

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

DEPARTMENT OF COMPUTER SCIENCE

COS 301 - SOFTWARE ENGINEERING

Software Requirements Specification

Authors:

Muller Potgieter

Sara Masilela

Lethabo Mogase

Stuart Andrews

Stephen Swanepoel

Renton McIntyre

Sibusiso Masemola

Student number:

u12003672

u10126202

u14103207

u12153983

u11032091

u14312710

u12270467

February 24, 2016

Contents

1	Introduction	1
2	Vision	1
3	Background	1
4	Architecture Requirements	2
4.1	Access Channel Requirements	2
4.2	Quality Requirements	2
4.3	Integration Requirements	4
4.4	Architecture Constrains	6
4.4.1	Since the system will be web-based, technologies that guide in building web pages will be needed, i.e.:	6
4.4.2	Architecture Patterns / Framework	6
5	Functional Requirements and application design	7
5.1	Use Case Prioritization	7
5.2	Use case/Services contracts	7
5.3	Functionality Requirements	13
5.3.1	Server Functional Requirements	13
5.3.2	Connection	13
5.3.3	Database Connection	13
5.3.4	Send paper to browser client	14
5.3.5	Get paper from browser client	14
5.3.6	Send Author list to browser client	15
5.3.7	User Identification	15
5.3.8	Process Specifications	17
5.3.9	Browser Client Functional requirements	17
5.3.10	Log in to server	17
5.3.11	Locate paper data	17
5.3.12	Manipulate paper	18
5.3.13	View author list	18
5.4	Domain Model	19

1 Introduction

This section deals with the software software requirements specification of the program. This includes:

- The vision.
- The background.
- The access channel requirements.
- The quality requirements.
- The integration requirements.
- The architecture requirements.
- Use cases.
- Required functionality.

2 Vision

The client wishes to create a program that allows users to keep track of their own work, as well as collaborate with other users, so that they can write papers together. Users will be able to specify the progress that they have made with their papers and alter them as needed. The program will keep a full record of all changes made to the papers, in order to create a time line of events. The program will be available as a desktop and mobile application, as well as being available as a web version.

3 Background

1. A specialized program, aimed at research papers does not exist that is not indigenous to the UP CS department
2. It can be used as a common platform for researchers around the world to easily manage their work and collaborate more easily
3. It will allow researchers easier access to their work and provides access to domain objects
4. It will allow researchers to keep track of their progress more easily
5. The program can be expanded to include researchers from other universities or scientific bodies

4 Architecture Requirements

4.1 Access Channel Requirements

1. A desktop application available in the form of a Windows (7/8/10) client, a Linux client and binaries ready to be built on either system with an interactive GUI
2. A web version compatible with all major browsers (eg. Mozilla Firefox, Google Chrome or Opera)
3. A mobile application developed for Android and compatible with all current and upcoming versions thereof

This will be accomplished by making use of RESTful web services (these being based on the REST, or REpresentational State Transfer architecture). The system itself will accept HTTP requests from any of these channels and create responses in the form of JSON strings, a format easily handled in any one of the aforementioned access systems. Additionally, the following access channels can be added:

1. A command line (terminal) based version of the desktop application, which could be suitable for the target audience, who are very technologically capable ready to be built on either system with an interactive GUI
2. A mobile application developed for Windows Phone and/or iOS.

4.2 Quality Requirements

The following assurances must be made in terms of quality:

- Performance
 1. The server must always provide the minimum data required to fulfill a request. That is to say, were a user to log in to the system, the server should only send the data pertaining to that user to be displayed, no papers related to his/her co-authors or other related parties.
 2. The system must be created with the most minimal and efficient coding practices possible, given that the result must still be reliable and robust.
 3. No actual files are to be stored in the system, lest it negatively affect the performance components of the system itself.
- Reliability
 1. The system must be thoroughly tested on both the client and server side, to ensure it will not cause faults or problems. It is important that no data is lost, thus the coding used to create the system must be defensive and thorough.

- Scalability

1. The system must be designed such that:
 - a - The client is able to handle and display details of a large, potentially infinite number of Publications.
 - b - The server is able to handle, display details pertaining to large, potentially infinite number of Users, Authors and Publications.
2. Modular programming should be used in order to ensure that there are no restrictions in terms of the system's ability to be extended and improved upon at later stages.

- Security

1. It should not be possible for individuals other than the actual Users to access or modify the system. This means that security has to be ensured in terms of password storage, secure login methods and user management (methods such as re-obtaining password via email should be very carefully guarded).
2. It should not be possible for Users to make changes to other Users' details, as it is with non-User Authors, unless they are one of a select few Super Users or Administrators.
3. A publication should not be able to be removed from a system, only edited, unless it is removed by an aforementioned Super User.
4. A User should not be capable of viewing or editing a publication for which they are not on the list of Authors.

- Flexibility

1. The system should be capable of reacting quickly to different stimuli. This means (as an example) that if multiple users are concurrently using the system and performing vastly differing tasks which make use of completely different parts of the same system, there should not be any noticeable loss of performance.
2. The system should be able to perform well even under bulk loads, without loss of data on the way.
3. It should be possible to add new components or fields to existing components in the system without making major changes.

- Maintainability

1. The system should be developed with current and maintained technologies, so as to avoid loss of support for as long as possible.
2. The system should be well documented so as to ensure future developers on the system are capable of maintaining the system without worry.
3. When changes are made in current technologies, the system should be updated as soon as possible to reflect relevant changes.
4. The modular design of the system must be such that if changes must be made to a part of the system, only that part itself should be changed.

- Monitorability
 1. All actions taken that have any affect on the databases stored server side are to be logged.
 2. All logs, current connections and current activity must be viewable by the Super Users in charge of the system.
- Integrability
 1. The system should be designed in such a manner (with modularity and common interfacing methods) that it is capable of having pieces or services plugged in and catered to with minimal effort, such as e-mail notification, which could be a logical future addition to the system for the sake of deadline maintenance.
- Cost
 1. The tools used to design the system should, as far as possible be open source, free and not require a license.
 2. In certain cases, paid and licensed software may be suitable for some individual pieces of the system, such as having a Database Management System (DBMS) to handle the storage of data as best possible.
 3. Costs may be created in the form of external hosting for the web service and database storage, should the client desire it to be so.
- Usability
 1. The Users, being staff members, must have easy access from any channel.
 2. The system should be designed in such a manner that the interface is easy to learn and use.
 3. The system should be minimal and avoid having unnecessary visuals that could impair a User's ability to use the system comfortably.

