Software Requirements Specification, Technology Neutral Process Design and Software Architecture Specification

Maree, Armand
12017800

Watt, Brenton
14032644

Meyers, Charl
14024633

Tom, Elton
13325095

Tswene, Kamogelo
12163555

Molefe, Keletso
14222583

Spazzoli, Lorenzo
13304862

March 11, 2016

Contents

1 Introduction

In this document it would be specified how a system would be developed for the department of computer science at the University of Pretoria in order for them to replace their current, not so efficient Microsoft Office Excel spreadsheet, system with a more concurrent and reliable option.

2 Vision

The client requires a system that would allow them to retrieve and submit meta-data about academic papers published by the department of computer science at the University of Pretoria. Certain users would be able to create new papers and the project leader would then be able to control certain properties of the project, like the current progress of the paper. It should also have a web interface where changes can be made and also an Android app.

3 Background

What lead to the project is the fact that that the client is facing a problem with the current system they are using. The spreadsheet which does fulfill its task of storing details about user's publications, does not have a user friendly interface, and does not allow researchers to see any changes made to the system immediately, it is also not a concurrent system and is deemed less reliable. The client wants a system that would improve the manner the users would be able to view their submissions and be able to modify the system to their liking, with changes immediately visible to researchers. Another aspect which lead to the development of a new system which would simplify the collaboration between the University of Pretoria users and users from other universities, is that instead of having a system which is local; a web interface could be used which is more accessible. The lack of ability to easily access previously entered user information data such as oneâĂŹs username or contact details also was a factor that lead to this project, as it would be more convenient and efficient to have to have an interface that would for example have a drop down list of all the user information data on the system.

4 Architecture Requirements

4.1 Access channel requirements

The system requires 2 interfaces:

- Web interface

- Android app (Could also be a mobile website)

4.2 Quality requirements

The following quality aspects needs to be addressed:

- **Scalability:** Less a 100 users will use the system.

- **Security:** Passwords will be hashed with the SHA512 hashing algorithm.

- **Reliability:** An automatic partial dump backup will be done everyday at 03:00 and each back up will be kept for a week. On top of this a full dump will be done every 3 days and will be kept for 2 weeks. Alternatively the backup will be done by an external program.

- **Audibility:** Every change should be logged and the person responsible for that log will be recorded. This will allow the administrators to track which users change what.

- **Maintainability:** Data that is deleted will only be moved to another database that might be a little slower. This will help alleviate some pressure off the main database.

- **Cost:** The total hours is estimated to be 120 at a cost of R500 per hour.

4.3 Integration requirements

The system should allow universities to connect to each other using an API. This would allow them to collaborate on projects and also to track the papers that are being written. The Android app should also be able to connect to the server via the API.

The protocol that will be used to transfer the data between the server and the clients will be the Hyper Text Transfer Protocol Secure (HTTPS). And the third parties that integrate with the system will access it either through the web interface, the mobile app or the provided API.

Some quality requirements that have to be considered are:

- **Security:** The data that is sent over the Internet should be encrypted and it has been decided to use the Secure Socket Layer (SSL) encryption algorithm.

- **Reliability:** The reliability of the transfer of the data is dependent on the reliability of the internet connection.

4.4 Architecture constraints

There are currently no architecture constraints that the client mentioned.

5 Functional requirements and application design

5.1 Use case prioritization

Critical:

- There must be some web hosting server in order to be able to host a web interface, this interface will be the main access channel to the system as it makes it easier to access the system on a computer or non-Android device. If there is no web-server then there will be no web interface that can be accessed from research groups all over the country. This defeats the whole purpose of the system.

- The web interface needs to be accessible from the Internet preferably so that other research groups outside the University are able to access the system.

- The system needs to be able to save data in some way. If there is no possible way to store data then the system might as well not exist, because the purpose of the system is to store meta data about research papers.

Important:

- Users must be able to download a bibliography of papers released in a selected period.

