

Alpha Releases 1, 2 & 3

# Technical Architecture

- Technical Architecture
  - SilverChat Development & Project Summary - Alpha Release 1
    - 1. Core Project Vision & Objective
    - 2. Alpha Phase (MVP) Goals
    - 3. Technical Stack & Architecture
    - 4. Development Methodology
    - 5. Project Structure & Key Files (Backend - Implemented)
    - 6. Database (PostgreSQL via Knex - Alpha 1 Schema Implemented)
    - 7. Development Workflow (Established)
    - 8. Future Vision (Post-Alpha)
  - 1. Phase 1: Alpha Development (Initial MVP)
  - 2. Technical Stack
  - 3. Development Methodology & Team
  - 4. Structure
  - 5. Potential Challenges & Mitigation
  - 6. SilverChat Project Documentation

## SilverChat Development & Project Summary - Alpha Release 1

This document summarizes `readme.md`, `technical-architecture.md`, `database-1-account-profile.md`, and `development-setup.md`, providing a holistic view of the project's purpose, the technical foundation laid, the current state of development, and the immediate next steps focused on building the Alpha MVP features using TDD.

# 1. Core Project Vision & Objective

- **Mission:** To be a premier online platform fostering **companionship, friendship, and mutual support** specifically for individuals aged **60 and over**.
- **Niche:** A distinct alternative to romance-focused platforms, prioritizing social connection based on shared interests and life experiences.
- **Core Pillars:**
  - **Companionship Focus:** Facilitating meaningful connections.
  - **Effortless Communication:** Simple, integrated text, voice, and video chat (via the 'SilverChat Agent' concept).
  - **Uncompromising Safety & Trust:** Robust member protection, including planned identity verification and AI monitoring ("AI Guard").
  - **Simplicity & Accessibility:** Intuitive, clear UI/UX designed for the target demographic.

---

[\[BACK TO INDEX\]](#)

## 2. Alpha Phase (MVP) Goals

- **Objective:** Develop a functional Minimum Viable Product (MVP) demonstrating core features:
  - User Authentication (Registration/Login)
  - Member Profile (Basic)
  - Member Connections (Request/Accept/Decline/View)
  - Basic 1-to-1 Text Chat
  - Basic Member Groups (View/Join/Leave)
  - Basic Notifications (Connection requests, new messages)
  - 'SilverChat Agent' UI Placeholder (Visual representation, no functionality yet)
  - Foundational Safety: Community Guidelines, basic reporting, secure password handling.
- **Purpose:** Validate the core concept, establish the technical base using TDD, and prepare for future enhancements.

---

[\[BACK TO INDEX\]](#)

## 3. Technical Stack & Architecture

- **Frontend:** Vanilla JavaScript (ES6+).
- **Backend:** Node.js (LTS recommended) with Express.js.
- **Database:** PostgreSQL.
- **DB Client/Interaction:** `pg` package, Knex.js (Query Builder, Migrations, Seeds).
- **Backend Testing:** Jest (Strict TDD approach).
- **Architecture:** MVC pattern for the backend. OOP principles with ES6+.
- **Environment:** `dotenv` for environment variables.
- **Tools:** Git/GitHub, VS Code, Insomnia.

## 4. Development Methodology

- **Strict Test-Driven Development (TDD):** Using Jest for backend tests (Red-Green-Refactor cycle). Tests written *before* implementation code.
- **Agile Principles:** Iterative development, small testable units.

## 5. Project Structure & Key Files (Backend - Implemented)

- **backend/ Directory:** Contains all backend code and configuration.
  - **src/:** Planned location for MVC components (`api`, `controllers`, `models`, `middleware`, etc.). *Note: MVC structure planned but not fully implemented in `src/` yet.*
  - **db/knex.js:** Initializes and exports the shared Knex instance.
  - **migrations/:** Contains `YYYYMMDDHHMMSS_create_alpha1_schema.js` defining initial tables.
  - **seeds/:** Contains `seed_initial_lookup_tables.js` for populating lookup tables.
  - **\_\_tests\_\_/:** Contains initial Jest tests (`database.test.js`, `server.test.js`).
  - **server.js:** Main server entry point (Express setup, basic root route, starts server). Exports `app`.
  - **knexfile.js:** Knex configuration (environments, DB connection via `.env`, paths, pool).
  - **jest.config.js:** Jest configuration (`testEnvironment: 'node'`, `clearMocks: true`).
  - **package.json:** Dependencies (`express`, `pg`, `dotenv`, `knex`), DevDependencies (`jest`, `nodemon`, `supertest`), Scripts (`dev`, `test`, `start`).
  - **.gitignore:** Ignores `node_modules/`, `.env`, etc.
  - **.env / .env.example:** Environment variable management.

