PANDA INC

Momentum - Multipy Active Dayz App

Architectural Design Document

Quinton Swanepoel	15245510
Azhar Patel	15052592
Tshepo Macebo Malesela	14211582
Keaton Pennels	14373018

STAKEHOLDERS

MMI Holdings: Phillip Kruger

Abstract

This document serves to describe the architecture used for the ActiveDayz application which is developed for Momentum (Multiply) as part of the Software Engineering Project (COS 301) at the University of Pretoria.

The document complies with the Architectural Design Document Software Engineering Standard, as set by the European Space Agency.

Contents

1 Introduction

1.1 Purpose

Active Dayz is part of Momentum's Multiply wellness and rewards programme. The Active Dayz programme tracks a user's steps taken and calories burned in order to provide users with relevant rewards. The programme currently only works with certain devices. An application is required that can log events such as marathons and other fitness events that users may attend, monitor users gym time, and track users steps taken and calories burned.

1.2 Scope

Active Dayz is aimed at providing momentum customers with a quick way to get rewards/active days by making use of location based services to track when a user goes to gym and how long they stay for.

The proposed system is to be integrated with a non-existing admin web interface. It should allow the integration of various services of which location based monitoring is the main objective. Other tools may include monitoring of steps take, calories burned and events attended.

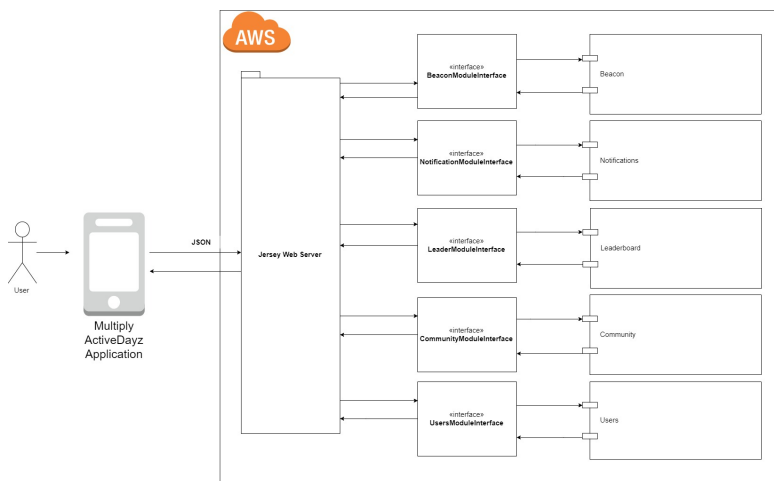## 1.3 Definitions and Abbreviations

1.3.1 Definitions

1.3.2 Abbreviations

1.4 References

1.5 Overview

2 Architectural Design

2.1   Architectural Patterns of ActiveDayz

- Service oriented architecture patterns (Micro services)

- Layered architecture patterns

- Event-Driven patterns

The first two design patterns were chosen such that the modules within our application would have the as few as possible and also so that we could ensure we achieve low coupling. The event driven pattern will aid in the connecting and ranging of the event driven beacons

2.2   Quality Requirements/Non functional Requirements

- Availability

- Security

- Reliability

- Maintainability

- Accessibility

The above list is the most integral quality requirements for our application and although it is not exhaustive, it encompasses what both our team and the client thought would be necessary for a seamless experience with the application

2.3   Design Requirements

- Our goal is to design a highly modular system which has minimal dependency between the modules

- The system is required to be designed such that if a module needs to be updated/modified or if a module needs to be added or removed. that this can be done with minimal effort

