UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

ARCHITECTURAL REQUIREMENTS
Mobile Monitoring App
Emilio Mumba

The 5 Concurrent Nodes

Khathutshelo Shaun Matidza
Sylvester Sandile Mpangane
Thabang Michael Letageng
Aluwani Augustine Simetsi
Matthew Nel

Department of Computer Science, University of Pretoria

May 2015

Contents

1 Introduction

This document contains the software (application) architectural requirements specification which is the infrastructure upon which the application will be developed.

The following requirements will be addressed:

- Quality requirements
- Architectural Patterns or Styles
- Access and Integration Channels
- Technologies

2 Android Application Requirements

2.1 Architectural Requirements

2.1.1 Critical Quality Requirements

Auditability

Description

Auditability refers to the ability to account for a system's usage by the user and be able to monitor events and create logs on what are the user's actions within the system.

Justification

The Mobile Monitoring Application main functionality is to monitor the user's device activities and report all logs within a particular device.

Mechanism

1. Strategy:

   Auditability can be achieved by:

   - Resource Monitoring: A tactic that registers all the events and resources that the application uses and creates logs for the resources usage.

2. Architectural Pattern(s):

   - To monitor the application events and resources a design pattern such as the observer pattern can be implemented by the application that can register all events and resources usage.

Security

Description

Security is a critical aspect of any system that deals with critical and confidential data, the strategies used for security provides mechanisms to protect data from unauthorized access and modification. For the Mobile Monitoring Application this means the following:

- No one (person or program) should be able to delete the data collected on device.
- When the data is transferred over the network, data should be protected from being intercepted and hacked.

Justification

Security is a critical for any system that deals with critical and confidential data as in the Mobile Monitoring Application. User collected data need to be protected from being tampered with and accessed by unauthorized individuals to keep data reliable and confidential.

Mechanism

1. Strategy:

   Security can be achieved by:

   - Authentication: The strategy is used to identify and confirm a user's identity.
   - Encryption: Data is converted to a secure format that cannot be easily read by unauthorized individuals.

2. Architectural Pattern(s):

   - Layering: This pattern decouples the system by dividing it into components (layers) that communicate with each other through message requests and responses, this control the access of the user level layer from directly make request to lower layers that provides critical data.

2.1.2 Important Quality Requirements

Maintainability

Description

A modular system design is used for this system so that components can be easily added and removed.

Justification

A modular design decouples a system into components that are easy to maintain and makes the system more adaptable. Decoupling the system will also ensure that it easy to do unit testing and integration testing.

Mechanism

1. Strategy:

   Maintainability ca be achieved by:

   - Decoupling: This strategy break the system into manageable components to achieve a proper structure and maintainable system.

2. Architectural Patterns(s):

   - Microkernel: The microkernel pattern improves maintainability because is separates high level services from low level services that is, it divides the systems into components that are maintainable and also allow for components to be easily removed or added to the system.

Performance

Description

Performance is a measure of a system responsiveness when executing some action.

Justification

The Mobile Monitoring Application need use the devices resources efficiently to increase the performance of the application which is an important quality requirement that will also make the system more reliable and increase throughput of the application.

Mechanism

1. Strategy:

   Performance can be achieved by:

   - Dynamic code optimization: This strategy focuses on the code design, quality and efficiency to improve performance on the system.

2. Architectural Patterns(s):

   - The best way to achieve performance is to manage the system's resources efficiently and use design patterns to optimize code and improve efficiency.

2.1.3 Nice-To-Have Quality Requirements

Testability

Description

Testability is a measure of how well a system allows one to test if a certain criteria is met by the system. This makes fault detection in the system easy and also the faults can be isolated in a timely manner.

Justification

It is vital that every component that is deployed on the system can be tested using unit testing and also integration testing so that faults can be detected as soon as possible and be fixed.

