

Mobile Storage Analytics

Low Level 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**](#_l0tebuiwovzb) **2**

[**1. Third Party Software**](#_u0316viqzgsi) **3**

[1.1 Overview](#_1u8vh8ykiqfw) 3

[1.2 Third Party Software Usage](#_jf4yfgloj0kf) 3

[**2. UML Diagram**](#_grmrvckbx9o0) **3**

[**3. Interfaces**](#_cn5gge1eai8x) **4**

[3.1 Internal Interfaces](#_dpou7todcq79) 4

[3.2 External Interfaces](#_271gnxxy9u3u) 4

# 

# 1. Third Party Software

## 1.1 Overview

In this application we are using a few different third party resources. Among those are Xamarin, Amazon Web Services RDS, and MySQL. Xamarin is a development platform that utilizes the C# programming language and allows for development on both iOS and Android mobile platforms. The Amazon Web Services RDS is a cloud based Relational Database service. MySQL is a relational database management system.

## 1.2 Third Party Software Usage

### 1.2.1 Xamarin

We decided to use Xamarin as our development platform. Xamarin is a software company that provides an IDE and platforms for mobile app development on both android and iOS. Xamarin utilizes C# in its IDE and has a lot of API’s available that would be useful in the development process.

### 1.2.2 Amazon Web Services RDS

We will be using Amazon Web Services’ Relational Database Service to store the necessary data for the application. AWS RDS is a web service that makes it easier to set up and utilize a relational database in the cloud. It is both cost efficient and has a resizeable capacity. The AWS relational database service is reliable and accessible anytime and anywhere.

### 1.2.3 MySQL

We are using MySQL as our relational database management system. MySQL is an open-source relational database management system. MySQL works really well with AWS RDS and is user-friendly.

# 2. UML Diagram

## 

Diagram 1: UML

This UML represents the presentation tier of the system. The various views (the log in screen, dashboard, and storage list) all implement the display interface. This allows them to toggle a side menu which has options to log the user out or display alerts. The storage list displays a list of storage systems associated with the current user, and that user can search or filter to return a new list of storage systems meeting their input criteria. The dashboard view can generate three types of dashlets, which all implement the dashlet interface. User is an interface implemented by the client and HP staff classes. This allows HP staff to view information on all storage systems while clients can only view the systems that they own. The log in screen is used to authenticate a user which affects the different systems that are shown to the user based on whether they are HP staff or a member of a company. Alerts are generated by a storage system and passed into the side menu from the alert class so they can be displayed to a user. The storage system is a structure to store necessary information about the storage system so it can be displayed to users via the storage list or dashboard view.

# 

# 3. Interfaces

## 3.1 Internal Interfaces

StorageSystem

|  |  |
| --- | --- |
| getName():String | Returns the name of the storage system |
| getCapacity():Int | Returns the current disk usage of the storage system |
| checkForAlerts():Alert[] | Returns an array of all alerts pertaining to the storage system, or an empty array if the storage system has no alerts |

Display

|  |  |
| --- | --- |
| toggle() | Changes display state from storage list to dashboard or vice versa |

SideMenu

|  |  |
| --- | --- |
| logout() | Logs the user out of the system |
| healthAlerts() | Shows the user the details of any health alerts |

StorageList

|  |  |
| --- | --- |
| display() | Displays detailed information about a storage system |
| filter(keyword:String):StorageSystem[] | Filters the list of storage system by the supplied filters |
| search(input:String):StorageSystem[] | Searches the storage systems based on an input string |

Alert

|  |  |
| --- | --- |
| createAlert() | Creates alert message to be displayed to user in the SideMenu |

## 3.2 External Interfaces

The only external interface will be a relation database running MySQL and hosted on AWS RDS. All communication with the database will utilize the C# System.Data.SQLClient class.

LogIn

|  |  |
| --- | --- |
| authenticate(user:String, pass:String) | Hashes the password and verifies it against a user database stored on AWS |

Dashlet

|  |  |
| --- | --- |
| getData(): dictionary(key: string, value: int) | Gets the dashlet data from the AWS database |

User

|  |  |
| --- | --- |
| getStorageSystems(id:String): StorageSystem[] | Returns a list of storage systems registered to a specific user from the AWS database |

# 