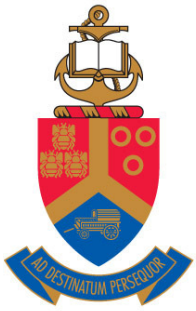


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS 301

Mini Project

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

Team Bravo

Student:	*Student number:*
Daniel King	u13307607
Azhar Mohungoo	u12239799
Andreas du Preez	u12207871
Banele Nxumalo	u12201911
Frederic Ehlers	u11061112
Diana Obo	u13134885
Bilal Muhammad	u13080335

Contents

1 Introduction **3**

2 Vision **3**

3 Background **3**

4 Architecture requirements **4**

4.1 Architecture requirements 4

4.2 Access channel requirements 4

4.2.1 Web Application 4

4.2.2 Mobile Application 4

4.2.3 Architectural scope 4

4.3 Quality requirements 5

4.4 Integration and access channel requirements 5

4.4.1 Integration channels 5

4.4.2 Protocols 5

4.4.3 Integration quality requirements 6

4.5 Architecture constraints 6

4.6 Architectural patterns or styles 7

4.6.1 MVC architecture 7

4.6.2 Three-tier architecture 7

4.7 Architectural tactics or strategies 8

4.8 Use of reference architectures and frameworks 9

4.8.1 Java Persistence API 9

4.8.2 ASP.NET MVC 10

4.9 Access and integration channels 10

4.10 Technologies 10

5 Functional requirements and application design **11**

5.1 Use case prioritization 11

5.1.1 Critical: 11

5.1.2 Important: 12

5.1.3 Nice-To-Have: 12

5.2 Use case/Services contracts 13
5.2.1 Login user 13
5.2.2 Register User 14
5.2.3 View paper 15
5.2.4 Modify paper 16
5.2.5 Publish paper 17
5.2.6 Terminate paper 18
5.2.7 Edit Account 19
5.2.8 Remove account 20
5.2.9 View user account 21
5.2.10 Search user 22
5.2.11 Assign co-author 23
5.2.12 View recent activities 24
5.2.13 System events 25
5.2.14 Set notification 26
5.2.15 Revive paper 27
5.2.16 Units accumulated from work on papers is reflected on ther user's profile 28
5.2.17 Data representation of units accumulated 29
5.3 Required functionality 30
5.3.1 User Gateaway 30
5.3.2 Research Paper Management 31
5.3.3 User Account Management Services 32
5.3.4 Log Services 33
5.3.5 Notification Services 34
5.3.6 Archival Services 35
5.4 Process specifications 36
5.4.1 Server Connection 36
5.4.2 Database Connection 37
5.4.3 Register User 38
5.4.4 Login 39
5.4.5 Add new User 40
5.4.6 View User Account 41

5.4.7 Remove Account 42
5.5 Domain Model 43

6 Open Issues **44**

References **45**

1 Introduction

This document aims to specify the functional and non-functional requirements of a document archiving system, as specified by Ms Vreda Pieterse of the Computer Science Department.

It will serve as a means of communication between the client and developers as well as providing an elaboration and a clear discription of it's implementation specifications.

2 Vision

We intend to create a system that will allow authors and their co-authors to work on their research papers in an environment that reassures collaborative work, which in turn diminishes the time spent on papers with multiple authors. The following are what we plan to achieve:

- Keep track of research papers.
- View meta-data of research papers.
- Allow multiple authors to collaborate on the same research paper.
- Different levels of authority, i.e. Admin, (Co)Author, User.
- View and edit the details, of a text based profile, of different researchers.
- Implemention as a website and an android application.

3 Background

We live in a world where time is valuable. We would like to do as much as we possibly can in the shortest amount of time. And if that's not possible, we work in teams to ensure that we achieve that goal.

Reseacher papers tend to be fairly lengthy, and if completed by only one author, it could be quite a tedious process. Hence we propose a system which would make the storage and collaboration of research articles and papers effortless by producing an archive system.

4 Architecture requirements

4.1 Architecture requirements

4.2 Access channel requirements

The different access channels for the system will be as follows:

4.2.1 Web Application

The system can be used via a web application that uses RESTful web services and bootstrap technology. The system will be fully supported on the following web browsers:

- Chrome 7.0.517 and up
- Firefox 3.6 and up
- Safari 4 and up
- Edge

Bootstrap will be used so that the web application will also be accessible by mobile web browsers that support HTML5 and JavaScript technology.

4.2.2 Mobile Application

The system will also be accessible via a mobile application and will be operational on the following mobile operating systems:

- Android 4.0 (Ice Cream Sandwich) and up.

4.2.3 Architectural scope

- There needs to be a display infrastructure which will handle how results of processes or data is displayed to the end user.
- There needs to be an infrastructure will can relay the system to the various mediums in will be used from (different browsers, andrioid).
- There needs to be a processing infrastructure that will perform the methods of the system, and store the results in the persistence architecture if needed.
- There needs be a reporting infrastructure to generate and store logs within the persistence infrastructure.
- There needs be a persistence architecture. This architecture will hold data pertaining to all the persistent parts of the system, namely:

  - User data (usernames, passwords, profile data, etc)
  - Group data (primary author, co-authors, paper, etc)
  - Paper data (location, type, published, etc)
  - Logs

