TextCasePro SaaS - Software Development Lifecycle (SDLC)

1. Software Requirement Specification (SRS)

1. SOFTWARE REQUIREMENT SPECIFICATION (SRS)

Project Name: TextCasePro SaaS

Purpose:

To provide a web-based SaaS API for converting text (uppercase/lowercase) using API keys and a credit-based usage system.

Actors:

- User
- Admin

Features:

- Register/Login with OAuth2
- Generate and use API Keys
- Convert text using API Key and available credits
- View and purchase credits
- Admin can approve credit requests

Non-Functional Requirements:

- Async performance (FastAPI + async SQLAlchemy)
- Secure authentication using JWT
- API key-based access control

2. High Level Design (HLD)

2. HIGH LEVEL DESIGN (HLD)

Architecture: Modular FastAPI app with services, routers, and schema separation.

Modules:

- `account`: Handles user auth, session, registration, email verification, password reset
- `converter`: Handles API key, credit, text conversion
- `db`: Handles DB setup using async SQLAlchemy 2.0

Technology Stack:

- Backend: Python 3.11, FastAPI
- ORM: SQLAlchemy 2.0 (async)
- DB: SQLite (dev), can upgrade to PostgreSQL
- Auth: OAuth2 + JWT (access/refresh tokens)
- API Key: Custom header + token model

3. Low Level Design (LLD)

3. LOW LEVEL DESIGN (LLD)

Main Classes:

- `User`, `RefreshToken` -> Auth & identity
- `APIKey`, `UserCredits`, `CreditRequest` -> Converter system
- `ConvertRequest` -> Operation-level schema

Relationships:

- One user -> One API key
- One user -> One UserCredits
- One user -> Many CreditRequests

Business Logic:

- `generate_user_api_key()`: deletes old key, saves new
- `handle_conversion()`: validates API key, deducts credit, returns result
- `submit_credit_request()` + `approve_credit_request()`: request/approve credits

4. UML Diagrams Summary

4. UML DIAGRAMS SUMMARY

Use Case Diagram

- Actors: User, Admin

- Use Cases: Register, Login, Convert, Request Credits, Approve Credits, Generate API Key

ER Diagram

- Tables: User, RefreshToken, APIKey, UserCredits, CreditRequest

- Relations: FK between user -> APIKey/UserCredits/CreditRequest

Sequence Diagram (Conversion Flow)

1. User -> `/convert` with API Key

2. System validates key, checks credits

3. If valid: deduct 1 credit, return converted result

5. Security Model

5. SECURITY MODEL

- OAuth2 token-based login (Access + Refresh tokens)
- API Key-based access with header: `X-API-Key`
- Token revocation (logout, refresh token expiry)
- Admin routes protected with `require_admin` dependency

6. Testing Strategy

6. TESTING STRATEGY

- Unit Tests: for services (`hash_password`, `create_tokens`, `convert_text`)
- Integration Tests: Auth + conversion API endpoints using `TestClient`
- Manual Testing: API key generation, credit handling
- Suggest CI/CD with Pytest + GitHub Actions

7. API Specification

7. API OVERVIEW

Auth Module (`/api/account/`)

- POST `/register`, `/login`, `/logout`

- GET `/me`, `/verify`, POST `/verify-request`, `/change-password`, `/forgot-password`, `/reset-password`

Converter Module (`/api/convert/`)

- POST `/generate-api-key`, `/convert`

- GET `/me/api-key`, `/me/credits`, `/credit-requests`

- POST `/buy-credits`, `/approve-credit/{id}`