Mechanism

1. Strategy:

    - White-box: This tactic is mainly used for unit testing and require the knowledge of the the application internal structure to create test cases for the application components.
    - Black-box: This tactic is mainly used for integration testing, it examines the functionality of the application against the specification and simplifies system components testing when plugged in a modular system.

2. Architectural Patterns(s):

    - Model View Controller: This pattern promotes separation of concern in a system by decoupling the system into components that can be tested independently.

2.1.4 Architectural Patterns or Styles

Model View Controller (MVC)

Description

Separates the applications concerns by separating the following responsibilities:

- Model: Provide business services and data.
- View: Provide view for information.
- Controller: Reacts to user events.

Justification

This pattern separate the system's concerns into components that can evolve independently which reduce the applications complexity. The Mobile Monitoring Application is to have low complexity and as a modular system it also needs to be maintainable which can be best achieved using the MVC pattern.

Benefits

- Simplification
- Improve maintainability
- Improve reuse
- Improve testability

2.1.5 Layered Architecture

Description

Partitions the applications concerns into a stacked group of layers.

Justification

Provides high level of abstraction which allows the application components to vary, this decouples the system which reduce complexity and improve the systems performance which is the objective for the Mobile Monitoring Application.

Benefits

- Improve cohesion
- Reduce complexity
- Improve maintainability
- Loose coupling
- Improve testability
- Improve reuse

2.2 Access and Integration Channels

2.2.1 Access Channels

Human Access Channel

Human access channels addresses all the different ways in which a human can interact with the Mobile Monitoring Application.

- Mobile device: The application only runs on android mobile devices, only minimal functionality is presented to the user as the application runs on the background.

2.2.2 Integration Channels

Channels

The Mobile Monitoring Application will need to access a MySQL database to store user information and data logs from device.

Protocols

The Mobile Monitoring Application will use the following protocols:

- HTTPS: This protocol will be used for security to ensure that a secure connection is maintained between the device and server and also that transported data cannot be easily intercepted.
- SMTP: Notifications for user forgotten password will use this protocol to allow user recover their information.

2.3 Technologies

2.3.1 Platform and IDE

- Android
- Android Studio

2.3.2 Programming Languages

- Java

2.3.3 Frameworks

- JUnit

2.3.4 Databases

- MySQL relational database

2.3.5 Web services

- REST

2.3.6 Others

- AJAX
- JSON

3 Web Application (Dashboard) Requirements

3.1 Architectural Requirements

3.1.1 Critical Quality Requirements

Scalability

Description

Scalability refers to the ability of a system to easily accommodate and handle a large amount of work at a single instance

Justification

The Mobile Monitoring Application dashboard needs to be able to handle as many concurrent users as possible without breaking.

Mechanism

1. Strategy:

   Scalability can be achieved by:

   - Clustering: Ensures that resources are not strained by running or maintaining many instances of the application over a cluster of servers.
   - Caching: Will reduce database workload by maintaining database query results on the application within a user session to avoid querying the database every time.

Security

Description

Security is a critical aspect of any system that deals with critical and confidential data, the strategies used for security provides mechanisms to protect data from unauthorized access and modification. For the Mobile Monitoring Application dashboard this means the following:

- Only authorized individuals may have access to the relevant data.
- Users should strictly be restricted by access levels

Justification

Security is a critical for any system that deals with critical and confidential data as in the Mobile Monitoring Application. User collected data need to be protected from being tampered with and accessed by unauthorized individuals to keep data reliable and confidential.

Mechanism

1. Strategy:

   Security can be achieved by:

   - Authentication: The strategy is used to identify and confirm a user's identity.
   - Encryption: Data is converted to a secure format that cannot be easily read by unauthorized individuals.

2. Architectural Pattern(s):

   - Layering: This pattern decouples the system by dividing it into components (layers) that communicate with each other through message requests and responses, this control the access of the user level layer from directly make request to lower layers that provides critical data.

3.1.2 Important Quality Requirements

Maintainability

Description

A modular system design is used for this system so that components can be easily added and removed.

