(`Bruker: ${params.userMessage}`);
  parts.push('');
  parts.push('Svar naturlig som din rolle basert på spillkonteksten over:');

  return parts.join('\n');
}
```

File: `lib/gemini/prompts.ts`

```
// Load from docs/AI_AGENT_SYSTEM_PROMPTS.md
export const AGENT_PROMPTS = {
  'anne-lise-berg': `
# ROLE
You are Kommunaldirektør Anne-Lise Berg...
[Complete prompt from documentation]
`,
  'bjorn-eriksen': `...`,
  'kari-andersen': `...`,
```

```
  'per-johansen': `...`,
};

export function getAgentPrompt(agentId: string): string {
  return AGENT_PROMPTS[agentId as keyof typeof AGENT_PROMPTS] || '';
}
```

PHASE 2: Core UI Components (Day 2-3, 10 hours)

2.1 Login Page (60 min)

File: `app/(auth)/login/page.tsx`

```
'use client';

import { useState } from 'react';
import { createClient } from '@/lib/supabase/client';
import { useRouter } from 'next/navigation';

export default function LoginPage() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const router = useRouter();
  const supabase = createClient();

  const handleLogin = async (e: React.FormEvent) => {
    e.preventDefault();
    setError('');

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

    if (error) {
      setError(error.message);
    } else {
      router.push('/dashboard');
    }
  };

  return (
    <div className="min-h-screen bg-gray-50 flex items-center justify-center">
      <div className="w-[400px] bg-white border border-gray-400 rounded-xl shadow-
lg p-8">
        {/* Logo */}
        <div className="flex justify-center mb-6">
```

```
          <div className="w-16 h-16 bg-primary rounded-full flex items-center
justify-center">
            <span className="text-white text-2xl font-bold">NHB</span>
          </div>
        </div>

        {/* Title */}
        <h1 className="text-center text-2xl font-bold text-gray-900 mb-2">
          Nye Hædda Barneskole
        </h1>
        <p className="text-center text-sm text-gray-600 mb-8">
          Prosjektledelses-simulator
        </p>

        {/* Form */}
        <form onSubmit={handleLogin} className="space-y-4">
          <div>
            <label className="block text-sm font-semibold text-gray-700 mb-1">
              E-postadresse
            </label>
            <input
              type="email"
              value={email}
              onChange={(e) => setEmail(e.target.value)}
              placeholder="din.epost@eksempel.no"
              className="w-full px-4 py-3 border border-gray-400 rounded-md text-
sm"
              required
            />
          </div>

          <div>
            <label className="block text-sm font-semibold text-gray-700 mb-1">
              Passord
            </label>
            <input
              type="password"
              value={password}
              onChange={(e) => setPassword(e.target.value)}
              placeholder="••••••••"
              className="w-full px-4 py-3 border border-gray-400 rounded-md text-
sm"
              required
            />
          </div>

          {error && (
            <p className="text-sm text-red-600">{error}</p>
          )}

          <button
            type="submit"
            className="w-full bg-primary text-white py-3 rounded-md font-semibold
hover:bg-blue-600"
          >
            Logg inn
          </button>
```

```
        </form>

        {/* Divider */}
        <div className="relative my-6">
          <div className="absolute inset-0 flex items-center">
            <div className="w-full border-t border-gray-400"></div>
          </div>
          <div className="relative flex justify-center text-sm">
            <span className="px-4 bg-white text-gray-600">eller</span>
          </div>
        </div>

        {/* Register Link */}
        <p className="text-center text-sm text-gray-600">
          Har du ikke en konto?{' '}
          <a href="/register" className="text-primary font-semibold">
            Registrer deg her →
          </a>
        </p>

        {/* Footer */}
        <div className="mt-6 p-4 bg-blue-50 border border-primary rounded-md">
          <p className="text-[11px] text-gray-600 text-center">
            🔒 Sikret med Supabase Authentication
          </p>
          <p className="text-[11px] text-gray-600 text-center">
            Budsjett: 390 MNOK låst + 310 MNOK tilgjengelig
          </p>
        </div>
      </div>
    </div>
  );
}
```

2.2 Three-Tier Budget Component (90 min)

File: components/budget/three-tier-budget.tsx