4.3 Integration Requirements

Given that the system in question is intended to use external systems to as minimal a degree as is possible, this section will instead be dedicated to the explanation behind why focusing on Integration Requirements is in fact a detrimental procedure to this project plan.

1. Firstly, it should be noted that this is a private system. This means that there is little need for a connection with any system which provides linkage to a major network of users.
2. The system itself is simplistic and requires no extra functionality provided by more complex external systems, as it is capable of functioning aptly with simplistic HTTP requests and a simple Object Data Type for communication. Although this

could be stretched to CORBA at a later stage and thus require some management of this system, as the system currently stands, it is sufficient to use basic JSON strings and work with these.

3. The system is personal to the degree that one may not view more than one's own profile. This implies that it would be redundant to make use of extra technologies, in a system which at its core is just a very small database management tool, with access only to one's own part of a database.

Regarding the future, it is important to note that the system is intended to be modular and adaptable, thus it could (possibly) be integrated into an external, or vice versa. The focus of such an endeavor would be on ensuring this does not compromise the system in any way. To guard against this, we must ensure the following:

1. The system is not dramatically affected by the changes made, in terms of performance. That is to say, given a new, external system being integrated into the existing system (such as e-mail notification), there should not be any noticeable performance drops that would hinder usage, as this would be detrimental to the system and possibly be a good reason to not perform such integrations in the first place.
2. The system is not to be compromised in terms of security. An example of this would be if an external system attempted to send or receive secure data via insecure means (eg. requiring raw, unencrypted data that should be sensitive).
3. It should still be possible to accurately monitor all external systems that are integrated with the original system.
4. The external system used should not be relied on too heavily by the original system, unless adequate reliability and safety redundancies can be assured.
5. The external system should not compromise the ability of the original system to be used in a scalable manner (that is to say, it should not make it less efficient or plausible to use the system for larger volumes of data or users).

With this in mind, the final note on this topic is as follows:

1. The protocols and systems used along with the original system that is to be created should never be overly complex or dramatic.
2. The system should, as a whole, remain independent and capable of being used in a modular and free manner.
3. An over reliance on external systems should be avoided and the integration between systems must be fluid and as loosely coupled as possible.

4.4 Architecture Constrains

4.4.1 Since the system will be web-based, technologies that guide in building web pages will be needed, i.e.:

- HTML 5
This is the overall language that will be used to develop the layout and functionality of the web page
- Bootstrap
This technology will be used to style the web page. Styling with this technology will reduce the work load giving us more time to work on the functionality of the system.
- Apache
We will use this as reference to how the final product will look, since we wont have access to the clients server until the system is completed.
- jQuery and Java Script
These two languages will be used for client side validation, i.e. validation of user credentials upon logging in to the system.
- PHP
PHP will be used for validation on the server side.
- SQL
This technology will be used to enter information to the database. Entries will be stored separately: Users, Authors and Co-Authors

The system will also have an Android application version. The technologies that will be used are:

- Android Studios
This will be the main technology used to build the main functionality of the system.
- Programming languages like Java, C++ and C will be used to assist in building the Android version of the system

4.4.2 Architecture Patterns / Framework

The application will make use of a four-tier layered pattern, which consists of:

1. The Presentation layer
This layer consists of an interface through which the users can access the application layer. This layer also captures the user's input, validates it and passes the information to the application layer.

2. The Application layer
This layer provides the back-end services of the system, i.e. the functionality of the system. Access to the web services layer is managed in this layer. The application layer may also be used as temporary storage when the database cannot be accessed.
3. The Web Services layer
Here the information that was processed in the application layer is received and passed down to the database. Access to the database will be read and write. This layer is where the sever is situated.
4. The Data layer
This layer is the actual database. The information is added to the database in this layer.

5 Functional Requirements and application design

5.1 Use Case Prioritization

1. User Login – Critical.
2. Author Login – Important.
3. Super-user Login – Critical.
4. User registration – Critical.
5. Author registration – Important.
6. Super-user Registration – Critical.
7. Creating a User – Critical.
8. Creating an Author – Important.
9. Creating a new publication – Critical.
10. Editing a publication – Nice-to-Have
11. Setting the status of a publication – Important.
12. Viewing Publications as an Author – Nice-to-Have.

5.2 Use case/Services contracts

1. User Login
 - (a) Pre-Conditions
 - i. A user must be registered as a user by admin before he/she is able to login to the Research Paper App.

- ii. In order to login a user must enter in his/her correct authentication details.
 - (b) Post-Conditions
 - i. The user has access to his/her profile and publications.
 - ii. The user may alter his/her publications.
 - iii. The user may access only his/her profile and publications and no others.
 - iv. A user can be an author.
2. Author Login
- (a) Pre-Conditions
 - i. An author must be registered by admin as an author before he/she may login to the Research Paper App.
 - ii. In order for an author to login he/she must enter in his/her correct authentication details
 - (b) Post-Conditions
 - i. The author has access to any profile that he/she co-authored.
 - ii. The author may not alter any publications that he/she was involved in.
 - iii. A user can be an author, but an author cannot be a user.
3. Super-user/admin Login
- (a) Pre-Conditions
 - i. A single user must be able to logon as admin or a super user.
 - ii. In order to login a user must enter in his/her correct authentication details.
 - (b) Post-Conditions
 - i. The super-user has access to any and all user and author profiles.
 - ii. The super-user is the only user capable of adding more users and authors.
 - iii. The super-user can alter any profile.
4. User Registration
- (a) Pre-Conditions
 - i. The admin or super-user is in charge of registering the users.
 - (b) Post-Conditions
 - i. The user receives his/her login details.
 - ii. The user has his/her privileges set.
 - iii. The user is registered in the user and author database tables.
 - iv. The user is able to logon to the Research Paper App with the login details supplied by the super-user/admin.

