Contents

SECTION 19: CONCURRENT CHAT & SPLIT-PANE UI (v3.6.0)

19.1 Concurrent Chat Overview

Industry-first feature allowing users to run multiple AI conversations simultaneously with split-pane interface.

19.2 Concurrent Chat Database Schema

```sql
-- migrations/028_concurrent_chat.sql

CREATE TABLE concurrent_sessions (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),
    session_name VARCHAR(100),
    layout_config JSONB NOT NULL DEFAULT '{"type": "horizontal", "panes": []}',
    max_panes INTEGER DEFAULT 4,
    is_active BOOLEAN DEFAULT true,
    created_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE concurrent_panes (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES concurrent_sessions(id) ON DELETE CASCADE,
    pane_index INTEGER NOT NULL,
    chat_id UUID REFERENCES chats(id),
    model VARCHAR(100),
    status VARCHAR(20) DEFAULT 'idle',
    last_activity TIMESTAMPTZ DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(session_id, pane_index)
);
```

```sql
CREATE TABLE concurrent_sync_state (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    session_id UUID NOT NULL REFERENCES concurrent_sessions(id) ON DELETE CASCADE,
    sync_mode VARCHAR(20) NOT NULL DEFAULT 'independent',
    shared_context TEXT,
    sync_cursor INTEGER DEFAULT 0,
    updated_at TIMESTAMPTZ NOT NULL DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_concurrent_sessions_user ON concurrent_sessions(tenant_id, user_id);
CREATE INDEX idx_concurrent_panes_session ON concurrent_panes(session_id);

ALTER TABLE concurrent_sessions ENABLE ROW LEVEL SECURITY;
ALTER TABLE concurrent_panes ENABLE ROW LEVEL SECURITY;
ALTER TABLE concurrent_sync_state ENABLE ROW LEVEL SECURITY;

CREATE POLICY concurrent_sessions_isolation ON concurrent_sessions USING (tenant_id = current_s
CREATE POLICY concurrent_panes_isolation ON concurrent_panes USING (
    session_id IN (SELECT id FROM concurrent_sessions WHERE tenant_id = current_setting('app.cu
);
CREATE POLICY concurrent_sync_isolation ON concurrent_sync_state USING (
    session_id IN (SELECT id FROM concurrent_sessions WHERE tenant_id = current_setting('app.cu
);
```

19.3 Concurrent Session Manager

```typescript
// packages/core/src/services/concurrent-session-manager.ts

import { Pool } from 'pg';

interface LayoutConfig {
    type: 'horizontal' | 'vertical' | 'grid';
    panes: PaneConfig[];
}

interface PaneConfig {
    id: string;
    size: number;
    model?: string;
    chatId?: string;
}

export class ConcurrentSessionManager {
    private pool: Pool;

    constructor(pool: Pool) {
        this.pool = pool;
```

```
}

async createSession(
    tenantId: string,
    userId: string,
    name?: string,
    initialPanes: number = 2
): Promise<string> {
    const layout: LayoutConfig = {
        type: 'horizontal',
        panes: Array(initialPanes).fill(null).map((_, i) => ({
            id: `pane-${i}`,
            size: 100 / initialPanes
        }))
    };

    const result = await this.pool.query(`
        INSERT INTO concurrent_sessions (tenant_id, user_id, session_name, layout_config)
        VALUES ($1, $2, $3, $4)
        RETURNING id
    `, [tenantId, userId, name, JSON.stringify(layout)]);

    const sessionId = result.rows[0].id;

    // Create pane records
    for (let i = 0; i < initialPanes; i++) {
        await this.pool.query(`
            INSERT INTO concurrent_panes (session_id, pane_index)
            VALUES ($1, $2)
        `, [sessionId, i]);
    }

    // Create sync state
    await this.pool.query(`
        INSERT INTO concurrent_sync_state (session_id)
        VALUES ($1)
    `, [sessionId]);

    return sessionId;
}

async addPane(sessionId: string, model?: string): Promise<number> {
    const session = await this.getSession(sessionId);
    if (session.layout_config.panes.length >= session.max_panes) {
        throw new Error('Maximum panes reached');
    }

    const newIndex = session.layout_config.panes.length;
```

```
        await this.pool.query(`
            INSERT INTO concurrent_panes (session_id, pane_index, model)
            VALUES ($1, $2, $3)
        `, [sessionId, newIndex, model]);

        // Update layout
        const newLayout = {
            ...session.layout_config,
            panes: [...session.layout_config.panes, { id: `pane-${newIndex}`, size: 100 / (new
        };

        // Rebalance sizes
        const equalSize = 100 / newLayout.panes.length;
        newLayout.panes = newLayout.panes.map(p => ({ ...p, size: equalSize }));

        await this.pool.query(`
            UPDATE concurrent_sessions SET layout_config = $2, updated_at = NOW() WHERE id = $
        `, [sessionId, JSON.stringify(newLayout)]);

        return newIndex;
    }

    async removePane(sessionId: string, paneIndex: number): Promise<void> {
        await this.pool.query(`DELETE FROM concurrent_panes WHERE session_id = $1 AND pane_ind

        // Reindex remaining panes
        await this.pool.query(`
            UPDATE concurrent_panes
            SET pane_index = pane_index - 1
            WHERE session_id = $1 AND pane_index > $2
        `, [sessionId, paneIndex]);

        // Update layout
        const session = await this.getSession(sessionId);
        const newPanes = session.layout_config.panes.filter((_, i) => i !== paneIndex);
        const equalSize = 100 / newPanes.length;

        await this.pool.query(`
            UPDATE concurrent_sessions
            SET layout_config = $2, updated_at = NOW()
            WHERE id = $1
        `, [sessionId, JSON.stringify({ ...session.layout_config, panes: newPanes.map(p => ({
    }

    async updatePaneModel(sessionId: string, paneIndex: number, model: string): Promise<void> 
        await this.pool.query(`
            UPDATE concurrent_panes SET model = $3 WHERE session_id = $1 AND pane_index = $2
        `, [sessionId, paneIndex, model]);
    }

    async setSyncMode(sessionId: string, mode: 'independent' | 'synchronized' | 'broadcast'): 
        await this.pool.query(`
            UPDATE concurrent_sync_state SET sync_mode = $2, updated_at = NOW() WHERE session_
        `, [sessionId, mode]);
    }

    async getSession(sessionId: string) {
        const result = await this.pool.query(`SELECT * FROM concurrent_sessions WHERE id = $1`
        return result.rows[0];
    }

    async getPanes(sessionId: string) {
        const result = await this.pool.query(`
            SELECT * FROM concurrent_panes WHERE session_id = $1 ORDER BY pane_index
        `, [sessionId]);
        return result.rows;
    }
}
```

