UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Software Requirements Specification
and
Technology Neutral Process Design
Research Paper Management System

University of Pretoria

Team Echo

Surname, First Name (Initial)	**Student Number**
Bode, Elizabeth (EF)	14310156
Bondjobo, Jocelyn (JM)	13232852
Broekman, Andrew (A)	11089777
Loreggian, Fabio (FR)	14040426
Schutte, Gerome (GC)	12031519
Sefako, Motsitsiripe (MG)	12231097
Singh, Emilio (E)	14006512

Table of Contents

1 Vision

The client, Vreda Pieterse, from the University of Pretoria has requested a system to keep track of research publications in the Department of Computer Science at the University of Pretoria. The scope of the system is managing the administration involved in tracking of research publications within the department. However collaboration on research papers is outside of the scope as a version control system is in use currently. The system is required to keep track of all publications and the associated meta data around the publications.

The department currently consists of various research groups, with various contributors, all producing research publications which should be tracked. As the University of Pretoria is rewarded for research produced, it is important for both the heads of research groups and the Head of Department to monitor the progress of current publications, and to gather other metrics to assist in other departmental planning.

The most important requirements as set out by the client are:

1. Monitoring the research output by the department through metrics by the Head of Department and heads of the research groups
2. Tracking the status of research publications
3. Being able to produce customized reports based on user defined values

2 Background

This project was commissioned based on the following three problems faced by the client

- Limitations of current system:
  - The current system used by the client involves utilizing one common Microsoft Excel spreadsheet document to keep track of the status of current research publications. A problem with this solution is that it causes data integrity problems due to the fact that it is not scalable.
- Increased productivity and transparency
  - A common issue faced by researchers in the Computer Science department is that an increased workload at due dates leads to the spreadsheet becoming unstable and subsequently crashing. Necessary data recovery on the spreadsheet leads to increased downtime, which, while also necessitating repeated data entry, decreases user productivity. Clearly, not a user friendly system, causing many a frustrated user.
- Monitoring and management of funding
  - Currently the department receives funding based on research produced. The funding model, and hence the amount, is based on various factors pertaining to the research publications, such as whether it is published, presented at a conference and so on. Therefore, having access to projections of the number and type of publications that are to be produced in coming years is very important to strategic departmental planning, as is monitoring historical performance of research groups and individual researchers.

3 Architecture requirements

3.1 Access channel requirements

3.1.1 Human Access Channels

The human access component must allow for both a desktop web client and an Android mobile client allowing the user to manage the meta data associated with research papers they are involved in.

The web client will be the primary contact point for users of the system as most research papers are created, edited and maintained on personal computers that either owned by the contributor in question or the University of Pretoria. The web client will be based on an MVC based SPA architectural pattern.

The Android application will be made available in the Android App Store to allow contributors from outside the University of Pretoria to easily gain access to the system. The Android application should be accessible by all mobile devices running Android version 4.2 and upwards.

3.1.2 System Access Channels

All system access will be made over a uniform interface which will assist in faster development time, ease of maintenance and lower overall cost associated with the system. The common interface that will be used, will be a RESTful API based on the dissertation Architectural Styles and the Design of Network-based Software Architectures of **fielding** All client side interaction will be done via the web client or Android application which will in turn use the RESTful API set of the system to accomplish the given task.

3.2 Quality requirements

3.2.1 Performance

1. **Description** Application performance can be defined as the amount of work that can be accomplished by an application in question in a measured time interval. The time interval is normally measured in seconds, where the amount of work can be defined as the throughput, latency or data transmission time.

   - **Throughput**
     The number of requests and responses which can be processed by the system in a given time interval.
   - **Latency**
     A time interval measured as the time it takes to service a request.
   - **Data transmission time**
     The time it takes to transmit a request to the server from a client or a response from the server to the client. This is heavily influenced by the size of the request, as well as the quality of the network medium.

   The aim for this system, is to increase the throughput and decrease the latency. As the developer has no control over the network medium used, he/she must aim for a minimal request and response payload as to decrease the data transmission time.

2. **Justification**

3. **Requirements**

   - Network requests must be benchmarked, monitored and aggregate statistics about network requests must be made available.
   - Function calls must be timed and benchmarked and this data should be logged.
   - Precompiled SQL must be used in statements.
   - Network responses should be cached on server side to lighten the load on database as well as decreasing the round trip time of request - response.

