System Test Documentation

for

Cloud Doctor

Version 1.0 approved

Prepared by

Halil Burak Noyan
Cüneyit Kiriş
Gökçen Nurlu
Can Carlak

BiGC^2

Table of Contents

1. Introduction

1.1. Problem Definition

This software product will be built in four main parts, and these parts will have their own design fashions. These parts are; embedded, intermediate, web server, mobile application, respectively. Distance between a patient to be monitored and the observer affects the quality of monitoring in terms of latency and sampling rates of vital data. Current solutions for a high-accurate health monitoring needs great workforce, expensive or limits the mobility of the patient. There is also no available affordable solution for a easy to setup, extensible and complete system, that could be used by caretakers and family members, to track toddlers and elders.

Moreover, the project aims to develop a health monitoring system so that, designed physiological sensing system can provide reliable vital signs measurements and incorporating real-time decision support. The project can be used to track a patient for a period of time and to detect immediate changes on individual that can be sign of an emergency. Modularity of this project will also provide different configurations for different targets, such as elders, adults and babies.

1.2 Purpose and Scope

The purpose of this document is to provide the test cases of the Cloud Doctor project. It defines the objective, scenario, expected outcomes and procedural requirements for each test case. It also includes a table showing which test case is related to which one. The software will be tested using guidance of this document. Although it covers all the test cases specifically in detail, a little portion of the details is subject to change in test phase.

2. Details for system test plan

This section describes the specific items to be tested at different levels and provides a Test Traceability Matrix that links the items to be tested with the requirements.

2.1. Test Items and Their Identifiers

Since the system consist of four subsystems which can be identified as components or levels, each subsystem is an object of tests. Integration of these components shall be included to tests as well.

At embedded component level, there are both hardware and software items which are subject to tests. There are currently three types of sensors available for measurements. Pulse sensor, temperature sensor and accelerometer are sensor items of embedded component. These identified sensors are connected to an Arduino Uno board. Arduino has an embedded software item for reading and packing sensor data. Arduino board also employs a bluetooth module for remote transmission of sensor data. Protocol used between Arduino board and intermediate component is identified as an object used for communication.

At intermediate component level, a Raspberry Pi B+ mini computer is used with Raspbian operating system which has bluetooth module and wi-fi module attached to it. Hardware items of intermediate component can be identified as listed above. Identified as intermediate software, a software item is implemented in order to obtain, analyze, pack and send sensor data at intermediate level. Software items can be categorized as five sub-components. They are bluetooth handler, analysis handler, tcp handler, configuration handler and self-management handler.

At cloud component level, four types of software, a database and a protocol can be identified. First software is for collecting user data directed from variable number of intermediate devices. Collected data is written to database. Second is detailed long-run analysis sub-component. Third software is web application that provides a user-interface to serve users for observation and configuration. Fourth is the notification sub-component. In case of emergencies, this piece of software works to send notifications to observers.

At mobile application level, application is an item under heavy development. Notifications are sent to mobile phone detached from the application.

2.2. Features to be Tested

Software in Arduino Board is going to be tested deployed with different combinations of sensors. Bandwidth of bluetooth module and serial port of Arduino will be in the features to be tested.

Bluetooth handler, analysis handler, tcp handler, configuration handler and self-management handler in Raspberry Pi will be tested both individually and integrated.

Database in web server will be tested for different cases. Data collection software will be tested connected to the database. Analysis tests are going to be held with differentiating conditions.

Web application will have its individual tests as well as notification software.
Phone application tests will be held for notifications and availability.
Protocols between components of system are subjects to the test. System-wide integration can be included in features to be tested as well.

2.3. Features not to be Tested

Device dependency tests for hardware architectures other than already stated ones will be omitted. Arduino board is able to employ more than three sensors but no additional sensors will be tried. Rationale behind that is lack of available hardware components and undecided design.

Third-party open source libraries usually conduct their own tests, so tests for those libraries will only take place in integration and unit tests of software implemented.

User stress tests for server will not exceed a certain limit because, for a brand new product, hardware requirements for scaling are subjects of successive test documents. Security tests will also be postponed for this version of document.

For each level, tests will be held using simulated inputs flow originated from lower levels since real world inputs are not sufficiently required.