5. Author Registration

(a) Pre-Conditions

- i. The admin or super-user is in charge of registering the authors.

(b) Post-Conditions

- i. The author receives his/her login details.
- ii. The author has his/her privileges set.
- iii. The author is registered in the author database table only.
- iv. The author is able to logon to the Research Paper App, as an author, with the login details supplied by admin or super-user.

6. Super-user/Admin Registration

(a) Pre-Conditions

- i. Upon system-initialization, a single user sets him/herself to the super-user/admin.
- ii. This user uses his/her authentication details to log in as the super-user/admin.

(b) Post-Conditions

- i. The super-user/admin can add and remove users
- ii. The super-user/admin can view all user and author profiles, as well as lists of publications associated with each.

7. Creating a User

(a) Pre-Conditions

- i. Prior to a user logging into his/her profile, the super-user/admin must have created the user profile, which the user logs onto.
- ii. Upon logging in, a user must have full access to his/her profile page.
- iii. Profile page must include
 - A. Full name of user.
 - B. Contact details
 - C. Cell phone number.
 - D. Telephone number.
 - E. Email Address.
 - F. Conference for whom the user is researching.
 - G. List of links to publications.
 - H. A list of co-authors per publication (if any).

(b) Post-Conditions

- i. The user must be able to edit his/her publication list as he/she sees fit.
 - A. Adding Publications.

B. Removing Publications.

- ii. The user may only view/edit his/her own profile and publications.

8. Creating an Author: Use Case Prioritization – Important

(a) Pre-Conditions

- i. Prior to the author logging into his/her profile the super-user/admin must assign him/her an author profile.
- ii. Upon logging in, the author must have full access to his/her profile.

(b) Post-Conditions

- i. Authors may not alter publications.
- ii. Displayed on their profile will be:
 - A. Full name of the author.
 - B. Contact details
 - C. Cell phone number.
 - D. Telephone number.
 - E. Email Address.
- iii. Authors will be able to see each of the publications that they co-authored.
 - A. No altering will be allowed.

9. Creating a new publication:

(a) Pre-Conditions

- i. The user or super-user must provide a publication title.
- ii. The supervisor of the paper must be included.
- iii. All the authors who worked under that supervisor, to co-author the paper, must be listed.
- iv. A deadline must be set.
- v. Progress of the paper must be specified.
 - A. Ongoing.
 - B. Terminated.
 - C. Completed.

(b) Post-Conditions

- i. The new publication will be viewable by the user, the super-user and all authors involved.
- ii. The publication may only be edited by the user who created the publication, or the super-user.

10. Editing a publication

(a) Pre-Conditions

- i. User or super-user must first successfully log on.
 - ii. Only the user who is the supervisor or owner of the paper may edit the publication.
 - An author who happens to be a user as well as an author may not edit another users publication.
 - iii. The super-user/admin may edit any profiles publication he/she wants.
- (b) Post-Conditions
- i. The user or the super-user/admin can alter the following:
 - A. The Author list,
 - B. The status of the paper,
 - C. The deadline of the paper,
 - D. The name of the paper,
 - ii. Or remove the publication all together.

11. Setting the status of a publication

- (a) Pre-Conditions
- i. If a user has successfully logged on, then he/she may view and alter the publications.
- (b) Post-Conditions
- i. Depending upon the status of the paper the user may alter it.
 - ii. Only a user, whose privileges allow it, may edit the publication.

12. Viewing Publications As an Author

- (a) Pre-Conditions
- i. Upon logging in, the author must have full access to his/her profile.
 - ii. Profile page must include
 - A. Full name of user.
 - B. Contact details
 - C. Cell phone number.
 - D. Telephone number.
 - E. Email Address.
 - F. Supervisor who the author is researching under.
 - G. List of links to publications for which he/she has co-authored.
- (b) Post-Conditions
- i. All publications that the author has co-authored must be available to view.
 - ii. Unless the author is also a user, he/she will be prohibited from altering any data

- Request and Results Data Structures

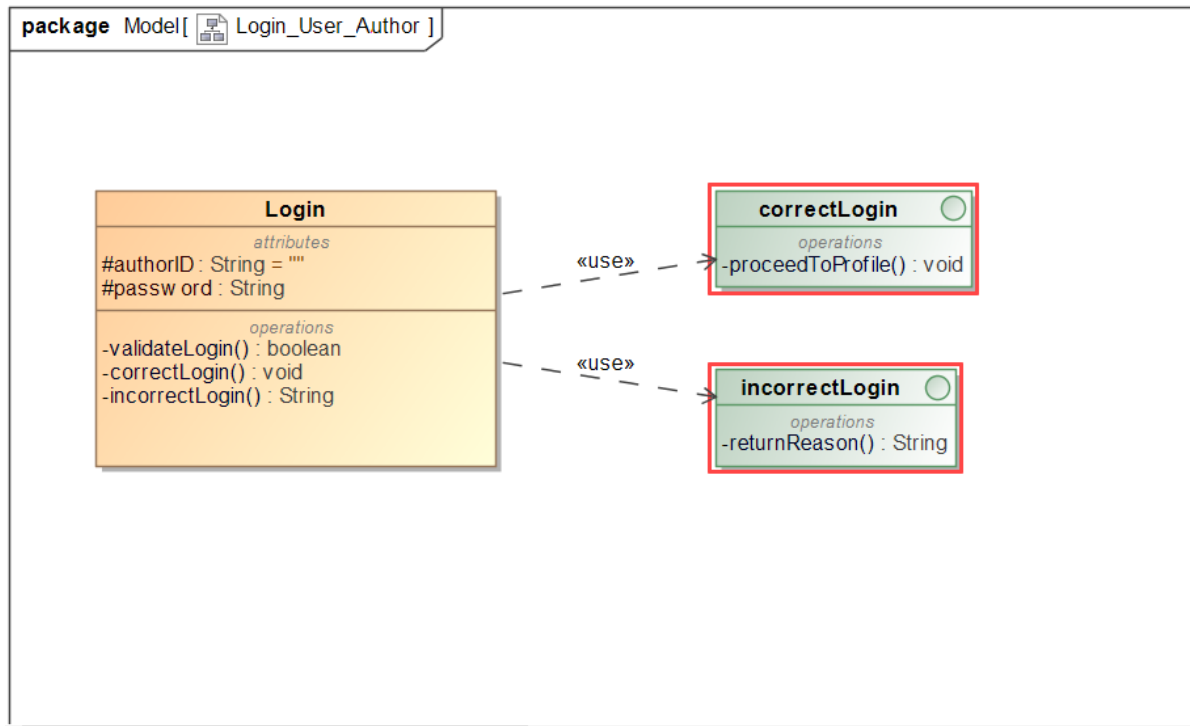


Figure 5.2.1: Login for Author and User.