Design Requirement The system is to be a modular system which allows for âŮŔ only a subset of modules to be deployed – minimally the system will require the core modules to be deployed. âŮŔ further modules, of which the navigation module is the most desirable, to be added at a later stage. To this end the system should be implemented with âŮŔ minimal dependencies between modules, âŮŔ no dependencies of core modules on any add-on modules, âŮŔ as far as possible making use of dependency injection.
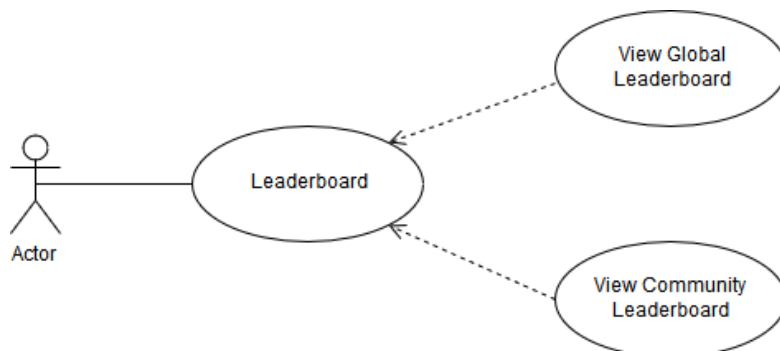
3 Component Descriptions

