Politecnico di Milano

AA 2018-2019

Computer Science and Engineering

Software Engineering 2 Project

By TrackMe

ITD

Implementation and test document

Link to the zip attachments:

https://github.com/tompere96/PeressonZiffer/tree/master/DeliveryFolder

Link to the source code:

https://github.com/tompere96/PeressonZiffer/tree/master/Implementation

https://github.com/tompere96/PeressonZiffer/tree/master/ImplementationMobileApp

Tommaso Peresson - 845427

Giacomo Ziffer - 920905

Version 1.1 - 16/01/2019

SUMMARY

1 INTRODUCTION

1.1 SCOPE OF THE DOCUMENT

This document is the Implementation and Test Document (ITD) for Data4Help. The document describes the characteristics of the developed system, the requirements that are met in the first release, the possible future additions, the adopted development frameworks, the structure of the code, tests and installation instruction.

1.2 DEFINITIONS, ACRONYMS, ABBREVIATIONS

1.2.1 Definitions

- Private Customer: a customer that applies to the service Data4Help as a provider of personal health data and that can subscribe to AutomatedSOS

- Business Customer: a customer that applies to the service Data4Help as a user of the data acquired. It can be a single individual, such as a doctor, but also a company

- Anonymized Request: request made by a business customer, who asks for data grouped according to different possible parameters (i.e. place, age, etc.)

- Individual Request: request made by a business customer, who requests data belonging to a specific private customer, in this case the request must be accepted by the pc.

1.2.2 Acronyms

- [BC] as Business Customer

- [PC] as Private Customer

- [SSC] as Social Security Number

- [CF] as Codice Fiscale

- [SM] as System Manager

- [Gn]: n-goal.

- [Dn]: n-domain assumption.

- [Rn]: n-functional requirement.

1.3 REVISION HISTORY
- 1.0 – First Release (11/01/2019)
- 1.1 – Mistake corrected in Section 2.2 (16/01/2019)

1.4 REFERENCE DOCUMENTS
- RASD version 1.1
- Design Document version 1.1

2 GENERAL DESCRIPTION OF THE FUNCTIONALITIES

In this section, the development team states those goals/features described in the RASD that are implemented in the first prototype and those that will (possibly) be implemented in the following updates. For the first release the development team focused all the effort on the production of a working prototype, leaving out all the features and requirements that weren't mandatory.

2.1 LEFT OUT FUNCTIONALITIES AND FEATURES
In this section the development team will describe all the features that aren't implemented in the first release, motivating why these were postponed.

- HTTPS – Right now all the communications are done in plain text HTTP
  - o Motivation: This is a very simple addition do, but it requires a certificate from a certification authority.

- Data cashing on client – Right now the mobile app generates internally random user data to send to the server, if the internet connection is absent, the data will be lost.
  - Motivation: Not mandatory for a working prototype. We preferred not to develop this functionality to focus instead on the basic functionalities for using the services.

- Billing services – Right now we don't interact with any billing service
  - Motivation: Not mandatory for a working prototype. We preferred not to develop this functionality to focus instead on the basic functionalities for using the services

- Security – Security as a general concept was not a main concern in our development process due to its complexity and our time constraints, we focused on the stability of the of Data4Help as a system rather than worrying about its security.

Although not fundamental for a prototype, these features are mandatory for a public release.

2.1.1 Discrepancy from Design Document

In the design document we've chosen AWS Elastic Beanstalk as our platform of choice to deploy our Node.JS backend for Data4Help. We didn't realize at the time that we couldn't run our application due to memory constraints imposed by the free tier of AWS Elastic Beanstalk. As a backup we've chosen Heroku to temporarily deploy Data4Help's backend. Since this happened only because we couldn't afford a higher tier on AWS Elastic Beanstalk, we still recommend it as our choice of deployment platform, due to its flexibility and cost effectiveness.

