*Biometrix*

***Use Case: "Store Information"***

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 11/12/2015 | 1.0 | Initial Template Revision | Troy J. Riblett |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Overview 1

1.1 Brief Description 1

1.2 Requirements Trace 1

1.3 Involved Actors 1

1.4 Preconditions 1

1.5 Post conditions 1

1.6 Invariants 1

2. Flow of Events 1

2.1 Basic Flow 1

2.2 Alternate Course – None 1

3. Extension Points 1

4. Scenarios 1

4.1 Happy Day 1

4.2 Rainy Day 1 – 2

Use Case: "Store Information"

# 

# Overview

## Brief Description

This use case describes the system’s storing of module information. Module information must be stored in order to be used, so this is vital to all.

## Requirements Trace

This use case maps to the following requirements: 1.4, 2.4, 3.1.4, 3.6, 3.7, 4.6, 5.3, 5.8, 6.1, 6.1.2, 6.2, 6.4, 6.5, 7.1, 7.2, 7.3, 7.4, 7.5,

## Involved Actors

User, system, webserver, and database.

## Preconditions

* This use case should be “called” by one of the “enter \_\_\_\_” use cases in order to store the information gathered

## Post conditions*.*

## Invariants

* User login information

# Flow of Events

## Basic Flow

This use case starts when the system generates a call to store information.

1. System stores information on local filesystem.
2. System makes call to webserver with request to store info.
3. Webserver determines if credentials are valid.
4. If user is logged in successfully, webserver generates SQL command.
5. Webserver sends SQL command to database.
6. Database returns status of command.
7. Webserver returns status to system.
8. System informs user if applicable.

## Alternate Course – None

# Extension Points

None.

# Scenarios

## Happy Day - “Successful Database Update”

Assumptions:

User Joseph has a login that is correct and is attempting to store sleep information. Joseph also has internet access.

Steps:

1. Joseph enters sleep information (see Sleep entry use case).
2. The system saves that information to the local filesystem.
3. The system calls the webserver with the sleep time information Joseph wants to save.
4. The webserver generates the SQL command that will update the sleep table with an entry for Joseph.
5. The webserver passes that SQL statement to the database.
6. The database executes the statement and adds the row to the table.
7. The database passes back to the webserver that the operation was a success.
8. The webserver passes back to the application that the operation was a success.
9. The system displays that information was saved correctly.

## Rainy Day – “Cannot Connect To Database”

Assumptions:

User Aelita is not connected to the internet, but she wants to store her diet information for the day.

Steps:

1. Aelita enters her diet information (calories etc. see Enter Diet use case)
2. The system saves that information on the local filesystem.
3. The system makes a call to the Webserver.
4. The system is unable to reach the webserver.
5. The system displays that it could not backup information to the database.

## Rainy Day – “Invalid login”

Assumptions:

User Jeremy wants to enter new medication information, but he has not created a database login or has invalid credentials.

Steps:

1. Jeremy enters his medication prescription information into the system (see use case Enter Medication)
2. The system saves that information on the local filesystem.
3. The system makes a call to the Webserver.
4. The Webserver determines that Jeremy is not setup as a user in the database.
5. The webserver informs that Jeremy the credentials were not valid.
6. The system informs Jeremy that he cannot backup his information due to incorrect login.

## Happy Day - “Successful File System”

Assumptions:

User Matt has opted out of backups, and wants to save his sleep times for the last night.

Steps:

1. Matt enters sleep information (see Sleep entry use case).
2. The system saves that information to the local filesystem.