Final Project Report

By:
Derek Campaniello, Derek_Campaniello@student.uml.edu
Alexander DiPaola, Alexander_DiPaola@student.uml.edu
Madhumathi Prakash, Madhumathi_Prakash@student.uml.edu
Johnathan Wydola, Jonathan_Wydola@student.uml.edu

Table of Contents

1. Introduction

1.1. Document Description

1.1.1. Introduction

Definitions/Abbreviations:

The following terms that are used in this document are defined as follows:

- The **client** refers to the user whose intention is to track their fitness goals.
- The **administrator** refers to the health care provider/clinician that is tracking the client's fitness data.
- The **sensor subsystem** refers to a system of sensors provided by the Pebble smartwatch.
- The **data-aggregator subsystem** refers to an Android powered tablet or smartphone that pre-processes data that will be inserted into a database system.
- The **cloud subsystem** refers to a system composed of a web user interface and database system.
- The **AWS** abbreviation refers to Amazon Web Services.
- The **VPS** abbreviation refers to virtual private servers.
- The **UI** abbreviation refers to a user interface.
- The **OS** abbreviation refers to operating systems.

Other technical definitions can be found in the glossary of this document.

Purpose:

- The purpose of this project is to display fitness data in a user friendly and functional web interface and to create a database system that will be shared by multiple subsystems. These subsystems include the sensor, data aggregator, and cloud.

Scope:

- The scope of this project is to provide a cloud system that stores fitness data received from the sensor and data-aggregator subsystems in a database. Also, the cloud system will retrieve and displays fitness data in a web user interface. The cloud is deployed on a scalable servers running on the Amazon Web Services (AWS).

References:

The following documents pertain to the contents of this document:

- Initial Feasibility Study – https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/FeasabilityStudy.pdf
- Product Backlog – *https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/ProductBacklog.pdf*
- Assignment 3 – *https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/Assignment3.pdf*

Abstract:

- The purpose of this document is to sum up what has been accomplished with this system and to convey the specifications and design of the system itself.

1.1.2. System Overview

The cloud system is composed of the following components:

- A **database** that contains a relevant database schema.
  - The database is created using MySQL server.
  - The database is then interfaced with the cloud, sensor and data-aggregator subsystems.
- A **login and user profiling system** that allows the client to register, login, and manage their personal account information including name, age, height, weight, sex, etc… This system will also contain the same features for the administrator.
- A **web user interface** that allows the client to view their personal fitness information and set goals. The web UI will also allow the administrator (with proper authentication) to view and manage the client's personal information

2. Design Considerations

2.1. Assumptions and Dependencies

The system is dependent on the following:

Hardware:

- A server that has adequate hardware to handle the task at hand in a timely and stable manner.
    - This may be composed of a single server or multiple servers to accomplish the task.
    - These servers may also be physical or VPS systems (such as AWS).

Operating Systems:

- The system can be deployed on any server OS and environment with adequate resources.
    - Appropriate operating systems include:
        - Unix based (Ubuntu Server, Debian, CentOS, Red Hat Enterprise Linux)
        - Windows based (Windows Server 2003, 2008, 2008 R2, 2012, 2012 R2)

Software:

The following software is required:

- HTTP/HTTPS server (i.e. Apache, nginx).
- PHP
- SMTP
- SSL
- Mysqli PHP driver
- Securimage PHP scripts (included)
- MySQL server.

2.2. General Constraints

There are multiple constraints that have impacted the design of the system. These include:

Availability of Resources:

- Since the overall system relies on this subsystem. It is imperative to maintain the stability and uptime of the system.

Interoperability Requirements:

- The cloud system is required to able to interface with other subsystems such as the sensor and data-aggregator subsystems. Therefore, the cloud system is required to communicate with multiple environments simultaneously.
- The cloud system is required to maintain a reliable connection with the database server.

Security Requirements:

- Since the system is dealing with the personal information of clients and administrators, security has made a huge impact on the design of the system.
- Secure data communications between subsystems has also made an impact on design.

Performance Requirements:

- The cloud system relies on optimal performance to deliver its intended purpose.

Network Communications:

- The cloud system requires a consistent and reliable network in order to communicate between subsystems.

Other Requirements:

- All system resources are limited by the available resources provided by VPS provider (such as AWS) at a given time. These resource requirements can be fulfilled by upgrading the server through dynamic scaling in a virtual environment.

2.3. Goals and Guidelines

The following goals were taken into account when designing the system:

- To create a visually appealing and easy to use interface to attract any general audience of client.
- Secure information using technologies such as Salt, Hashing, SSL, HTTPS and encryption.
- To ensure that all data is only visible to authorized administrators and clients.
- To create a cloud system that maintains a high quality of standard when referring to uptime and stability.
- To create and maintain a stable operating environment.

- To create a system with maintenance standards in mind.

2.4. Development Methods

This software system was designed using an incremental method with some reuse. The incremental method allowed the project to be broken into smaller and more manageable sections. Each member was assigned sections to complete and maintain as the project progressed. While working in increments, a reuse model was also used. This allowed each member to take some of the code from previous projects and integrate it into this product, allowing each member to focus on more important and higher priority features.

