Nilla Technical Documentation

This document provides a comprehensive technical overview of the Nilla platform, including architecture, database schema, API endpoints, frontend structure, and development setup.

Table of Contents

Project Overview

The Problem

70% of first-time open source contributors abandon their PRs. The journey from "I want to contribute" to "I shipped a PR" is filled with friction:

- Finding the right issue is overwhelming
- Understanding codebases takes time
- Motivation fades without accountability
- No structured path to follow

The Solution

Nilla is an AI-powered commitment coach that helps first-time open source contributors build sustainable contribution habits through:

- **7-Day Commitments**: Time-bound goals with milestone tracking
- **AI-Powered Guidance**: Personalized issue explanations and coaching
- **Smart Issue Discovery**: AI-recommended issues matched to skill level
- **Gamification**: XP, levels, streaks, and badges for motivation

Core Value Proposition

> "From wanting to contribute to actually shipping PRs"

Tech Stack

Frontend

Technology	Purpose	Version
Next.js	React framework with App Router	15.x
React	UI library	19.x
TypeScript	Type safety	5.x
Tailwind CSS	Utility-first styling	3.x
Radix UI	Accessible component primitives	Latest
Zustand	Client state management	4.x
TanStack Query	Server state & caching	5.x
Lucide React	Icon library	Latest

Backend

Technology	Purpose
Next.js API Routes	Server-side endpoints
Supabase	PostgreSQL database + Auth + Vector storage
Zod	Runtime schema validation

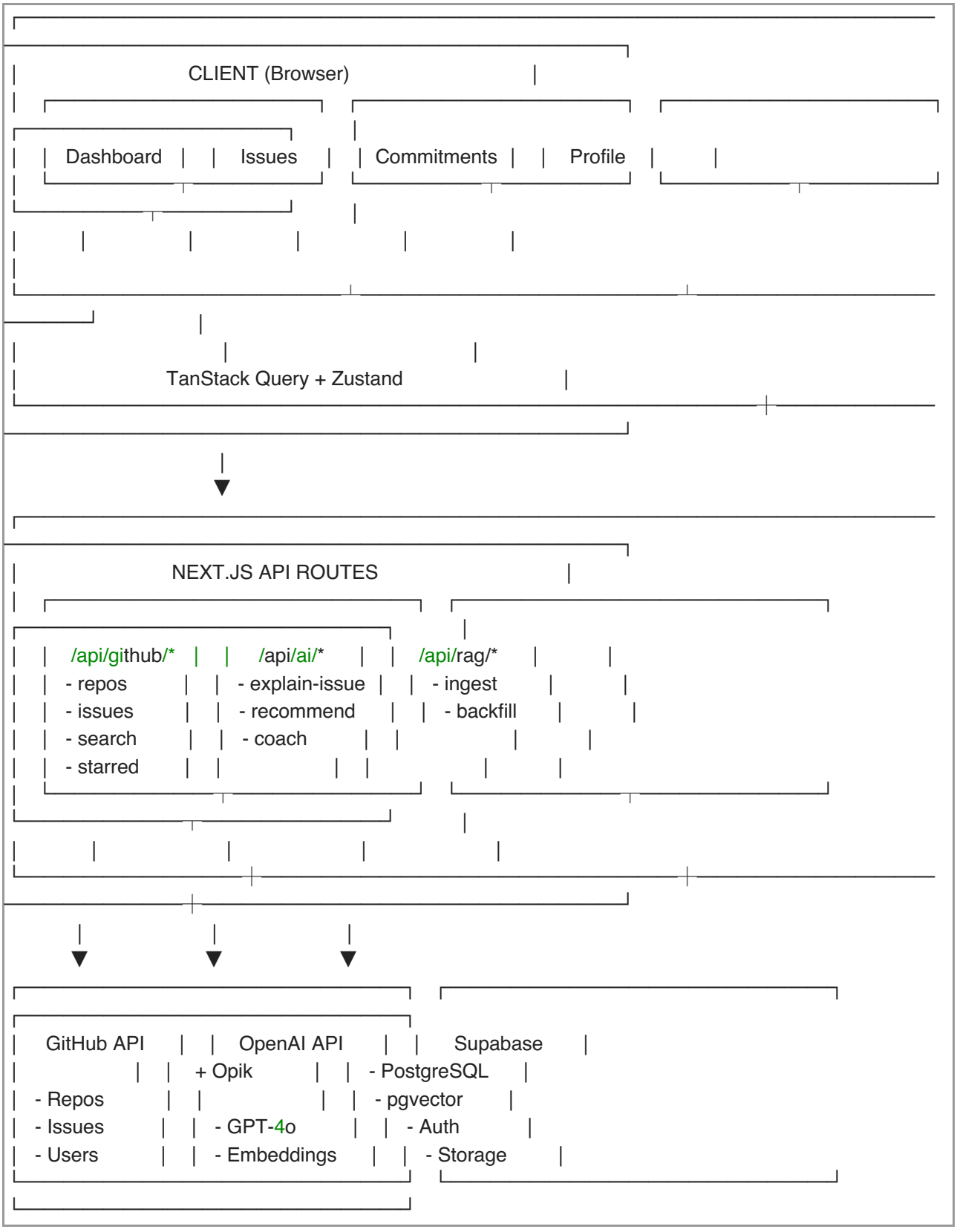
AI/ML

Technology	Purpose
OpenAI GPT-4o	LLM for agents
OpenAI Embeddings	text-embedding-3-small (1536 dims)
Opik	LLM observability & evaluation

