Mobile Storage Analytics

Software Design Document

Team CodeStars

Project Members

• Hisham Alhussain • Matthew LeBlanc

• Joy Liu • Yueying Liu

• Paul Maurais • Aisiri Murulidhar

• Siddharth Naidu • Varun Sharma

• Andrew Sharp • Quyen Tran

• Cody Dyl (manager)

# Table of Contents

[**Table of Contents**](#_xrh4uh7yf6s) **2**

[**1. Overview**](#_u0316viqzgsi) **3**

[1.1 Purpose](#_euci9gq3h4bz) 3

[1.2 Context Level Diagram](#_2766d0ub639z) 3

[**2. Architectural Design/High Level View**](#_fpudg8vo7gov) **4**

[2.1 System Description](#_unc5tm5ejs8u) 4

[2.2 System Architecture Design](#_dfb2u05o36qv) 5

[2.2.1 Elements catalog - System Architecture Design](#_b6j47oh6vebs) 6

[2.3 High Level View](#_lfim3f4qzhr4) 7

[2.3.1 Elements catalog - High Level View](#_v3up243ibc71) 8

[**3. Architectural Design Rationale**](#_z48q9rr04d1o) **9**

[3.1 Architecture Rationale](#_4rz2oh1sb7du) 9

[3.2 Technology Rationale](#_aprmkspw03p9) 9

[3.2.1 Xamarin](#_d1p9cr425xb1) 9

[3.2.2 C#](#_qhafmtxp0aop) 9

[3.2.3 Amazon Web Services RDS](#_qvdcsc5xqfuf) 9

[3.2.4 MySQL](#_8hh315m0b1n) 9

[**4. ER Diagram**](#_31tzcc9jlce0) **10**

[**5. High Level Technical Decisions**](#_hvwfx83l5yjh) **11**

[5.1 Third Party Software References](#_afks76se3kcl) 11

[5.1.1 Xamarin](#_4fh9lfzfu2ca) 11

[5.1.2 C#](#_ewnso5qnr0ub) 11

[5.1.3 Amazon Web Services RDS](#_t83z7p3ng6m4) 11

[5.1.4 MySQL](#_yjvxlpyif8qb) 11

[5.2 Protocols](#_vdwulfhocmgz) 11

[5.2.1 Application layer](#_i5klkljwkut) 11

[5.2.2 Transport layer](#_l9taldktki8p) 11

[5.2.3 Network layer](#_3jju2yr1kks4) 11

# 

# 1. Overview

## 1.1 Purpose

The software design document’s purpose describes the implementation details of the HP Mobile Storage Analytics. The Mobile Storage Analytics is a mobile representation of the website based implementation of HP’s Storage Analytics. Once the user logs into the application they have view access to all the relevant information regarding their storage. This is an additional resource that users have access to view their storage analytics alongside the already existing website.

## 1.2 Context Level Diagram



# 2. Architectural Design/High Level View

## 2.1 System Description

The Mobile Storage Analytics comprises of two parts. The first component is the client-side mobile application and the second component is a database of attributes and information about client’s storage units.

The client-side mobile application essentially functions to provide data and information about a user’s storage units in a readable manner. The application delivers information based upon the storage unit and data format selected by the user to view.

The database of information about storage units interacts with functions that use the provided data to create readable user friendly visual and graphical forms of the data. The database receives, stores and parses data from HP and is updated periodically. Ideally, the database will be automatically updated with real-time data but for security, our access to HP’s real-time data is currently limited. The real-time data is used to create visuals and graphics on the client side and provides services such as health alerts on storage units for clients.

We have decided to use Xamarin to implement the system. Xamarin is a platform used to build mobile applications on iOS, Android and Mac. Because Xamarin is our primary platform for the application, we will primarily be writing in C# for both components of the mobile application.

## 

## 2.2 System Architecture Design

The architecture has three main tiers - the presentation tier on top, the business tier in the middle and the data tier on the bottom. The presentation tier is implemented as a MVC model. The controller of the top tier is comprised of the on screen buttons and text input of the user to change the UI’s condition.

The view is the client’s observations of each state of the UI. The model is the data construct representing each of UI state the user experiences.

The business tier acts as a bridge between the presentation and data tier. It dictates the instructions and restrictions of the method calls, processes, and data structures. The business tier interacts with the data tier and processes the data for use in the presentation tier.

The final tier is the data tier. This tier represents the storage of the users. MySQL will probably be used to access the storage provided by HP. Users will communicate their modifications to their storage through a client server model, where the clients are the users and the server is the databases provided by HP.