```
'use client';

interface BudgetData {
  tier1_used: number;
  tier1_available: number;
  tier2_locked: number;
  tier3_total: number;
}

export function ThreeTierBudget({ budget }: { budget: BudgetData }) {
  const tier1Percentage = (budget.tier1_used / budget.tier1_available) * 100;
  const tier3Used = budget.tier2_locked + budget.tier1_used;
  const tier3Percentage = (tier3Used / budget.tier3_total) * 100;
  const tier1Remaining = budget.tier1_available - budget.tier1_used;
```

```
  const getProgressColor = (percentage: number) => {
    if (percentage > 100) return '#ef4444';
    if (percentage > 90) return '#f59e0b';
    if (percentage > 70) return '#f59e0b';
    return '#10b981';
  };

  return (
    <div className="space-y-4">
      {/* Header */}
      <div>
        <h2 className="text-base font-bold text-gray-900">Prosjektbudsjett</h2>
        <p className="text-xs text-gray-600">
          3 forhandlbare pakker | 12 låste pakker (390 MNOK)
        </p>
      </div>

      {/* TIER 1: Tilgjengelig */}
      <div className="p-4 bg-gray-50 border border-gray-400 rounded-md">
        <div className="flex justify-between mb-2">
          <span className="text-[13px] font-medium text-gray-700">
            Tilgjengelig budsjett (3 forhandlbare pakker)
          </span>
          <span className="text-lg font-bold text-gray-900">
            {Math.round(budget.tier1_used / 1000000)} /{' '}
            {Math.round(budget.tier1_available / 1000000)} MNOK
          </span>
        </div>

        {/* Progress bar */}
        <div className="w-full bg-gray-300 rounded-full h-5 mb-1">
          <div
            className="h-5 rounded-full transition-all"
            style={{
              width: `${Math.min(tier1Percentage, 100)}%`,
              backgroundColor: getProgressColor(tier1Percentage),
            }}
          />
        </div>

        <p className="text-[10px] text-gray-500 text-center">
          {tier1Percentage.toFixed(0)}% brukt -{' '}
          {Math.round(tier1Remaining / 1000000)} MNOK gjenstår
        </p>
      </div>

      {/* TIER 2: Låst */}
      <div className="p-4 bg-gray-200 border border-gray-400 rounded-md">
        <div className="flex justify-between mb-1">
          <span className="text-[13px] font-medium text-gray-700">
            Låst budsjett (12 kontraktfestede pakker)
          </span>
          <span className="text-[13px] font-medium text-gray-700">
            {Math.round(budget.tier2_locked / 1000000)} MNOK
          </span>
        </div>
        <p className="text-xs text-gray-600">
```

```
          Dette budsjettet er allerede forpliktet og kan ikke endres
        </p>
      </div>

      {/* TIER 3: Totalt */}
      <div className="p-4 bg-gray-50 border border-gray-400 rounded-md">
        <div className="flex justify-between mb-1">
          <span className="text-[13px] font-medium text-gray-700">
            Totalt budsjett
          </span>
          <span className="text-lg font-bold text-gray-900">
            {Math.round(tier3Used / 1000000)} /{' '}
            {Math.round(budget.tier3_total / 1000000)} MNOK
          </span>
        </div>
        <p className="text-xs text-gray-600 text-right">
          {tier3Percentage.toFixed(0)}% brukt |{' '}
          {Math.round((budget.tier3_total - tier3Used) / 1000000)} MNOK gjenstår ✓
        </p>
      </div>

      {/* Warning Banner */}
      <div className="p-4 bg-yellow-100 border-2 border-yellow-500 rounded-md">
        <p className="text-[13px] font-bold text-yellow-900 mb-1">
          ⚠ VIKTIG: Budsjettutfordring
        </p>
        <p className="text-xs text-yellow-900">
          3 forhandlbare pakker estimert til 345 MNOK (105+60+180)
        </p>
        <p className="text-xs text-yellow-900">
          Tilgjengelig budsjett: 310 MNOK → Underskudd: 35 MNOK
        </p>
      </div>
    </div>
  );
}
```

2.3 WBS Items List (120 min)

File: `components/wbs/wbs-item-card.tsx`