External Services

Service	Purpose
GitHub API	Repository and issue data
Resend	Transactional emails
Vercel	Hosting & cron jobs

Architecture Overview



Data Flow Patterns

1. User Authentication

```
GitHub OAuth → Supabase Auth → Session Cookie → Middleware Validation
```

2. Issue Discovery

User Request → GitHub API → Cache **in** Supabase → Return to Client

3. AI Coaching

User Context → AI Agent → OpenAI API → Opik Trace → Structured Response

4. RAG Retrieval

Issue **Text** → Embed **Query** → Vector Search → **Context** Injection → Agent Response

---

Project Structure

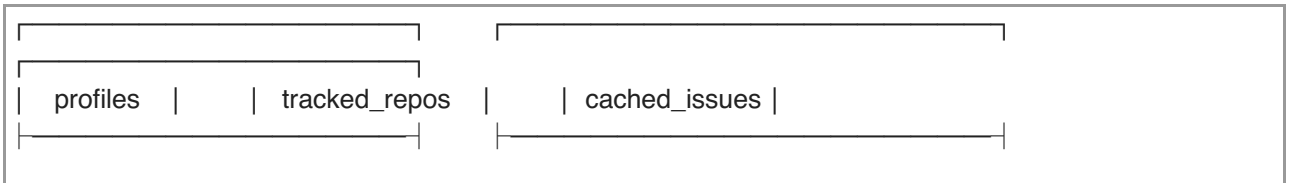
```
nilla/
├── app/                          # Next.js App Router
│   ├── (auth)/                   # Auth route group (no layout)
│   │   ├── callback/page.tsx     # OAuth callback handler
│   │   ├── login/page.tsx        # Login page
│   │   └── onboarding/page.tsx   # New user onboarding
│   │
│   ├── (dashboard)/              # Dashboard route group (shared layout)
│   │   ├── layout.tsx            # Dashboard layout with sidebar
│   │   ├── commitments/page.tsx  # Active commitments
│   │   ├── dashboard/page.tsx    # Main dashboard
│   │   ├── issues/page.tsx       # Browse issues
│   │   ├── profile/page.tsx      # User profile & badges
│   │   └── repos/
│   │       └── [owner]/[name]/   # Dynamic repo detail page
│   │           └── page.tsx
│   │
│   ├── api/                      # API routes
│   │   ├── ai/                   # AI agent endpoints
│   │   │   ├── commitment-coach/route.ts
│   │   │   ├── explain-issue/route.ts
│   │   │   └── recommend-issue/route.ts
│   │   ├── cron/
│   │   │   └── reminders/route.ts    # Daily reminder cron
│   │   ├── github/               # GitHub API proxy
│   │   │   ├── issues/route.ts
│   │   │   ├── repos/route.ts
│   │   │   ├── search/route.ts
│   │   │   └── starred/route.ts
│   │   └── rag/
│   │       ├── ingest/route.ts   # Document ingestion
│   │       └── backfill/route.ts # Backfill embeddings
│   │
│   ├── globals.css               # Global styles
│   ├── layout.tsx                # Root layout
│   └── page.tsx                  # Landing page
│
├── components/                   # React components
│   ├── ai/                       # AI feature components
```