2.2 FIRST RELEASE OF DATA4HELP

[G1] Allow a Visitor to become a Private Customer.
Requirements met: [R1] [R2] [R3]

[G2] Allow a Visitor to become a Business Customer.
Requirements met: [R2] [R3] [R5]

[G4] Allow a Private Customer to review personal data.
Requirements met: [R3] [R8]

[G5] Allow a Business Customer to monitor data from Data4Help.
Requirements met: [R3] [R9] [R10] [R11]

[G6] Allow a Business Customer to request data from Data4Help.
Requirements met: [R3] [R12] [R13]

[G7] Allow a Private Customer to share his real time position and health status by a Business Customer.
Requirements met: [R3] [R14] [R15] [R16]

[G8] Allow a Business Customer to subscribe to a data source like a specific PC or a geographical area.

    Requirements met: [R3] [R12] [R17] [R18] [R19]

[G10] Allow a System Manager to do operations of system maintenance.

    Requirements met: [R22]

    [G10.1] Allow a SM to verify and accept the request of appliance from a BC.

        Requirements met: [R23] [R24]

See RASD v1.1 to have more information about the requirements.

3 ADOPTED DEVELOPMENT FRAMEWORK

In this section the development team describes more in depth the frameworks used, to develop Data4Help client and server.

3.1 CLIENT SIDE

The client application is designed to run on Android devices. The code was written in *Java* using IntelliJ as IDE using the Android SDK to build the application. Debugging was done on an Android Device using *adb* (Android Debug Bridge). Gradle was used as source and build manager. Other framework used:

- *AsyncHttpClient:* To handle all the http requests between the server and the client.

3.2 SERVER SIDE

The server application is designed to run over *JavaScript V8 Runtime.* The code was written in *JavaScript* for the server environment *Node.JS,* using IntelliJ as IDE. *Npm* was used as a package manager for Node.JS allowing a seamless integration with all the external dependencies such as:

- *Express: t*his library provides a robust set of features for web and mobile applications, as well as the ability of creating robust API quickly and easily.
- *Body-Parser:* library used to parse incoming request bodies in a middleware before handling the request.
- *MySQL:* library used to connect and interact with Data4Help's MySQL database.
- *Logger:* middleware used to track API requests.
- *Passport:* middleware used to implement login functionalities for the web app.

Using this well-known and maintained modules we were able to focus all the effort on developing the main functionalities of Data4Help. This approach however might render our application vulnerable by relaying on code written by others.

4 STRUCTURE OF THE SOURCE CODE

Data4Help is composed of a two-tier architecture with distributed data and logic.

4.1 CLIENT SIDE

The source code of the client is available at the path */ImplementationMobileApp,* in the app module we can find two folders:

- */com/trackme/data4help :* containing al the code managing the lifecycle and the user interface of the application.
- */HttpClient :* containing all the code that handles the http connection with the server.

4.2 SERVER SIDE

The source code of the server is available at the path */Implementation.* Inside its subfolders the code is structured by its functionalities:

- *Controller:* containing the main functions that Data4Help exports.
- *Model:* containing the representation of the data and the data access layer that maps object in memory to the database.
- *Database:* containing the code that manages the database connection.
- *Routes*: containing all the code that handles the HTTP endpoints.

5 TESTING

The reader is invited to read Section 5 of the Design Document, which contains information about procedures and frameworks used to test the written code.

If you want to run the test suite you should run '*npm test'* command in */Implementation* directory.

Our system is mainly focused on the management of health data, and each of the functions is interfaced with the DBMS to collect information or to add new ones. In testing the system for the first release, the team focused primarily on the correctness of its functions, checking their behavior in different situations.

5.1 TEST CASES

The most significant test cases follow here below.

5.1.1 Case 1 Get Queries

Components Tested	Request Services, DBMS
Description	Request Services requests all the anonymous request that belong to a Business Customer
Input specification	Email of some Business Customers
Output specification	Check consistency in information stored in the database
Result	PASSED (260 ms)

5.1.2 Case 2 Get Query Data

Components Tested	Request Services, DBMS
Description	Request Services requests the data belonging to a specific request of a Business Customer
Input specification	Email of a Business Customer and titles of its requests
Output specification	Check consistency in information stored in the database and data are anonymized correctly (num > 1000)
Result	PASSED (220 ms)

5.1.3 Case 3 Make a valid Anonymous Request

Components Tested	Request Services, DBMS
Description	Should create a new Anonymous Request if all parameters respect consistency
Input specification	Email of a Business Customer and parameters of its requests
Output specification	Check that the request's submission was successful
Result	PASSED (90 ms)