2.4. Approach

Every level of test have it's individual approach. Although detailed information can be found in chapter 4, test approaches are summarized below:

- Embedded Component Level: Behaviour of embedded software shall be stable. Considering this requirement, it is expected to obtain unaltered sensor measurements. But real world measurements tend to be non-deterministic. Not automated but manual observation is required within scope of this level, so tests will be held using analysis.

- Intermediate Component Level: For bluetooth handler and tcp handler, black-box integrity tests shall be held. Analysis handler is expected to identify emergency. Black-box technique can be employed in order to test correctness of analysis. Configuration handler and self-management handler tests use white-box method, because internal mechanism of program is affected.
- Cloud Component Level: Database integrity tests shall be held using black-box method. Long-run analysis tests will be conducted using black-box technique in order to observe analysis accuracy. Web application and notification system outputs will be observed with action inputs, so analysis technique will be employed. For data-collection software, white-box test method will be used.

- Mobile Application Component Level: All tests regarding to mobile application will be handled using analysis test method.

2.5. Item Pass/Fail Criteria

There are several test categories which specify fail criteria for different tests:

- Integrity Checks: If integrity of values are not preserved during relay operations, test is failed.
- Connectivity Tests: If packages or connection is lost, test is failed.
- Analysis Accuracy Checks: If analysis accuracy is lower than 70%, test is failed.

- Integration Tests: For integration tests, predecessor component shall be available for test status to be passed, otherwise it is failed.
- Performance Tests: There is no specification of delays that is not negligible in this version of document.
- Unit Tests: Unit crash or permanent failure means that test is failed.

2.6. Suspension Criteria and Resumption Requirements

During integration tests, if one lower level fails, tests associated with the lower level are invalidated. In such a case, tests for both levels shall be repeated. For system integration tests, any type of component crash requires this test to be suspended. Unit tests for crashed item shall be conducted again.

3. Test Management

Best way to test the whole system is creating sub-system tests. This approach is superior to complete system testing because sub components are pretty big already. Their interactions and communications with each other should be well defined and work steadily. Finally the complete system integration and testing must be performed. Thus the workload of testing schema is as follows:

3.1. Testing a sub component within itself

This process ensures that the component is working without any interaction by any other sub components. To be able to conduct this test procedure for the embedded sub-system, in addition to cores of the hardware such as micro-controller, pulse sensor, temperature sensor, accelerometer/gyroscope, we need a serial connection to a workstation. This will allow us to be able analyze & debug the embedded code much more efficiently. On the other hand, remaining sub-systems do not depend on specialized hardware/software parts. Nevertheless, their unit tests are written by corresponding programming languages(Python / C) and deployed individually on them. For instance, Raspberry Pi runs intermediate sub-system tests on PyCharm IDE.

3.2. Testing the communication protocols and interactions between adjacent sub-system components

Purpose of this part is stabilize inter sub-system communications. Techniques are mostly composed of erroneous situation generation. As defects reveals the real life situations more realistically. Sending an invalid JSON object from intermediate component to cloud server might be an example. In fact, when exactly an exception occurs and how sub-systems reacts those are the heart of this test activity.

3.3. Integration of the complete system and testing

This is the final test scenario and describes how will the complete system work when it is deployed in real life. Thus, conducting stress tests for the system is necessary obviously. To make stress testing effective, several duplicate simulators are designed and executed. Those simulators mimic realistic behaviors of the sub-systems. As an example, embedded component simulators generate data for the intermediate components.

4. System Test Levels

4.1. Embedded Component Level

4.1.1. Embedded Test 1

Test Case Identifier	EMBEDDED-TEST-01
Objective	Correctness of temperature sensor measurements
Scenario	Heating and cooling the temperature sensor via external inputs
Input	Holding the analog temperature sensor for a while and blowing the cold air to the sensor afterwards.
Outcome	Temperature output of the micro-controller smoothly rises first and falls later on.
Requirements	Avoidance of extra ordinary noise interruption

4.1.2. Embedded Test 2

Test Case Identifier	EMBEDDED-TEST-02
Objective	Correctness of pulse sensor measurements
Scenario	Breathing heavily when equipped with pulse sensor on the finger
Input	Increasing rate of the person's heart rate
Outcome	Heart beat per minute data output of the micro-controller rises
Requirements	Equipping the pulse sensor properly so that no extra environmental light disturbs the led of the sensor