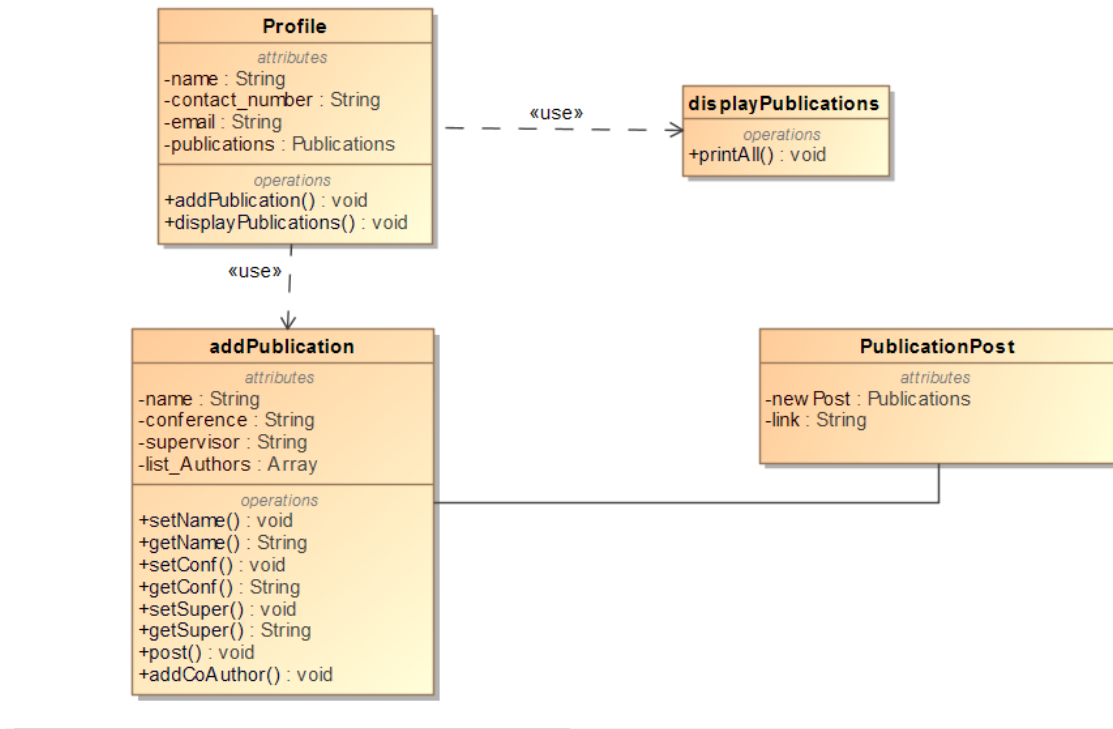


Figure 5.2.2: User posting publication.

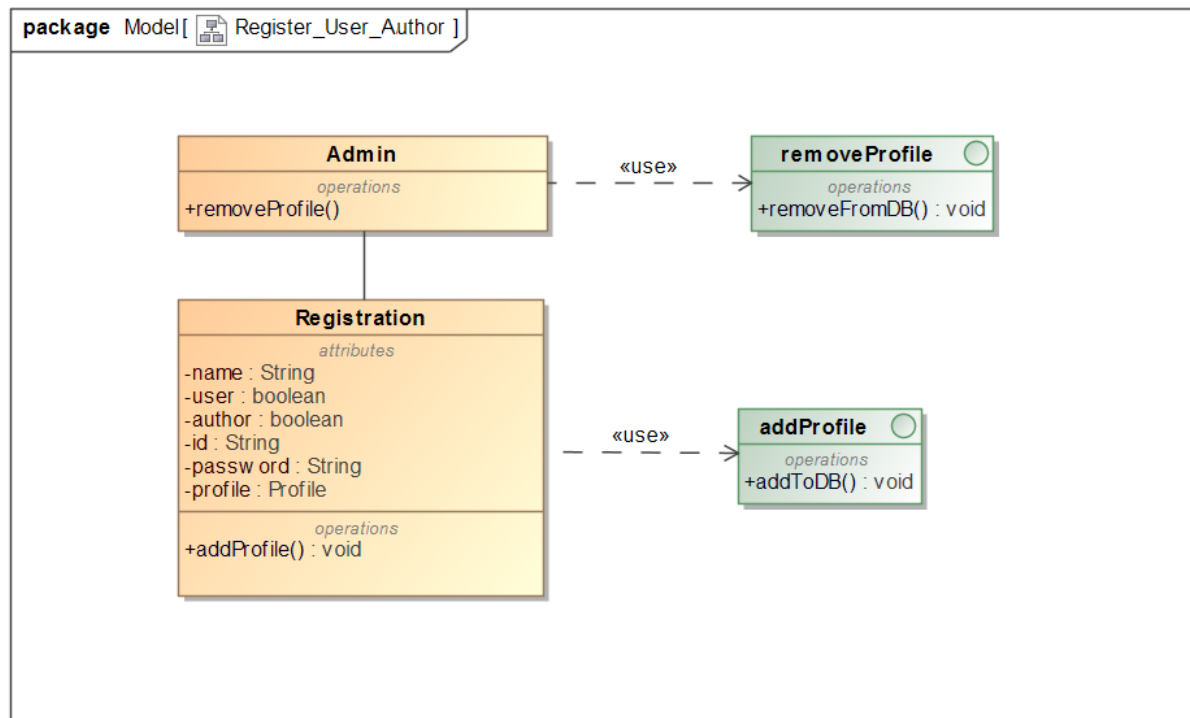


Figure 5.2.3: Registration

5.3 Functionality Requirements

5.3.1 Server Functional Requirements

5.3.2 Connection

This connection provides communication between client, server and the relational database, achieving this through a protocol that allows multiple access (100) to the server, at one time.