- An Android app must be available that can access the system to make it easier to access the system on the go.

- Users must create their user profile manually and will not be able to link the system with a web service in order to create their profile.

- No item should ever be deleted in the system. Everything is kept. Old unaccessed data may be archived but never deleted. There needs to be some sort of recovery or backup system that enables a user to restore accidentally deleted data.

- The system must be able to handle a max of at least 50 concurrent users as there is a possibility that at least 50 users may access the system all at once. There will also be an estimate of at least a 1000 users. The system must be able to handle that amount of users.

Nice-to-have:

- The system must be lightweight.

- The system must show the completion of a paper in percentage that a user specifies after editing the paper.

- The system may link with Google Calender to synchronize due dates and alerts for the paper a user is busy working on.

- The web interface can be designed to work on mobile phones for those who want to access the system on the go but do not have an Android device.

5.2 Use case/Services contracts

- **Use Case:** Add a paper.

  Primary Actor: Registered author or HOD.

  Brief: Only an author or head of department can add new papers.

  Postconditions:

  - The article is saved to the database.
  - All authors are notified of their new paper.
  - The changes are logged.
  - Paper is displayed.

  Triggers: The user invokes the Add New Paper request.

  Basic flow:

  1. Present the actor with a new paper form.
  2. Actor fill in the form and clicks submit.
  3. The data is sent to the server and added to the database and logs the event.
  4. Actor is presented with a success message.
  5. Paper is presented to the actor.
  6. Authors are notified.

- **Use Case:** Editing a paper.

  Primary Actor: Author of paper or HOD.

  Brief: Only an author involved with the paper or head of department can edit the paper.

  Preconditions:

  - Author is presented with the paper in edit mode.

Postconditions:

- The article is saved to the database.
- All authors are notified of the changes.
- The changes are logged.
- Paper is displayed.

Triggers: The user invokes the Edit Paper request.

Basic flow:

1. Present the actor with the paper in edit mode.
2. Actor makes all desired changes and clicks submit.
3. The data is sent to the server and added to the database and logs the event.
4. Actor is presented with a success message.
5. Paper is presented to the actor.
6. Authors are notified.

- **Use Case:** Viewing a paper.

  Primary Actor: Author of paper or HOD.

  Brief: Only an author involved with the paper or head of department can view the paper.

  Preconditions:

  - Author is presented with the paper in read-only mode.

  Triggers: The user invokes the View Paper request.

  Basic flow:

  1. Present the actor with the paper in read-only mode.
  2. When the actor is done viewing they simply navigate away. No changes or events has to be logged.

Request and Results Data Structures: See figure 1.
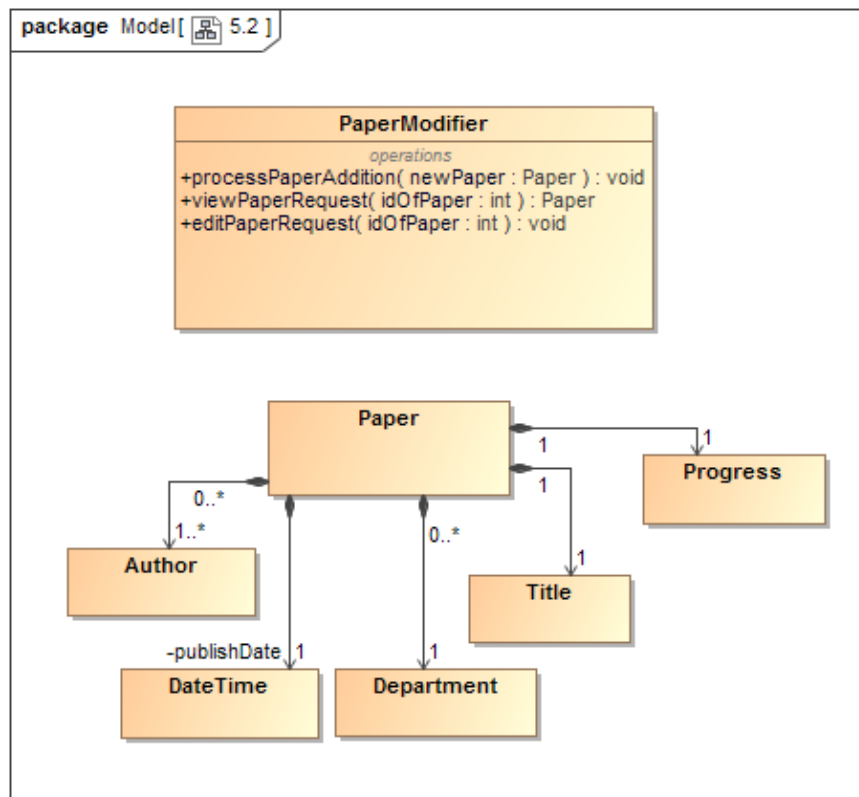
Figure 1: UML diagram for Request and Results Data Structure.

