WIT MSc Communications Software Architecture Description (AD)

WIT Design Patterns Assignment AD: Pacemaker Android Application

<< Note: This Architecture Description (AD) template from the Open Group TOGAF™ 9.1 architecture framework has been adapted to the purposes of the WIT MSc module on IT Architecture Patterns.>>.

Table of Contents

Document Information

Project Name:	Pacemaker Android Application		
Prepared By:	Colm Carew - 20053766	**Document Version No:**	1.0
Title:	Architecture Description (AD)	**Document Version Date:**	2016-05-02
Reviewed By:	Colm Carew	**Review Date:**	2016-05-03

Distribution List

From	Date	Phone/Fax/Email
Colm Carew	2016-05-02	N/A

To	Action*	Due Date	Phone/Fax/Email
Siobhan Drohan	Review	2016-05-12	sdrohan@wit.ie
Eamon De Leastar	Review	2016-05-12	edealeastar@wit.ie
Pierre Peclier	Review	2016-05-12	ppeclier@wit.ie

* Action Types: Approve, Review, Inform, File, Action Required, Attend Meeting, Other (please specify)

Document Version History

Version Number	Version Date	Revised By	Description	Filename
1.0	May. 02nd	Colm Carew	This document files some of the Architecture Views describing the Android and Play Application Assignment of the Design Patterns module: Pacemaker Android Application.	pacemaker_architecture.archimate

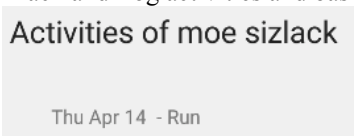
Document Overview

The following document shows the Architecture Views of the Pacemaker Android Application for the Design Patterns Module Assignment on Architecture Styles & Patterns into a TOGAF Architecture Description.

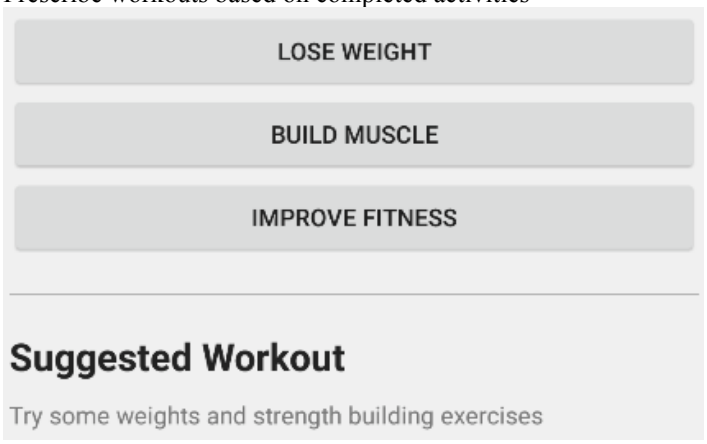
A) Problem Description

A.1 PROBLEM SCOPE

A.1.1 PROBLEM SUMMARY

Project Name	Pacemaker Android Application
Industry Domain	Activity Tracking
Problem Description	A Play Application was created in the Agile Software Development Module. This assessment seeks to create an Android Application which interfaces with the Play Application using the REST API. The Android Application will mirror the Play Application as closely as possible as well as add some features specific to the Android Application. The Android Application should: 1. Track and Log activities and basic stats  2. Have progress reports based on completed activities  3. Connect with friends and compare workouts with friends

4. Prescribe workouts based on completed activities



The Android Application should also exhibit and number of code design patterns.

A.1.2 DOMAIN GLOSSARY

User	A registered Pacemaker User can track their activities, connect with friends, compare workouts and generate prescribes workouts based on their completed activities. The User can log in via colmcarew.com or via the Android Application. Comparing and prescribing workouts are features solely for the Android Application.

Activities	Activities are exercised based activities which hold data such as date, duration and distance. These figures are then used for prescribing workouts and progress reports.

A.2 STAKEHOLDERS CONCERNS

Initiative Sponsors	Pierre Peclier, Eamon De Leastar, Siobhan Drohan
Project Development Team	Colm Carew
System Administrator	Colm Carew

A.3 REQUIRED QUALITY PROPERTIES

Performance	System must have acceptable performance and response time for requests. Proximity of components should remain close to not lose performance due to communication overhead.
Usability	The Android Application must be simple to understand and use. The user should be able to navigate through all the Android Application's features without any impediments.
Executability	The Play Application should be easily deployable to another server without large issues. The Play Application should run as cheaply as possibly due to limited resources.
Portability	The system should be relatively easily built upon for a different platform such as iOS.
Topology	The main application should not become monolithic.

B) Solution Description

Below are the key characteristics of the solution design.