```
'use client';

interface WBSItem {
  id: string;
  code: string;
  name: string;
  supplier: string;
  duration: string;
  baseline: number;
  negotiable: boolean;
  status: 'pending' | 'negotiating' | 'committed';
}
```

```
export function WBSItemCard({
  item,
  onContactClick,
}: {
  item: WBSItem;
  onContactClick?: () => void;
}) {
  if (!item.negotiable) {
    // Locked item (gray)
    return (
      <div className="p-4 bg-gray-200 border border-gray-400 rounded-md opacity-
70">
        <div className="flex items-center gap-2">
          <div className="w-1 h-12 bg-gray-500 rounded" />
          <div className="flex-1">
            <p className="text-[13px] font-medium text-gray-600">
              🔒 12 låste pakker (kontraktfestet)
            </p>
            <p className="text-xs text-gray-600 mt-1">
              Eksempler: Prosjektering (30 MNOK), RIB (25 MNOK), Elektrisk (35
MNOK)...
            </p>
            <p className="text-[10px] text-gray-500 mt-1">
              Totalt 390 MNOK fordelt på 12 pakker som allerede er forhandlet
            </p>
          </div>
        </div>
      </div>
    );
  }

  // Negotiable item (blue)
  const statusIcons = {
    pending: '🟣',
    negotiating: '🟡',
    committed: '🟢',
  };

  const statusText = {
    pending: 'Venter',
    negotiating: 'Under forhandling',
    committed: 'Forpliktet',
  };

  return (
    <div className="p-4 bg-blue-50 border-2 border-primary rounded-md relative">
      {/* Left accent */}
      <div className="absolute left-0 top-0 bottom-0 w-1 bg-primary rounded-l" />

      <div className="flex items-start justify-between">
        <div className="flex items-start gap-3">
          {/* Dot indicator */}
          <div className="w-4 h-4 bg-primary rounded-full mt-1" />

          <div>
            <h3 className="text-[13px] font-bold text-gray-900">
```

```
              {item.code} {item.name}
            </h3>

            {/* Badge */}
            <div className="inline-block mt-1 px-3 py-1 bg-blue-200 border border-
primary rounded-full">
              <span className="text-[10px] font-semibold text-blue-900">
                💬 Kan forhandles
              </span>
            </div>

            <p className="text-xs text-gray-600 mt-2">
              Leverandør: {item.supplier} | Varighet: {item.duration}
            </p>
            <p className="text-xs text-gray-600">
              Baseline: {Math.round(item.baseline / 1000000)} MNOK
            </p>
          </div>
        </div>

        <div className="flex flex-col items-end gap-2">
          {/* Status */}
          <span className="text-[13px] font-medium text-gray-700">
            {statusIcons[item.status]} {statusText[item.status]}
          </span>

          {/* Contact button */}
          {item.status !== 'committed' && (
            <button
              onClick={onContactClick}
              className="px-4 py-2 bg-primary text-white text-[13px] font-semibold
rounded hover:bg-blue-600"
            >
              Kontakt
            </button>
          )}
        </div>
      </div>
    </div>
  );
}
```

File: components/wbs/wbs-list.tsx

```
'use client';

import { WBSItemCard } from './wbs-item-card';

export function WBSList() {
  const negotiableItems = [
    {
      id: '1',
      code: '1.3.1',
      name: 'Grunnarbeid',
```

```
    supplier: 'Bjørn Eriksen',
    duration: '3.5 mnd',
    baseline: 105000000,
    negotiable: true,
    status: 'pending' as const,
  },
  {
    id: '2',
    code: '1.3.2',
    name: 'Fundamentering',
    supplier: 'Kari Andersen',
    duration: '2.5 mnd',
    baseline: 60000000,
    negotiable: true,
    status: 'pending' as const,
  },
  {
    id: '3',
    code: '1.4.1',
    name: 'Råbygg',
    supplier: 'Per Johansen',
    duration: '4.0 mnd',
    baseline: 180000000,
    negotiable: true,
    status: 'pending' as const,
  },
];

const lockedPreview = {
  id: 'locked',
  code: '',
  name: '',
  supplier: '',
  duration: '',
  baseline: 390000000,
  negotiable: false,
  status: 'committed' as const,
};

return (
  <div className="space-y-4">
    <div>
      <h2 className="text-base font-bold text-gray-900">
        WBS-pakker (Work Breakdown Structure)
      </h2>
      <p className="text-xs text-gray-600">
        3 forhandlbare (blå) + 12 låste (grå)
      </p>
    </div>