4.1.3. Embedded Test 3

Test Case Identifier	EMBEDDED-TEST-03
Objective	Correctness of accelerometer measurements
Scenario	Tilting, pushing, pulling and moving the accelerometer
Input	Changing accelerometer data with respect to the 3 axises
Outcome	Rapid changes for the accelerometer output of the micro-controller
Requirements	Calibration and sensitivity settings(4g) should be properly configured for the digital accelerometer sensor

4.1.4. Embedded Test 4

Test Case Identifier	EMBEDDED-TEST-04
Objective	Correctness of gyroscope measurements
Scenario	Tilting, pushing, pulling and moving the gyroscope
Input	Changing gyroscope data with respect to 3 rotational axises
Outcome	Rapid changes for the gyroscope output of the micro-controller
Requirements	Calibration and sensitivity settings should be properly configured for the digital gyroscope sensor

4.1.5. Embedded Test 5

Test Case Identifier	EMBEDDED-TEST-05
Objective	Bluetooth connection establishment
Scenario	Bringing the peer device closer and send it away later on
Input	Increasing the power level of the bluetooth signal, then decresing it.
Outcome	First, micro-controller sets up a bluetooth connection with the peer and sends measured data to it. Whenever the peer is far away from the micro-controller, connectivity drops. The whole scenario is looped again when the peer is getting closer.
Requirements	Bluetooth pass-key must be known by the pair prior to the connection

4.1.6. Embedded Test 6

Test Case Identifier	EMBEDDED-TEST-06
Objective	Simulated data models real life situation
Scenario	Simulation program runs on a workstation and generates continuous data for the intermediate sub component.
Input	Simulation convergence points are required before execution. In this way we can control the upper and lower limits of the generated data. Moreover falling rate can also be adjusted.
Outcome	Simulator should behave just like a real time embedded hardware. It should generate steady data most of the time, however at random time urgent conditions should be generated.
Requirements	Python interpreter and bluetooth module

4.2. Intermediate Component Level

4.2.1. Intermediate Component Test 1

Test Case Identifier	INTERMEDIATE-TEST-01
Objective	Configurations
Scenario	Configurations written to configuration file in intermediate device are

	being fetched by the software.
Input	“Config.json” which include system id, sensor and analyzer info, device and server identification, logger and cacher specifications
Outcome	Observing desired configurations in runtime using white-box method.
Requirements	Presence of syntactically correct config.json JSON file in required path.

4.2.2. Intermediate Component Test 2

Test Case Identifier	INTERMEDIATE-TEST-02
Objective	Getting Sensor Data
Scenario	Sensor data is obtained from embedded device. After this procedure, it is used by Analyzer threads. Analyzer threads shall be able to get the data concurrently and correctly.
Input	Data queue of a sensor periodically enqueued with new sensor data
Outcome	Periodically obtaining the same data from queue concurrently
Requirements	A simulator which generates sensor data or embedded device input is required

4.2.3. Intermediate Component Test 3

Test Case Identifier	INTERMEDIATE-TEST-03
Objective	Mutual Exclusion of Analyzer Threads and Bluetooth Handler Threads
Scenario	Bluetooth Handler pushes data to public queues which are available to Analyzer threads. Analyzers and Bluetooth Handler shall work mutually exclusive.
Input	Real world or simulated sensor data.
Outcome	Threads work concurrently and without starvation
Requirements	INTERMEDIATE-TEST-01, INTERMEDIATE-TEST-02 shall be successful

4.2.4. Intermediate Component Test 4

Test Case Identifier	INTERMEDIATE-TEST-04
Objective	Mutual Exclusion of Analyzer Threads and TCP Handler Threads
Scenario	Analyzers push data to public queues which are available to TCP Handler threads. Analyzers and TCP Handler shall work mutually exclusive.
Input	Real world or simulated sensor data pushed to input queues of analyzers
Outcome	Threads work concurrently and without starvation
Requirements	INTERMEDIATE-TEST-01, INTERMEDIATE-TEST-02 shall be successful