B.1 ARCHITECTURE VIEWS

VIEW NAME	PURPOSE	TARGET STAKEHOLDER(s)
CONTEXT View	The purpose of this view is to outline the main elements at play in the design of the Pacemaker dashboard: 1.What is the purpose of the system 2.What are the systems features	Data Analyst, CIO
BEHAVIORAL View	List processing flows at play for the rendering Pacemaker's dashboard. The purpose of this View is to outline: 1.How data gets collected and prepared for storing 2.How analytics data is served to users	CIO Data Analysts
INFORMATION STRUCTURE View	Describes how the Pacemaker system stores and manages data. 1. Analytics Data Flow - how the application processes and calculated data 2. Rendering Data Flow – how data is stored and retrieved	Data Analysts Developers
FUNCTIONAL View	This specifies the key components of the Pacemaker system and their relationships with one another.	Developers
INFRASTRUCTURE View	The purpose of this view is to outline the infrastructure behind the design such as servers, databases, applications, how they connect to one another and how the user can connect to the applications.	System Administrators

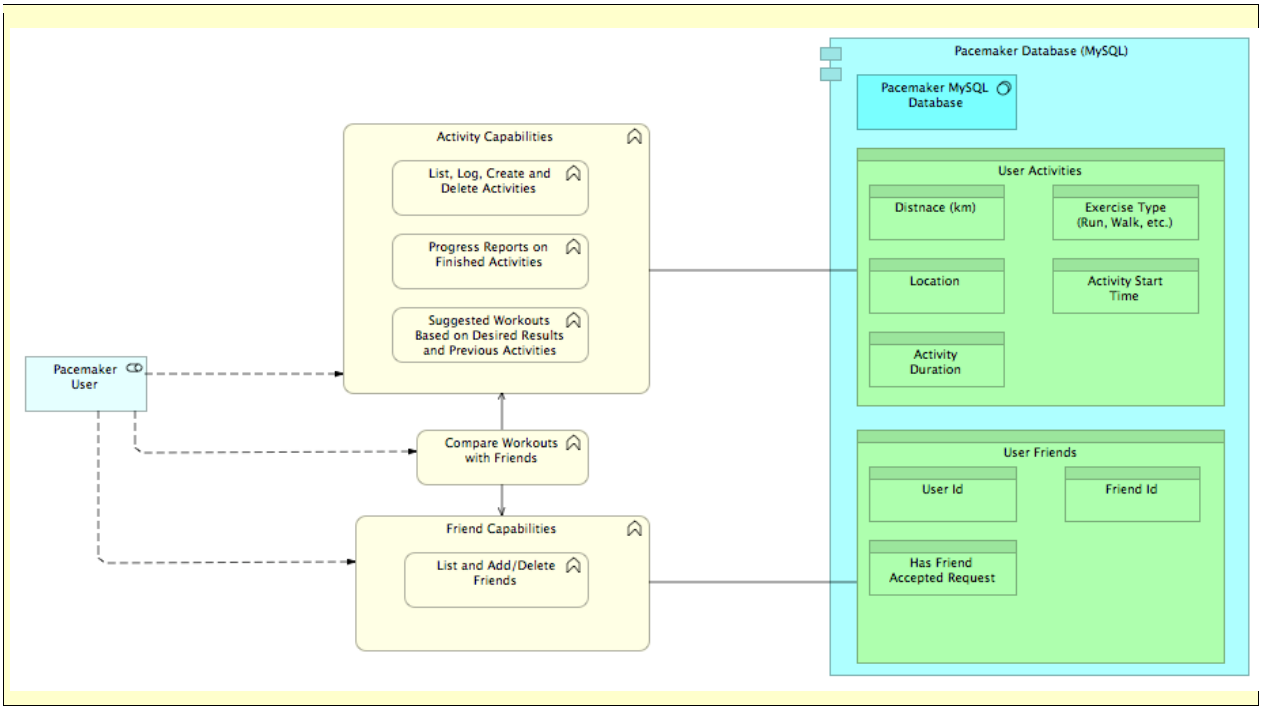
B.2 CONTEXT VIEW

B.2.1 VIEW INTENT

MODEL NAME	PURPOSE
SYSTEM CONTEXT Model	The purpose of this model is to show the solution in terms of system components.
SYSTEM FEATURES Model	The purpose of this model is to show the solution in terms of it’s features.