    {negotiableItems.map((item) => (
      <WBSItemCard
        key={item.id}
        item={item}
        onContactClick={() => {
          window.location.href = `/game/chat?wbs=${item.code}`;
        }}
```

```
        />
      ))}

      {/* Locked items preview */}
      <WBSItemCard item={lockedPreview} />
    </div>
  );
}
```

2.4 Chat Interface (180 min)

File: `components/chat/chat-interface.tsx`

```
'use client';

import { useState, useRef, useEffect } from 'react';
import { MessageBubble } from './message-bubble';
import { OfferBox } from './offer-box';
import { AcceptRejectButtons } from './accept-reject-buttons';

interface Message {
  id: string;
  role: 'user' | 'assistant';
  content: string;
  timestamp: string;
}

interface Offer {
  wbsId: string;
  cost: number;
  duration: number;
  description: string;
}

export function ChatInterface({ agentId, wbsId }: { agentId: string; wbsId: string
}) {
  const [messages, setMessages] = useState<Message[]>([]);
  const [input, setInput] = useState('');
  const [isLoading, setIsLoading] = useState(false);
  const [currentOffer, setCurrentOffer] = useState<Offer | null>(null);
  const messagesEndRef = useRef<HTMLDivElement>(null);

  const scrollToBottom = () => {
    messagesEndRef.current?.scrollIntoView({ behavior: 'smooth' });
  };

  useEffect(scrollToBottom, [messages]);

  const handleSendMessage = async () => {
    if (!input.trim() || currentOffer) return;

    const userMessage: Message = {
      id: Date.now().toString(),
```

```
      role: 'user',
      content: input.trim(),
      timestamp: new Date().toISOString(),
    };

    setMessages((prev) => [...prev, userMessage]);
    setInput('');
    setIsLoading(true);

    try {
      // Call backend API
      const response = await fetch('/api/chat', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          agentId,
          wbsId,
          message: userMessage.content,
          conversationHistory: messages,
        }),
      });

      const data = await response.json();

      const aiMessage: Message = {
        id: (Date.now() + 1).toString(),
        role: 'assistant',
        content: data.response,
        timestamp: new Date().toISOString(),
      };

      setMessages((prev) => [...prev, aiMessage]);

      // Check if response contains an offer
      if (data.offer) {
        setCurrentOffer(data.offer);
      }
    } catch (error) {
      console.error('Chat error:', error);
    } finally {
      setIsLoading(false);
    }
  };

  const handleAcceptOffer = async () => {
    if (!currentOffer) return;

    // Call commitment API
    await fetch(`/api/sessions/${sessionId}/commitments`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({
        wbsId: currentOffer.wbsId,
        cost: currentOffer.cost,
        duration: currentOffer.duration,
      }),
    });
```

```
    // Redirect to dashboard
    window.location.href = '/dashboard';
  };

  const handleRejectOffer = () => {
    setCurrentOffer(null);
  };

  return (
    <div className="flex flex-col h-full">
      {/* Messages */}
      <div className="flex-1 overflow-y-auto p-4 space-y-4">
        {messages.map((message) => (
          <MessageBubble key={message.id} message={message} />
        ))}

        {currentOffer && (
          <>
            <OfferBox offer={currentOffer} />
            <AcceptRejectButtons
              onAccept={handleAcceptOffer}
              onReject={handleRejectOffer}
              offer={currentOffer}
            />
          </>
        )}

        {isLoading && (
          <div className="text-center text-sm text-gray-600">
            AI skriver...
          </div>
        )}

        <div ref={messagesEndRef} />
      </div>

      {/* Input */}
      <div className="p-4 border-t">
        <div className="flex gap-2">
          <input
            type="text"
            value={input}
            onChange={(e) => setInput(e.target.value)}
            onKeyPress={(e) => e.key === 'Enter' && handleSendMessage()}
            placeholder={
              currentOffer
                ? 'Du må godta eller avslå tilbudet...'
                : 'Skriv en melding...'
            }
            disabled={isLoading || !!currentOffer}
            className="flex-1 px-4 py-2 border border-gray-400 rounded-md text-sm"
          />
          <button
            onClick={handleSendMessage}
            disabled={isLoading || !!currentOffer || !input.trim()}
            className="px-6 py-2 bg-primary text-white rounded-md font-semibold
```