## 6. Database (PostgreSQL via Knex - Alpha 1 Schema Implemented)

- **Setup:** User (`DB_USER`), Password (`DB_PASSWORD`), and Database (`DB_NAME`) created manually; credentials in `.env`.
- **Schema:** Managed via Knex migrations. `snake_case` naming.
- **Tables Created:**
  - **Lookups:** `title`, `member_type`, `verification_state`, `member_status`, `gender`, `marital_status`. (Populated via seeds).
  - **Core:** `member` (stores profile info, links to lookups, includes status IDs like `identity_verification_status_id`), `address` (1:1 with `member`).
- **Key Relationships:** FKs link `member` to lookups; 1:1 `member` <-> `address`.

- **Data Integrity:** Constraints ( `PK`, `FK`, `NOT NULL`, `UNIQUE`, `DEFAULT` ) and indexes applied.

---

[\[BACK TO INDEX\]](#)

## 7. Development Workflow (Established)

- **Run Dev Server:** `npm run dev` (Nodemon for auto-restart).
- **Run Migrations:** `npx knex migrate:latest`.
- **Run Seeds:** `npx knex seed:run`.
- **Run Tests:** `npm test` (Jest).

---

[\[BACK TO INDEX\]](#)

## 8. Future Vision (Post-Alpha)

- Full 'SilverChat Agent' (integrated voice/video).
- Advanced 'AI Guard' for safety monitoring.
- Enhanced group features and offline meetups.
- Potential "No-Scam Guarantee".

---

[\[BACK TO INDEX\]](#)

# 1. Phase 1: Alpha Development (Initial MVP)

**Objective:** To develop a functional Minimum Viable Product (MVP) demonstrating SilverChat's core user experience and foundational features using a strict Test-Driven Development (TDD) methodology. This phase validates the core concept and establishes the technical base.

**Methodology:** Development will rigorously follow TDD principles using Jest. Tests will be written *before* implementation code for all functionalities, ensuring code quality, maintainability, and facilitating future development. The application will be structured using the Model-View-Controller (MVC) pattern.

### Alpha Features & TDD Application:

#### 1. User Authentication (Backend & Frontend):

- **Functionality:** Secure user registration and login.
- **TDD Approach:** Test registration success/failure (e.g., duplicate email), login success/failure, password security, session management. Test frontend validation.

#### 2. Member Profile:

- **Functionality:** Basic profile creation (username, bio, interests). View own/others' profiles.
- **TDD Approach:** Test profile data storage, retrieval, and display logic.

#### 3. Member Connections:

- **Functionality:** Send, receive, accept, decline connection requests. View connections list.

- *TDD Approach*: Test connection state changes, database updates, and correct list display.

#### 4. Basic Text Chat:

- **Functionality**: One-to-one text chat between connected members.
- *TDD Approach*: Test message sending, receiving, storage, and history retrieval.

#### 5. Member Groups (Basic):

- **Functionality**: View predefined groups, join/leave groups, view group members.
- *TDD Approach*: Test membership logic, database updates, list retrieval.

#### 6. Notifications (Basic):

- **Functionality**: In-app notifications for connection requests, new messages.
- *TDD Approach*: Test notification generation, retrieval, display, and read-status updates.

#### 7. 'SilverChat Agent' Placeholder:

- **Functionality**: A static UI element within the chat or profile interface visually representing where the future voice/video controls (as shown in the concept mock-up) will reside. No actual calling functionality in Alpha.
- *TDD Approach*: Test the conditional rendering of these placeholder UI elements.

#### 8. Safety and Moderation (Alpha Implementation)

- **Focus on foundational elements**:
  - Drafting clear Community Guidelines.
  - Implementing a basic user reporting mechanism.
  - Secure password handling and input sanitisation.
  - *Note*: Advanced features like ID verification, AI Guard, and geographical fencing are planned for post-Alpha phases but inform the overall design philosophy.

**Prototype**: A demonstration involving ~5 mock user profiles illustrating key user journeys (registration, connection, chat) and frontend-backend interaction.

[\[BACK TO INDEX\]](#)

## 2. Technical Stack

- **Frontend**: Vanilla JavaScript (ES6+) - *Client-side logic and DOM manipulation.*
- **Backend**: Node.js with Express.js framework - *Server-side logic, API routing, and request handling.*
- **Database**: PostgreSQL - *Relational database for persistent data storage.*
- **Database Interaction & Migrations**: Knex.js - *SQL query builder for interacting with the database in code and a robust migration tool for managing database schema changes.*
- **Backend Testing**: Jest - *JavaScript testing framework for implementing Test-Driven Development (TDD) for backend logic.*
- **API Development/Testing Tool**: Insomnia (or similar like Postman) - *Used during development for sending requests to the backend API and inspecting responses.*
- **Architecture**: Model-View-Controller (MVC) - *Backend architectural pattern for separation of concerns.*

[\[BACK TO INDEX\]](#)

### 3. Development Methodology & Team

- **Methodology:** Strict Test-Driven Development (TDD) using Jest. Backend development will employ Object-Oriented Programming (OOP) principles using modern ES6+ JavaScript. Agile principles (iterative development, small testable units) applied. Red-Green-Refactor cycle enforced.
- **Team:** Louie Morais (Founder)
  - Roles: Product Owner, UX/UI Designer, Full-Stack JavaScript Developer.