B.2.2 VIEW MODELLING ARTIFACTS

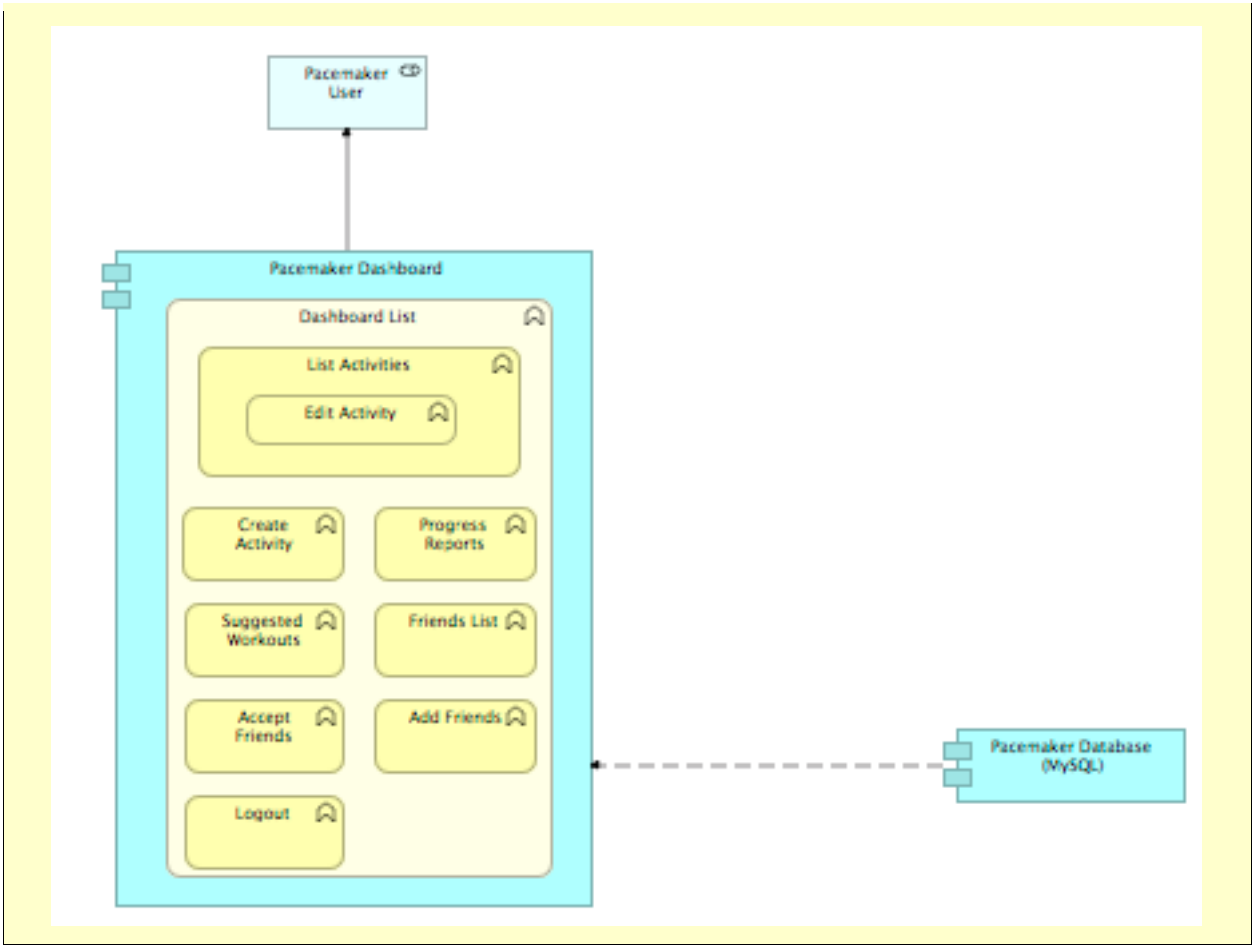
B.2.2.1 SYSTEM CONTEXT MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Context View Type	To illustrate the relationship between the solution and the environment.
MODEL KIND USED	**JUSTIFICATION / INTENT**
System Context Modelling	This model shows the Pacemaker Dashboard capabilities in the centre of the model. All the user data is stored and collected from the MySQL database. Analytics data is calculated at run time and is not stored. Data used to calculated the analytics are the finished activities which are stored in the database.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**
Data-driven Style: Agile Data Processing	User Analytics data is calculated at run time and is not persisted to any database. Data regarding activities is pulled from the database when the user requests analytics calculations so all calculations are near real time.
Domain Pattern: Shared Database	User data, activities and friends are all stored within the same database. Data can be pulled or pushed from external applications into this database via REST with the Pacemaker

	Play Application.