3.1 Beacon Module
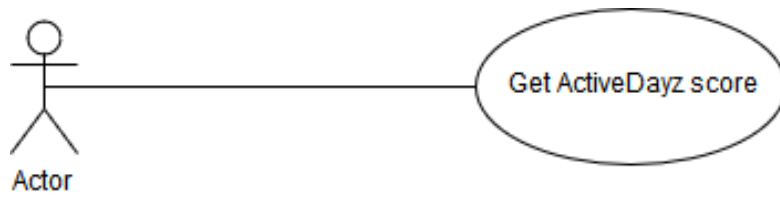
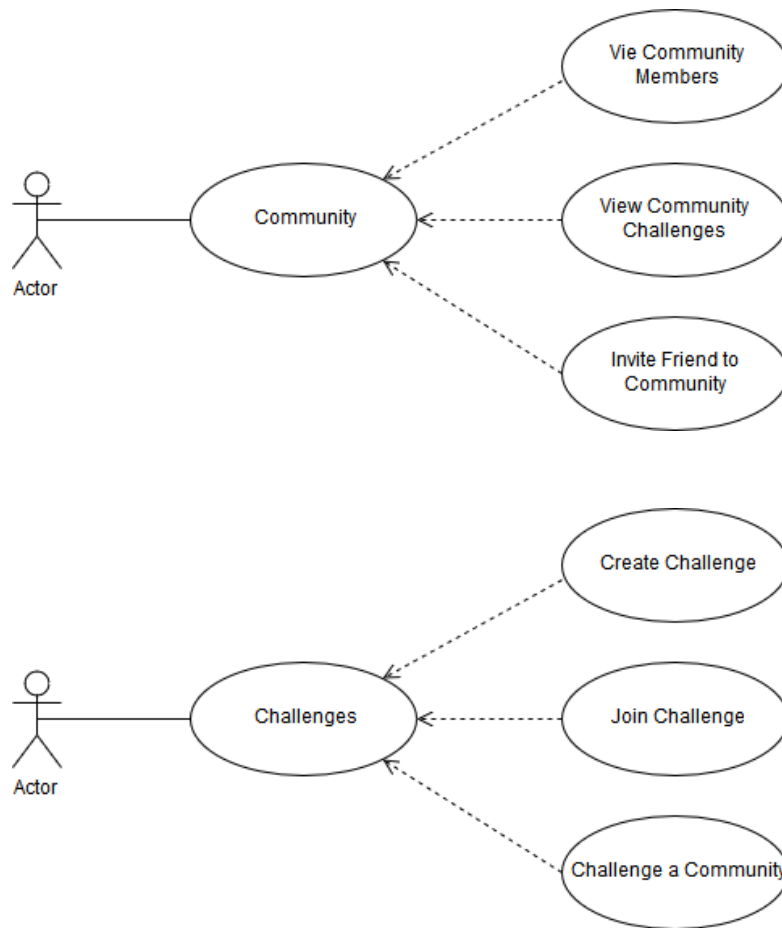## 3.2 Notification Module
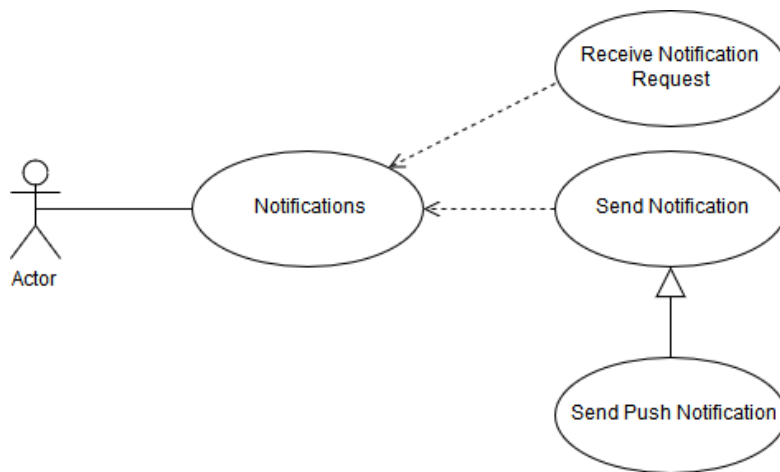
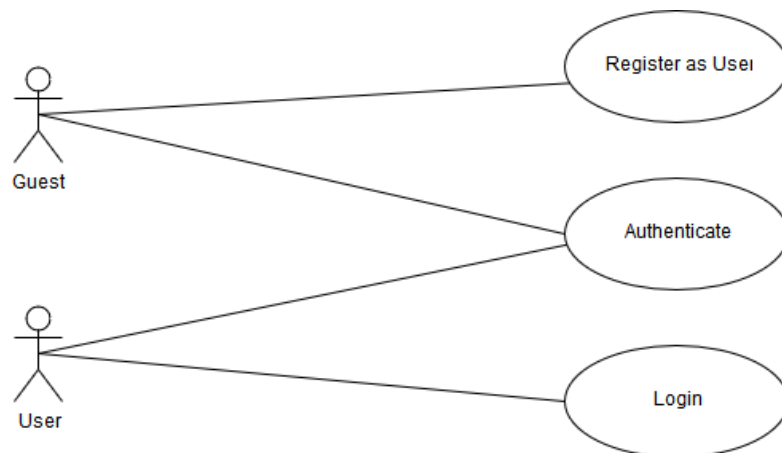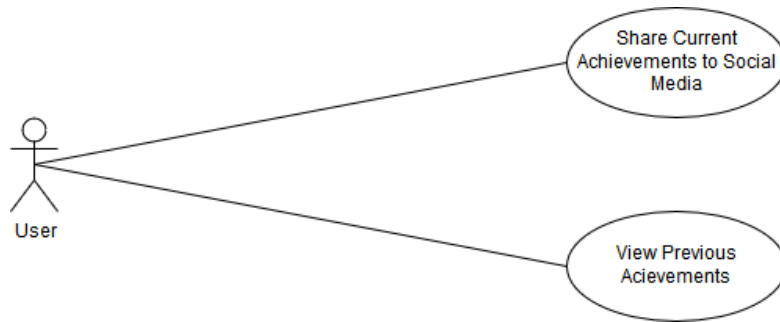## 3.3 Leader Module

3.4   Community Module

3.5   Users Module

3.6 Social Module

4 Feasibility and Resource Estimates

5 Technologies Used

5.1 Server

JavaEE eclipse oxygen
Jersey restful web service
Tomcat App server
Amazon aws
JSON communication to ensure low coupling
Maven build automation

5.2 Web develpement

polymer
node bower and polymer
webcomponents
js
mysql db
beacons estimotes
docker
wildfly