Software Design Specification for Blood-Pressure Track

Version 1.0

Prepared by

Arisara Suriyawong Poon	**6522781564**	**6522781564@g.siit.tu.ac.th**

Date: **July 9, 2025**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0		BP-Track is a health-focused platform designed to simplify and secure blood pressure monitoring for both patients and doctors. It offers a comprehensive set of features including image-based data extraction using OCR, detailed tracking, and doctor-patient access management. With OTP-based authentication, and role-specific workflows, the system ensures data privacy, accessibility, and smooth collaboration between users and healthcare providers.	

1 Introduction

1.1 Document Purpose

The purpose of this document is to provide a comprehensive description of the software system that allows users to engage in blood pressure records. Additionally, this platform allows user interactions through requests for permission and gives permission to other users. The SRS covers all aspects of the system, ensuring a secure and user-friendly platform that prevents fraudulent activities and secure users information. This document will guide developers, designers, and stakeholders throughout the project lifecycle.

1.2 Product Scope

The blood pressure record application is a mobile application that enables users to record their blood pressure and share records to authenticate users. The platform gives 2 different roles to users: patient and doctor, allowing patients to record their blood pressure and give permission to doctors, and allowing doctors to view authenticated blood pressure records and request patients for permission.

This system prioritizes privacy, data integrity, and healthcare collaboration, the system incorporates features such as OTP-based authentication, role-specific access control, and secure session management. The platform ensures trust between users by providing transparency in access permissions, secure communication, and protection against unauthorized data exposure.

1.3 Intended Audience and Document Overview

This Software Requirements Specification (SRS) document is primarily intended for the client and the professor:

- Clients: Individuals who will use the Blood-Pressure Track application to record blood pressure and share records to authenticated doctors. This document helps them understand the platform's capabilities and security features.
- Professor (Instructor): The evaluator of the project, ensuring that the system meets software engineering principles, functional requirements, and project objectives.
  The SRS is organized into the following sections:
  1. Introduction: Overview of the Blood-Pressure Track application, its purpose, and objectives.
  2. Overall Description: Explanation of system functionalities, scope, and design constraints.
  3. Specific Requirements: Detailed functional requirements, user interactions, and system capabilities.
  4. Other Non-Functional Requirements: Performance, security, and software quality attributes.
  5. Requirements Traceability: Mapping of requirements for verification and project tracking.
  6. Appendices: Supporting documents such as a data dictionary and team collaboration logs.

Recommended Reading Sequence

- Clients should begin with the Introduction and Overall Description to understand the platform's purpose and functionalities.
- Professor should review the Introduction and then focus on Specific Requirements to evaluate the completeness and feasibility of the system design.

1.4 Definitions, Acronyms and Abbreviations

- API: Application Programming Interface is the set of rules and protocols for building and interacting with software applications.
- AI: Artificial Intelligence is the simulation of human intelligence in machines that are programmed to think and learn.
- MB: Megabyte is a unit of data storage equal to 1,024 kilobytes (KB) or approximately one million bytes.
- KB: Kilobyte is a unit of data storage equal to 1,024 bytes, often used to measure the size of small files or storage capacity.
- IEEE: Institute of Electrical and Electronic Engineers is a professional association for electronic engineering and electrical engineering.
- IP: Intellectual Property is a creation of the mind such as inventions, literary and artistic works, designs, symbols, names, and images used in commerce.
- OTP: One Time Password is a password that is valid for only one login session or transaction, used for security purposes.
- PDPA: Personal Data Protection Act is a law that governs the collection, use, and disclosure of personal data by organizations.

- QR: Quick Response is a type of matrix barcode (or two-dimensional barcode) that can be read by a QR scanner or a smartphone camera.
- SMS: Short Message Service is a text messaging service component of most telephone, internet, and mobile device systems.
- SRS: Software Requirements Specification is a document that describes what the software will do and how it will be expected to perform.
- UI: User Interface is the space where interactions between humans and machines occur.
- UX: User Experience is a person's emotions and attitudes about using a particular product, system, or service.
- UML: Unified Modeling Language is a standardized modeling language in the field of software engineering.
- 2FA: Two-Factor Authentication is a security process that requires two different authentication factors to verify a user's identity.
- JPG: Joint Photographic Experts Group is a commonly used method of lossy compression for digital images.
- PNG: Portable Network Graphics is a raster-graphics file format that supports lossless data compression.

1.5 Document Conventions

This document follows these formatting requirements:

- The standard font used throughout the document is Arial.
- The font size for general text is 11.
- Font size 12 may be used for emphasis in special cases.
- Headers, topics, and section titles are in bold.

1.6 References and Acknowledgments

The following references have been used in the creation of this SRS:

- COMET Method for Software Design: A structured approach to software design and modeling.
- UML Modeling Guidelines: Standards and best practices for Unified Modeling Language (UML) diagrams.
- Security Best Practices for Online Platforms: Guidelines for securing user authentication, transactions, and intellectual property.
- Laws and Regulations Related to Digital Data: Compliance requirements for digital data protection, intellectual property management, and fraud prevention.