5.3 Required functionality
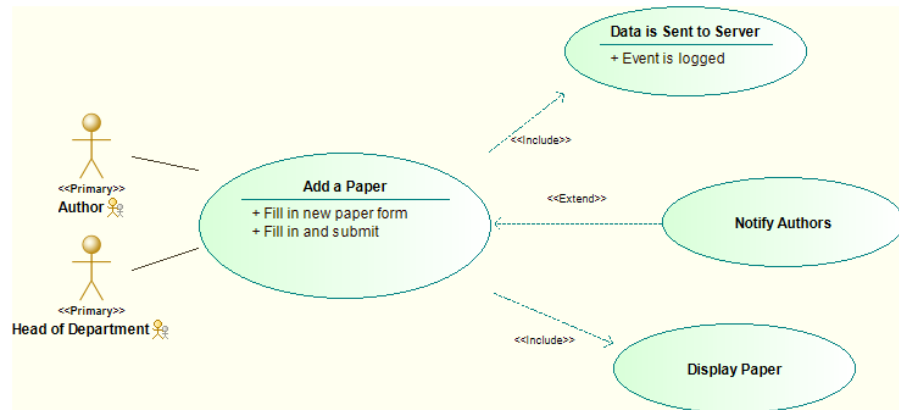
- Add a paper Use case: See figure 2

Figure 2: Use case diagram for adding a paper
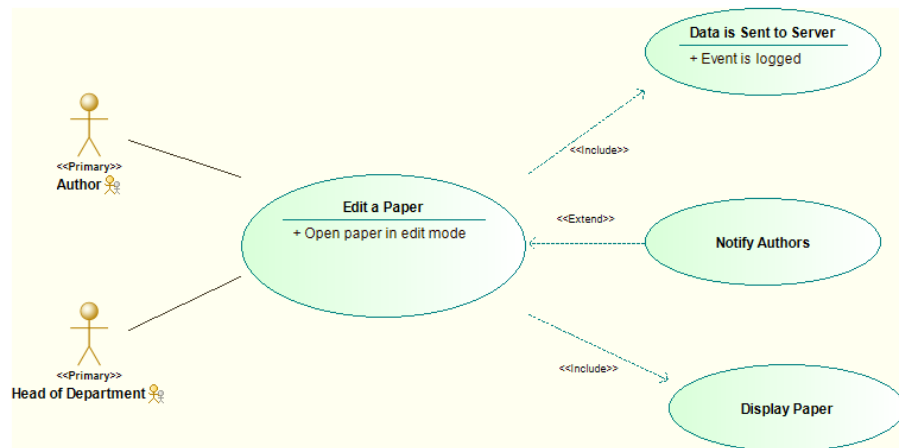
- Edit a paper Use case: See figure 3

