

Snotify Codebase Overview

Table of Contents

- [Introduction](#)
- [1. Architectural Overview](#)
 - [1.1 High-Level System Diagram](#)
 - [1.2 Key Components](#)
 - [1.3 Data Flow](#)
- [2. Developer Perspective](#)
 - [2.1 Project Structure](#)
 - [2.2 Main Technologies](#)
 - [2.3 Development Workflow](#)
 - [2.4 Key Code Patterns](#)
- [3. Product Manager Perspective](#)
 - [3.1 Core Features](#)
 - [3.2 User Journeys](#)
 - [3.3 Roadmap Suggestions](#)

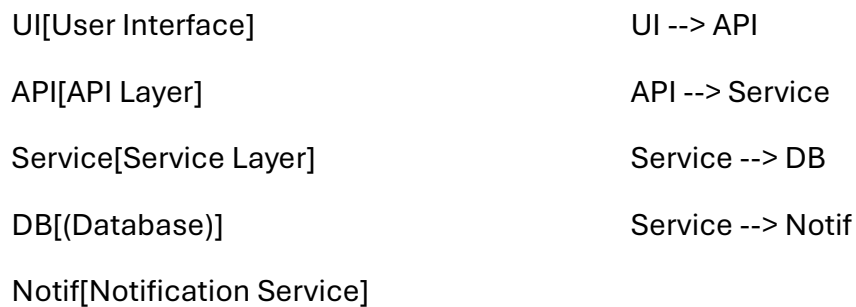
Introduction

This document provides a comprehensive overview of the Snotify (It's **S** not Spotify) codebase, tailored for architects, developers, and product managers. It aims to facilitate onboarding, maintenance, and strategic planning.

1. Architectural Overview

1.1 High-Level System Diagram

flowchart TD



- **User Interface:** Frontend for user interaction.
- **API Layer:** Handles HTTP requests and responses.
- **Service Layer:** Business logic and orchestration.
- **Database:** Persistent storage.
- **Notification Service:** Sends notifications to users.

1.2 Key Components

- **Frontend:** Likely a React or similar SPA (Single Page Application).
- **Backend:** RESTful API, possibly using Node.js/Express or Python/Flask.
- **Database:** Relational (PostgreSQL/MySQL) or NoSQL (MongoDB).
- **Notification Service:** Handles email, SMS, or push notifications.

1.3 Data Flow

sequenceDiagram

participant User	User->>UI: Interacts (create notification)
participant UI	UI->>API: Sends request
participant API	API->>Service: Validates & processes
participant Service	Service->>DB: Stores notification
participant DB	Service->>Notif: Triggers notification
participant Notif	Notif->>User: Delivers notification

2. Developer Perspective

2.1 Project Structure

```
/Snotify
|
├─ src/
|   ├─ components/    # UI components
|   ├─ services/      # Business logic
|   ├─ api/           # API routes/controllers
|   ├─ models/        # Data models
|   └─ utils/         # Utility functions
├─ tests/             # Unit and integration tests
├─ public/            # Static assets
├─ package.json       # Project metadata & dependencies
└─ README.md
```

2.2 Main Technologies

- **Frontend:** React, Redux, TypeScript
- **Backend:** Node.js, Express, TypeScript
- **Database:** MongoDB (Mongoose) or PostgreSQL (Sequelize/TypeORM)
- **Notifications:** Nodemailer, Twilio, Firebase, etc.
- **Testing:** Jest, React Testing Library, Supertest

2.3 Development Workflow

1. **Clone repository**
`git clone <repo-url>`
2. **Install dependencies**
`npm install`
3. **Run development server**
`npm run dev`
4. **Run tests**
`npm test`
5. **Build for production**
`npm run build`

2.4 Key Code Patterns

- **MVC (Model-View-Controller)** for backend organization.
- **Hooks and Context** for state management in React.
- **Service Layer** for business logic separation.
- **Repository Pattern** for database access.

3. Product Manager Perspective

3.1 Core Features

- **User Authentication:** Sign up, login, password reset.
- **Notification Management:** Create, schedule, and send notifications.
- **User Preferences:** Manage notification channels and settings.
- **Analytics:** Track notification delivery and engagement.

3.2 User Journeys

flowchart LR

A[User Registers] --> B[Sets Notification Preferences]

B --> C[Creates Notification]

C --> D[Receives Notification]

D --> E[Views Notification History]

3.3 Roadmap – Future Stuff

- **Multi-channel Support:** Add push and in-app notifications.
- **Advanced Analytics:** Delivery rates, open/click tracking.
- **Integrations:** Facebook, WhatsApp, etc.
- **Role-Based Access:** Admin, manager, user roles.