Justification

The dashboard application must be decoupled into components that are easy to maintain and makes the system more adaptable. Decoupling the system will also ensure that it easy to do unit testing and integration testing.

Mechanism

1. Strategy:

   Maintainability ca be achieved by:

   - Decoupling: This strategy break the system into manageable components to achieve a proper structure and maintainable system.

2. Architectural Patterns(s):

   - Microkernel: The microkernel pattern improves maintainability because is separates high level services from low level services that is, it divides the systems into components that are maintainable and also allow for components to be easily removed or added to the system.

Performance

Description

Performance is a measure of a system responsiveness when executing some action.

Justification

The dashboard application need use resources efficiently to increase the performance of the application which is an important quality requirement that will also make the system more reliable and increase throughput of the application.

Mechanism

1. Strategy:

   Performance can be achieved by:

   - Dynamic code optimization: This strategy focuses on the code design, quality and efficiency to improve performance on the system.

3.1.3 Nice-To-Have Quality Requirements

Testability

Description

Testability is a measure of how well a system allows one to test if a certain criteria is met by the system. This makes fault detection in the system easy and also the faults can be isolated in a timely manner.

Justification

It is vital that every component that is deployed on the system can be tested using unit testing and also integration testing so that faults can be detected as soon as possible and be fixed.

Mechanism

1. Strategy:

    - White-box: This tactic is mainly used for unit testing and require the knowledge of the the application internal structure to create test cases for the application components.
    - Black-box: This tactic is mainly used for integration testing, it examines the functionality of the application against the specification and simplifies system components testing when plugged in a modular system.

2. Architectural Patterns(s):

    - Model View Controller: This pattern promotes separation of concern in a system by decoupling the system into components that can be tested independently.

3.2 Architectural Patterns or Styles

3.2.1 Model View Controller (MVC)

Description

Separates the applications concerns by separating the following responsibilities:

- Model: Provide business services and data.
- View: Provide view for information.
- Controller: Reacts to user events.

Justification

This pattern separate the system's concerns into components that can evolve independently which reduce the applications complexity.

Benefits

- Simplification
- Improve maintainability
- Improve reuse
- Improve testability

3.2.2 Layered Architecture

Description

Partitions the applications concerns into a stacked group of layers.

Justification

Provides high level of abstraction which allows the application components to vary, this decouples the system which reduce complexity and improve the systems performance which is the objective for the Mobile Monitoring Application dashboard

Benefits

- Improve cohesion
- Reduce complexity
- Improve maintainability
- Loose coupling
- Improve testability
- Improve reuse

3.3 Access and Integration Channels

3.3.1 Access Channels

Human Access Channel

Human access channels addresses all the different ways in which a human can interact with the Mobile Monitoring Application.

The dashboard application is accessible through a web browser, hence can be access using any of the following devices:

- Desktop Computer
- Mobile Device
- Tablet
- Laptop

3.3.2 Integration Channels

Channels

The dashboard application will need to access a MySQL database to read user information and data logs from device.

Protocols

The dashboard application will use the following protocols:

- HTTPS: This protocol will be used for security to ensure that a secure connection is maintained between the application and server and also that transported data cannot be easily intercepted.
- SMTP: Notifications for user forgotten password will use this protocol to allow user recover their information.

3.4 Technologies

3.4.1 Platform

- JavaEE

3.4.2 APIs

- JAX-RS 2.0
- JPA (Java Persistence API)
- JTA (Java Transition API)

3.4.3 Persistence Provider

- Hibernate

3.4.4 Application Server

- GlassFish

3.4.5 Programming Languages

- Java

3.4.6 Frameworks

- JUnit

3.4.7 Dependency Injector

- CDI (Context and Dependency Injector)

3.4.8 Dependency Management

- Apache Maven

3.4.9 Databases

- MySQL relational database

3.4.10 Web services

- REST

3.4.11 Others

- JSON
- Java Server Faces
- Servlets