[\[BACK TO INDEX\]](#)

### 4. Structure

```
silverchat/
├── .gitignore    # Specifies intentionally untracked files that Git should ignore
├── README.md    # Project overview, setup instructions, etc.
├── backend/     # Contains all Node.js/Express backend code
│   ├── src/     # Source code for the backend application
│   │   ├── api/ # API route definitions (the 'V'iew layer in API context)
│   │   │   ├── index.js # Main router to aggregate all other route files
│   │   │   ├── auth.routes.js
│   │   │   ├── users.routes.js
│   │   │   ├── connections.routes.js
│   │   │   ├── messages.routes.js
│   │   │   ├── groups.routes.js
│   │   │   └── notifications.routes.js
│   │   ├── controllers/ # Handles incoming requests, interacts with models/services
│   │   │   │   (the 'C'ontroller)
│   │   │   ├── auth.controller.js
│   │   │   ├── users.controller.js
│   │   │   ├── connections.controller.js
│   │   │   ├── messages.controller.js
│   │   │   ├── groups.controller.js
│   │   │   └── notifications.controller.js
│   │   ├── models/ # Handles data logic and database interaction (the 'M'odel)
│   │   │   ├── user.model.js
│   │   │   ├── connection.model.js
│   │   │   ├── message.model.js
│   │   │   ├── group.model.js
│   │   │   ├── notification.model.js
│   │   │   └── db.js # Database connection setup (e.g., PostgreSQL pool)
│   │   └── middleware/ # Custom Express middleware (e.g., authentication checks,
│   │   │   validation)
│   │   │   ├── requireAuth.js
│   │   │   └── validateInput.js
│   └── ...
```

```

| | | └─ services/    # Optional: For complex business logic separating it from
controllers
| | |
| | | └─ config/      # Configuration files (database credentials, environment
variables)
| | |   └─ index.js
| | |   └─ database.config.js
| | |
| | | └─ utils/       # Utility/helper functions
| | |   └─ passwordUtils.js
| | |
| | | └─ app.js       # Express application setup (middleware, routes)
| | |   └─ server.js  # Server initialisation (starts listening for requests)
| | |
| | └─ __tests__/     # Backend tests using Jest (Updated folder name convention)
| |   └─ unit/        # Unit tests for models, utils, services
| |   └─ integration/ # Integration tests for controllers, API endpoints
| |     └─ setup.js   # Global test setup/teardown (e.g., test database connection)
| |
| | └─ .env           # Environment variables (DB connection strings, secrets) - MUST be in
.gitignore
| |   └─ .env.example # Example environment variables file for reference
| |   └─ package.json # Backend dependencies and scripts
| |   └─ jest.config.js # Jest configuration for the backend
| |
└─ frontend/         # Contains all vanilla JS frontend code
  └─ public/          # Static assets served directly by the web server
    └─ index.html     # Main entry point for the application
    └─ css/           # CSS stylesheets
      └─ main.css
    └─ images/        # Image assets
      └─ favicon.ico
  └─ src/             # Frontend JavaScript source code
    └─ js/            # Main JavaScript files
      └─ main.js      # Main script, initializes the app
      └─ apiService.js # Functions for making requests to the backend API
      └─ auth.js      # Handles login/registration forms and logic
      └─ chat.js      # Handles chat interface logic
      └─ profile.js   # Handles profile display/editing logic
      └─ uiComponents.js # Functions to create/update parts of the UI
      └─ utils.js     # Frontend utility functions
    └─ __tests__/     # Frontend tests (using Jest, potentially with JSDOM) (Updated
folder name convention)
      └─ unit/
      └─ setup.js
      # Optional: package.json for frontend dev dependencies (linter, jest)
      # Optional: jest.config.js for frontend tests

```

[\[BACK TO INDEX\]](#)

## 5. Potential Challenges & Mitigation

- **Technical Complexity:** Implementing even basic real-time features requires careful planning. *Mitigation:* Strict adherence to TDD, breaking down tasks, focusing on core MVP logic.
- **Scope Management:** Balancing the ambitious vision with achievable Alpha goals. *Mitigation:* Rigorous adherence to the defined Alpha scope, logging future ideas.
- **TDD Implementation:** Ensuring effective TDD practice. *Mitigation:* Consistent application of the Red-Green-Refactor cycle, starting with simple tests.

[\[BACK TO INDEX\]](#)

## 6. SilverChat Project Documentation

- [SilverChat Project Scope \(README\) | GitHub](#)
- **THIS DOCUMENT:** *SilverChat Technical Architecture*
- [SilverChat Database Schema 1: Member Account and Profile | GitHub](#)
- [SilverChat Database Schema 2: Member Networking | GitHub](#)
- [SilverChat Database Schema 3: Member Messaging | GitHub](#)

[\[BACK TO INDEX\]](#)