B.2.2.2 SYSTEM FEATURES MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Context View Type	The purpose of this view is to outline the main features of the Android Application.
MODEL KIND USED	**JUSTIFICATION / INTENT**
System Feature Modeling	This model shows that the Application is composed of one large component, the Pacemaker Dashboard. This model shows how features of the solution are decomposed and allocated within the main components. Most of the features require some data being pulled and manipulated from the database or data being edited/created.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**

Data-driven Style: Agile Data Processing	User Analytics data is calculated at run time and is not persisted to any database. Data regarding activities is pulled from the database when the user requests analytics calculations so all calculations are near real time.
Domain Pattern: Shared Database	User data, activities and friends are all stored within the same database. Data can be pulled or pushed from external applications into this database via REST with the Pacemaker Play Application.

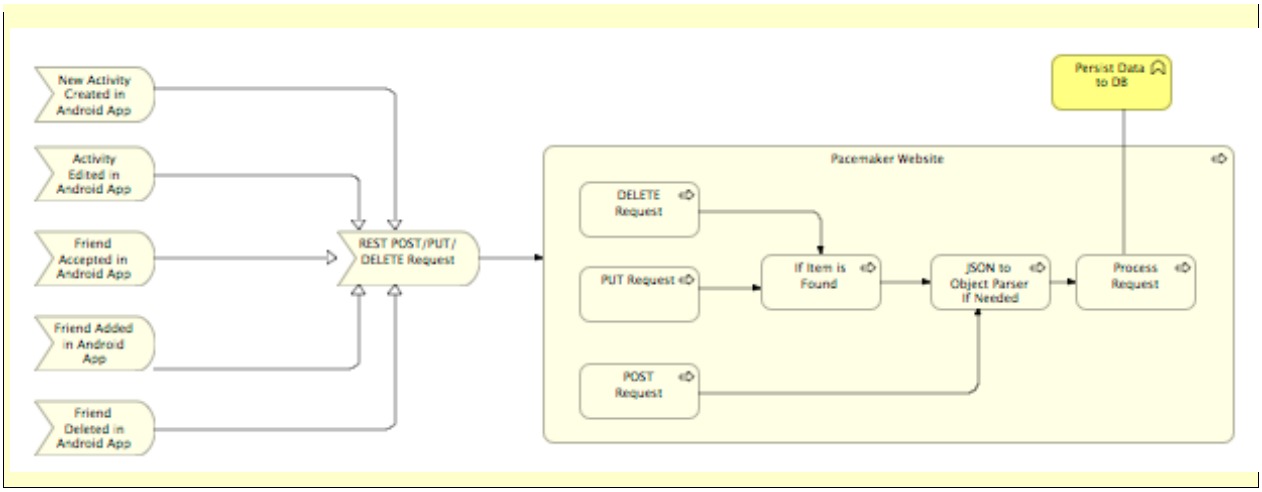
B.3 BEHAVIORAL VIEW

B.3.1 VIEW INTENT

MODEL NAME	PURPOSE
Data Persistence	This model describes the data persistence process. The process is triggered every time a User edits/create data(e.g. a new activity, a modification of an older activity) in the Android Application. The flow describes how all data is created, edited and deleted in this Pacemaker Android solution.
Data Rendering	This model describes the type of tasks the Pacemaker Dashboard performs to fetch the activity data from the Pacemaker website (REST communication), and transform the data to render it on the Android screen.