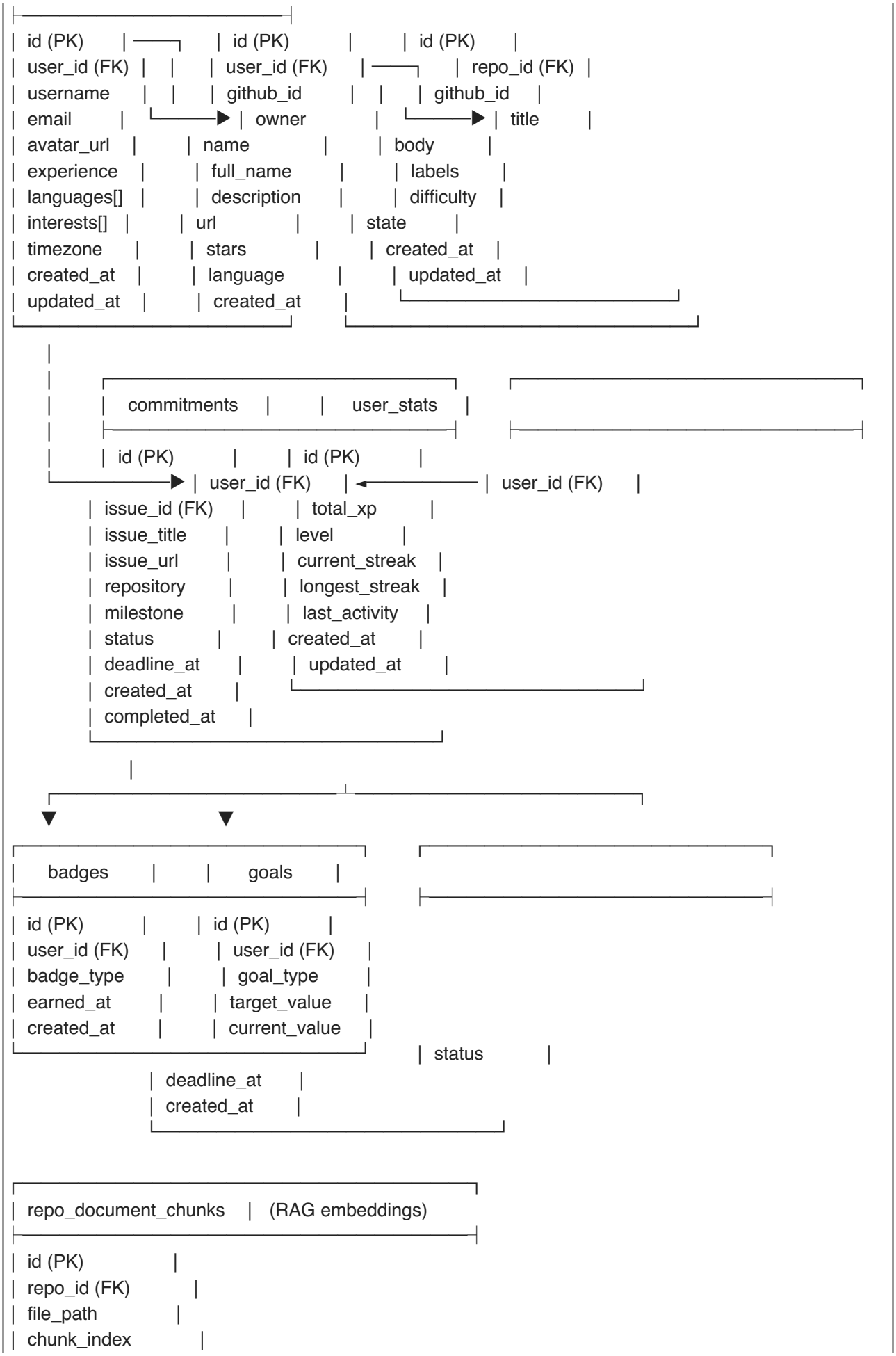
```
│   │   ├── issue-explainer.tsx
│   │   ├── commitment-coach.tsx
│   │   └── issue-recommender.tsx
│   ├── issues/                 # Issue-related components
│   │   ├── issue-card.tsx
│   │   ├── issue-list.tsx
│   │   └── issue-filters.tsx
│   ├── layout/                 # Layout components
│   │   ├── navbar.tsx
│   │   ├── sidebar.tsx
│   │   └── mobile-nav.tsx
│   ├── profile/                # Profile components
│   │   ├── badge-display.tsx
│   │   ├── streak-counter.tsx
│   │   └── xp-progress.tsx
│   └── ui/                     # Base UI components (Radix-based)
│       ├── button.tsx
│       ├── card.tsx
│       ├── dialog.tsx
│       ├── input.tsx
│       └── ...
│
├── lib/                        # Business logic & utilities
│   ├── ai/
│   │   ├── openai.ts           # OpenAI + Opik client setup
│   │   ├── index.ts            # Type exports
│   │   └── agents/
│   │       ├── index.ts        # Re-exports
│   │       ├── commitment-coach.ts   # Commitment Coach agent
│   │       ├── issue-explainer.ts    # Issue Explainer agent
│   │       ├── recommend-issue.ts    # Issue Recommender (agentic)
│   │       └── tools/
│   │           ├── fetch-repo-stats.ts
│   │           └── analyze-complexity.ts
│   │
│   ├── constants/              # Application constants
│   │   ├── badges.ts           # Badge definitions
│   │   ├── goals.ts            # Goal types
│   │   ├── languages.ts        # Programming languages
│   │   └── xp-values.ts        # XP reward values
│   │
│   ├── email/
│   │   └── resend.ts           # Resend email client
│   │
│   ├── github/
│   │   └── api.ts              # GitHub API wrapper
│   │
│   ├── hooks/                  # React hooks
│   │   ├── use-ai.ts           # AI feature hooks
│   │   └── use-toast.ts        # Toast notifications
│   │
│   ├── opik/
```

```
│   │   ├── client.ts              # Opik client setup
│   │   ├── evaluations/
│   │   │   ├── datasets.ts        # Test datasets (36 cases)
│   │   │   ├── judges.ts          # LLM-as-judge functions
│   │   │   └── index.ts
│   │   └── experiments/
│   │       ├── issue-recommender.ts
│   │       ├── commitment-coach.ts
│   │       ├── issue-explainer.ts
│   │       └── run-evaluations.ts
│   │
│   ├── rag/
│   │   ├── fetch-repo-docs.ts     # GitHub docs fetcher
│   │   ├── ingest.ts              # Chunking & embedding
│   │   └── retrieve.ts            # Vector search
│   │
│   ├── supabase/
│   │   ├── client.ts              # Browser client
│   │   ├── server.ts              # Server client
│   │   └── admin.ts               # Service role client
│   │
│   └── utils/                     # Utility functions
│       ├── cn.ts                  # Class name merger
│       └── format.ts              # Formatters
│
├── supabase/
│   └── migrations/                # Database migrations
│       ├── 001_initial_schema.sql
│       ├── 002_add_stats_functions.sql
│       └── 003_add_repo_embeddings.sql
│
├── types/                         # TypeScript type definitions
│   ├── database.ts                # Supabase generated types
│   ├── github.ts                  # GitHub API types
│   └── index.ts                   # Shared types
│
├── middleware.ts                  # Auth middleware
├── next.config.ts                 # Next.js configuration
├── tailwind.config.ts             # Tailwind configuration
├── tsconfig.json                  # TypeScript configuration
└── package.json                   # Dependencies & scripts
```