Figure 3: Use case diagram for editing a paper
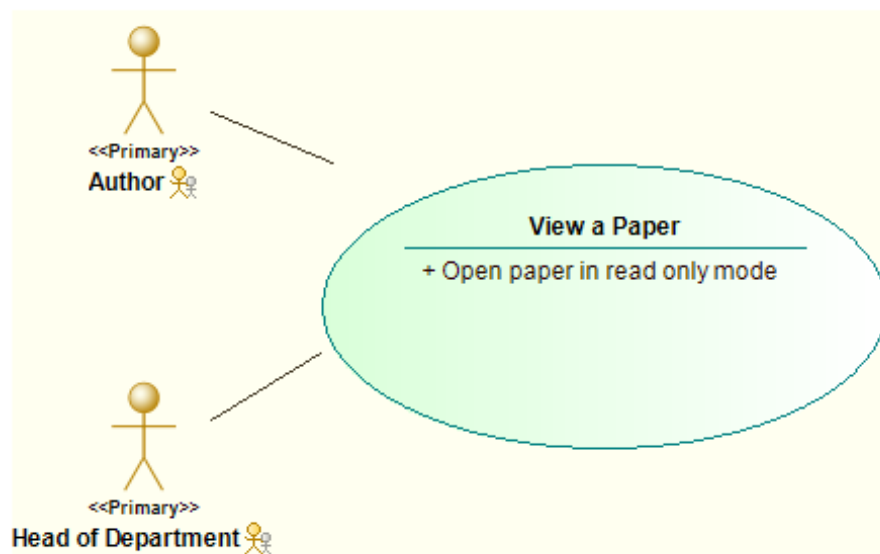
- View a paper Use case: See figure 4

Figure 4: Use case diagram for viewing a paper

5.4   Process specifications

- Activity diagram to add a paper: See figure 5
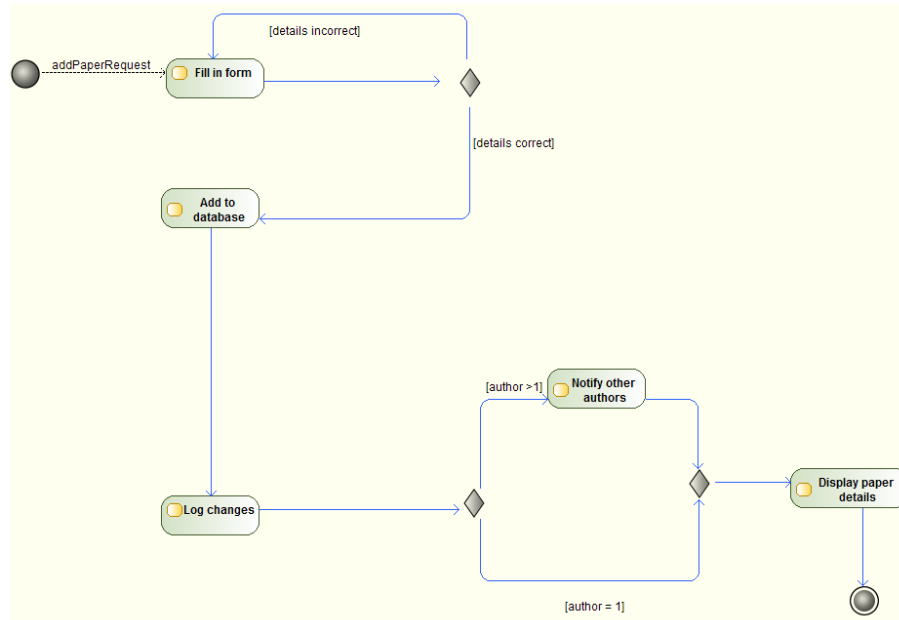