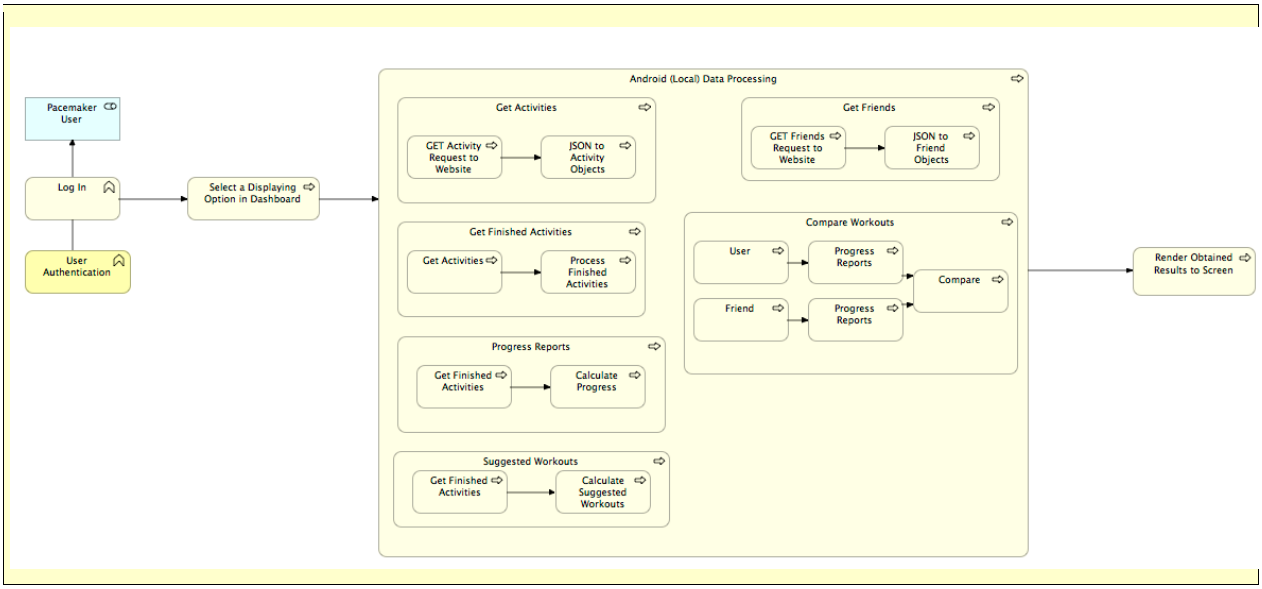
B.3.2 VIEW MODELLING ARTIFACTS

B.3.2.1 DATA PERSISTENCE MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Behavioral View Type	The purpose of this view is to show communication mechanisms required to coordinate operations between functional elements. It also outlines triggers, events, and flow-logic..
MODEL KIND USED	**JUSTIFICATION / INTENT**
Process Flow Modeling	This model shows how a user request in the Android Application for editing/creating data persists to the database through REST requests.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**
Service Registry Pattern	The Android Application needs to determine the location of the Pacemaker website in order for it to to send REST requests.

B.3.2.2 DATA RENDERING MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Behavioral View Type	To outline the process by which the activity and friend data is fetched and rendered in the Android Application. It also outlines the steps that manipulate data before it is displayed to the user.
MODEL KIND USED	**JUSTIFICATION / INTENT**
Process Flow Modeling	This model shows the steps involved in activity and user data being obtained from the Pacemaker website via REST. The

	data is then processed via the Android Application and displayed on screen for the user to view.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**
MVC Pattern	Model-View-Controller pattern to render Android Activity pages based on the data collected and manipulated from the user activities and friends. The model for this pattern is the data (the user or the activity), the controller is the logic behind processing and manipulating the data and the view is the Android Activity the user sees.

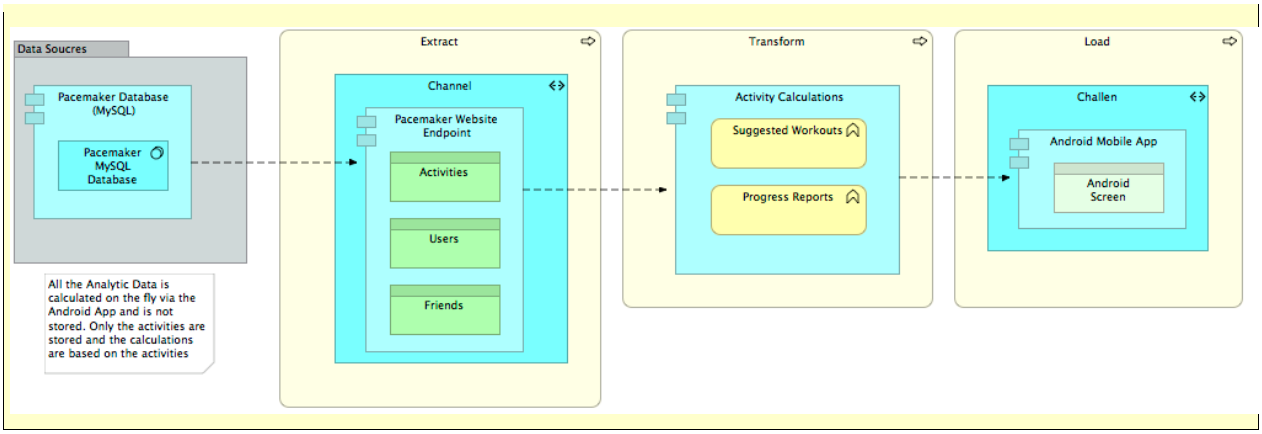
B.4 INFORMATION VIEW

B.4.1 VIEW INTENT

MODEL NAME	PURPOSE
Analytics data flow	This model outlines the components that transform data pulled from the web application. The data flow described in this model is a similar to an ETL model as the data is extracted from the database (REST from the web application), transformed via domain logic in the Android Application and finally loaded onto the users screen.
Rendering data flow	This model presents the flow of requests and responses of objects throughout application components. It is triggered by a user performing an action in the Android Application which causes data to be requested via REST.