Database Schema

Entity Relationship Diagram




id (PK)
user_id (FK)
username
email
avatar_url
experience
languages[]
interests[]
timezone
created_at
updated_at
id (PK)
user_id (FK)
github_id
owner
name
full_name
description
url
stars
language
created_at
id (PK)
repo_id (FK)
github_id
title
body
labels
difficulty
state
created_at
updated_at
commitments
id (PK)
user_id (FK)
issue_id (FK)
issue_title
issue_url
repository
milestone
status
deadline_at
created_at
completed_at
user_stats
id (PK)
user_id (FK)
total_xp
level
current_streak
longest_streak
last_activity
created_at
updated_at
badges
id (PK)
user_id (FK)
badge_type
earned_at
created_at
goals
id (PK)
user_id (FK)
goal_type
target_value
current_value
status
deadline_at
created_at
repo_document_chunks
(RAG embeddings)
id (PK)
repo_id (FK)
file_path
chunk_index

```
| content                |
| embedding (vector)     |
| created_at             |
└────────────────────────────────────────────┘
```

Core Tables

profiles

User profile information and preferences.

```sql
CREATE TABLE profiles (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES auth.users(id) ON DELETE CASCADE,
  username TEXT NOT NULL,
  email TEXT,
  avatar_url TEXT,
  experience_level TEXT CHECK (experience_level IN ('beginner', 'intermediate', 'advanced')),
  preferred_languages TEXT[] DEFAULT '{}',
  interests TEXT[] DEFAULT '{}',
  timezone TEXT DEFAULT 'UTC',
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(user_id)
);
```

tracked_repos

Repositories the user is following.

```sql
CREATE TABLE tracked_repos (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES profiles(user_id) ON DELETE CASCADE,
  github_id BIGINT NOT NULL,
  owner TEXT NOT NULL,
  name TEXT NOT NULL,
  full_name TEXT NOT NULL,
  description TEXT,
  url TEXT NOT NULL,
  stars INTEGER DEFAULT 0,
  language TEXT,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(user_id, github_id)
);
```

commitments

7-day contribution commitments with milestone tracking.

```
CREATE TYPE milestone_type AS ENUM (
  'not_started',
  'read_issue',
  'ask_question',
  'work_on_solution',
  'open_pr',
  'completed'
);

CREATE TYPE commitment_status AS ENUM (
  'active',
  'completed',
  'expired',
  'abandoned'
);

CREATE TABLE commitments (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES profiles(user_id) ON DELETE CASCADE,
  issue_id BIGINT NOT NULL,
  issue_title TEXT NOT NULL,
  issue_url TEXT NOT NULL,
  repository TEXT NOT NULL,
  current_milestone milestone_type DEFAULT 'not_started',
  milestones_completed milestone_type[] DEFAULT '{}',
  status commitment_status DEFAULT 'active',
  deadline_at TIMESTAMPTZ NOT NULL,
  last_activity_at TIMESTAMPTZ,
  day3_reminder_sent BOOLEAN DEFAULT FALSE,
  day6_reminder_sent BOOLEAN DEFAULT FALSE,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  completed_at TIMESTAMPTZ
);
```

user_stats

XP, level, and streak tracking.

```
CREATE TABLE user_stats (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES profiles(user_id) ON DELETE CASCADE,
  total_xp INTEGER DEFAULT 0,
  level INTEGER DEFAULT 1,
  current_streak INTEGER DEFAULT 0,
  longest_streak INTEGER DEFAULT 0,
  last_activity_at TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT NOW(),
  updated_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(user_id)
);
```