4.3 Quality requirements

- Security - The system shall identify all of its client applications before allowing them access to those enities. Possible measurement methods: Success rate in authentication, percentage of successful attacks, encryption level and probability/time/resources needed to attack the system.
- Usability - The degree of ease of use and training needed for end users. Possible measurement methods: Time it took a user to find a report (search functionality), percentage of deadlines met (notification functionality) and time it took a user to perform certain tasks.
- Auditability - The degree to which transactions can be traced. Possible measurement methods: The number and precision of logs generated in a period of time (metadata accessed/deleted/added).
- Portability - Measure ability of the system to run under different computing environments. Possible measurement methods: Number of targeted software environments (ie different browsers and operating systems) and proportion of platform specific functionality.
- Maintainability - Measures ability to make changes quickly and cost effectively. Possible measurement methods: Degree of complexity to make changes to the format of the metadata and mean time to add or change certain functionalities (ie new report types and new types of users).
- Availability - Percentage of time that the system is up and running correctly. Possible measurement methods: Length of time between failures and length of time needed to resume operation after a failure.
- Performance - Possible measurement methods: How well the system perform under high workload and number of events processed/denied in some interval of time.

4.4 Integration and access channel requirements

4.4.1 Integration channels

- System will be integrated on two platforms: a website and an android application.

4.4.2 Protocols

- Protocols needed for the website: HTTPS, Data Query Protocol
- Protocols needed for the application: Session Initiation Protocol (SIP)

4.4.3 Integration quality requirements

- Reliability - The service must not crash, it must run effectively and efficiently allowing users to make use of the service at all times.
- Auditability - Have logs record all activity that occurs thus all actions and occurrences can be traced.
- Security - Giving the users reassurance that the system will protect their details and content. Allow secure and safe communication between user and the system.
- Maintainability - Provide easy, effective and efficient maintenance to the system allowing minimal or no downtime of the system.
- Usability - Allow users to use the system with ease and minimal need for tutorials. Ensure that the system is self-explanatory, efficient and effective.
- Portability - To provide compatibility with as many systems as possible without integration failure or function failure.
- Availability - Minimise downtime of the system. Quick recoveries from crashes, preventive measures to minimise chances of system crash.
- Performance - The rate of work done when the system has either a high or low workload and minimise amount of denied events.
- Seamless transitions between platforms - Implementation between the integrated platforms must not vary greatly to avoid confusion and allow ease of use.

4.5 Architecture constraints

- The platform must exist in 2 mediums, namely web and android application.
- It is an internal network to be used by the Computer Science department, so it need not to source any information from the web.
- Any form of database can be used, but the team has decided that MySQL would be the best option from the current skillset.
- PHP and JavaScript will be used for the web and Android Studio for the android application.
- The system should cater for and including 100 users as a maximum.
- There should be a log that records all activity on the system

4.6 Architectural patterns or styles

4.6.1 MVC architecture

The MVC (Model View Control) architecture allows change to the system's state. The model is completed through a common interface(the control). The view is where developments are made to user interfaces and these changes are independent to the system providing extensibility at a low cost.

Reasons to use MVC:

- It encapsulates the interaction from the user and transforms those interactions, in the form of requests into business logic that interacts with the system mode to produce reponses.
- A multi-layered approach encapsulated by the MVC which enables the separation of business logic providing the need for sub-systems and internal and external APIs.

The MVC architecture will be based on SOA (Service Oriented Architecture) reference architecture. It is a loosely-coupled architecture where services exist independently from each other. They communicate with each other by means of protocols and standard-based interfaces such as SOAP and XML schema to define messages passed between objects. Services and functionality can easily be added to the existing system during progession.

Reasons to use SOA:

- Reuse of objects
- Flexible and agile
- Simplicity of implementation
- System is open to change and easy alteration

4.6.2 Three-tier architecture

The three-tier architecture is a client-server software architecture pattern where the user interface (presentation), functional process logic(business rules), computer data storage and data access are developed and maintained as independent modules, most often on separate platforms.

Reasons to use three-tier architecture:

- In web development, three-tier is often used to refer to websites, commonly electronic commerce websites, which are built using three-tiers:
  - A front-end web server. In web based application, front-end is the content rendered by the browser. The content may be static or generated dynamically.

  - A middle dynamic content processing and generation level application server for example JavaEE, PHP, Python platfrom.
  - A back-end database or data store, comprising both data sets and the database management system software that manages and provides access to the data.

4.7 Architectural tactics or strategies

Availability is best defined by the proportion of time a system is functional and working as required. It can be affected by system errors, infrastructure problems, malicious attacks and system load. Here are some key issues:

- The server on which the database is kept can fail or become unresponsive, causing the entire system to fail. We will have to consider how to design failover support in the system. For example, we can use Network Load Balancing for Web servers to distribute the load and prevent requests being directed to a server that is down. Also, consider using a RAID mechanism to mitigate system failure in the event of a disk failure.
- Denial of service attacks, which prevent authorised users from accessing the system, can interrupt operations if the system cannot handle massive loads in a correct and timely manner, often due to the processing time required, or network configuration and congestion. To minimize interruption from such attacks, identify malicious behaviour, use application instrumentation to expose unintended behaviour, and implement comprehensive data validation. We will consider using the Circuit Breaker or Bulkhead patterns to increase system resiliency.

Conceptual integrity is the consistency and coherence of the overall design. This means that we have to be consistent in our various modules designs as well as coding styles and variable names. A coherent system is one that is easy to maintain because you will know what is consistent in the modules. Things to look at:

- Lack of collaboration and communication between different groups involved in the application lifecycle. We should establish a development process integrated with tools to facilitate process workflow, communication, and collaboration.
- Lack of design and coding standards. A guideline for design and coding standards should be made, and we can incorporate code reviews into our development process to ensure guidelines are followed.