B.4.2 VIEW MODELLING ARTIFACTS

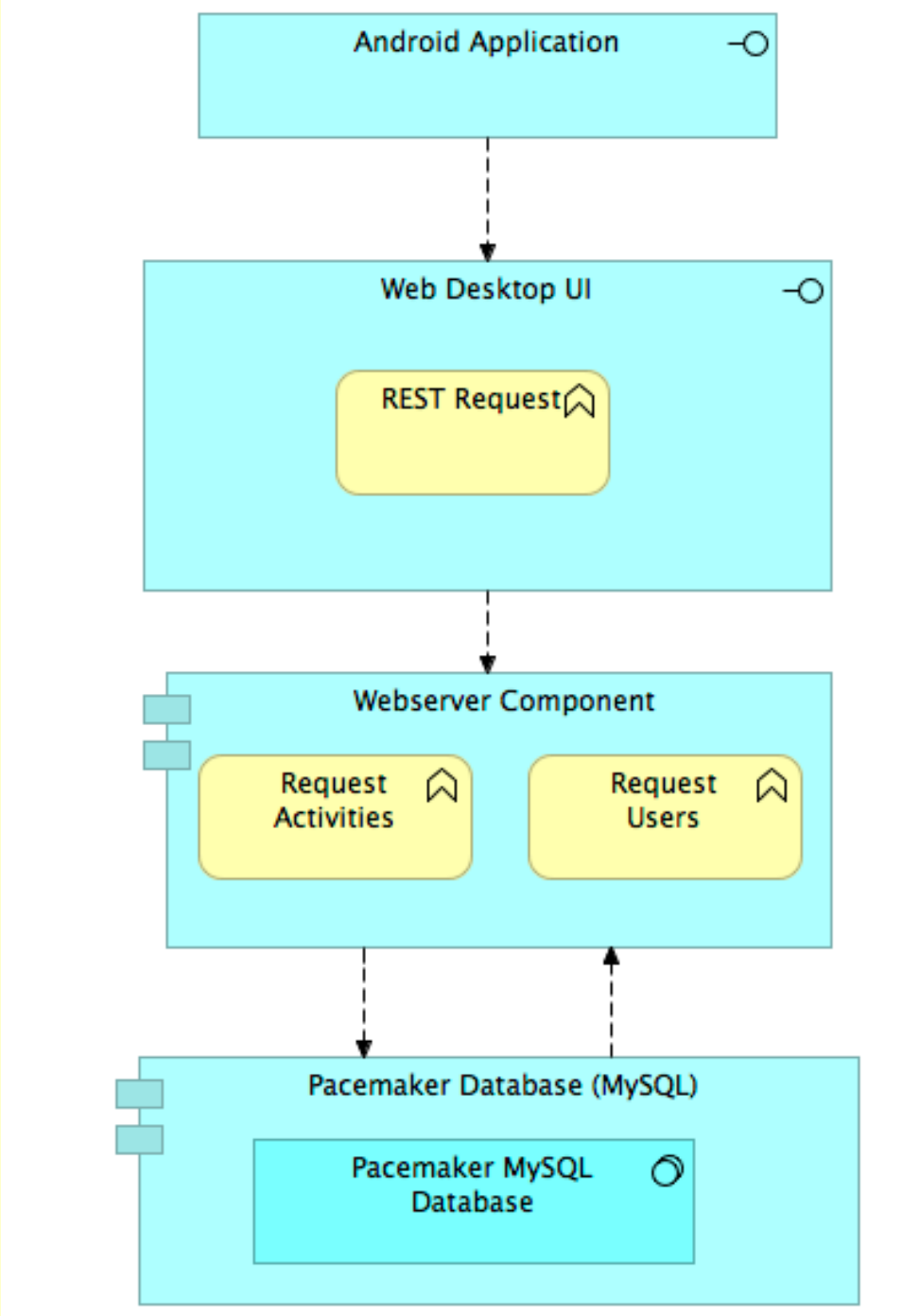
B.4.2.1 ANALYTICS DATA FLOW MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Information Services View Type	Describes how the architecture stores data and define any

	significant data structures used within the system. It also shows the data exchange between functional elements
MODEL KIND USED	**JUSTIFICATION / INTENT**
ETL Orchestration Modelling	To describe the roles of application components in the extract, transform and load phase of data.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**
Pipes & Filters Pattern	IThe purpose of this pattern is to implement a message processing mechanism which is platform independent. Thus making it re-usable for future views. IT also identifies the type of transformation functions and data objects involved in the input and output of each pipe and filter.

B.4.2.2 RENDERING DATA FLOW MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Information Services View Type	Describes how the architecture stores data and define any significant data structures used within the system. It also shows the data exchange between functional elements
MODEL KIND USED	**JUSTIFICATION / INTENT**
Data Flow Modelling	This model shows the flow of data when the Android Application makes a REST request to the Play Web Application.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**
Synchronous Point to Point Processing pattern	In this case this pattern sends a message from the Android Application to the one receiver which is currently the Raspberry Pi running the Pacemaker Web Application. There is only one receiver for now so only one receiver can receive the message.