badges

Earned achievement badges.

```
CREATE TYPE badge_type AS ENUM (
  'first_steps',
  'pr_pioneer',
  'merged',
  'getting_started',
  'week_warrior',
  'consistent_contributor',
  'open_source_hero',
  'commitment_keeper',
  'quick_starter',
  'helping_hand',
  'explorer',
  'polyglot'
);

CREATE TABLE badges (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id UUID REFERENCES profiles(user_id) ON DELETE CASCADE,
  badge_type badge_type NOT NULL,
  earned_at TIMESTAMPTZ DEFAULT NOW(),
  created_at TIMESTAMPTZ DEFAULT NOW(),
  UNIQUE(user_id, badge_type)
);
```

repo_document_chunks

RAG embeddings for repository documentation.

```
CREATE TABLE repo_document_chunks (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  repo_id UUID REFERENCES tracked_repos(id) ON DELETE CASCADE,
  file_path TEXT NOT NULL,
  chunk_index INTEGER NOT NULL,
  content TEXT NOT NULL,
  embedding VECTOR(1536),
  created_at TIMESTAMPTZ DEFAULT NOW()
);

-- Vector similarity search index
CREATE INDEX ON repo_document_chunks
USING ivfflat (embedding vector_cosine_ops)
WITH (lists = 100);
```

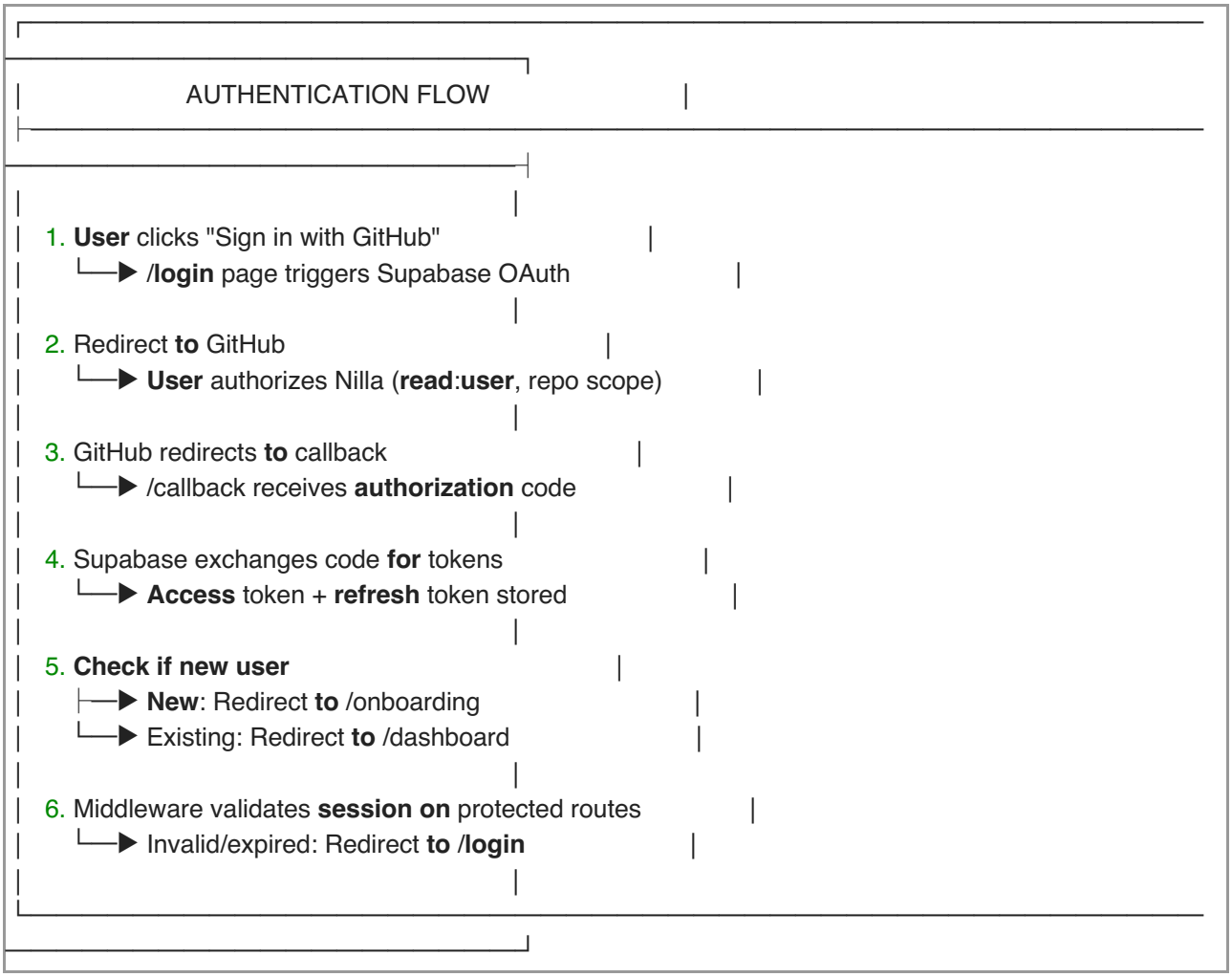
Database Functions

```sql
-- Match repository documents by similarity
CREATE FUNCTION match_repo_documents(
  query_embedding VECTOR(1536),
  match_threshold FLOAT DEFAULT 0.7,
  match_count INT DEFAULT 5,
  p_repo_id UUID DEFAULT NULL
)
RETURNS TABLE (
  id UUID,
  repo_id UUID,
  file_path TEXT,
  chunk_index INTEGER,
  content TEXT,
  similarity FLOAT
)
LANGUAGE plpgsql
AS $
BEGIN
  RETURN QUERY
  SELECT
    rdc.id,
    rdc.repo_id,
    rdc.file_path,
    rdc.chunk_index,
    rdc.content,
    1 - (rdc.embedding <=> query_embedding) AS similarity
  FROM repo_document_chunks rdc
  WHERE (p_repo_id IS NULL OR rdc.repo_id = p_repo_id)
    AND 1 - (rdc.embedding <=> query_embedding) > match_threshold
  ORDER BY rdc.embedding <=> query_embedding
  LIMIT match_count;
END;
$;
```