4.2.5. Intermediate Component Test 5

Test Case Identifier	INTERMEDIATE-TEST-05
Objective	Sensor data analysis Threads shall work effectively
Scenario	Analysis threads are required to obtain data from input queues, analyze them for any case of emergencies.
Input	Continuously filled input queues
Outcome	Emergent situations are successfully identified
Requirements	INTERMEDIATE-TEST- 03 shall be successful

4.2.6. Intermediate Component Test 6

Test Case Identifier	INTERMEDIATE-TEST-06
Objective	Regular Sender Periodic Send Operation
Scenario	Given a period by configuration file, regular sender thread works every N seconds to wipe out the output sensors, pack the sensor data and send them to cloud.
Input	Regular sensor data.
Outcome	Threads work concurrently and without starvation
Requirements	INTERMEDIATE-TEST- 05 shall be successful

4.2.7. Intermediate Component Test 7

Test Case Identifier	INTERMEDIATE-TEST-07
Objective	Urgent Sender Immediate Send Operation
Scenario	In case of emergency identified by analyzer, urgent sender thread sends the emergent situation data immediately to the server.
Input	Urgent sensor data.
Outcome	If network delays are ignored, there is no delay when sending data to cloud
Requirements	INTERMEDIATE-TEST- 05 shall be successful

4.2.8. Intermediate Component Test 8

Test Case Identifier	INTERMEDIATE-TEST-08
Objective	Bluetooth Connection Loss Recovery
Scenario	Bluetooth connection loss happens because of exceeding range or hardware, signal faults
Input	-
Outcome	Exception handler thread tries to establish the connection periodically until the connection is successfully established.
Requirements	-

4.2.9. Intermediate Component Test 9

Test Case Identifier	INTERMEDIATE-TEST-09
Objective	TCP Connection Loss Recovery
Scenario	TCP connection loss happens because of any kind of network failure
Input	-
Outcome	Exception handler thread tries to establish the connection periodically until the connection is successfully established.
Requirements	-

4.2.10. Intermediate Component Test 10

Test Case Identifier	INTERMEDIATE-TEST-10
Objective	Thread Exception Recovery
Scenario	Thread failures occur because of unhandled exceptions
Input	-
Outcome	In case of any thread failure, related thread is restarted.
Requirements	

4.2.11. Intermediate Component Test 11

Test Case Identifier	INTERMEDIATE-TEST-11
Objective	Caching of Sensor Data in Persistent Memory
Scenario	TCP connection loss happens because of any kind of network failure. In those situations, connection may not be established for a long time. It results with hogging the memory. In order to prevent this, sensor data shall be transferred to persistent memory in absence of tcp connection.
Input	-
Outcome	Sensor data is cached in persistent memory and pushed to cloud when connection is re-established
Requirements	INTERMEDIATE-TEST- 06, 07 shall be successful

4.2.12. Intermediate Component Test 12

Test Case Identifier	INTERMEDIATE-TEST-12
Objective	Logging and Send Logs
Scenario	Failures, warnings and necessary info required to be archieved. Failure types such as bluetooth connection loss or device fault recovery shall be known by the server.
Input	Noteworthy event
Outcome	Noteworthy events such as bluetooth connection loss or device fault recovery are logged to the server.
Requirements	INTERMEDIATE-TEST- 05 shall be successful

4.3. Cloud Component Level

4.3.1. Cloud Register Test 1

Test Case Identifier	Cloud- Register - 01
Objective	To test the register component of the system
Scenario	1. User clicks “Create New Account” button on application launch page. 2. User fills the required fields with valid data. 3. User clicks “Register” button.
Input	name, surname, address,phone email, password
Outcome	• A new user is created in the system. • User is redirected to the application launch page
Requirements	The application shall be started

4.3.2. Cloud Register Test 2

Test Case Identifier	Cloud- Register - 02
Objective	To test the register component of the system with an invalid e-mail
Scenario	1. User clicks “Create New Account” button on application launch page. 2. User fills the required fields : e-mail with an invalid data. 3. User clicks “Register” button.
Input	name, surname, address,phone email, password
Outcome	• The user is warned to enter a valid e-mail on register page
Requirements	The application shall be started