5.1.4 Case 4 Make a not valid Anonymous Request

Components Tested	Request Services, DBMS
Description	Should not create a new Anonymous Request if a parameter doesn't respect consistency
Input specification	Email of a Business Customer and parameters of its requests
Output specification	Check that the request's submission was unsuccessful
Result	PASSED (2 ms)

5.1.5 Case 5 Make a valid Private Request

Components Tested	Request Services, DBMS
Description	Should create a new Private Request if all parameters respect consistency
Input specification	Email of a Business Customer and email of a Private Customer
Output specification	Check that the request's submission was successful
Result	PASSED (180 ms)

5.1.6 Case 6 Make a not valid Private Request

Components Tested	Request Services, DBMS
Description	Should not create a new Private Request if a parameter doesn't respect consistency
Input specification	Email of a Business Customer and email of a Private Customer
Output specification	Check that the request's submission was unsuccessful
Result	PASSED (30 ms)

5.1.7 Case 7 Registration of a Private Customer

Components Tested	User Services, DBMS
Description	Should create a new Private Customer if all the provided data are consistent
Input specification	Data of a new Private Customer
Output specification	Check that the request's submission was successful
Result	PASSED (178 ms)

5.1.8 Case 8 Authentication

Components Tested	User Services, DBMS
Description	Should authenticate the Private Customer if the credentials are correct
Input specification	Email and password of a Private Customer
Output specification	Check that the authentication was successful
Result	PASSED (82 ms)

5.1.9 Registration of a Business Customer

Components Tested	User Services, DBMS
Description	Should register a new Business Customer only if all the provided data are consistent
Input specification	Data of a new Business Customer
Output specification	Check that the request's submission was successful and that the status is 0 (before System Manager's acceptance)
Result	PASSED (176 ms)

5.1.10 Login of a Business Customer

Components Tested	User Services, DBMS
Description	Should register a new Business Customer only if all the provided data are consistent
Call	POST /login
Input specification	Email and password of a specific Business Customer
Output specification	Check that the authentication was successful, redirectTo('/')
Result	PASSED (830 ms)

6 INSTALLATION INSTRUCTIONS

6.1 APP INSTALLATION

If you have an Android phone you can download the apk file from the */DeliveryFolder.*

Otherwise you need to install IntelliJ and the Android SDK, including the Android Simulator. Download the latest Android API to ensure compatibility with our application.

After you've cloned *"PeressonZiffer"* repository from GitHub, import the project in IntelliJ using Gradle as your build/source manager. IntelliJ should take care of everything for you.

Now you can run the application on the simulator, by clicking the green arrow(▶) in the top-right corner.

6.2 SERVER INSTALLATION

You can find our web app deployed on *http://data4help.herokuapp.com* .

If you want to import and install it on your computer you need to download the latest release of IntelliJ, Node.JS with npm and the Node.JS plugin for IntelliJ.

After you've cloned *"PeressonZiffer"* repository from GitHub run the '*npm install'* command in the */Implementation* directory to download all the dependencies (alternatively you can extract the *Server.rar* file find in */DeliveryFolder* into a folder of your preference and then run *"npm install"*).

To run the backend, you can either use IntelliJ and click the green arrow on the top-right corner (after you've imported the project) or run the command '*npm start'*.

Than after a few seconds the website and the backend will be available at *http://localhost:3000*

6.2.1 System Manager usage

Since the System Manager has a crucial role in the registration of a Business Customer, we will share the credentials needed to access its functionalities.

At the login page on the web app insert:

Email: system@manager

Password: password

6.2.2 Warnings

We have implemented a scheduler that periodically checks and updates all the periodical Anonymous Requests. We highly advise against leaving the server running on your local machine since this may cause some duplications and conflicts in the database with the instance running on Heroku. This is because every instance of the server shares the same database.

7 EFFORT SPENT

Time spent on this document: 10h

- Giacomo Ziffer 5h
- Tommaso Peresson 5h

Time spent on implementation and testing: 150h

- Tommaso Peresson 75h
- Giacomo Ziffer 75h