- A client is able to log in a request
- A client is able to send a request which goes through the server to the database.
- Conversion of request to a database query is performed.

5.3.3 Database Connection

Server must be connected to database, every time it is in use. No information is left on server and not recorded on database.

- Database is accessible any time by server, on client request.

5.3.4 Send paper to browser client

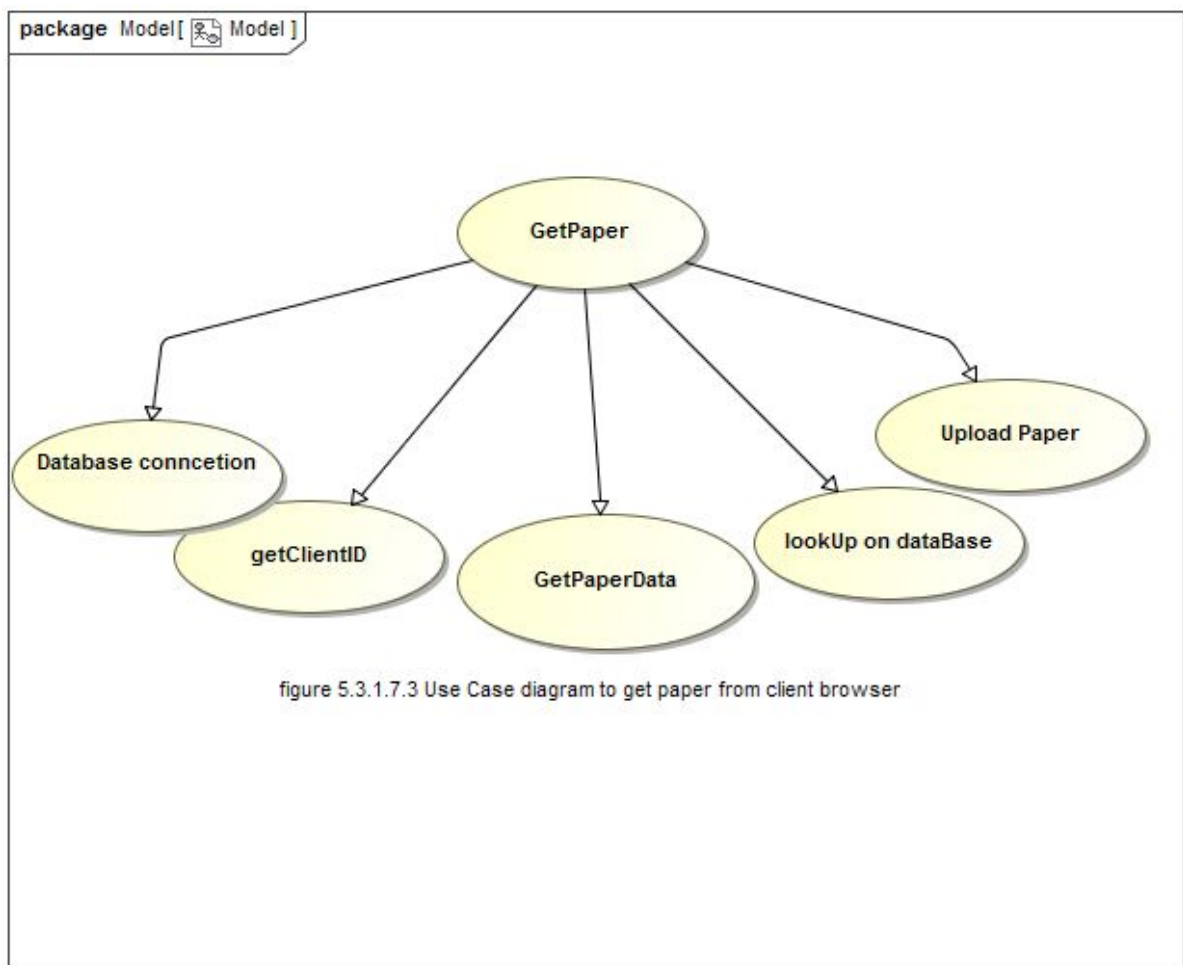
Sending paper to browser, provides a requested paper data to the web browser. It is a step by step process

- Perform validation on whether the paper exists in the database.
- Retrieves the paper from the database, to server, to web browser

5.3.5 Get paper from browser client

This process provides receiving paper data from the browser client, This is a result of pushing the paper data to the database after it has been edited.