Maintainability is the ability of the system to undergo changes with a degree of ease. These changes could impact components, services, features, and interfaces when adding or changing the applications functionality in order to fix errors, or to meet new business requirements. We will tackle this challenge in the following way:

- This documentation will serve as the basic requirements of the system which will make it easier to upgrade and maintain the system at all times.

- The technologies we will be using like PHP, JavaScript, etc are well known languages in the programming world, making it easy to find programmers that can interpret existing code as well as add on to it.
- Logical errors can be difficult to trace in such huge systems, and can cause unnecessary dependencies on other components. Rather we will try to design components so that they are cohesive by having low coupling and thus reducing dependencies.

Manageability is about how easy it is for the system administrators (tech team) to manage the application. This is important for key factors like monitoring systems, debugging and performance tuning. We will try to tackle this with the following approach:

- System logs should be backed up yearly, making sure that they are safely kept on a machine other than the server.
- We should capture and report sufficient information about errors and state changes in order to enable accurate monitoring, debugging, and management. Also, we should consider creating management packs that administrators can use in their monitoring environments to manage the application.
- We should be able to create snapshots of the system so that in case of system failure we can revert to the most recent system state instead of having lost all previous work.

Performance is an indication of how responsive a system is and how it goes about in executing specific actions in a given time and interval. It can be measured by in terms of latency and throughput. Latency is the time taken to respond to any event. Throughput is the number of events that take place in a given amount of time. The application performance is directly proportional to its scalability. Thus lack of scalability can affect performance. This can be avoided by reducing the likelihood of contention for shared resources. The main factor affecting system performance is the demand for a specific action and the systems response to the demand. Here are some key issues:

4.8 Use of reference architectures and frameworks

As the system will be using the Model View Controller (MVC) architecture ,it is important to make use of the Java EE platform which provides a runtime environment for developing and running large-scale ,multi-tiered and secure network applications. It also provides various frameworks which will be discussed below, for utilising services for multi-tier applications that produces manageability and accessibility needed by the system.

4.8.1 Java Persistence API

The java Persistence API provides services such as object/relational mapping(ORM) facility for managing relational data in Java application and it also allows relational database to be viewed as object-oriented database. The java persistence consists of :

- Persistence API
- The query language
- ORM

4.8.2 ASP.NET MVC

The ASP.NET MVC is a web application framework which implements the Model View Controller pattern. The ASP.NET MVC supports web pages ,web forms and MVC which is needed by the system.It supports all ASP.NET functionalities like data binding, user controls,master pages etc. Hence ASP.NET MVC framework is ideal for this system which use MVC architecture.

4.9 Access and integration channels

- Java Persistence API
- Simple Access Project Protocol (SOAP) is a protocol which is platform and language independent and allows programs that run on different operating system to communicate using HTTP and XML. It's quite useful in handling asynchronous processing and supports many other protocols and technologies
- The system will be accessible by using a web interface which is bootstraped to cater for all devices of different sizes
- Access to the system will be provided using the HTTPS protocol to ensure security, along with IPsec (Internet Protocol Security) which will allow for a secure IP and to ensure that no malicious data is sent to the servers
- Web Service Definition Language (WSDL) will be used to describe the functionality and the operations provided by the web-based service
- PHP along with MySQL API will be used to query the database
- A RESTful web services API which will allow users to access the system without knowledge of what happens in the backend

4.10 Technologies

5 Functional requirements and application design

5.1 Use case prioritization

This section specifies the level of importance of different use cases, prioritized in terms of the following 3 catogories: Critical, Important and Nice-To-Have. Each of which has a lesser importance than the previous one.

5.1.1 Critical:

- User gateaway
  - Register user
  - Login user
- Research paper management
  - View paper
  - Publish paper
  - Terminate paper
  - Assign research paper's conference venue
  - View research paper's venue
- User account management
  - View user account
  - Remove account
  - Create new account
  - Search user
  - Modify account
  - Assign co-author
- Log services
  - View recent activities
  - System events
  - Determine date and time
  - Identify user
- Other
  - System should only allow one primary author for each research paper. The rest should be co-authors
  - Superuser, one to be above all other users to have complete control and access to the entire system
  - Only metedata of the document will be posted, no documents will be posted
  - Research documents are archived rather than deleted

5.1.2 Important:

- Notification Services
  - Create notification
- Archival Services
  - Terminate paper
  - Revive paper
- Units accumulated from work on papers is reflected on the user's profile