3. Architectural Strategies

Use of Particular Products:

- **Amazon Web Services (AWS)** – AWS was used because of its dynamic scaling of resources.
- **PHP** – PHP is easy to learn, does not use a lot of system resources and is scalable.
- **MySQL** – MySQL is the industry standard and most popular database server and it is easy to use.

Reuse of Existing Software Components:

- The reuse of certain existing PHP and HTML scripts were used to save time and allow the team to work on higher priority tasks.

Error Detection:

- Python is used for unit and stress testing of the operating environment because Python is easily integrated and is useful for software testing.

Memory/Storage Management:

- Policies were put in place to manage the data types and size of columns in the MySQL tables to preserve system resources.

Distribution of Data:

- Data is distributed between the sensor, data-aggregator and cloud subsystems in order to create seamless integration of all components.

Communication Mechanisms:

- Communication has been used through the project when developing the system. This comes in the form of weekly meetings in order to stay on track and successfully complete the project.

Management of Other Resources:

- Integration between the database and web server is handled by PHP because of its stability and ease of use.

- Code is managed through Github to keep track of changes between incremental deliveries.

4. System Architecture

The cloud subsystem is broken into multiple different subsystems. These include:

- The **User Account/Profile System** which is responsible for handling registrations, logins and profile information of the client.
- The **Device Registration System** which allows the client to add, edit and delete devices that can record data to their account.
- The **User Authorization System** which allows the client to grant an administrator access to their account details.
- The **Database Data Collection and Analysis Systems** which store, retrieve, and analyze data which is in turn displayed within a graphic UI.
- The **Administrator/User Device Registration System** allows the administrator to register devices to a client's account with proper authentication.
- The **Administrator Account/Profile System** which is responsible for handling registrations, logins and profile information of administrators.

The cloud subsystem can be described using the **Model View Controller (MVC)** design pattern. The MVC model can be described using the following components:

- The **Model** in this system is the MySQL database server.
- The **View** in this system is dynamic pages which are generated using database contents.
- The **Controller** in this system is HTTP request processed when accessing these pages. The HTTP server serves these requests.

5. Policies and Tactics

Use of Particular Products:

- PHP and MySQL were used for communication between the database and webserver.

Coding Guidelines and Conventions:

- Comments and documentation were used in code to provide better readability and insight.
- Generic naming conventions were used to easily identify variables.

Plans for Ensuring Requirements Traceability:

- Along the way of the development of the cloud system, detailed documentation was created in order to track original and updated requirements.

Plans for Testing the Software:

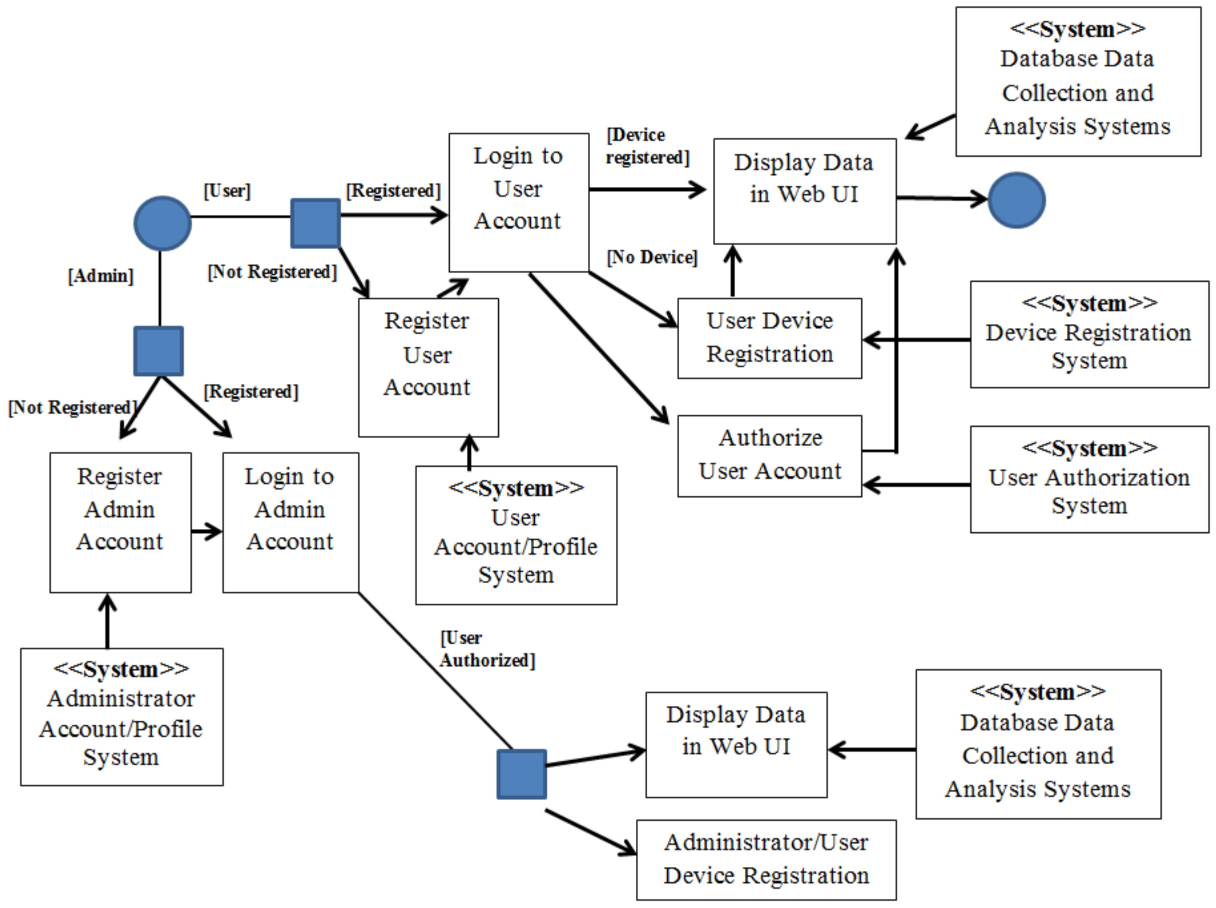
- Python was chosen to test the system because of its flexibility and its ease of use in testing.