3.2.2 Reliability

1. **Description** The design of this system is largely based on the fact that the current system used by the client is not reliable and suffers from system crashes and data loss. The newly designed system needs to be accessible from both inside University of Pretoria campuses as well as from other networks, especially on other campuses on the TENET network. The system should be reliable in both its accessibility and in the data that the system will use to calculate contributor metrics.

2. **Justification**

3. **Requirements**

   - To enable offline activity and access of data from the Android app, data synced must be made differentiable with timestamps.
   - The system must resolve synchronization conflicts using timestamps.
   - Hot swapping of system modules should not affect system service reliability.

3.2.3 Scalability

1. **Description** Scalability refers to the application in question's ability to handle an above normal workload for extended time periods and the mechanisms employed to facilitate this. The client has specified minimum conditions under which the system needs to function. However the client has indicated that due to the business requirements, the system will experience higher workloads during the end of the month.
2. **Justification**
3. **Requirements**
    - The system should be able to handle 10 research groups with each research group containing 100 members.
    - The system should be able to facilitate 50 concurrent user sessions at any time.

3.2.4 Security

1. **Description** Security in application software refers to authentication, authorization, data security and accounting. Authentication refers to the systems' ability to provide a way of identifying a user, normally with a combination of a user name and password, or by using access tokens. Authorization refers to the systems' ability to determine whether the user in question has the required authority to execute certain commands or tasks. Data security refers to data only being accessible through authorised channels. Finally if the user has been authenticated and has the required authorization then accounting needs to be done to be able to determine how the users action has changed the system.
2. **Justification**
3. **Requirements**
    - System should be resistant to SQL injections.
    - A user group hierarchy system should be used to manage user access rights.
    - Authentication credentials such username and password should not be stored on Android user device. A token based authentication approach should be utilized so that only tokens are stored on end user device, allowing for the easy revocation of device access if a device is lost. Available technologies, such as OAuth, should be utilized.
    - Password hosting with a unique salt for each user should be used.
    - A key derivation function should be used with passwords.
    - After N invalid attempts, account should be locked for M min, where N and M would be agreed upon customer and developer based on current best practices.
    - Database access must require authentication.

3.2.5 Flexibility

Flexibility refers to the ability of the system to be changed dynamically either by hot swapping certain components in a live system or by extending the system with some kind of plugin.

1. **Description**
2. **Justification**
3. **Requirements**
    - The system should allow unregistered users to use existing third party web service login credentials to register or log in to the service.
    - The system should be decoupled from the database technology it uses and allow the client to select and change the database it uses in future.
    - Authentication mechanisms used should be decoupled from the system, allowing them to be interchangeable.
    - Modules should be decoupled from one another, allowing the system to be extensible without a break in service which is achieved by integrating new modules and swapping out existing ones.

3.2.6 Maintainability

The system is to be designed in such a way that it is easily updated, modified or exitable by the client in the future. In order to achieve these requirements, design patterns and best practices such as coding style guides are normally used to ensure uniformity and modularity across the system.

1. **Description**
2. **Justification**
3. **Requirements**
    - All code should be documented in the applicable language documentation framework, such as JavaDocs for a Java based system, DOxygen for a C/C++ based system, etc.
    - A coding style guide/manual should be set up and associated with the project, such that all developers use similar coding styles and conventions, to allow for more readable code that is easier to maintain.
    - System should be separated in distinct, concise and independent modules relating to separate concerns, to allow for easier maintenance.

3.2.7 Auditability/Monitorability

1. **Description**
2. **Justification**
3. **Requirements**
    - Data in database should be always be consistent. This implies that all data should adhere to constraints placed on the data by the data model, such as regex patterns, minimum and maximum length, non nullable fields, etc.
    - No data should ever be deleted.
    - It should always be possible to see which user created which objects at what time.
    - It should always be possible to see which user last changed objects and at what time.
    - An immutable log of user actions should be kept and should only be accessible by administrative staff.
    - Logging of stack traces and crash analytics should be implemented in the mobile client, to ensure the developers can see to client reliability.

3.2.8 Integrability

1. **Description** The system should allow for future external integration with other platforms such as security authentication providers, other external research meta-data databases etc.
2. **Justification**
3. **Requirements**
    - The system should allow technology neutral importing and exporting of data.
    - The back-end system should integrate with a desktop web client and Android mobile app clients.
    - The system should be able to integrate with different back-end authentication services.