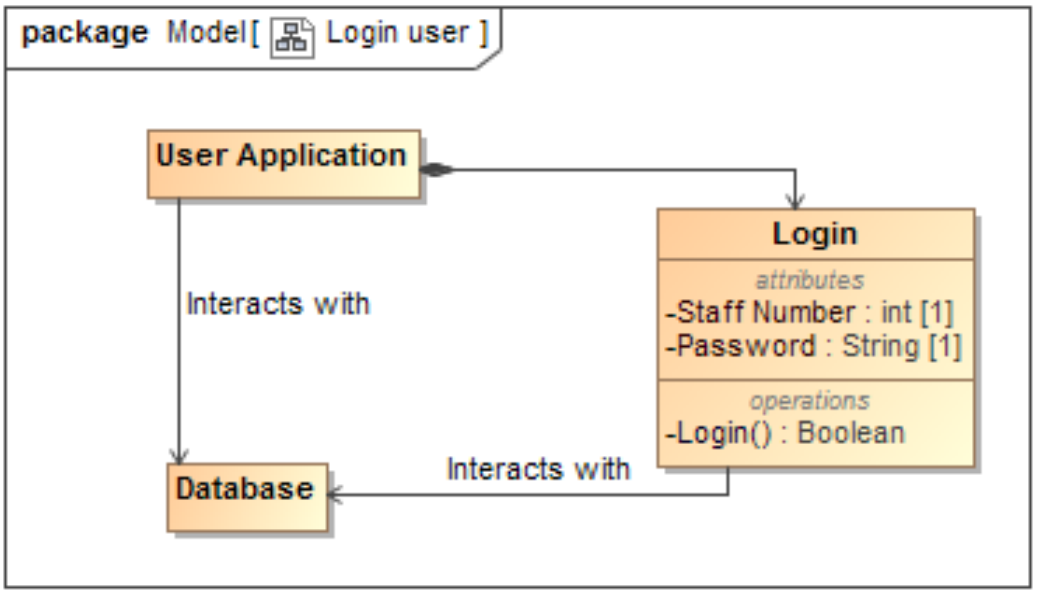
5.1.3 Nice-To-Have:

- Users can search for other users attending the same conference
- Data representation via bar charts of the units accumulated by researchers throughout the past years

5.2 Use case/Services contracts

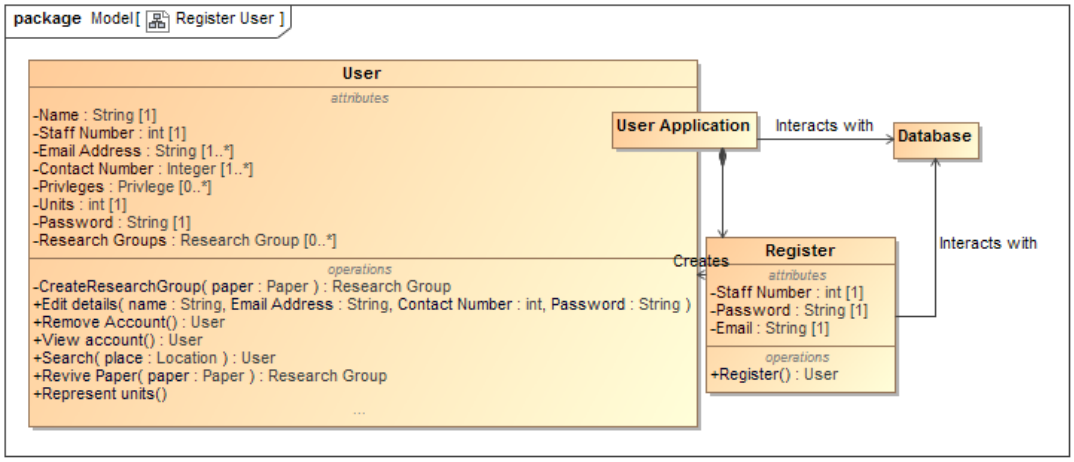
5.2.1 Login user

- **Description:** The ability for a user to login to the system
- **Pre-Condition:** A user name and password must be provided by the user attempting to login. The username must exist in the database and the password needs to be the same as the one found within the database.
- **Post-Condition:** The user will be notified the login was successful and now has the privledges associated with the account.



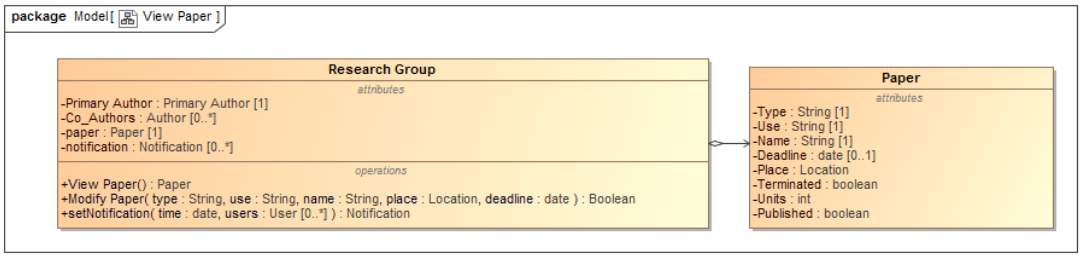
5.2.2 Register User

- **Description:** The ability to create a new account within the system
- **Pre-Condition:** The user must provide a username, password and email address that will be associated with the created account. The username must also not already exist within the database.
- **Post-Condition:** The account needs to be created and exist within the database



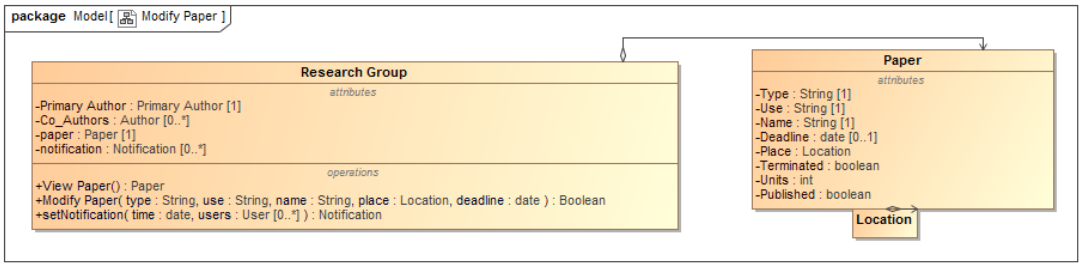
5.2.3 View paper

- **Description:** The ability to view the meta-data related to a research paper
- **Pre-Condition:** The user attempting to view the paper must be an author of the paper or have sufficent privledge.
- **Post-Condition:** The data of the paper in question must be displayed to the user.