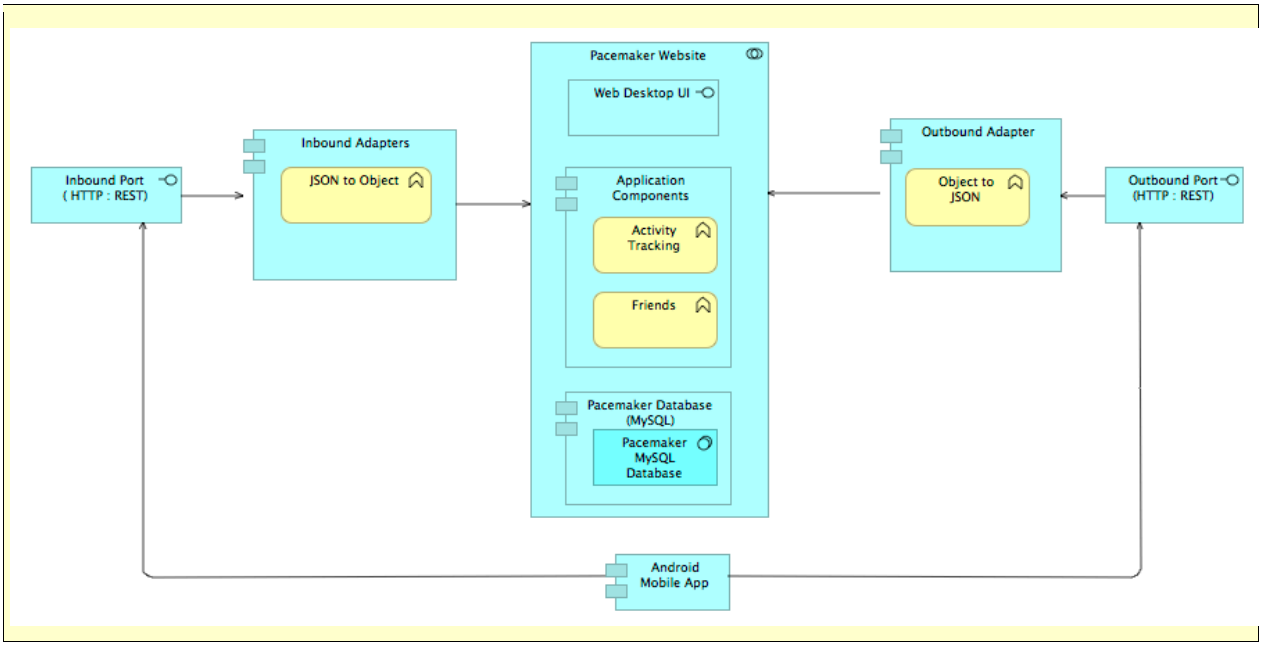
B.5 FUNCTIONAL VIEW

B.5.1 VIEW INTENT

MODEL NAME	PURPOSE
Ports and Adapters	The purpose of this view is outline the structure of the Pacemaker Dashboard. The main reason behind using this style is to allow clients (regardless if client is a database, a user or another system) to interact with an application in a consistent way. In the case here the interaction with the Play Application is via the Android Application using the REST API.

B.5.2 VIEW MODELLING ARTIFACTS

B.5.2.1 PORTS AND ADAPTERS MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Functional View Type	Describes the functional elements of the system in terms of their static functional structure. As illustrated by the image the main function of this system is the Play Application with the Android Application being a front for some of the features.
MODEL KIND USED	**JUSTIFICATION / INTENT**
Application Structure Modelling	The purpose of this structure is to allow the various applications to communicate with the main application (Play Application) in a consistent manner.
STYLE/PATTERNS USED	**JUSTIFICATION / INTENT**
Ports and Adapters Style	This styles aims to provide a level of abstraction by reducing coupling of the environment. It allows an application to be driven by users, applications and automated scripts.