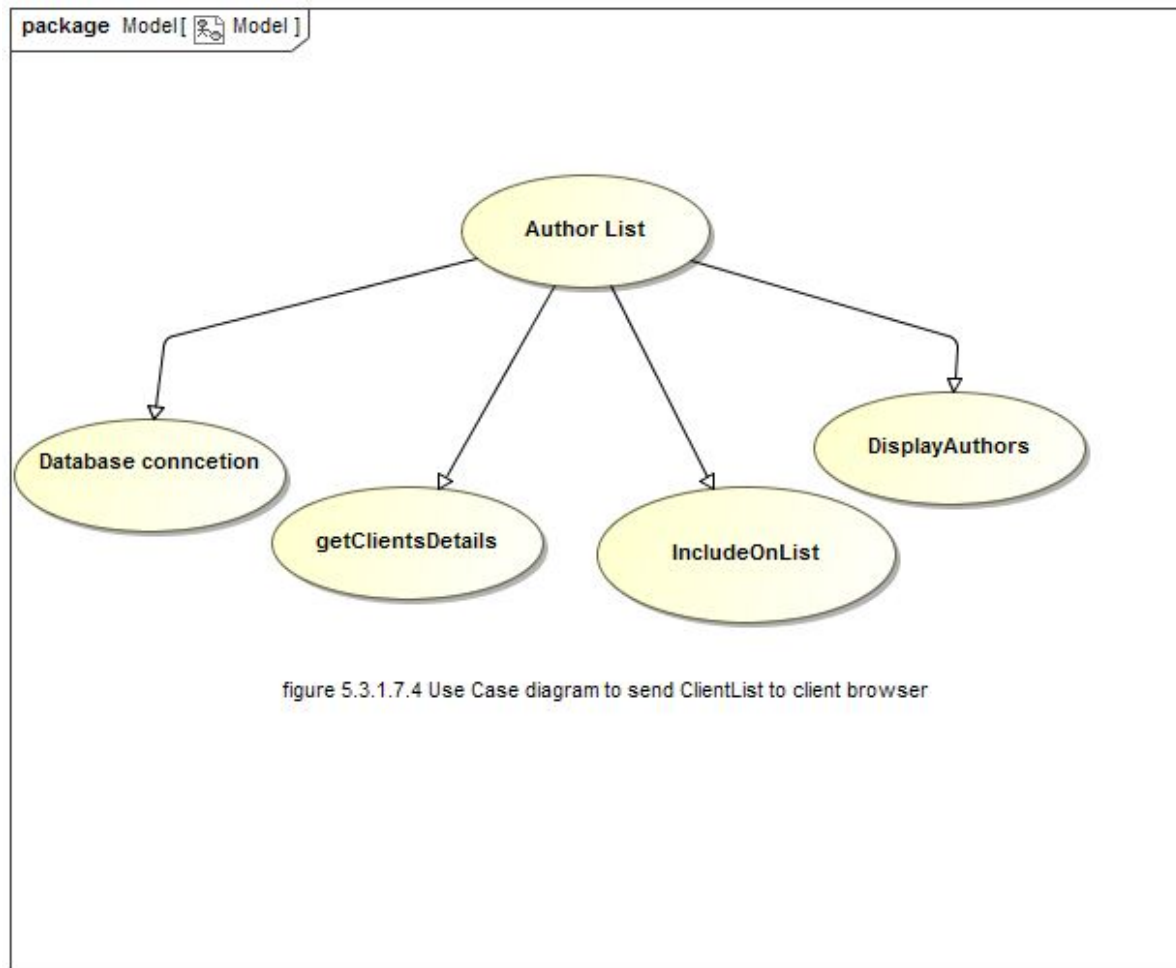
- Perform validation on paper if it is recent.(date modified).
- Replace paper data on the database, with the new one.



5.3.6 Send Author list to browser client

Providing the list of authors in the system. Results in a navigable list.

- Get all authors from the database, to the server, to the client browser.
- Allow viewing to navigate through authors.

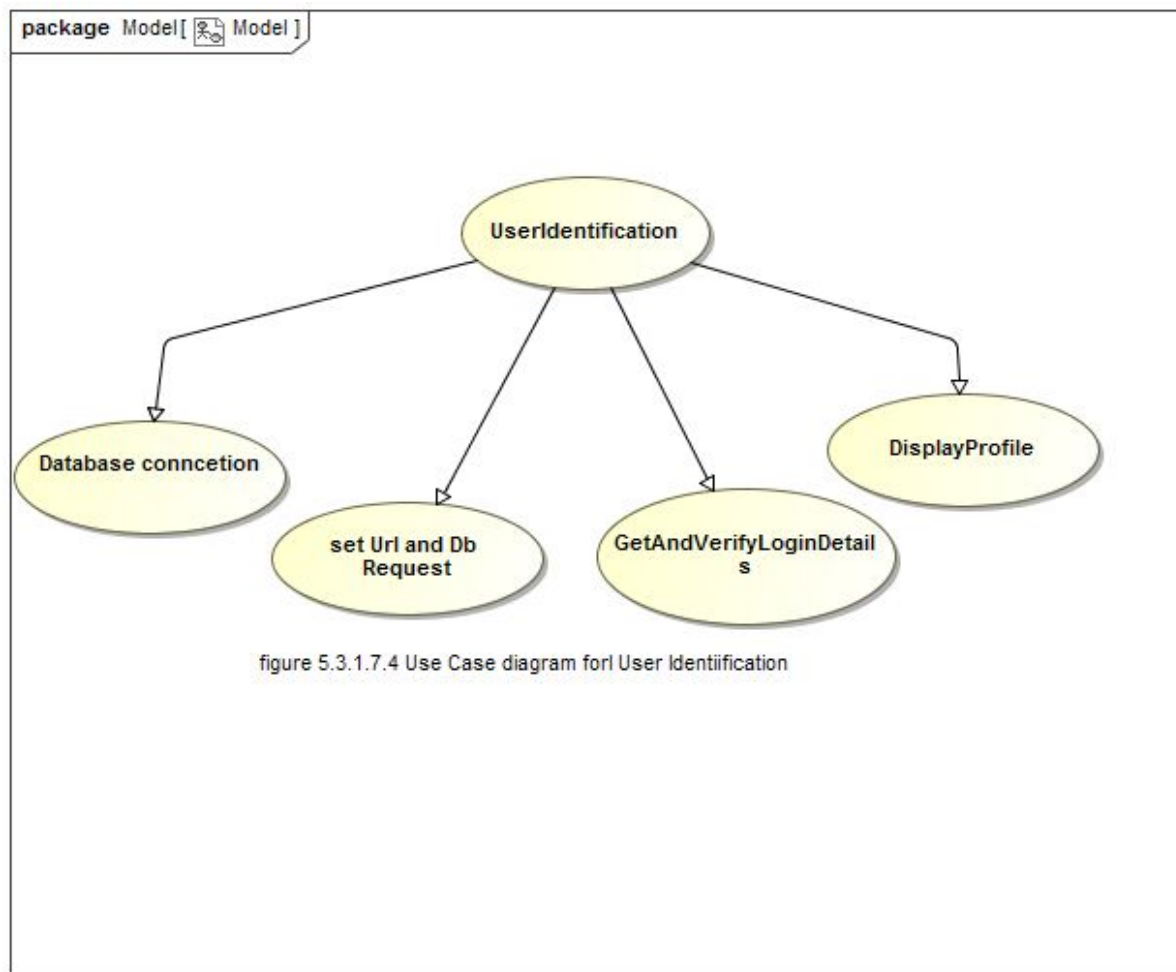


5.3.1.7 Use case diagrams

5.3.7 User Identification

User Identification provides the means to get client (user) credentials and log them in the system to their appropriate profile.