Figure 5: Adding a paper to the database

- Activity diagram to edit a paper: See figure 6

- Activity diagram to Login: See figure 7

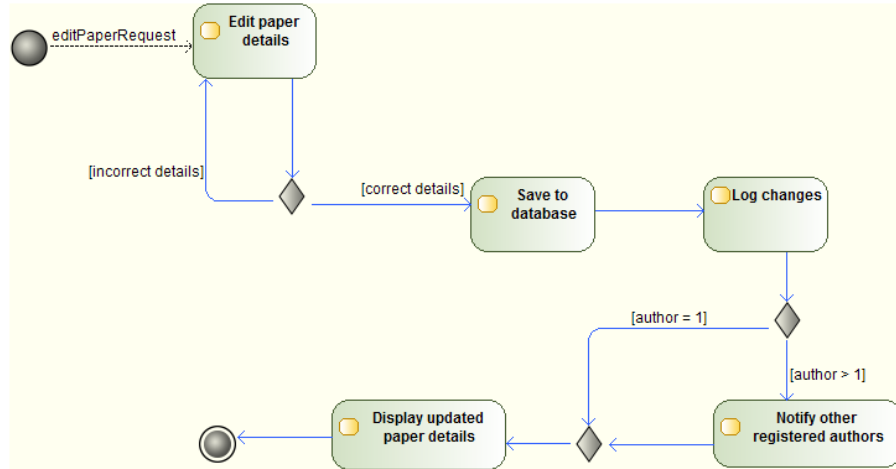- Activity diagram to Request document details: See figure 8
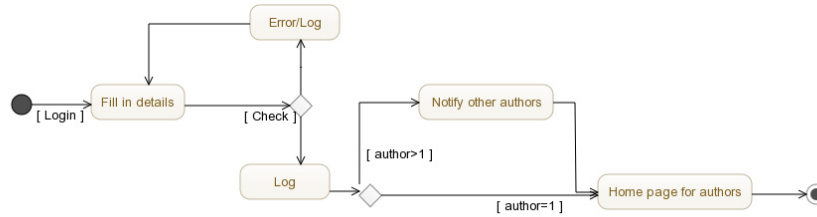
Figure 6: Editing a paper
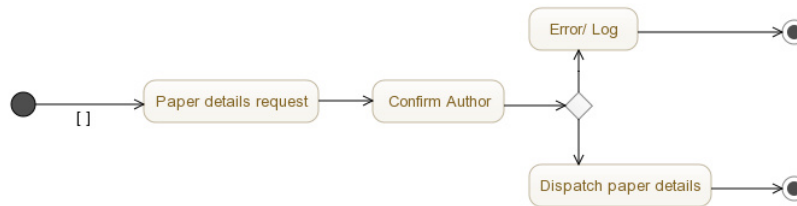

Figure 7: Login to system

Figure 8: Request paper details

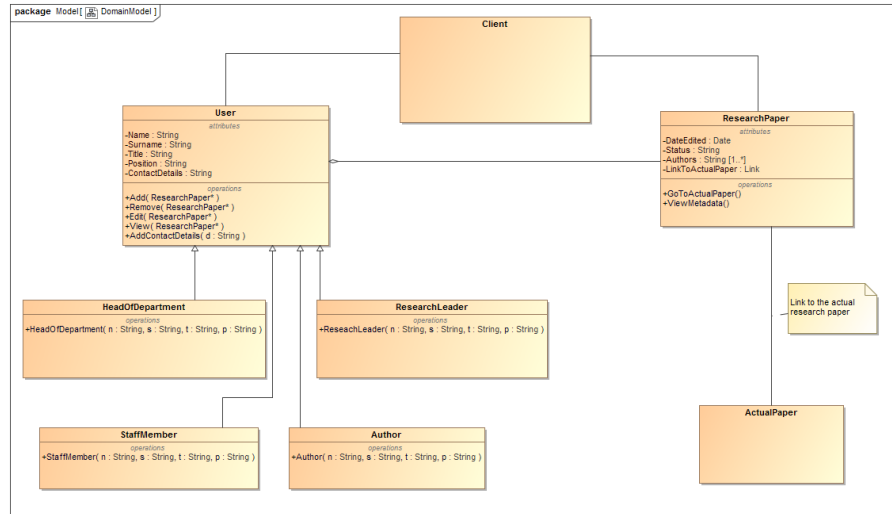## 5.5 Domain Model

- Domain Model of the system: See figure 9

Figure 9: Domain Model

6 Software Architecture Documentation

6.1 Architecture requirements

6.1.1 Architectural scope

The architectural responsibilities that need to be handled by the software architecture are as follows:

- A main database that will store the papers that are added.

