CASE (working name)

Design Document

Surendra Jammishetti

0.1.0

2026-02-10

This document outlines the design, architecture, and implementation plan for CASE (working name).

Table of Contents

1 Project Goals

CASE (working name) intends to be an offline-first task and goal tracking application that supports multiple devices and multiple user interfaces.

1.1 Primary Goals

1. **Goal 1:** Local-First
   - Success metric: Airplane mode functionality
2. **Goal 2:** Multidevice capability
   - Success metric: Sync'ed data between Mac and iPhone.
3. **Goal 3:** Idiomatic UI
   - An idiomatic UI is ergonomic UI that the user can expect to be standard with other tools in the same platform category.
4. **Goal 4:** Integrations
   - The app should have integrations with capability to source tasks from platforms like canvas, email, GitHub, etc.

1.2 Secondary Goals

Nice-to-have objectives that aren't critical for the initial release:

- Secondary objective 1
- Secondary objective 2
- Secondary objective 3

1.3 Non-Goals

- Sharing / Sharing tasks with friends: I simply don't care.
- More as I figure out what I don't feel like doing.

2 Project Components

2.1 Component Overview

Component	**Description**	**Status**	**Priority**
Frontend	User interface and client-side logic	Planned	High
Backend API	Server-side business logic	Planned	High
Database	Data persistence layer	Planned	High
Auth System	User authentication and authorization	Planned	Medium

2.2 Component 1: [Name]

Purpose: What this component does and why it's needed

Technology Stack:

- Technology 1
- Technology 2
- Technology 3

Key Features:
1. Feature 1: Description
2. Feature 2: Description
3. Feature 3: Description

Dependencies:
- Depends on Component X for Y

2.3 Component 2: [Name]

Purpose: What this component does and why it's needed

Technology Stack:
- Technology 1
- Technology 2

Key Features:
1. Feature 1: Description
2. Feature 2: Description

2.4 Component 3: [Name]

Purpose: What this component does and why it's needed

Technology Stack:
- Technology 1
- Technology 2

Key Features:
1. Feature 1: Description
2. Feature 2: Description

hitecture

2.5 High-Level Architecture

Insert architecture diagram here or describe the overall system structure

```
┌─────────────┐
│   Client    │
│  (Browser)  │
└──────┬──────┘
       │
       │ HTTPS
       ▼
┌─────────────┐
│  API Layer  │
│   (REST)    │
└──────┬──────┘
       │
       ├────────────┬────────────┐
       ▼            ▼            ▼
  ┌────────┐  ┌─────────┐  ┌─────────┐
  │Database│  │ Cache   │  │ Storage │
  └────────┘  └─────────┘  └─────────┘
```

2.6 Data Flow

Describe how data moves through the system:

1. User initiates action in frontend
2. Request sent to API layer
3. API validates and processes request
4. Data persisted/retrieved from database
5. Response returned to client
6. UI updates to reflect changes

2.7 Technology Stack

2.7.1 Frontend

- **Framework:** React / Vue / Svelte / etc.
- **State Management:** Redux / Zustand / etc.
- **Styling:** Tailwind / CSS Modules / etc.
- **Build Tool:** Vite / Webpack / etc.

2.7.2 Backend

- **Runtime:** Node.js / Python / Go / etc.
- **Framework:** Express / FastAPI / Gin / etc.
- **API Style:** REST / GraphQL / gRPC

2.7.3 Database

- **Primary:** PostgreSQL / MongoDB / etc.

- **Caching:** Redis / Memcached
- **Search:** ElasticSearch / Algolia (if applicable)

2.7.4 Infrastructure

- **Hosting:** AWS / GCP / Azure / Vercel / etc.
- **CI/CD:** GitHub Actions / GitLab CI / etc.
- **Monitoring:** Datadog / New Relic / Sentry / etc.

3 Security Considerations

3.1 Authentication & Authorization

- How users will authenticate
- What authorization model we'll use (RBAC, ABAC, etc.)
- Token management strategy

3.2 Data Protection

- Encryption at rest and in transit
- PII handling and privacy concerns
- Compliance requirements (GDPR, CCPA, etc.)

3.3 Security Best Practices

1. Input validation and sanitization
2. SQL injection prevention
3. XSS protection
4. CSRF protection
5. Rate limiting
6. Security headers

Security Review: This design should undergo security review before implementation begins.

4 Testing Strategy

4.1 Unit Testing

- Coverage target: 80%+
- Key areas requiring unit tests
- Testing framework and tools

4.2 Integration Testing

- API endpoint testing
- Database integration tests
- Third-party service integration tests

4.3 End-to-End Testing

- Critical user flows to test
- Testing tools (Playwright, Cypress, etc.)
- Test environment setup

4.4 Performance Testing

- Load testing approach
- Performance benchmarks
- Scalability targets