```
disabled:opacity-50"
          >
            Send
          </button>
        </div>
      </div>
    </div>
  );
}
```

File: components/chat/accept-reject-buttons.tsx

```
'use client';

interface AcceptRejectButtonsProps {
  onAccept: () => void;
  onReject: () => void;
  offer: {
    cost: number;
    duration: number;
  };
}

export function AcceptRejectButtons({
  onAccept,
  onReject,
  offer,
}: AcceptRejectButtonsProps) {
  return (
    <div className="space-y-2">
      <div className="flex gap-4">
        {/* Accept button (green, large) */}
        <button
          onClick={onAccept}
          className="flex-1 max-w-[280px] h-[50px] bg-success text-white text-
[15px] font-semibold rounded-md hover:bg-green-600 transition"
        >
          ✓ Godta tilbud: {Math.round(offer.cost / 1000000)} MNOK, {offer.duration}
mnd
        </button>

        {/* Reject button (gray, smaller) */}
        <button
          onClick={onReject}
          className="w-[180px] h-[50px] bg-gray-200 text-gray-700 text-[13px] font-
semibold rounded-md border border-gray-400 hover:bg-gray-300 transition"
        >
          ✗ Avslå og reforhandle
        </button>
      </div>

      <p className="text-xs text-success">
        Ved å godta, oppdateres budsjettet ditt automatisk
      </p>
```

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

---

5. Testing Strategy

5.1 POC Testing (2-3 hours)

Goal: Validate core functionality works end-to-end

Test Scenarios:

1. **Authentication & Session**
    - Log in → Dashboard loads
    - Session created in database
    - Budget initialized correctly
2. **Dashboard Display**
    - 3-tier budget shows 0/310/390/700
    - Warning banner displays
    - 3 blue negotiable items
    - 1 gray locked preview
3. **Chat with AI**
    - Click "Kontakt" → Chat opens
    - Send message → AI responds in Norwegian
    - Offer appears in green box
4. **Explicit Accept/Reject**
    - Accept button shows offer details
    - Click accept → Confirmation modal
    - Budget updates after confirm
    - No automatic acceptance
5. **Owner Time Rejection**

    - Switch to "Eier" mode
    - Request time extension
    - Verify 100% rejection

Success Criteria:

- All 5 scenarios pass
- No console errors
- Budget calculations accurate
- AI responses in Norwegian

5.2 Functional Testing (6-8 hours)

Complete Test Suite:

1. **Authentication**
    - Sign up, login, logout, session persistence
2. **Budget Management**
    - All budget tiers update correctly
    - Cannot exceed limits
    - Progress bars accurate
    - Color changes at thresholds
3. **AI Negotiation Quality**
    - Test all 4 agents
    - Verify personalities match
    - Check hidden parameters (concession rates, patience)
    - Owner never extends time (100% rejection)
4. **Timeout Mechanic**
    - 6 disagreements trigger timeout
    - Countdown timer works
    - Agent unlocks after 10 minutes
5. **Visualizations**
    - Gantt chart displays correctly

- Precedence diagram shows dependencies
- History view shows before/after

6. **Error Handling**
    - Budget exceeded errors
    - Timeline exceeded errors
    - Network errors
7. **Performance**
    - Page load <2 seconds
    - AI response 1-3 seconds
    - Lighthouse score >85

5.3 UAT (User Acceptance Testing)

Participants: 5-10 Norwegian engineering students

Process:

1. Provide access to deployed POC
2. Ask users to complete full game (45-60 min)
3. Survey after completion

Survey Questions:

- Did the AI agents feel realistic? (1-5 scale)
- Was the budget challenge clear? (1-5 scale)
- Did you understand the 3-tier budget model? (Y/N)
- Were the explicit accept/reject buttons helpful? (Y/N)
- How satisfied are you with the experience? (1-5 scale)

Success Metrics:

- 80%+ satisfaction score
- 90%+ understood budget model
- 100% found accept/reject buttons helpful

---

6. Quality Assurance Checklist

Visual QA (Match Mockups)

Login Page:

- ☐ Logo centered (NHB in circle)
- ☐ Email/password fields styled correctly
- ☐ "Logg inn" button blue (#3b82f6)
- ☐ Footer with budget info

Dashboard:

- ☐ 3-tier budget matches mockup exactly
- ☐ Progress bars accurate
- ☐ Warning banner yellow with border
- ☐ 3 negotiable items blue
- ☐ "💬 Kan forhandles" badge correct
- ☐ 12 locked preview gray
- ☐ Right sidebar with 3 sections

Chat:

- ☐ Chat header with avatar
- ☐ Toggle "Leverandør / Eier" works
- ☐ User messages right, AI left
- ☐ Offer box green
- ☐ Accept button green, large
- ☐ Reject button gray, smaller
- ☐ Right sidebar shows budget impact

Functional QA

- ☐ Authentication works (sign up, login, logout)
- ☐ 3-tier budget updates correctly
- ☐ WBS items display properly (3 negotiable + 12 locked)
- ☐ Chat with all 4 agents works
- ☐ Explicit accept/reject flow works (no auto-acceptance)

- ☐ Owner NEVER extends time
- ☐ Timeout mechanic after 6 disagreements
- ☐ Visualizations render correctly
- ☐ Error messages in Norwegian

Accessibility (WCAG AA)

- ☐ Keyboard navigation works
- ☐ Screen reader compatible
- ☐ Color contrast ratios meet 4.5:1
- ☐ Focus indicators visible
- ☐ Form labels associated

Security

- ☐ JWT not in localStorage
- ☐ RLS policies prevent data leaks
- ☐ Input validation on all forms
- ☐ No XSS vulnerabilities
- ☐ Rate limiting on Gemini API

---

7. Deployment Guide

7.1 Pre-Deployment

- ☐ All tests passed
- ☐ Code reviewed
- ☐ No console errors in production build
- ☐ Environment variables configured
- ☐ Database migrations run
- ☐ RLS policies enabled
- ☐ Gemini API quota sufficient

7.2 Deploy to Vercel