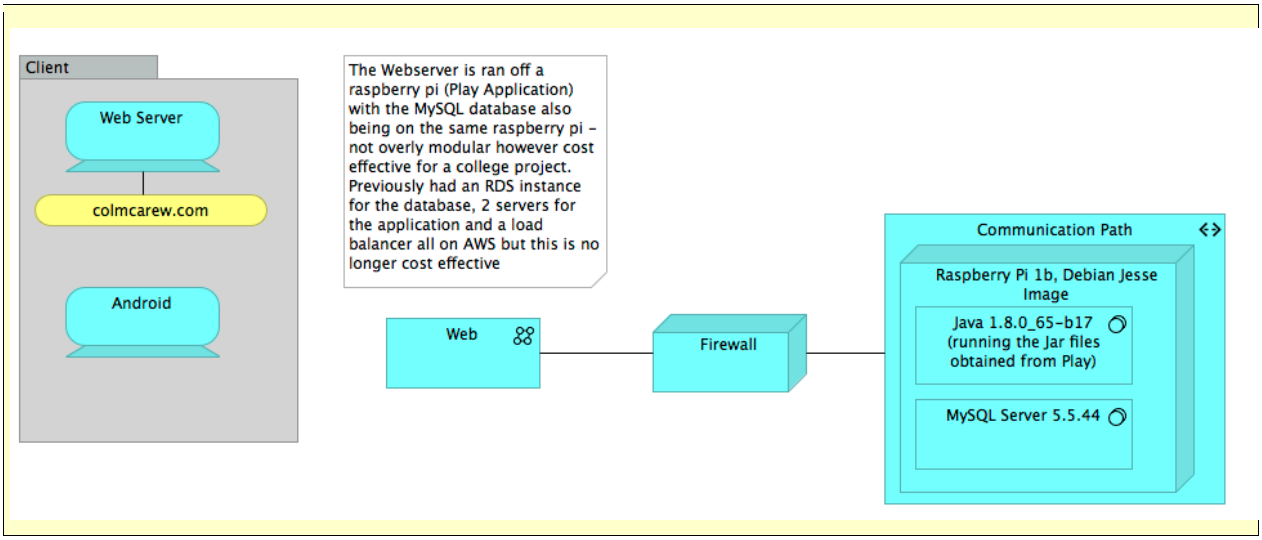
B.6 INFRASTRUCTURE VIEW

B.6.1 VIEW INTENT

MODEL NAME	PURPOSE
Infrastructure	The purpose of this view is outline the infrastructure required to run the system. It shows all applications and databases and what server/framework each is run on.

B.6.2 VIEW MODELLING ARTIFACTS

B.6.2.1 INFRASTRUCTURE MODEL



VIEWPOINT USED	JUSTIFICATION / INTENT
Infrastructure View Type	Describes the hardware and software used for the system to be operational.
MODEL KIND USED	**JUSTIFICATION / INTENT**
Application Structure Modelling	The purpose of this structure is to allow the various infrastructural components used to be transparent. Should be designed so reconfiguration of the infrastructure is not an issue (e.g. adding a server as an application handler).

B.7 ARCHITECTURE PERSPECTIVES

B.7.1 PERFORMANCE PERSPECTIVE

B.7.1.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	**System must have acceptable performance and response time for requests**	**Decision**: All components of the architecture must be kept in close proximity such that the Web Application has a solid connection to the database. **Trade-off:** All components are in one location with no backup currently. Moving any of the components will caused

		performance issues due to communication overhead.

B.7.2 USABILITY PERSPECTIVE

B.7.2.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	**The Android Application must be simple to understand and use.** **The user should be able to navigate through all the Android Application's features without any impediments.**	**Decision**: Android Application must be kept simple in order for people of all age groups and computer skills to be able to use it. **Trade-off**: Application may become too simplified such that it does not serve a purpose or perform any novel task such as to entice new customers.

B.7.3 EXECUTABILITY PERSPECTIVE

B.7.3.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	**The Play Application should be easily deployable to another server without large issues.** **The Play Application should also run as cheaply as possibly due to limited resources.**	**Decision:** The application and database are run off the same server which is a Raspberry Pi model 1b. This is cost effective as the Pi consumes little power and there is only one server. **Trade-off**: Risk of architecture of becoming monolithic as all functional or the application are in one place with ever other front end application connecting to this possible monolith.

B.7.4 PORTABILITY PERSPECTIVE

B.7.4.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	**The system should be relatively easily built upon for a different platform such as iOS.**	**Decision**: The Play Application should store all information relating to any of Pacemaker applications. This allows all new applications being developed to obtain up to date data via REST. **Trade-off**: Main application can become monolithic over time and data storage may become an issue depending on number of users and uploads.

B.7.5 TOPOLOGY PERSPECTIVE

B.7.5.1 RECORDING of DESIGN DECISIONS / TRADE-OFFS

.ID	DECISION ITEM	DECISION MADE
#1	**The main application should not become monolithic.**	**Decision**: Using the Ports and Adapters Styles opens the design up to becoming monolithic. This is due to there being one main node. **Trade-off**: For design not to become monolithic the features set will need to remain humble rather than verbose as all feature additions will cause main node to grow.