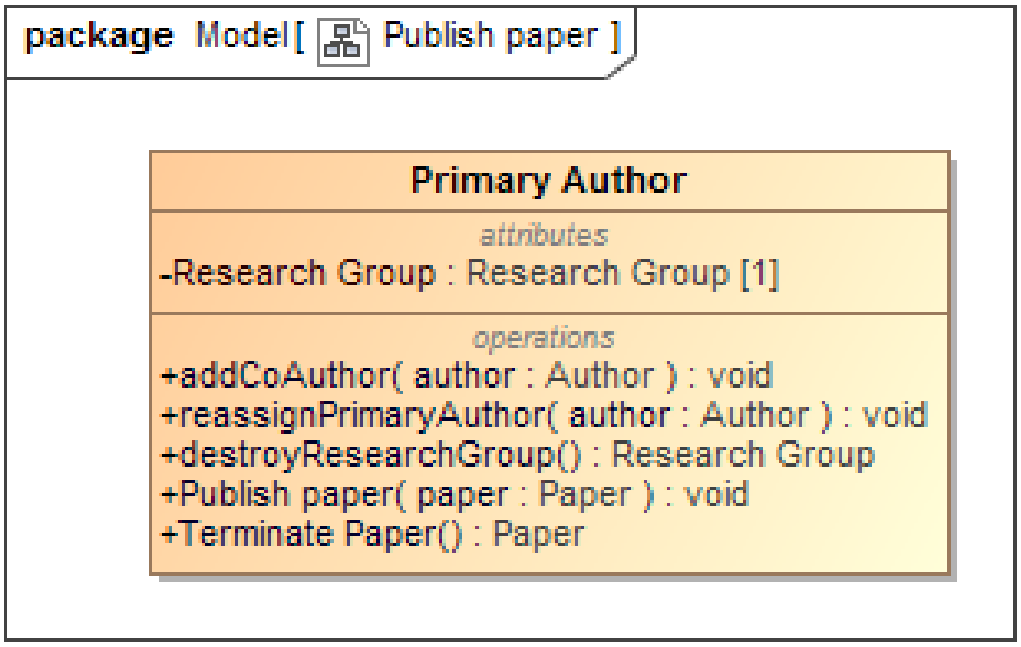
5.2.4 Modify paper

- **Description:** The ability to edit the meta-data related to a research paper.
- **Pre-Condition:** The user attempting to edit the paper must be an author of the paper or have sufficent privledge.
- **Post-Condition:** The paper's meta-data needs to be altered in the desired way.



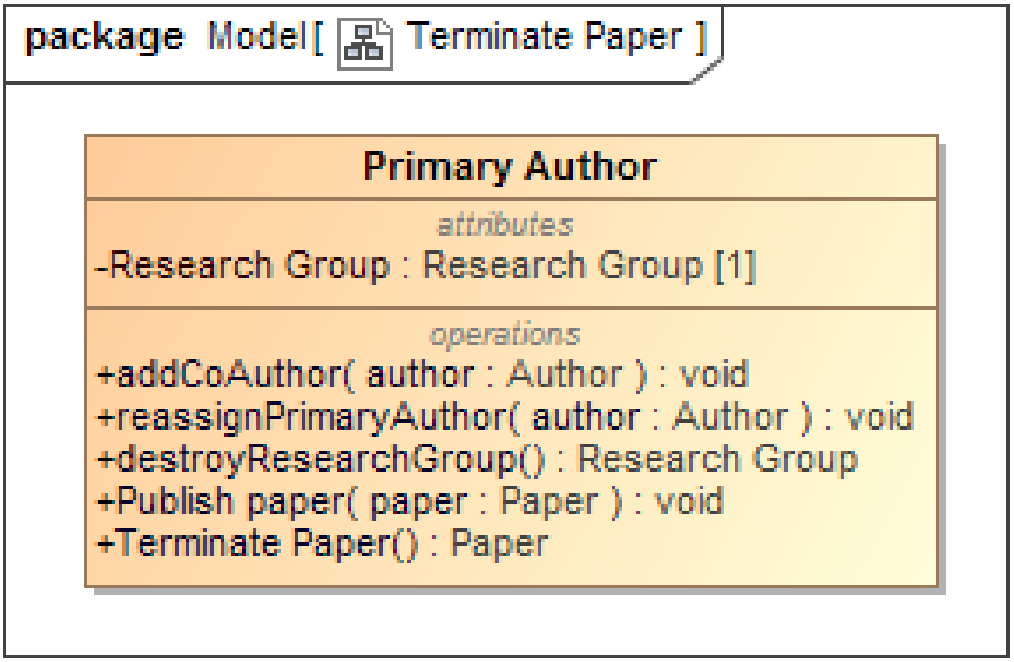
5.2.5 Publish paper

- **Description:** The ability to set a paper within the system as published and discontinue work on it.
- **Pre-Condition:** A user with sufficient priveledge should be making this action
- **Post-Condition:** The paper is set as published and the group is dissolved.