19.4 Split-Pane React Component

```
// apps/admin-dashboard/src/components/concurrent/SplitPane.tsx

'use client';

import React, { useState, useCallback, useRef } from 'react';
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query';
import { Card, CardContent, CardHeader, CardTitle } from '@/components/ui/card';
import { Button } from '@/components/ui/button';
import { Select, SelectContent, SelectItem, SelectTrigger, SelectValue } from '@/components/ui
import { Plus, X, ArrowLeftRight, ArrowUpDown, Grid } from 'lucide-react';

interface Pane {
    id: string;
    paneIndex: number;
    chatId?: string;
    model?: string;
    status: string;
}

interface SplitPaneProps {
    sessionId: string;
    onSendMessage: (paneIndex: number, message: string) => void;
}
```

```
export function SplitPane({ sessionId, onSendMessage }: SplitPaneProps) {
    const queryClient = useQueryClient();
    const [layout, setLayout] = useState<'horizontal' | 'vertical' | 'grid'>('horizontal');
    const [sizes, setSizes] = useState<number[]>([]);
    const containerRef = useRef<HTMLDivElement>(null);

    const { data: session } = useQuery({
        queryKey: ['concurrent-session', sessionId],
        queryFn: async () => {
            const res = await fetch(`/api/admin/concurrent/${sessionId}`);
            return res.json();
        }
    });

    const { data: panes = [] } = useQuery({
        queryKey: ['concurrent-panes', sessionId],
        queryFn: async () => {
            const res = await fetch(`/api/admin/concurrent/${sessionId}/panes`);
            return res.json();
        }
    });

    const addPaneMutation = useMutation({
        mutationFn: async () => {
            const res = await fetch(`/api/admin/concurrent/${sessionId}/panes`, { method: 'POS
            return res.json();
        },
        onSuccess: () => queryClient.invalidateQueries({ queryKey: ['concurrent-panes', sessio
    });

    const removePaneMutation = useMutation({
        mutationFn: async (paneIndex: number) => {
            await fetch(`/api/admin/concurrent/${sessionId}/panes/${paneIndex}`, { method: 'DE
        },
        onSuccess: () => queryClient.invalidateQueries({ queryKey: ['concurrent-panes', sessio
    });

    const updateModelMutation = useMutation({
        mutationFn: async ({ paneIndex, model }: { paneIndex: number; model: string }) => {
            await fetch(`/api/admin/concurrent/${sessionId}/panes/${paneIndex}`, {
                method: 'PATCH',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ model })
            });
        },
        onSuccess: () => queryClient.invalidateQueries({ queryKey: ['concurrent-panes', sessio
    });
```

```
const handleResize = useCallback((index: number, delta: number) => {
    setSizes(prev => {
        const newSizes = [...prev];
        newSizes[index] = Math.max(10, Math.min(90, newSizes[index] + delta));
        newSizes[index + 1] = Math.max(10, Math.min(90, newSizes[index + 1] - delta));
        return newSizes;
    });
}, []);

const getLayoutClasses = () => {
    switch (layout) {
        case 'vertical': return 'flex-col';
        case 'grid': return 'grid grid-cols-2';
        default: return 'flex-row';
    }
};

return (
    <div className="h-full flex flex-col">
        {/* Toolbar */}
        <div className="flex items-center justify-between p-2 border-b bg-gray-50">
            <div className="flex items-center gap-2">
                <Button
                    variant={layout === 'horizontal' ? 'default' : 'outline'}
                    size="sm"
                    onClick={() => setLayout('horizontal')}
                >
                    <ArrowLeftRight className="h-4 w-4" />
                </Button>
                <Button
                    variant={layout === 'vertical' ? 'default' : 'outline'}
                    size="sm"
                    onClick={() => setLayout('vertical')}
                >
                    <ArrowUpDown className="h-4 w-4" />
                </Button>
                <Button
                    variant={layout === 'grid' ? 'default' : 'outline'}
                    size="sm"
                    onClick={() => setLayout('grid')}
                >
                    <Grid className="h-4 w-4" />
                </Button>
            </div>

            <Button
                size="sm"
                onClick={() => addPaneMutation.mutate()}
```

```
                    disabled={panes.length >= 4}
                >
                    <Plus className="h-4 w-4 mr-1" /> Add Pane
                </Button>
            </div>

            {/* Panes Container */}
            <div ref={containerRef} className={`flex-1 flex ${getLayoutClasses()} gap-1 p-1`}>
                {panes.map((pane: Pane, index: number) => (
                    <React.Fragment key={pane.id}>
                        <div
                            className="flex-1 min-w-0 border rounded-lg overflow-hidden"