2 Overall System Design

2.1 Architectural Design

Presentation Layer (Frontend)

- Provides the application interface for users (patients, doctors).
- Handles user inputs, displays blood pressure records, requests and grants permission.
- Technologies: React/Vue.

Application Layer (Backend Logic)

- Manages business logic such as bidding rules, user authentication, artwork posting, and transaction handling.
- Implements APIs that are consumed by the frontend.
- Technologies: FastAPI/Python.

Database Layer (Data Repository)

- Stores user profiles, blood pressure records, patients and doctors relations, OTP code, and access request data.
- Maintains data integrity and supports queries across user and artwork data.
- Technologies: PostgreSQL.

Third-Party Integration Subsystem

- Includes APIs for email services and SMS services.
- Interacts with external systems for secure authentication.

Security and Monitoring Module

- Handles 2FA, role-based access control (RBAC), and encryption of sensitive data.

2.2 Hardware Interface Design

As Art-Horizon is a web-based system, hardware requirements are minimal and mostly on the client side and server infrastructure.

Hardware Interfaces:

- **Client Devices:** Desktops, laptops, tablets, or smartphones with internet access and modern web browsers.
- **Web Server/Cloud Hosting:** Hosted on cloud infrastructure (e.g., AWS EC2, Azure VMs) to serve application requests.
- **Database Server:** Cloud-managed or dedicated server for running a relational or NoSQL database.

2.3 Software Interface Design

The system interacts with several software interfaces and external APIs:

Software/API Interfaces:

- **RESTful APIs:** Backend exposes endpoints for login, registration, user profile, bidding, blood pressure record, patient view, and doctor view.
- **OAuth 2.0:** Used for social login via Google or Facebook accounts.
- **Notification Services:** Email APIs (e.g., SendGrid, Mailgun), SMS APIs

2.4 Data Design

2.4.1 Data Description

The system uses a relational database model to structure its information domain. Each entity in the system is mapped to a table, with attributes defined as columns. Relationships between entities are enforced via foreign keys.

Key Design Principles:

1. Storage:
   - Core entities (e.g., Users, BloodPressureRecord, OTPCode, and UserSession) are stored in normalized tables to minimize redundancy.
   - Relationships include doctor-patient connection and access control requests using join tables that are managed via foreign keys.

2. Processing:
   - Login sessions are managed with UserSession, allowing token expiration, IP/device tracking, and rate-limiting for sensitive routes to mitigate abuse.
   - Uploaded blood pressure images are processed using Gemini AI OCR, which extracts systolic, diastolic, and pulse values. The extracted data is stored in the BloodPressureRecord table.
   - Doctor-Patient 2 ways authentication: 1. Doctors can send access requests via the AccessRequest table. Patients can view, accept, or reject them. 2. Patients can also proactively authorize doctors via the DoctorPatient table, instantly granting access.

3. Organization:
   - Tables are indexed on their primary keys (e.g., users.id, blood_pressure_records.id) and on high-frequency query fields (e.g., otp_codes.contact_target, access_requests.status, doctor_patients.doctor_id) to support fast lookups.
   - The DoctorPatient and AccessRequest tables provide a modular design by decoupling access control from the main user record, enabling many-to-many relationships and fine-grained access tracking.
   - Timestamps like created_at and updated_at are included across tables to support audit trails, activity monitoring, and temporal analysis.

Databases/Storage:

- Primary Database: Relational database for structured data.
- File Storage: Cloud storage for images and digital artwork.
- Caching Layer: Redis for high-frequency queries (e.g., active auctions, user sessions).

1. User Entity → Users Table

- Stores registration information, profile details, and authentication credentials.
- Each user has a unique ID and role such as doctor and patient.
- Passwords are stored securely using hashing algorithms.

2. Blood Pressure Entity → BloodPressureRecord Table

- Includes blood pressure id, user id, systolic, diastolic, pulse, measurement date, and measurement time.
- Blood pressure records themselves are stored in a cloud-based object storage system (e.g., Google Cloud SQL).

3. Doctor-Patient Entity → DoctorPatient Table

- Linked doctor and patient relation.
- Each record includes doctor ID, patient ID, and optional hospital affiliation.
- Supports many-to-many linkage between users by referencing the Users table twice.

4. Access Request Entity → AccessRequest Table

- Manages doctor requests to access a patient's blood pressure records.
- Each entry includes doctor ID, patient ID, request status (pending, accepted, rejected), and creation timestamp.
- Supports patient-controlled access control, allowing approval or denial of doctor requests.

5. User session Entity → UserSession Table

- Tracks active login sessions for each user across devices.
- Stores session token, device metadata, IP address, and expiration time.
- Used for multi-device login tracking, logout, and session validation.