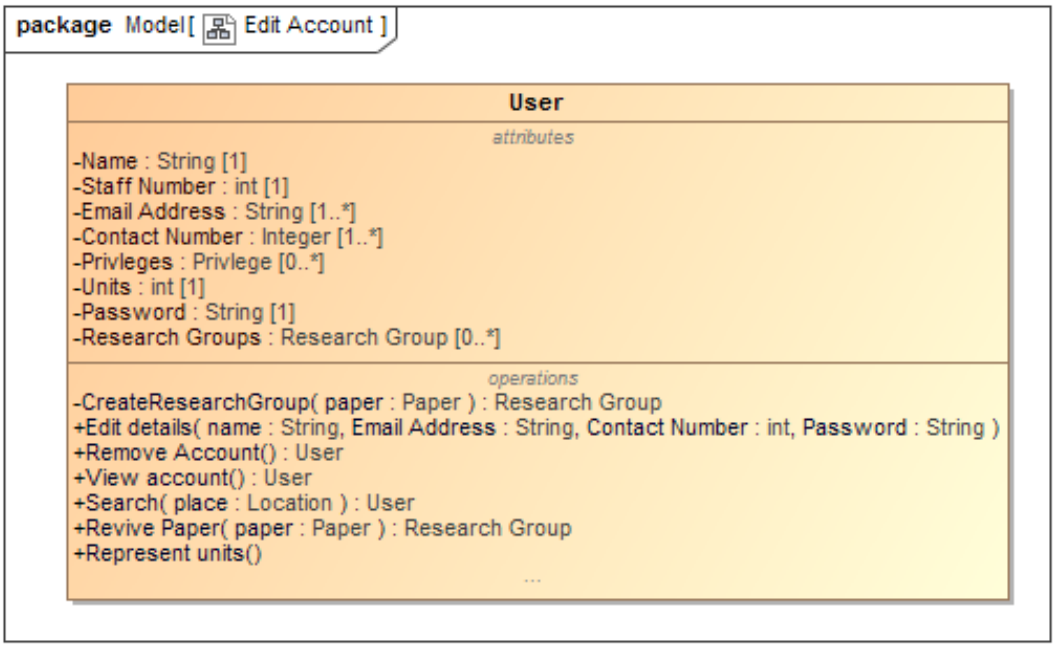
5.2.6 Terminate paper

- **Description:** The paper is discontinued by the current research group
- **Pre-Condition:** The request has to be made by the primary-author, the admin, the Superuser or the Head of Department
- **Post-Condition:** The paper is listed as discontinued



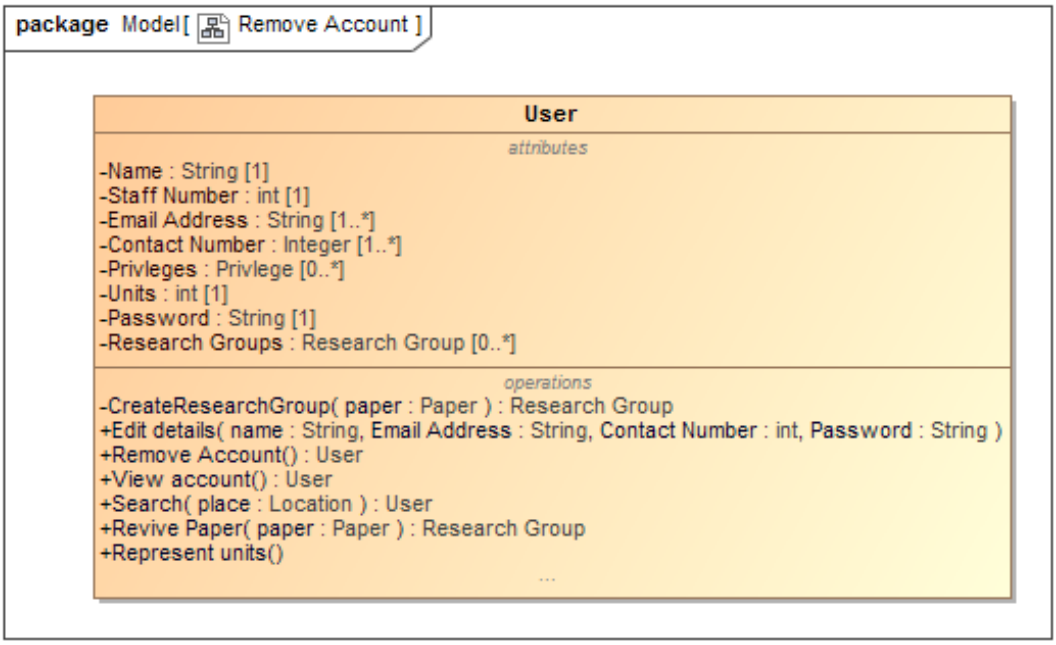
5.2.7 Edit Account

- **Description:** The ability for users to edit their profiles
- **Pre-Condition:** The user attempting to edit the profile must either have sufficient priveledge or must be the owner of the account
- **Post-Condition:** The profile has been edited in the desired way