Authentication Flow

Nilla uses GitHub OAuth via Supabase Auth for secure, read-only access to user's GitHub data.

OAuth Flow



Middleware Configuration

```
// middleware.ts
import { createMiddlewareClient } from '@supabase/auth-helpers-nextjs';
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';

export async function middleware(req: NextRequest) {
  const res = NextResponse.next();
  const supabase = createMiddlewareClient({ req, res });

  const { data: { session } } = await supabase.auth.getSession();

  // Protected routes
  const protectedPaths = ['/dashboard', '/repos', '/issues', '/commitments', '/profile'];
  const isProtected = protectedPaths.some(path => req.nextUrl.pathname.startsWith(path));

  if (isProtected && !session) {
    return NextResponse.redirect(new URL('/login', req.url));
  }

  return res;
}

export const config = {
  matcher: ['/((?!apil_next/staticl_next/imagelfavicon.ico).*)'],
};
```

GitHub Scopes

Scope	Purpose
read:user	Access user profile information
repo	Read repository and issue data

API Endpoints

GitHub Proxy Endpoints

These endpoints proxy requests to GitHub API using the user's OAuth token.

GET /api/github/repos

Fetch user's repositories or search public repos.

```
// Query Parameters
interface ReposQuery {
  type?: 'owned' | 'starred' | 'search';
  query?: string;  // For search
  page?: number;
  per_page?: number;
}

// Response
interface ReposResponse {
  repos: Repository[];
  total_count?: number;
}
```

GET /api/github/issues

Fetch issues for a repository.

```
// Query Parameters
interface IssuesQuery {
  owner: string;
  repo: string;
  labels?: string;  // Comma-separated
  state?: 'open' | 'closed' | 'all';
  page?: number;
  per_page?: number;
}

// Response
interface IssuesResponse {
  issues: Issue[];
}
```

GET /api/github/starred

Fetch user's starred repositories.

```
// Response
interface StarredResponse {
  repos: Repository[];
}
```

AI Agent Endpoints

POST /api/ai/explain-issue

Get AI-powered issue explanation.

```
// Request Body
interface ExplainIssueRequest {
  issue: {
    title: string;
    body?: string;
    labels: string[];
    repository: string;
    url: string;
  };
  experienceLevel: 'beginner' | 'intermediate' | 'advanced';
}

// Response
interface ExplainIssueResponse {
  summary: string;
  expectedOutcome: string;
  repoGuidelines: string[];
  beginnerPitfalls: string[];
  suggestedApproach: string;
  keyTerms: Array<{ term: string; definition: string }>;
  confidenceNote: string;
}
```

POST /api/ai/recommend-issue

Get AI-recommended issues.

```
// Request Body
interface RecommendIssueRequest {
  user: {
    id: string;
    username: string;
    skillLevel: 'beginner' | 'intermediate' | 'advanced';
    preferredLanguages: string[];
    interests?: string[];
    pastContributions?: number;
    availableHoursPerWeek?: number;
  };
  issues: Array<{
    id: string;
    title: string;
    body?: string;
    labels: string[];
    repository: string;
    language?: string;
    url: string;
  }>;
}

// Response
interface RecommendIssueResponse {
  recommendedIssue: {
    id: string;
    title: string;
    repository: string;
    url: string;
  };
  explanation: string;
  riskLevel: 'low' | 'medium' | 'high';
  riskFactors: string[];
  alternativeIssues: Array<{
    id: string;
    title: string;
    reason: string;
  }>;
  rankedIssues: Array<{
    issueId: string;
    difficultyScore: number;
    fitScore: number;
    reasoning: string;
  }>;
}
```