                            style={{ flex: sizes[index] ? `0 0 ${sizes[index]}%` : 1 }}
                        >
                            <PaneContent
                                pane={pane}
                                onRemove={() => removePaneMutation.mutate(pane.paneIndex)}
                                onModelChange={(model) => updateModelMutation.mutate({ paneInd
                                onSendMessage={(message) => onSendMessage(pane.paneIndex, messa
                                canRemove={panes.length > 1}
                            />
                        </div>
                        {index < panes.length - 1 && layout !== 'grid' && (
                            <div
                                className={`${layout === 'horizontal' ? 'w-1 cursor-col-resize
                                onMouseDown={(e) => {
                                    const startPos = layout === 'horizontal' ? e.clientX : e.cl
                                    const onMouseMove = (moveEvent: MouseEvent) => {
                                        const currentPos = layout === 'horizontal' ? moveEvent
                                        handleResize(index, (currentPos - startPos) / 5);
                                    };
                                    const onMouseUp = () => {
                                        document.removeEventListener('mousemove', onMouseMove)
                                        document.removeEventListener('mouseup', onMouseUp);
                                    };
                                    document.addEventListener('mousemove', onMouseMove);
                                    document.addEventListener('mouseup', onMouseUp);
                                }}
                            />
                        )}
                    </React.Fragment>
                ))}
            </div>
        </div>
    );
}

interface PaneContentProps {
    pane: Pane;
    onRemove: () => void;
    onModelChange: (model: string) => void;
    onSendMessage: (message: string) => void;
    canRemove: boolean;
}

function PaneContent({ pane, onRemove, onModelChange, onSendMessage, canRemove }: PaneContentPr
    const [input, setInput] = useState('');

    const models = [
        { id: 'claude-opus-4', name: 'Claude Opus 4' },
        { id: 'claude-sonnet-4', name: 'Claude Sonnet 4' },
        { id: 'gpt-4o', name: 'GPT-4o' },
        { id: 'grok-4', name: 'Grok 4' },
        { id: 'gemini-2', name: 'Gemini 2' }
    ];

    return (
        <div className="h-full flex flex-col">
            {/* Pane Header */}
            <div className="flex items-center justify-between p-2 border-b bg-white">
                <Select value={pane.model || ''} onValueChange={onModelChange}>
                    <SelectTrigger className="w-40">
                        <SelectValue placeholder="Select model" />
                    </SelectTrigger>
                    <SelectContent>
                        {models.map(m => (
                            <SelectItem key={m.id} value={m.id}>{m.name}</SelectItem>
                        ))}
                    </SelectContent>
                </Select>

                {canRemove && (
                    <Button variant="ghost" size="sm" onClick={onRemove}>
                        <X className="h-4 w-4" />
                    </Button>
                )}
            </div>

            {/* Chat Area */}
            <div className="flex-1 overflow-auto p-4 bg-gray-50">
                {/* Messages would render here */}
                <div className="text-gray-500 text-center">
                    {pane.model ? `Ready to chat with ${pane.model}` : 'Select a model to star
                </div>
            </div>
            {/* Input */}
            <div className="p-2 border-t bg-white">
                <div className="flex gap-2">
                    <input
                        type="text"
                        value={input}
                        onChange={(e) => setInput(e.target.value)}
                        placeholder="Type a message..."
                        className="flex-1 px-3 py-2 border rounded-lg"
                        onKeyDown={(e) => {
                            if (e.key === 'Enter' && input.trim()) {
                                onSendMessage(input);
                                setInput('');
                            }
                        }}
                    />
                    <Button
                        onClick={() => {
                            if (input.trim()) {
                                onSendMessage(input);
                                setInput('');
                            }
                        }}
                    >
                        Send
                    </Button>
                </div>
            </div>
        </div>
    );
}
```