- A secondary database (separate from the first) that will act as the backup database where the partial and complete dumps will be stored.

- An infrastructure that will allow for notification to contributing authors that changes have been made to a paper they collaborated on, this infrastructure will also create a log of changes and who made them so that there is accountability.

- An infrastructure that will allow the users of the system to create and add papers to the database mentioned above as well as being able to edit and view them later.

- A mobile version of the above previously mentioned infrastructure must also exist.

6.1.2 Quality requirements

In order of importance our quality requirements are:

- **Reliability:** This is highly important as this might be the only place research could be backed up to, therefore, the system needs to be reliable. Reliability in this system is measured against the effectiveness of the backup measures in place as well as the system be available on demand. This will be achieved by the an automatic partial dump backup occurring daily in the early hours of the morning and a full dump once every 3 days.
  Strategy used:
  Maintain Backup, Checkpoint Rollback
  Design Patterns:
  Memento design pattern may be used to roll back database to previous state in the event of some type of failure. Visitor Pattern will be used for creating the partial and full backup dumps.

- **Security:** The system needs to be secure from anyone who does not (or should not) have permission to use, therefore, passwords will be secured using the SHA512 hashing algorithm.
  Strategy used:
  Limit access strategy, by using Authentication and Authorization strategies.

Design Patterns:
Authenticator and Authorization Enforcer Patterns used for managing and delegating the authorizing and authenticating processes of users.

- **Audibility:** This is measured in terms of is every change noted and logged somewhere. This will be done through means of recording, when a change is made, the system will log the change as well as the user name that submitted the change, therefore everyone will be held accountable.
  Strategy used:
  No specific strategy tactic will be used for this
  Design Patterns:
  No specific design pattern will be used for this

- **Scalability:** There will not be a huge amount of users making use of the system but it is important that if need be every user can make use of the system concurrently. This will be achieved by implementing the system in such a way that it is multi-threaded. In terms of two users trying work on the same documents meta-data simultaneously, when a user tries to upload an edited version of the documents meta-data, if the version on the system differs from the original document then the user will be alerted that there is a synchronization error and he should update the local version of the document and add the changes. This will avoid data corruption.
  Strategy used:
  Thread pooling and Use Concurrencies strategies
  Design Patterns:
  No specific design pattern will be used for this

- **Maintainability:** Measured against how effectively the system choses to relocate data due to age. When data has been on the system for a predetermined length of time, it will be moved from the main system and relocated to a different (most likely slower) database and if a user wishes to make use of it again it can be called from there.
  Strategy used:
  No specific strategy tactic will be used
  Design Patterns:
  Visitor design pattern will be used for relocating data to a different database

6.1.3 Integration and access channel requirements

The Integration and access channel requirements are:

- **Centre of integration:** This will be the main Server, probably an HTTP/HTTPS Server that will be feeding data to other sub-parts of the system. The main Server will be directly connected to all the

13

Databases. It will be the source of data. To prevent anyone one to connect to the Server password will have to be provided, checks will be done to check if the sub-part of the System is authorized, this could be anyone from outside who will be trying to connect to the server, e.g the Android Application, Client as (Website) and windows application.

- **sub-parts of the system:** This will be anyone who will be trying to gain access to the Server. The sub-part of the system will have to prove to the server that it is authorized to gain access to the information held by the server. Login details and passwords will be provided to the server to allow access.