- User name of the client(client browser)
- User password of the client (client browser)
- Validate with users in database.



5.3.8 Process Specifications

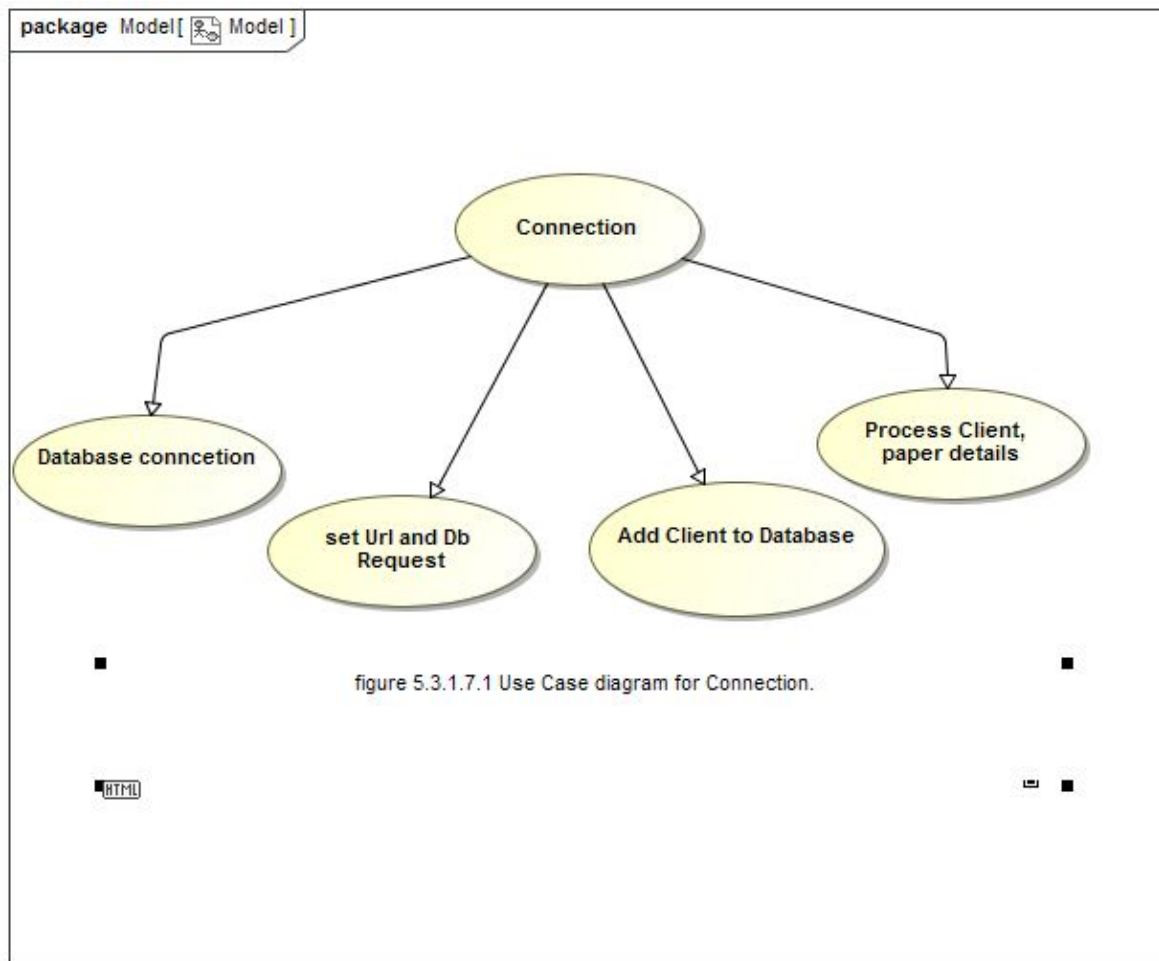


Figure 5.2.3: Registration

5.3.9 Browser Client Functional requirements

5.3.10 Log in to server

Log in allows user (client browser) to log in to the system, resulting in their profile information and current work loaded on the browser.