POST /api/ai/commitment-coach

Get personalized coaching for a commitment.

```
// Request Body
interface CommitmentCoachRequest {
  commitment: {
    id: string;
    issueTitle: string;
    issueUrl: string;
    repository: string;
    createdAt: string;
    deadlineAt: string;
    currentMilestone: Milestone;
    milestonesCompleted: Milestone[];
    lastActivityAt?: string;
  };
  user: {
    username: string;
    totalCommitments?: number;
    completedCommitments?: number;
    timezone?: string;
  };
}

// Response
interface CommitmentCoachResponse {
  nextAction: {
    action: string;
    why: string;
    estimatedMinutes?: number;
  };
  nudge: {
    message: string;
    tone: 'encouraging' | 'motivating' | 'celebratory' | 'urgent' | 'supportive';
  };
  riskAssessment: {
    level: 'on_track' | 'needs_attention' | 'at_risk' | 'critical';
    reason: string;
    daysRemaining: number;
    hoursRemaining: number;
  };
  warning?: {
    message: string;
    suggestion: string;
  };
  progress: {
    currentMilestone: Milestone;
    milestonesRemaining: number;
    percentComplete: number;
  };
}
```

RAG Endpoints

POST /api/rag/ingest

Ingest repository documentation for RAG.

```
// Request Body
interface IngestRequest {
  repoId: string;
  owner: string;
  name: string;
}

// Response
interface IngestResponse {
  success: boolean;
  chunksCreated: number;
  filesProcessed: string[];
}
```

Cron Endpoints

GET /api/cron/reminders

Daily cron job for commitment reminders (called by Vercel Cron).

```
// Headers
{
  'Authorization': 'Bearer ${CRON_SECRET}'
}

// Response
interface RemindersResponse {
  day3Sent: number;
  day6Sent: number;
  expired: number;
}
```

Frontend Structure

Route Groups

(auth) - Authentication routes (no shared layout)

- /login - GitHub OAuth login
- /callback - OAuth callback handler
- /onboarding - New user setup wizard

(dashboard) - Protected routes (shared dashboard layout)

- /dashboard - Main dashboard with stats and recent activity
- /repos - Repository management
- /repos/[owner]/[name] - Individual repo with issues
- /issues - Browse all issues across tracked repos

- /commitments - Active and past commitments
- /profile - User profile, badges, and settings

Key Components

Layout Components

- Navbar - Top navigation with user menu
- Sidebar - Desktop side navigation
- MobileNav - Bottom navigation for mobile

AI Components

- IssueExplainer - Displays AI issue explanations
- CommitmentCoach - Shows coaching messages and next actions
- IssueRecommender - Displays recommended issues with reasoning

Gamification Components

- XPProgress - XP bar with level indicator
- StreakCounter - Current streak display with flame icon
- BadgeDisplay - Grid of earned/locked badges

State Management

Zustand Stores:

- useUserStore - User profile and preferences
- useRepoStore - Tracked repositories
- useCommitmentStore - Active commitments

TanStack Query:

- Used for all server data fetching
- Automatic caching and background refetching
- Optimistic updates for mutations

AI Agents

Nilla uses three specialized AI agents. For detailed architecture, see [AI Architecture Documentation (./3.NILLA-AGENT-ARCHITECTURE.md)](./3.NILLA-AGENT-ARCHITECTURE.md).

Agent Summary

Agent	Type	RAG	Purpose
Issue Explainer	Structured	Yes	Break down issues for skill level
Commitment Coach	Structured	No	Personalized coaching messages
Issue Recommender	Agentic	No	Multi-step issue matching

Opik Integration

All agents are traced with Opik for observability:

```
import { createTrackedAI, flushTraces } from '@/lib/ai/openai';

const trackedAI = createTrackedAI('agent-name');

const completion = await trackedAI.chat.completions.create({
  model: 'gpt-4o',
  messages: [...],
});

await flushTraces();
```

RAG Pipeline

Ingestion Flow

```
Repository Added
      |
      ▼
Fetch Documentation (GitHub API)
  ├── README.md
  ├── CONTRIBUTING.md
  └── docs/*.md
      |
      ▼
Chunk Documents
  ├── Target: ~400 tokens per chunk
  ├── Overlap: 50 tokens
  └── Split on paragraph boundaries
      |
      ▼
Generate Embeddings (text-embedding-3-small)
      |
      ▼
Store in Supabase (pgvector)
```

Retrieval Flow

Gamification System

XP Values

Action	XP
Create commitment	10
Read issue milestone	5
Ask question milestone	10
Work on solution milestone	15
Open PR milestone	25
PR merged milestone	50
Complete commitment	30
Complete early bonus	20
3-day streak	15
7-day streak	30
14-day streak	50
30-day streak	100
Earn badge	50
Complete goal	100

Level Thresholds

Level	XP Required
1	0
2	100
3	250
4	500
5	1,000
6	2,000
7	3,500

8	5,500
9	8,000
10	11,000

Badges

Badge	Criteria
First Steps	Created first commitment
PR Pioneer	Opened first PR
Merged!	Got first PR merged
Getting Started	3-day streak
Week Warrior	7-day streak
Consistent Contributor	14-day streak
Open Source Hero	30-day streak
Commitment Keeper	5 on-time completions
Quick Starter	Completed in under 3 days
Helping Hand	Contributed to 3+ repos
Explorer	Added 10 repositories
Polyglot	Contributed in 3+ languages