- **Protocols to be used(Access channels):** HTTP/HTTPS will be used, the sub-parts of the system will have to go through the HTTP/HTTPS to gain access to the server. If the subsystem fails to prove to the Server through HTTPS/HTTP it will not be allowed access to the Server's data.

- **Gaining of data from Server:** The server will contain an API that will allow the sub-parts of the system to gain access of the data through function calls, and authentication will be taking during the communication of the Server and the client(sub-part of the system). To make the security reliable, authentication will have to place regularly.

6.1.4 Architectural constraints

6.2 Architectural patterns or styles

6.3 Architectural tactics or strategies

Performance   To better manage resource intensive tasks thread pooling will be used. There will be an object that receives all tasks and allocates them to a specified thread when a thread becomes available.

Exception handling   Exceptions should be dealt with as soon as and as best as possible, but since it cannot be afforded that the system crashes there should be some form of "last resort" that attempts to deal with unhandled exceptions in such a way that the system as a whole keep functioning.

Passive Redundancy   There will be 2 databases that will both contain all the information. This will increase the reliability of the system by having a back up database that can take over at any point when the primary database fails. This will be transparent to the user.

Transactions Changes to the database will be done in transactions that will allow any changes to be rolled back to s checkpoint should an error occur. This will cause atomicity in the system and increase reliability.

Security Encryption to increase security HTTPS protocols will be used to transmit data and all private information (like passwords) will be hashed using the SHA512 hashing algorithm.

Integrability In order to increase integrability a modular approach will be taken when implementing the system by using dependency injection as far as possible.

6.4   Use of reference architectures and frameworks

- Bootstrap for styling the Web interface of the system. Will be used for alignment of the elements of the web interface, any alignment for tables (if any) and responsive design so that the web interface can be viewed on mobile devices.

- AngularJS will be used for the front end of the website along with bootstrap by using UI Bootstrap. For example will be used for DOM manipulation and data-binding can be used to send data to the server. Most notable and obvious use is when a new user creates an account, AngularJS will handle and send the data entered by the user to the server.

- PhoneGap (renamed to Apache Cordova, we will use PhoneGap in this document) will be used as a wrapper for the above website in mobile form. Meaning the web interface will be used and PhoneGap will create a hybrid Android app that communicates with the server and only acts as an interface to enter data.

- Ionic framework will be used together with AngularJS and PhoneGap and replace Bootstrap for the hybrid android app. ionic makes better and responsive apps when used with PhoneGap and requires angularJs. The reason it will replace Bootstrap for the mobile app is because Bootstrap and Ionic do not play well because of name conflicts.

- Spring framework will be mainly used for dependancy injection on the server side of the system. The MVC nature of Spring will also be of good use for the needs of the system. Spring is also some kind of wrapper for Java EE.

- Java EE can be seen as a reference architecture that supports scalability, MVC and dependancy injection. All are needed fo be implemented in the CS Research System.

6.5   Access and integration channels

The different access channels that will enable users to use the system are a web interface, which can be accessed from either a computer or mobile phone as well as a mobile app, preferably an Android app.

The integration channel through which our system will integrate with external systems is a database, which will be able to store data about the users as well as the details of the papers that the users contribute to. The database organises large amounts of data, that is then easier to access, manage and update. An API will also be used so that users from different universities can connect to the system and contribute to any project that they are working on.

The protocol that will be used is the Hypertext Transfer Protocol Secure (HTTPS), because it enables for data to be sent securely through to the server and also because the privacy and integrity of the data that is sent between the server and the system is protected, which is regarded as important as the content of the system should be safe from unauthorized users.

For the integration channel, the technology that will be used for the database is MySql, a database used on the web and that most people are familiar with. It is also easy to use, reliable, fast and it can cater for large applications.

6.6   Technologies

Technologies used will be MySQL for the database , it is very light weight and will have sufficient access control for the given product. The server will be a mix of PHP and Java, both will connect to MySQL , the Java will be the host for the android connection and the PHP will handle requests from the web interface.

7   Open Issues

There are currently no open issues.