Low Level Design

# Realtime Health Guardian

|  |  |
| --- | --- |
| **Written By** | Sachin moze |
| **Document Version** | 0.1 |
| **Last Revised Date** |  |

**DOCUMENT CONTROL**

## Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **AUTHOR** | **COMMENTS** |
| 0.1 | 15- April -  2024 | Sachin Moze | Introduction and document written |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSION** | **DATE** | **REVIEWER** | **COMMENTS** |
|  |  |  |  |

**Approval Status:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **VERSION** | **REVIEW**  **DATE** | **REVIEWED BY** |  | **APPROVED BY** | **COMMENTS** |
|  |  |  |  |  |  |

# Contents

#### Introduction 04

* 1. **What is Low-Level Design Document? 04**

|  |  |  |
| --- | --- | --- |
| **1.2** | | **Scope 04** |
| **2.** | **Architecture 05** | |
| **3.** | **Architecture Description 08** | |
|  | **3.1** | **Data Description 08** |
|  | **3.3** | **Data Transformation 08** |
|  | **3.4** | **Data insertion into database 08** |
|  | **3.5** | **Connection with SQL server 08** |
|  | **3.5** | **Export Data from database 12** |
|  | **3.6** | **Deployment 12** |
| **4.** | **Unit** | **test cases 15** |

# Introduction

## What is Low-Level design document?

The goal of the LDD or Low-level design document (LLDD) is to give the internal logic design of the actual program code for the realtime health guardian application. LDD describes the class diagrams with the methods and relations between classes and programs specs. It describes the modules so that the programmer can directly code the program from the document.

## Scope

This document focuses on the technical implementation details of the Realtime Health Guardian application. It covers the architecture, data description, web scraping, data transformation, database operations, deployment.

# Architecture

# The architecture of the Realtime Health Guardian application follows a client-server model. It consists of a front-end interface for users, a back-end server for processing data, and integration with wearable devices for real-time health monitoring.

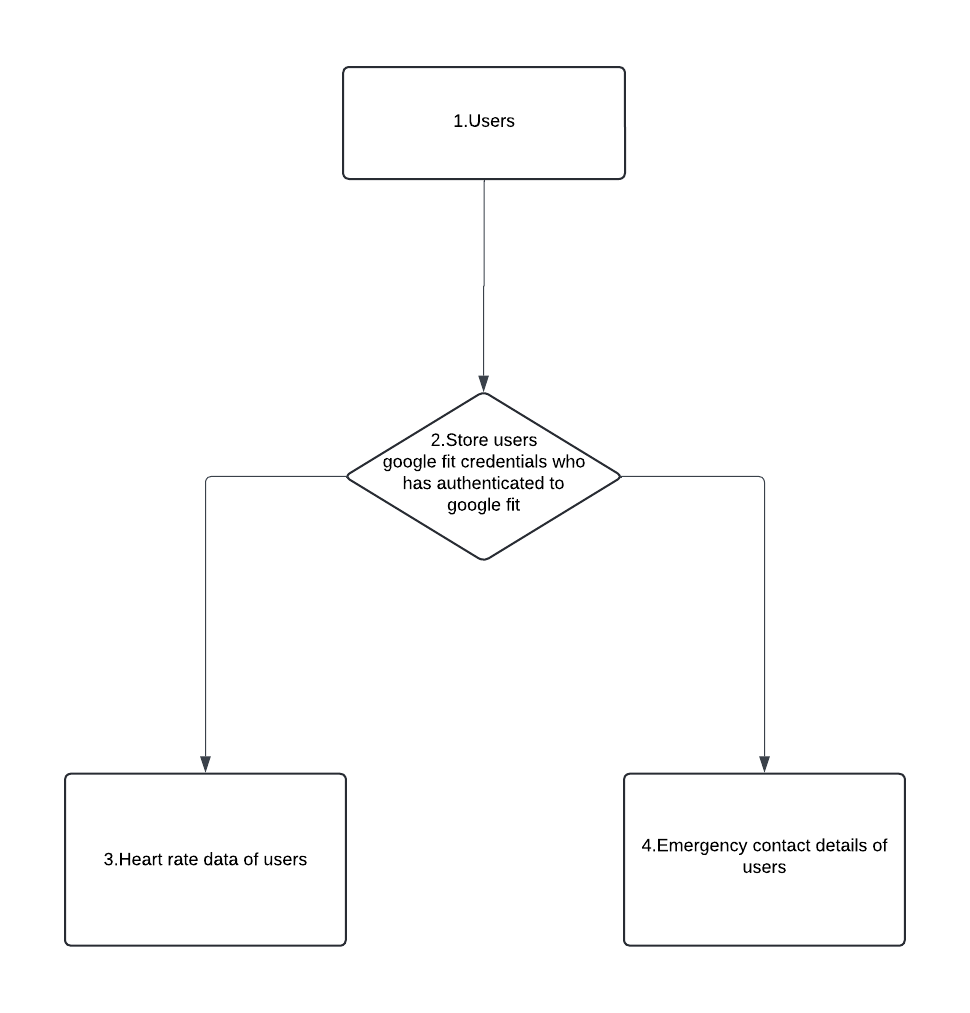
# A diagram of a smart watch Description automatically generated

**Database Architecture**

The following diagram shows Database architecture:

**LOW LEVEL DESIGN**

**6**



# Architecture Description

## Data Description

Data collected from wearable devices includes heart rate readings, timestamps, and user identification. This data is transmitted to the server for processing and analysis.

## Data Transformation

Upon receiving data from wearable devices and web scraping, data transformation processes are applied to clean, filter, and format the data for analysis. This involves removing duplicates, handling missing values, and standardizing data formats.

## Data Insertion into Database

1. The transformed data is inserted into a relational database for storage and retrieval. SQL queries are used to create tables, define schemas, and perform data insertion operations.

## Make the SQL connection and set up the data source

## The application establishes a connection with the SQL server using database management system (DBMS) libraries such as SQLAlchemy in Python. This connection allows the application to execute SQL queries and retrieve data from the database.

## Export Data from Database

Data stored in the database can be exported in various formats such as CSV, JSON, or Excel for further analysis or reporting purposes. Export functionalities are implemented to enable users to extract insights from the stored data.

## Deployment.

The Realtime Health Guardian application is deployed on OnRender cloud platform. Deployment involves configuring servers, setting up databases, and ensuring scalability and availability of the application.

**12**