Email System

Provider

Resend - Transactional email service

Reminder Schedule

Reminder	Timing	Template
Day 3	3 days after commitment	Supportive check-in
Day 6	1 day before deadline	Urgency reminder

Cron Job

```
// Runs daily at 9 AM UTC via Vercel Cron
// vercel.json
{
  "crons": [{
    "path": "/api/cron/reminders",
    "schedule": "0 9 * * *"
  }]
}
```

Observability

Opik Integration

All LLM calls are traced with Opik:

- **Cost tracking** per agent
- **Latency monitoring**
- **Prompt/response logging**
- **Hierarchical tracing** for agentic workflows
- **LLM-as-judge evaluations**

Evaluation Metrics

Each agent is evaluated on 4-5 dimensions:

Issue Recommender:

- Match quality
- Difficulty calibration
- Explanation clarity
- Risk assessment

Commitment Coach:

- Tone appropriateness
- Actionability
- Risk accuracy
- Urgency calibration

Issue Explainer:

- Clarity
- Accuracy
- Level appropriateness
- Actionability

Running Evaluations

```
pnpm experiment:recommender  # Issue Recommender
pnpm experiment:coach        # Commitment Coach
pnpm experiment:explainer    # Issue Explainer
pnpm eval                    # All evaluations
```

Environment Setup

Required Environment Variables

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://your-project.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key
SUPABASE_SERVICE_ROLE_KEY=your-service-role-key

# App
NEXT_PUBLIC_APP_URL=http://localhost:3001

# OpenAI
OPENAI_API_KEY=sk-...

# Resend (Email)
RESEND_API_KEY=re_...

# Cron Authentication
CRON_SECRET=your-cron-secret

# Opik (Observability)
OPIK_URL_OVERRIDE=https://opik.example.com  # Optional
OPIK_PROJECT_NAME=nilla
OPIK_API_KEY=your-opik-key
OPIK_WORKSPACE=your-workspace
```

Local Development Setup

```
# 1. Clone the repository
git clone https://github.com/your-org/nilla.git
cd nilla

# 2. Install dependencies
pnpm install

# 3. Copy environment variables
cp .env.example .env.local
# Edit .env.local with your values

# 4. Set up Supabase
# Option A: Use Supabase Cloud
# Option B: Run locally with Docker
supabase start

# 5. Run database migrations
supabase db push

# 6. Start the development server
pnpm dev

# 7. Open http://localhost:3001
```

Development Workflow

Available Scripts

```
# Development
pnpm dev              # Start dev server (port 3001, Turbopack)
pnpm build            # Production build
pnpm start            # Run production server
pnpm lint             # Run ESLint

# Evaluations
pnpm eval             # Run all agent evaluations
pnpm experiment:recommender  # Issue Recommender evaluation
pnpm experiment:coach        # Commitment Coach evaluation
pnpm experiment:explainer    # Issue Explainer evaluation

# Database
supabase db push      # Push migrations
supabase db reset     # Reset database
supabase gen types    # Generate TypeScript types
```

Code Quality

- **TypeScript** - Strict mode enabled
- **ESLint** - Next.js recommended rules
- **Prettier** - Code formatting (optional)
- **Zod** - Runtime validation for all API inputs/outputs

Git Workflow

```
# Feature branches
git checkout -b feature/your-feature

# Commit convention
git commit -m "feat: add new feature"
git commit -m "fix: resolve bug"
git commit -m "docs: update documentation"

# Pull request
gh pr create --title "feat: your feature" --body "Description"
```

Deployment

Vercel Configuration

```
// vercel.json
{
  "crons": [
    {
      "path": "/api/cron/reminders",
      "schedule": "0 9 * * *"
    }
  ]
}
```

Next.js Configuration

```
// next.config.ts
import type { NextConfig } from 'next';

const nextConfig: NextConfig = {
  images: {
    remotePatterns: [
      {
        protocol: 'https',
        hostname: 'avatars.githubusercontent.com',
      },
    ],
  },
  webpack: (config, { isServer }) => {
    if (isServer) {
      // Externalize OpenAI and Opik for server-side
      config.externals.push('openai', 'opik', 'opik-openai');
    }
    return config;
  },
};

export default nextConfig;
```

Environment Variables in Vercel

Add all environment variables from the Environment Setup section to your Vercel project settings.

Deployment Checklist

- [ ] All environment variables configured
- [ ] Supabase project created and migrations applied
- [ ] GitHub OAuth app configured with production callback URL
- [ ] Resend domain verified
- [ ] Opik project created
- [ ] Cron job configured in vercel.json

Additional Resources

- Supabase Documentation (https://supabase.com/docs)

- Next.js Documentation (https://nextjs.org/docs)
- Opik Documentation (https://www.comet.com/docs/opik/)
- OpenAI API Reference (https://platform.openai.com/docs/api-reference)