# Architectural Requirements Notes

P.A.P.E.R.S (Publication And Papers Electronic Repository System)

University of Pretoria - Team Charlie

**Notes By:**

Claudio Da Silva - 14205892
Dillon Heins - 14035538

GitHub Repository - Team Charlie

For more information, please click here

5 March 2016

# Contents

# List of Figures

# 1 Architecture requirements

## 1.1 Architectural scope

## 1.2 Quality requirements

- Performance
  - Workload is a maximum of 100 users concurrently.
  - No implementation of concurrent editing of document entries - last saved edit is written to the database.
  - The system should be able to support 100 users updating information at the same time as updating is more intensive than reading.
  - The response time of the system should be fast enough that a user is able to complete their work without frustration. Due to the system being off-line it is reasonable to expect the system's response time to be only limited by the speed of the network.
- Reliability
  - The system should not fail whilst providing critical or important use cases.
  - The system should handle all requests and respond with appropriate result objects for each.
- Scalability
  - Ability for multiple external systems to connect to the system's API.
  - The system should be able to support a large amount of historical document entries being added to the database.
- Security
  - A hierarchical system will be used to determine the security privileges of users of the system.
  - Passwords are to be hashed using at least sha256 and should be stored as such within the database along with a salt.
  - An inactive user session should be terminated after a period of 10 minutes with no activity.
  - A user who has forgotten their passwords can use a password reset option which will send a one time password to their registered email address so that they may login once using it and reset their password.
- Flexibility
  - The client has stated that the system is not needed to be able to extend to accommodate a greater number of departments.
- Maintainability
  - The system should have as few bugs as possible so as to prevent having to constantly maintain it in the future.
  - The system should be built in a modular way so that all services are decoupled in such a manner that allows for the extension of the system at a later stage.
- Auditability/monitorability
  - Every action performed by a user should be logged and all details about said action should be stored.
  - These actions should be visible to admin users.

- Integrability
  - User's document entries should not be able to be deleted, if it is a case where the document will not be completed it should remain in the system and be terminated.
  - A user with no admin rights should not have access to admin privileges so that the system's data may remain integrable and safe.
  - The system should not ever be in a state where it is under pressure and the data is at risk of becoming corrupted. The system should be designed to handle the pressure for which it has been specified to handle.
- Cost
  - All software used should not be proprietary but rather open source so as to minimise cost as much as possible.
- Usability
  - The interface should be lightweight.
  - The interface should be intuitive to use as well as obey Human Computer Interaction guidelines so that it is efficient and easy to use.

## 1.3 Integration and access channel requirements

The system will make use of:

- Django (Model View Controller based framework)
- HTML5 compliant front-end to ensure compatibility with Android
- Lungo (Android framework based on HTML5 an CSS3)
- Sencha Touch (HTML5 Framework for Android)
- Android SDK (Final changes to Android app to ensure compatibility)

The different access channels through which the system's services will be made available to users as well as other systems are as follows:

- An Application Program Interface residing on a server which will be interfaced with by clients in order to supply services to them. Clients referring to:
  - Human users via an interface
  - External systems using the services provided by the API
- Human users can interface with the system via the use of:
  - a web-based application service
  - an android based mobile application

The interface is required to be lightweight.

## 1.4 Architectural constraints

- The system will be a server based API accessible via a web browser as well as an Android application.
- The system may not be browser or operating system specific, and should be able to run on any system chosen.
- The system should be handled locally and should not rely on outside internet sources in order to function.
- The database used must be a relational database as the information being stored requires both a standard structure as well as relations on fields.
- Modules must be decoupled as far as possible, allowing as much pluggability and further editing as need be when the system grows larger.
- A proper framework such as Django will probably be used in favour of a standard PHP approach during the program's design.
- The technology used may not be proprietary and should be free and available to use for all.