3.2.9 Cost

1. **Description** The cost of the system entails any initial expenses as well as any ongoing expenses which the client may incur at some point. Such expenses arise from software licenses, external computing resources required as well as future maintenance of the system in terms of time.
2. **Justification**
3. **Requirements**
    - System should be cheap to operate, maintain and extend. If the quality and maturity of technologies available allow it, technologies used must be freely available/usable.
    - As far as possible, open source compatible, mature technologies should be used, to ensure system stability as far as possible.

3.2.10 Usability

1. **Description** Usability refers to ease with which humans, and to a lesser extent, servers, interact with the system in question. Usability can be measured in various ways such as using quantifiable scientific measures or more subjective measures with a key question point being if the API follows conventions and so on.
2. **Justification**
3. **Requirements**
    - Each view in the desktop and mobile clients should be related to a single topic only.
    - The web client should render properly and be fully functional in modern web browsers.
    - Mobile devices running Android 4.2 and upwards should be fully supported.
    - Material design UI guidelines prescribed by Google must be used, to ensure that the clients feel modern and familiar.
    - Android app and web client should support the full back-end API specifications.
    - The user must be allowed to elect whether they would like their data to be available offline.
    - Mobile client users should be able choose between downloading data over Wi-Fi or 3G.

3.3 Integration requirements

The web interface should be accessible to most modern PCs with a modern browser. The website will be making use of an SPA MVC based design which will query a RESTful API, resulting in a very light, mobile and responsive web interface that will it to work across various devices.

The Android application will be built to work on all Android version 4.2 and upward devices. The mobile application on the Android device should be sensitive to the amount of data that is transmitted over the network due to the high cost of mobile internet in South Africa. To allow for minimal communication but maximum functionality, the MVC based design will be followed in the Android application design. Furthermore, the use of accessing RESTful APIs will allow objects to be transmitted on the wire with a very minimal representation, thereby saving on client bandwidth costs.

The RESTful API will be implemented using the HTTP protocol as a high level protocol. The reason that HTTP is chosen is that it is a widely supported and deployed protocol, allowing a lot of flexibility. Furthermore, most JavaScript libraries, and the Android platform itself, have excellent support for RESTful APIs over HTTP.

3.4 Architecture constraints

- Representational State Transfer (REST) will be used, it depends on cacheable, client server and on stateless communications protocol, and uses the HTTP protocol in most cases. It is an architectural style used to design networked applications. All the four CRUD operations are used in REST.
- Service Oriented Architecture (SOA). SOA makes it easier for software components on computers that are connected on a network to work together with no need to make changes to the underlying program itself. The concept of service is what SOA is based on.
- Spring will be used. It is a technology that is associated with developing secure websites and interacting with databases in conjunction with Hibernate. Spring is simply a usable API, with the idea being for the developer to represent Java objects with Spring Beans.
- Dependency Injection (DI) will be used because it eases software testing and re-usability of software by using a design based on independent classes/components.
- Unit Testing to automate the following tasks: compilation of Java source code, running test cases and generating project documentation.

3.4.1 Architectural Patterns

- Layered Architectural Pattern
- Representational State Transfer (REST) Architectural Pattern
- Services Orientated Architecture/ Microservices Architecture
- Client-Server Architectural Pattern

4 Functional requirements and application design

4.1 Use case prioritization

4.1.1 Critical

- Create User
- View/Edit User
- User Authentication[Login/Logout/Permissions]
- Create Paper
- View/Edit Paper
- Web Interface
- Create Authors/Researchers
- View/Edit Authors/Researchers
- Add/Remove Authors/Researchers to Paper
- Create Research Groups
- View/Edit Research Groups
- Add/Remove Users to Research Groups
- Add Venue
- View/Edit Venue

4.1.2 Important

- Import Historical Paper [Linked with Create Paper]
- Create Bibliography
- View/Edit Bibliography
- Research Paper Units
- User Units [Earnings]

4.1.3 Nice to have

- Dates for paper linking to Google Calender
- Android Interface
- Paper Progress Tracking
- User Hierachy
- Home View

4.2 Use case/Services contracts

4.2.1 User Login - User holds no tokens

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to initiate user login via the back-end service.
- Pre-Conditions
  1. The user is already registered with the back-end service.
  2. User account is not locked.
  3. User currently does not have required tokens.
- Post-Conditions
  1. The user will be logged in.
  2. The user will be issued with an access and refresh token.
- Service Contract

4.2.2 User Login - User holds access and refresh tokens

- Description
  This use case will be used by the REST clients, specifically the Android app, to achieve safe long term login without storing user credentials on device.
- Pre-Conditions
  1. The user is already registered with the backend service.
  2. User account is not locked.
  3. User currently has access and refresh tokens.
- Post-Conditions
  1. The user will be logged in.
  2. The user access token will be updated using the refresh token.
- Service Contract