```
# Install Vercel CLI
npm install -g vercel

# Login
vercel login

# Link project
cd frontend
vercel link

# Set environment variables
vercel env add NEXT_PUBLIC_SUPABASE_URL
vercel env add NEXT_PUBLIC_SUPABASE_ANON_KEY
vercel env add GEMINI_API_KEY

# Deploy
vercel --prod
```

7.3 Post-Deployment

1. Visit production URL
2. Run POC tests in production
3. Verify Supabase connection
4. Verify Gemini API works
5. Run Lighthouse audit
6. Test from mobile
7. Monitor error logs for 24 hours

---

8. Timeline & Resources

Time Estimates

- **Phase 1:** Foundation (Day 1, 4 hours)
- **Phase 2:** Core UI (Day 2-3, 10 hours)
- **Phase 3:** Game Mechanics (Day 4, 6 hours)
- **Phase 4:** Visualizations (Day 5, 6 hours)
- **Phase 5:** Testing & Polish (Day 6, 8 hours)

Total: ~34 hours over 6 days

Testing Time

- POC Testing: 2-3 hours
- Functional Testing: 6-8 hours
- UAT: 4-6 hours

Total Testing: 12-17 hours

Resources Needed

- **Developer:** 1 full-stack developer
- **Designer:** UX mockups already complete
- **Testers:** 5-10 students for UAT
- **Infrastructure:**
  - Supabase (free tier sufficient for POC)
  - Gemini API quota (~$20-50 for testing)
  - Vercel (free tier)

Appendix A: Converting to PDF

To convert this markdown file to PDF:

Option 1: VS Code + Markdown PDF Extension

1. Install "Markdown PDF" extension in VS Code
2. Open this file
3. Press Ctrl+Shift+P → "Markdown PDF: Export (pdf)"

Option 2: Pandoc (Command Line)

```
pandoc COMPLETE_IMPLEMENTATION_PLAN_V3.md -o implementation-plan.pdf
```

Option 3: Online Converter

- https://www.markdowntopdf.com/
- Upload this file and download PDF

Document Version: 3.0 **Status:** Ready for Review **Next Steps:** Review → Approve → Execute Implementation