Plans for Maintaining the Software:

- The overall system was designed with maintenance in mind. Therefore, it was designed to be easily updated with changing requirements.

6. Detailed System Design

This diagram explains the process for the cloud subsystem using the process model and additional information can be found in Assignment 3 (https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/Assignment3.pdf) which further explains the system design in more detail. (**Note:** This was approved by Professor Cao, himself.)



<<System>> Database Data Collection and Analysis Systems
[Device registered]
Login to User Account
Display Data in Web UI
[User]
[Registered]
[Admin]
[Not Registered]
[No Device]
Register User Account
User Device Registration
<<System>> Device Registration System
[Not Registered]
[Registered]
Authorize User Account
<<System>> User Authorization System
Register Admin Account
Login to Admin Account
<<System>> User Account/Profile System
<<System>> Administrator Account/Profile System
[User Authorized]
Display Data in Web UI
<<System>> Database Data Collection and Analysis Systems
Administrator/User Device Registration

7. System Installation and Execution

The following instructions that explain the configuration and installation of the cloud system:

1. Install a stable and reliable physical server or VPS environment with adequate hardware to handle tasks and a proper operating system environment.
    a. If running a virtual environment on AWS. Please follow the following instructions:
    b. Create an EC2 instance with the proper system resources and operating system image.
    c. Apply security settings including opening inbound ports 80, 21, 443, 25 and 20.
    d. Give the instance an elastic IP so that it retains its static IP after restart.
2. Install desired HTTP server (i.e. Apache, nginx, etc…).
3. Configure HTTP server to quality standards including adding SSL security to make a proper HTTPS connection so data isn't transmitted in plain text.
4. Install PHP so that the system can function.
5. Install PHP mysqli libraries. Do not use mysql functions to connect to the database as these are deprecated and will be removed in a future version of PHP.
6. Install and Configure a MySQL server. Be sure to setup proper security features such as disabling root access from remote locations and giving passwords to MySQL users. Also, add SSL security when communicating with MySQL server.
7. Move PHP files in HTTP public folder to be displayed.
8. Change config.php to personal database information.

8. Glossary

The following are definitions of technical terms used within the document:

- **HTTP/HTTPS (Hypertext Transfer Protocol/Hypertext Transfer Protocol over SSL)**
    - **HTTP** is a data protocol used in the communication between a computer and web server. HTTP sends data communication in plain text.
    - **HTTPS** is a more secure data protocol used for the communication between a computer and web server. HTTPS utilizes SSL in order to encrypt data transfer.
- **SSL (Secure Socket Layer) –** A form of encryption used to encrypt data transmissions.
- **PHP –** A server-side scripting language used in web development.
- **MySQL Server –** An open source database system.
- **Mysqli –** A database driver used in PHP to interface with a MySQL server.
- **Salt –** A random string used as one way encryption that hashes a password.

- **Hash –** A one way function that scrambles plain text using a particular algorithm.
- **Python –** A high-level programming language.
- **Virtual Private Server (VPS) –** A virtual server that runs its own OS and is functionally equivalent to a physical server.
- **Amazon Web Services (AWS) –** A cloud computing platform that offers remote services.
- **SMTP (Simple Mail Transfer Protocol) –** A data protocol for the transmission of emails.

9. Bibliography

Related References:

- **Feasibility Study -** *https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/FeasabilityStudy.pdf*
- **Product Backlog -** *https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/ProductBacklog.pdf*
- **Assignment 3 -** *https://github.com/UMLproject/Software_Engineer_91.411_2/blob/master/3_Cloud/Document/Assignment3.pdf*
- **PHP Documentation -** *http://php.net/manual/en/*
- **MySQL Documentation -** *http://dev.mysql.com/doc/refman/5.6/en/index.html*
- **AWS Documentation -** *http://aws.amazon.com/documentation/*
- **Apache Documentation -** *http://httpd.apache.org/docs/2.4/*
- **Python Documentation -** *https://docs.python.org/3/*