4.3.3. Cloud Register Test 3

Test Case Identifier	Cloud- Register – 03
Objective	To test the register component of the system with a not unique e-mail.
Scenario	a) User clicks “Create New Account” button on application launch page. b) User fills the fields name, surname, and password with valid data, e-mail with a previously registered user’s e-mail data. c) User clicks “Register” button.
Input	name, surname, address,phone email, password
Outcome	• The user is warned that entered e-mail address is already registered.
Requirements	The application shall be started

4.3.4. Cloud Register Test 4

Test Case Identifier	Cloud- Register – 04
Objective	To test the register component of the system with an invalid password.
Scenario	1. User clicks “Create New Account” button on application launch page. 2. User fills the required fields (name, surname, email) with valid data.

	3. User fills the “password” field with a password shorter than six characters. 4. User clicks “Register” button
Input	name, surname, address,phone email, password
Outcome	- “Invalid Password” error is displayed and a new user is not created.
Requirements	The application shall be started

4.3.5. Cloud Login Test 1

Test Case Identifier	Cloud- Login – 01
Objective	To test the login component of the system with valid data
Scenario	1. User clicks “Log In” button on application launch page. 2. User fills the required fields (email, password) with valid data. 3. User clicks “Log In” button.
Input	email, password
Outcome	- The user is logged in to the application. - New session is created for the user. - The user is redirected to main page.
Requirements	The application shall be started and user must be registered.

4.3.6. Cloud Login Test 2

Test Case Identifier	Cloud- Login – 02
Objective	To test the login component of the system with invalid credentials
Scenario	1. User clicks “Log In” button on application launch page. 2. User fills the required fields (email, password) with invalid data. 3. User clicks “Log In” button.
Input	email, password
Outcome	- The user is not logged in to the application. - A warning indicating that login credentials are wrong is shown. - The user stays on the login page.
Requirements	The application shall be started.

4.3.7. Cloud Forgot Password Test 1

Test Case Identifier	Cloud- Forgot Password – 01
Objective	To test the reset password feature of the system with valid data
Scenario	1. User clicks “Log In” button on application launch page. 2. User fills the email field with valid data. 3. User click “Forgot Password?” button on login page.
Input	email
Outcome	- New password is generated randomly and login credentials are updated in the database. - An email with randomly generated new password is sent to user. - A warning indicating that password has changed is shown. - The user stays on the login page.

Requirements	The application shall be started and user must be registered.

4.3.8. Cloud Forgot Password Test 2

Test Case Identifier	Cloud- Forgot Password – 02
Objective	To test the reset password feature of the system with invalid data
Scenario	1. User clicks “Log In” button on application launch page. 2. User fills the email field with invalid data. 3. User click “Forgot Password?” button on login page.
Input	email
Outcome	• New password is not generated • A warning indicating that credentials are wrong is shown. • The user stays on the login page.
Requirements	The application shall be started and user must be registered

4.3.9. Cloud Change Password Test 1

Test Case Identifier	Cloud- Change Password – 01
Objective	To test the change password functionality
Scenario	1. User clicks “Settings” tab in application launch page. 2. User clicks “Change Password” button. 3. User is redirected to a new page. 4. User enters a new password in the required field. 5. User clicks “Change Password” button.
Input	Password
Outcome	• The password associated with the user is changed in the database
Requirements	The application shall be started and user must be logged

4.3.10. Cloud Logout Test 1

Test Case Identifier	Cloud- Logout– 01
Objective	To test the logout functionality.
Scenario	1. User clicks “Logout” tab in page.
Input	None
Outcome	• The session should be invalidated. • The user is redirected to application launch page.
Requirements	The application shall be started and user must be logged

4.3.11. Cloud Personal Info Change Test 1

Test Case Identifier	Cloud- Personal Info Change– 01
Objective	To test the changing personal info functionality.
Scenario	1. User clicks “Settings”. 2. User clicks “Personal Info” tab in page 3. User change the necessary fields then click “Save” button
Input	Age, weight, height, disease, body mass index
Outcome	• User configuration updated in database
Requirements	The application shall be started and user must be logged

4.3.12. Cloud Notification Test 1

Test Case Identifier	Cloud- Notification System– 01

Objective	To test the notification system platform change functionality.
Scenario	1. User clicks “Settings”. 2. User clicks “Notification” tab in page 3. User change the platforms then click “Save” button
Input	Mobile Notification, E-mail Notification, Desktop Notification
Outcome	- User configuration updated in database
Requirements	The application shall be started and user must be logged