### 2.2.1 Elements catalog - System Architecture Design

|  |  |
| --- | --- |
| **Element Name** | **Description** |
| Presentation Tier | Provides user with a graphical interface to interact with the system. Follows the model-view-controller(MVC) model. |
| Business Tier | Acts as a bridge between the presentation and data tier. Dictates instructions and restrictions of method calls, processes, and data structures and modifies elements in the data tier if permitted. |
| Data access Tier | The data access tier acts as a bridge to the data tier to ensure the business and presentation tier do not receive private knowledge of the system’s databases or schema. It contains mappers to the data in the system and is responsible for communicating between the business tier and the data tier. |
| Data Tier | The data tier contains all information about the users and storage systems. Represented in this tier are client data, such as login credentials, and storage data, such as the product family, model, and capacity info. |

## 2.3 High Level View

## 

The system’s data input module pulls new storage system data from the HP servers and processes it to be included in our database. The user can use the controller module to select specific systems by searching or filtering or to select information that they want to view in dashlet view. It is also used by the user to change what view they see. The controller module will then query the database for the required information. The data aggregation module then compiles the information into the requested form (e.g. the correct format for specific dashlets) and passes it to the view module. The view module then takes the data it is being passed and displays it back to the user in whatever format was requested. There is also an alert manager module that periodically queries the database for any potential issues and generates alerts that the view module will display as notifications to the user.

### 2.3.1 Elements catalog - High Level View

|  |  |
| --- | --- |
| **Element Name** | **Description** |
| HP Server | Centralized server. Able to update database by sending new data to data input. |
| Data Input | Receives data to be parsed and updated into the database. |
| Database | Information home for all storage systems. Contains information about all systems such as capacity info, product family, model, etc. Aggregates requested data and is able to send alert data to the alert manager. |
| Data Aggregation | Information from the database pertaining to user requests. Reports data to the view. |
| User | The entity that is using the application. User interacts with the controller to request information. |
| View | What the user sees. Includes info dashlets such as storage lists and individual systems. |
| Controller | Acts as the middleman between the user and the database. Sends data request from the user to the database. |
| Alert Manager | Requests database for any possible alerts and receives from database alert data. Sends health alerts to the view. |

# 

# 3. Architectural Design Rationale

## 3.1 Architecture Rationale

The architecture design that was chosen separates concerns between the presentation tier, business tier and data tier. The MVC model for the application allows for the separation of concerns between the data and views and acts as the presentation tier. The controller acts as the middleman between the models and views and requests for data from the server. The business tier works between the presentation tier and the data tier. This tier modifies data from the data tier if needed and has different methods to facilitate the flow of data to the presentation tier. The data tier is comprised of the data access tier and data tier. This is where all the data is stored pertaining to the users and given to the business tier when needed.

## 3.2 Technology Rationale

### 3.2.1 Xamarin

Rather than using a framework, we decided to use Xamarin as a development platform. The platform uses C# and comes with an IDE. C# similar in syntax to Java, which all of our developers have a background in. Additionally, code written in Xamarin can be targeted at multiple platforms including Android and IOS.

### 3.2.2 C#

C# will be used to develop the application because we chose to use the Xamarin platform

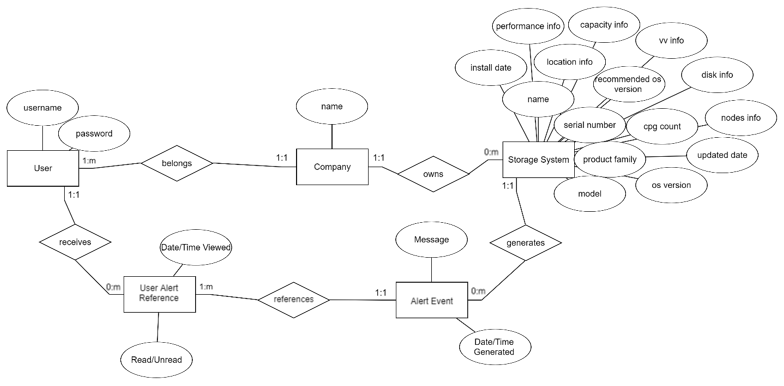
### 3.2.3 Amazon Web Services RDS

Amazon Web Services offers a relational database service. We chose to use AWS RDS because of it reliability and accessibility. With RDS, none of the database hardware or network infrastructure needs to be maintained in house. RDS is accessible anywhere, anytime and is inexpensive.

### 3.2.4 MySQL

We chose to use MySQL as a relational database management system because of its simplicity and ease of use. Additionally, MySQL deploys instantly on AWS RDS

# 4. ER Diagram



# 

# 5. High Level Technical Decisions

## 5.1 Third Party Software References

### 5.1.1 Xamarin

https://www.xamarin.com/

### 5.1.2 C#

https://docs.microsoft.com/en-us/dotnet/csharp/

### 5.1.3 Amazon Web Services RDS

https://aws.amazon.com/rds/

### 5.1.4 MySQL

https://www.mysql.com/

## 5.2 Protocols

### 5.2.1 Application layer

Our application will be using Hypertext Transfer Protocol Secure (HTTPS) to ensure the security and confidentiality of the data returned from web server. For any actions that require fetching data from the database, we will be using Structured Query Language (SQL) Server Protocols.

### 5.2.2 Transport layer

To guarantee reliability for users attempting to access their systems list we will be using Transfer Layer Protocol (TCP).

### 5.2.3 Network layer

For the network layer, we will be using Internet Protocol (IP).