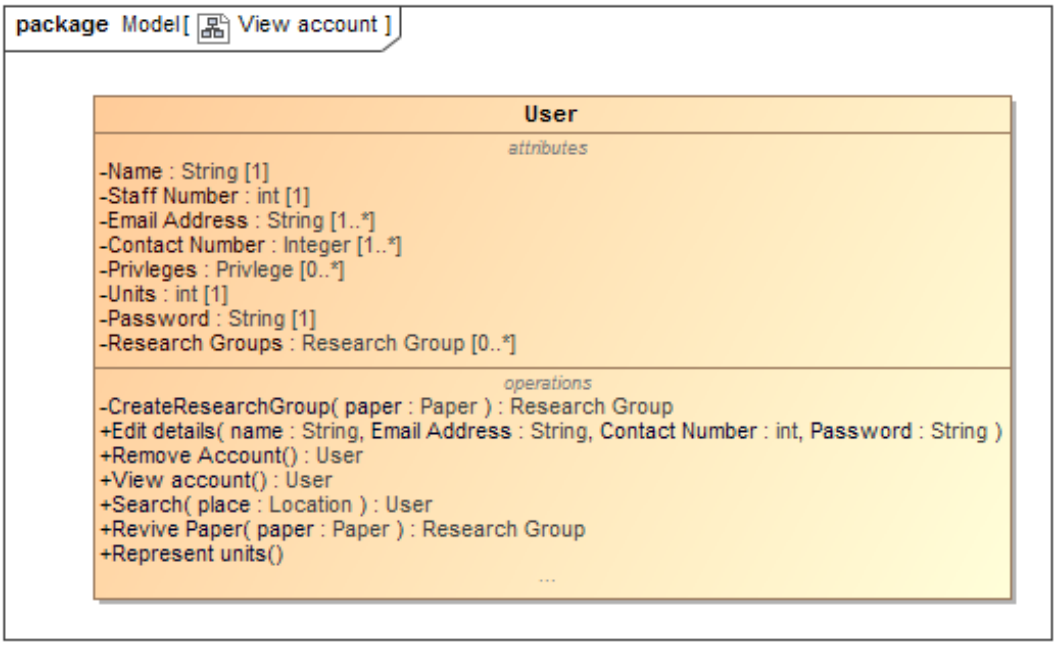
5.2.8 Remove account

- **Description:** Delete a user account from the system
- **Pre-Condition:** The user attempting to delete the profile must either have sufficient priveledge or must be the owner of the account
- **Post-Condition:** The profile no longer exists within the system



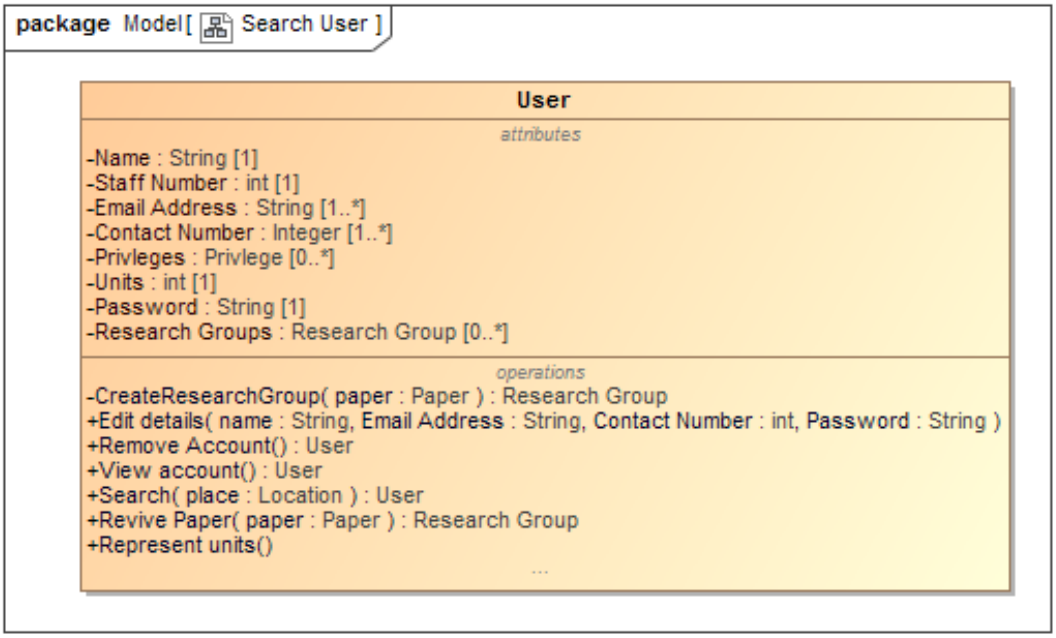
5.2.9 View user account

- **Description:** The ability for a user to view an account
- **Pre-Condition:** The user must be logged into their account or the user attempting to view the account must be an admin, super admin, or head of department.
- **Post-Condition:** The user's account needs to be displayed for the requesting user