4.3.13. Cloud Add Observer Test 1

Test Case Identifier	Cloud- Add Observer– 01
Objective	To test the adding observer to patient functionality.
Scenario	1. User clicks “Settings” tab in page 2. User clicks “Add Observer” tab in page. 3. User enter email with valid data 4. Observer request sent to this email to verify and register the system
Input	Email
Outcome	- The observer which received invitation ,verify the email and register the system - The user can see pending/confirmed invitation
Requirements	The application shall be started and user must be logged

4.3.14. Cloud Remove Observer Test 2

Test Case Identifier	Cloud- Remove Observer– 02
Objective	To test the removing observer to patient functionality.
Scenario	1. User clicks “Settings” tab in page. 2. User clicks “Remove Observer” tab in page 3. User select observer(s) to remove from list
Input	None
Outcome	- Selected user(s) removed from list and the observer(s) are notified
Requirements	The application shall be started and user must be logged

4.3.15. Cloud Display Real Time Information Test 1

Test Case Identifier	Cloud- Display Real Time Info– 01
Objective	To test the display RT graphics is displayed correctly.
Scenario	1.User clicks “Dashboad” tab in page.
Input	None
Outcome	- 3 different RT graph which belongs to user are displayed - Each graph has own statistics on it -
Requirements	The application shall be started and user must be logged

4.4. Mobile Application Component Level

4.4.1. Mobile Test 1

Test Case Identifier	MOBILE-TEST-01
Objective	Testing the Notification System’s Registration procedure
Scenario	When a user installs the mobile application and runs it, he or she will be welcomed with a login page, where one can use same credentials with web application.
Input	Providing the username and password
Outcome	The mobile application registers the mobile phone to server automatically.
Requirements	User has a mobile phone, with notification application installed. Internet connection on mobile phone exists.

4.4.2. Mobile Test 2

Test Case Identifier	MOBILE-TEST-02
Objective	Testing the Notification
Scenario	An event, such as ‘dangerously high temperature’, occurred that needs a notification to be sent to an ‘observer’ type user. Then for such cases, our server sends notifications to related users.
Input	-
Outcome	A notification appears on notification bar of the smartphone.
Requirements	User has a mobile phone, with notification application installed, and the application has been launched once (to get registered to server to get notifications). Internet connection on mobile phone exists.

4.5. System Integration Level

4.5.1. System Test 1

Test Case Identifier	SYSTEM-TEST-01
Objective	Testing the whole data flow from sensor to web server
Scenario	Sensors on hardware are working and sampling data with an arbitrary rate.
Input	Sensor’s Input
Outcome	Data is transmitted to the web server(database actually)
Requirements	User has sensor hardware, intermediate device working. Web server is running. Internet connection on mobile phone exists.

4.5.2. System Test 2

Test Case Identifier	SYSTEM-TEST-02
Objective	Testing the flow with critical sensor data
Scenario	Sensors on hardware are working and sampling data with an arbitrary rate. At a moment, sensors sample critical value.
Input	Sensor’s Input

Outcome	Data is transmitted to the web server with additional information.
Requirements	User has sensor hardware, intermediate device working. Web server is running. Internet connection on mobile phone exists.

4.5.3. System Test 3

Test Case Identifier	SYSTEM-TEST-03
Objective	Testing the flow with from sensor to web server in an environment with unstable internet
Scenario	Sensors on hardware are working and sampling data with an arbitrary rate. Data will be waited and stored when there is connection available. Ultimately, data will arrive with correct datetime stamp.
Input	Sensor's Input
Outcome	Data is transmitted to the web server with correct datetime stamps.
Requirements	User has sensor hardware, intermediate device working. Web server is running.

4.5.4. System Test 4

Test Case Identifier	SYSTEM-TEST-04
Objective	Testing the data flow when sensor is stopped
Scenario	When sensor is stopped, webserver must be aware of that.
Input	Sensor's Input **Absence**
Outcome	Log is transmitted to webserver.
Requirements	User has sensor hardware stopped. Intermediate device working. Web server is running.