4.2.3 Create User

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Create a user of the system
- Pre-Conditions
  1. The user does not exist in the system
  2. All required details have been provided
  3. All details are valid
- Post-Conditions
  1. The user will be created in the database
- Service Contract

4.2.4 View/Edit User

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to View and Edit the currently logged in User
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. Editted details are valid
- Post-Conditions
  1. The user details will be editted
  2. User page will be displayed
- Service Contract

4.2.5 Create Paper

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Create a Research Paper
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. Paper details are filled in and valid
- Post-Conditions
  1. Paper succesfully created
- Service Contract

4.2.6 View/Edit Paper

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to View and Edit a Research Paper
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an Author of the Paper
  3. The user is an HOD or a Research Group Leader for the Group of the Paper
  4. The new details are valid
- Post-Conditions
  1. The papers details will be editted
  2. The paper details will be displayed
- Service Contract

4.2.7 Create Author

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Create Authors
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The details for the Author are valid
- Post-Conditions
  1. The Author is succesfully created
- Service Contract

4.2.8 View/Edit Author

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to View and Edit an Author
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The new details are valid
- Post-Conditions
  1. The Authors details will be editted
  2. The Authors details will be displayed
- Service Contract

4.2.9 Add/Remove Author to Research Paper

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Add and Remove Authors of a Research Paper
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an Author of the Paper
  3. The user is an HOD or a Research Group Leader for the Group of the Paper
  4. The new details are valid
- Post-Conditions
  1. The Papers Authors have changed sucessfully
- Service Contract

4.2.10 Create Research Group

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Create a Research Group
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an HOD or a Research Group Leader for the Group of the Paper
- Post-Conditions
  1. The Research Group is sucessfully added
- Service Contract

4.2.11 View/Edit Research Groups

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to View/Edit a Research Group
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an HOD or a Research Group Leader for the Group of the Paper
  3. The new details are valid
- Post-Conditions
  1. The Research Group have changed sucessfully
- Service Contract

4.2.12 Add/Remove Users to Research Groups

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Add/Remove Users to a Research Group
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an HOD or a Research Group Leader for the Group of the Paper
  3. The user is valid
- Post-Conditions
  1. The User is sucessfully added to a Research Group
- Service Contract

4.2.13 Import Historical Paper

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Import a Historical Paper
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The paper details are correct
- Post-Conditions
  1. The Historical Paper is sucessfully added
- Service Contract

4.2.14 Create Bibliography

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Create a Bibliography for a Paper
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an auhtor of the paper
- Post-Conditions
  1. The Bibliography is succesfully added to the paper
- Service Contract

4.2.15 View/Edit Bibliography

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to View/Edit a Bibliography for a Paper
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The user is an auhtor of the paper
  3. The new details are valid
- Post-Conditions
  1. The Bibliography is succesfully ediited
- Service Contract

4.2.16 Create Venue

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to Create a Venue
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. Venue details are valid
- Post-Conditions
  1. The Venue is succesfully ediited
- Service Contract

4.2.17 View/Edit Venue

- Description
  This use case will be used by the REST clients, specifically the web client and Android app, to View/Edit a Venue
- Pre-Conditions
  1. The user is succesfully logged in to the system
  2. The new details are valid
- Post-Conditions
  1. The Venue is succesfully ediited
- Service Contract

4.3 Required functionality

4.4 Process specifications

4.5 Domain Model

5 Open Issues

5.1 GitHub Repository

Team Echo Repository: `https://github.com/andrewbroekman/echo`
This repository contains:

1. All work done by team members.
2. CONTRIBUTORS.md file outlining which members where involved in which phases of the project.

Appendix A Data Dictionary

Term	Definition
Android	A Linux based operating system developed by Google Inc. and the Open Handset Alliance for smart phones, tablets, and other mobile computing devices. Android applications are developed in Java.
HTTP	HyperText Transfer Protocol, standarised by RFCs 7230-7237
ISO	International Organization for Standardization
MVC	Model-View-Controller
OAuth	An open standard for authorization, using an access token approach, allowing resource owners to authorize third-party clients to access protected user resources on a resource server, without the client providing their credentials to the third-party in question.
RDBMS	Relational DataBase Management System
SPA	Single Page Application
SQL	Structured Query Language is a programming language, standardised by ISO for managing data in RDBMS. Actions allow for the creation, reading/retrieving, updating and deletion of data in the management system.