2.4.2 Data Dictionary

Entity	Type	Description
User	Table	Stores user profiles (id, full name, email, phone number, role, user details).
BloodPressureRecord	Table	Store blood pressure records (systolic, diastolic, pulse)
DoctorPatient	Table	Store relationship between doctor and patient.
AccessRequest	Table	Manages doctor requests to access a patient's blood pressure records.
UserSession	Table	Store when and where user login

Functions and Parameters

Functions	Parameters	Purpose
generate_request_id()	-	Generates a unique UUID string for tracking requests.
generate_otp()	-	Generates 6 digits OTP for registration, login, password reset, and contact verification.
hash_otp()	otp: str	Hashes the OTP using SHA-256 for secure storage.
hash_password()	password: str	Hashes a user password using bcrypt.
verify_password()	password: str, hashed: str	Verifies a password against its bcrypt hash.
create_access_token()	data: dict, expires_delta: Optional[timedelta] = None	Generates a JWT access token with expiration.
create_refresh_token()	data: dict	Generates a refresh token valid for 30 days.
send_email_otp()	recipient_email: str, otp: str, purpose: str	Sends OTP via configured email provider.
send_sms_otp()	phone: str, otp: str, purpose: str	Sends OTP via configured SMS provider.

create_standard_response()	status, message, data, meta, errors, request_id	Generates a consistent API response schema.
get_image_metadata()	image_path: str	Extracts EXIF metadata from an image file.
verify_api_key()	api_key: str	Checks if the provided API key is in the list of valid keys.
get_current_user()	credentials: HTTPAuthorizationCredentials, db: Session	Validates JWT and returns the current authenticated user.
is_account_locked()	user: User	Checks if the user's account is temporarily locked.
lock_account()	user: User, db: Session	Locks a user account for 30 minutes after failed logins.
read_blood_pressure_with_gemini()	image_path: str	Uses Gemini API to extract BP values from images and return structured OCR results.

API endpoints and Parameters

Functions	Parameters	Purpose
POST /api/v1/auth/request-otp	email/phone_number, purpose	Request OTP for registration, login, password reset, and contact verification.
POST /api/v1/auth/verify-otp	email/phone_number, otp_code, purpose	Verifies an OTP sent to email or phone.
POST /api/v1/auth/register	full_name, password, email/phone_number, role, optional profile fields	Registers a new user (patient or doctor), requires prior OTP verification.
POST /api/v1/auth/login	email/phone_number, password, remember_me	Authenticates a user and returns JWT token pair.
POST /api/v1/auth/logout	(JWT token)	Logs out and deactivates sessions.
POST /api/v1/auth/change-password	current_password, new_password, confirm_new_password	Changes password for authenticated users.

POST /api/v1/auth/reset-password	email/phone_number, otp_code, new_password, confirm_new_password	Resets password using OTP.
POST /api/v1/auth/verify-contact	otp_code, email/phone_number, purpose	Verifies a new email or phone number for an existing user.
GET /api/v1/users/me	(JWT token)	Retrieves the current user's profile.
PUT /api/v1/users/me	profile fields (e.g., blood type, height, etc.)	Updates the authenticated user's profile.
GET /api/v1/users/search	q (query string), role (optional: "doctor" or "patient"), page, per_page	Searches for users by full name, ID, or medical license.
GET /api/v1/bp-records	page, per_page, start_date, end_date	Lists blood pressure records of the user with pagination.
POST /api/v1/bp-records	systolic, diastolic, pulse, measurement_date, measurement_time, notes.	Creates a new BP record.
GET /api/v1/bp-records/{record_id}	record_id	Retrieves a specific BP record.
PUT /api/v1/bp-records/{record_id}	record_id, update fields	Updates a specific BP record.
DELETE /api/v1/bp-records/{record_id}	record_id	Deletes a specific BP record.
POST /api/v1/ocr/process-image	image file (UploadFile)	Uses Google Gemini AI to extract BP values from an image.
GET /api/v1/stats/summary	days (default: 30, max: 365)	Returns summary statistics (average, min, max) of systolic, diastolic, and pulse values over the last N days.
POST /api/v1/patient/authorize-doctor	doctor_id	Patient gives a doctor direct access to their BP records.

GET /api/v1/patient/authorized-doctors	(JWT token)	Patient views a list of doctors they have authorized to access their blood pressure records.
DELETE /api/v1/patient/authorized-doctors/{doctor_id}	doctor_id	Patient revokes access previously granted to a doctor.
GET /api/v1/patient/access-requests	(JWT token)	Patient views pending access requests from doctors.
POST /api/v1/patient/access-requests/{request_id}/approve	request_id	Patient approves a doctor’s access request.
POST /api/v1/patient/access-requests/{request_id}/reject	request_id	Patient rejects a doctor’s access request.
POST /api/v1/doctor/request-access	patient_id	Doctor requests access to a patient's blood pressure data.
GET /api/v1/doctor/access-requests	(JWT token)	Doctor views their sent access requests (pending, approved, or rejected).
GET /api/v1/doctor/patients	page, per_page	Doctor views a paginated list of patients who authorized access.
GET /api/v1/doctor/patients/{patient_id}/bp-records	page, per_page, start_date, end_date	Doctor views blood pressure records of an authorized patient.
DELETE /api/v1/doctor/access-requests/{request_id}	request_id	Doctor deletes a pending access request.