- Enter their username
- Enter their password

5.3.11 Locate paper data

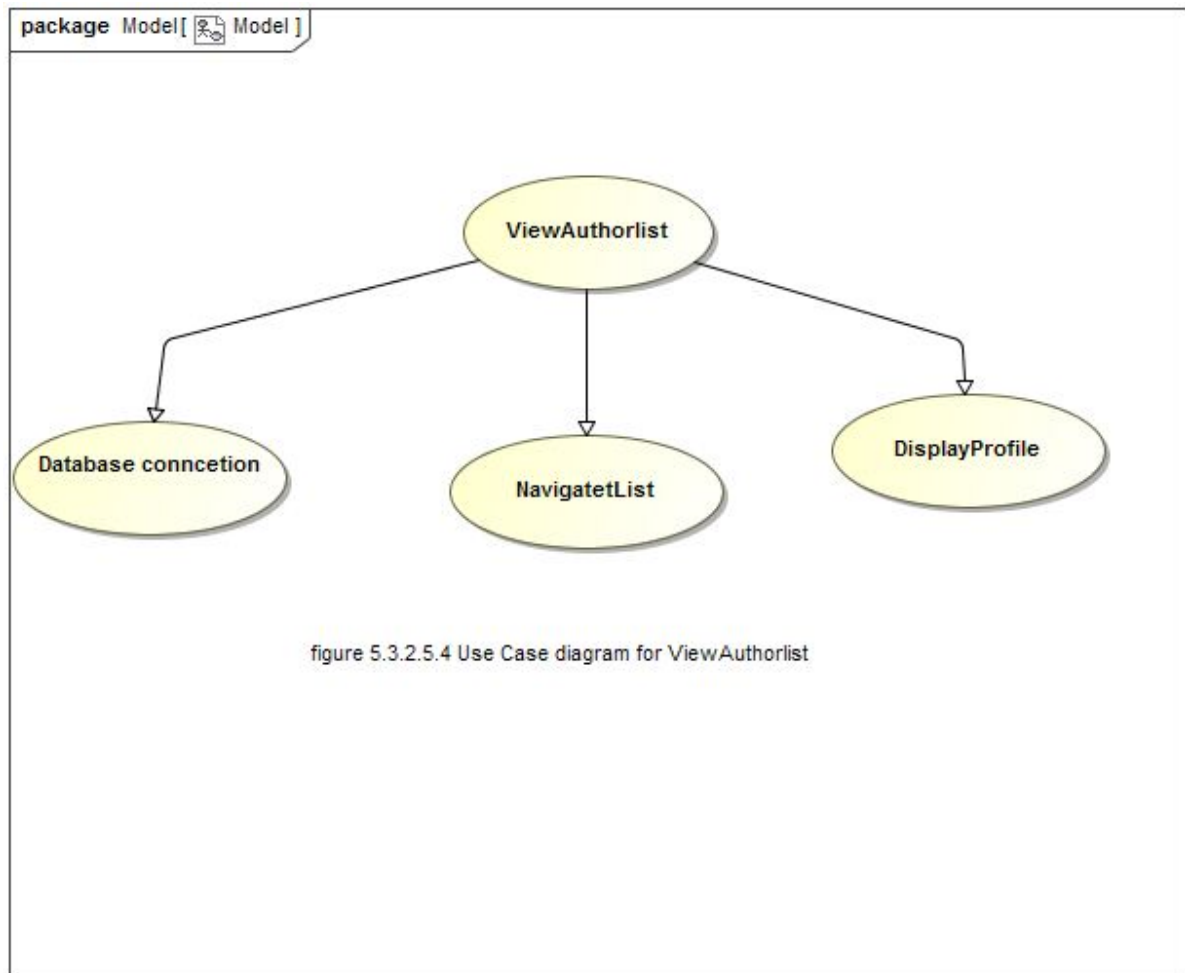
Loads paper upon selection/request from the system, as the user (client) may be involved in multiple papers. Upon request the server will load correct paper data.

5.3.12 Manipulate paper

This is to provide user (Client browser) to be able to manipulate their current work

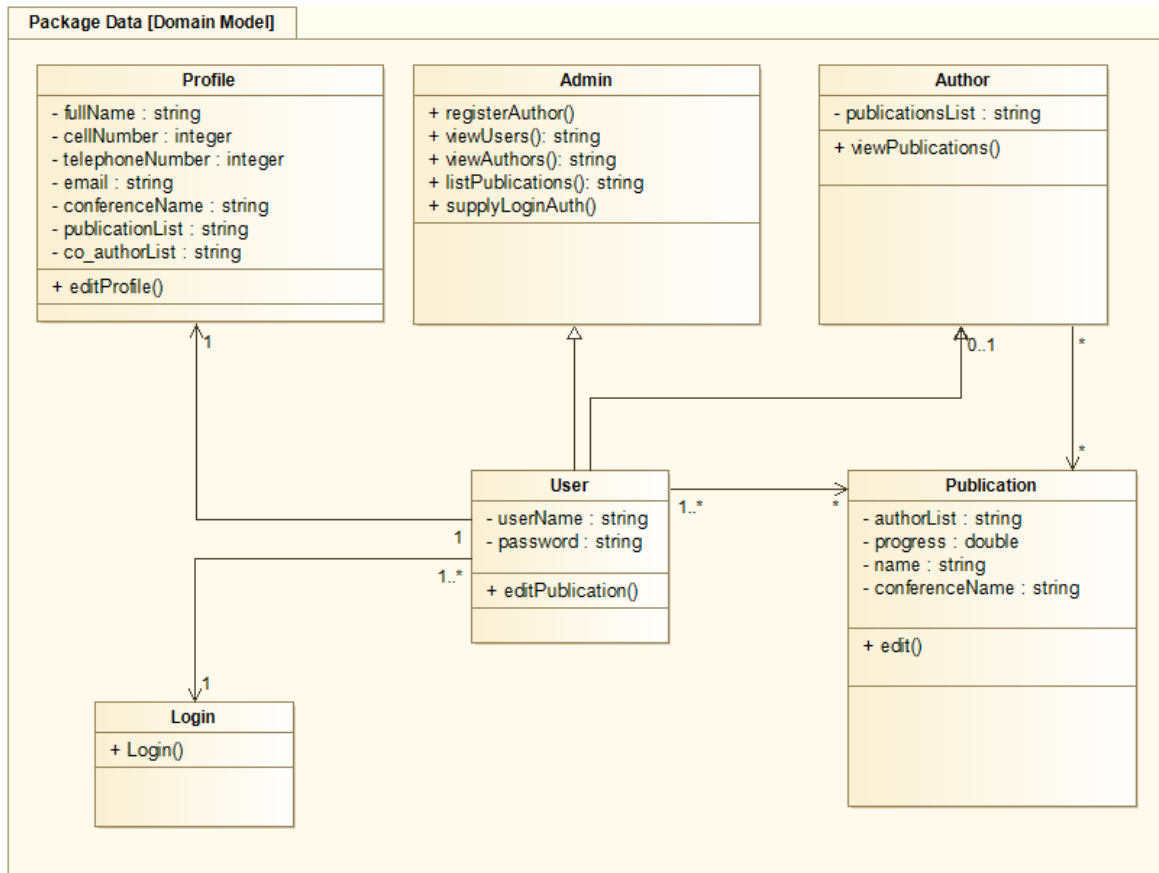
5.3.13 View author list

This is to provide user (Client browser) with the list of authors from the server.



5.3.2.5 Use Case diagram

5.4 Domain Model



6 Open Issues

- User name of the client(Client browser)
- User password of the Client (Client browser)
- Validate with users in database.