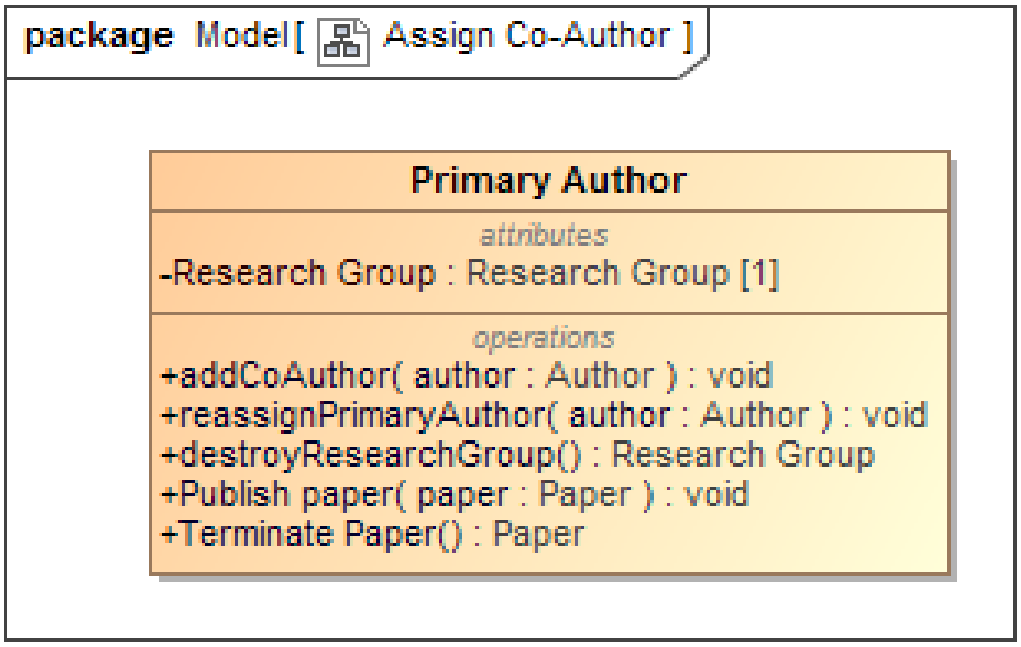
5.2.10 Search user

- **Description:** The ability to search for users working on paper's for the same conference as well as admins, Superusers and the Head of Department to search for users based on certain criteria
- **Pre-Condition:** The criteria for the desired group of users must be specified
- **Post-Condition:** The user/s are presented to the user requesting



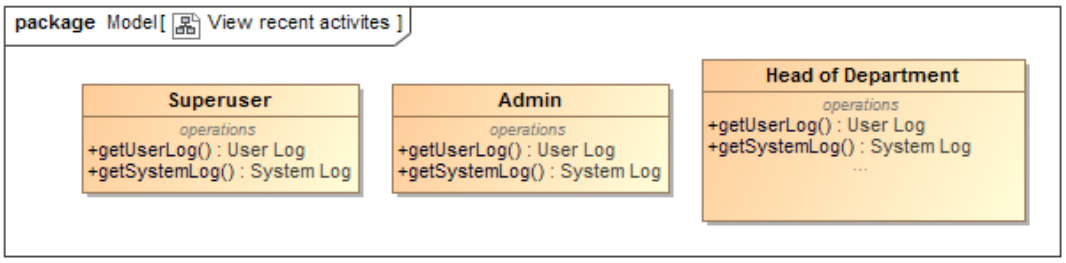
5.2.11 Assign co-author

- **Description:** A co-author is assigned to a research group
- **Pre-Condition:** The user assigning the co-author must be a primary-author/admin/Superuser or Head of Department
- **Post-Condition:** The co-author is linked to the research group



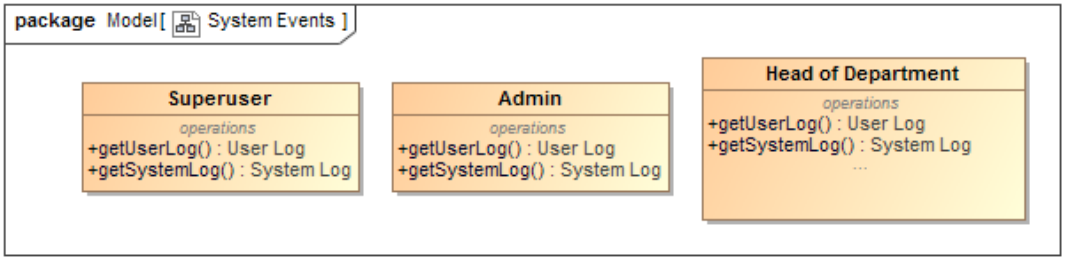
5.2.12 View recent activities

- **Description:** The ability to view recent actions taken by the user on the system
- **Pre-Condition:** User requesting must be of sufficient priveledge
- **Post-Condition:** The activities are presented to the user requesting them



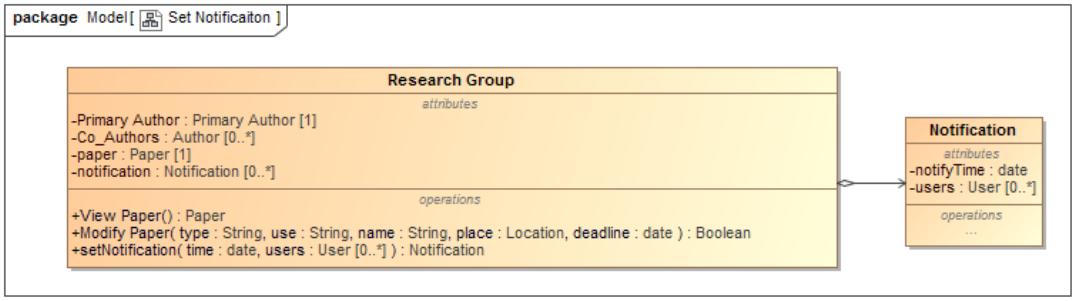
5.2.13 System events

- **Description:** The ability to view logs of events that have taken place within the system
- **Pre-Condition:** User requesting must be an Admin, Superuser or Head of Department
- **Post-Condition:** The log of system events is presented to the user



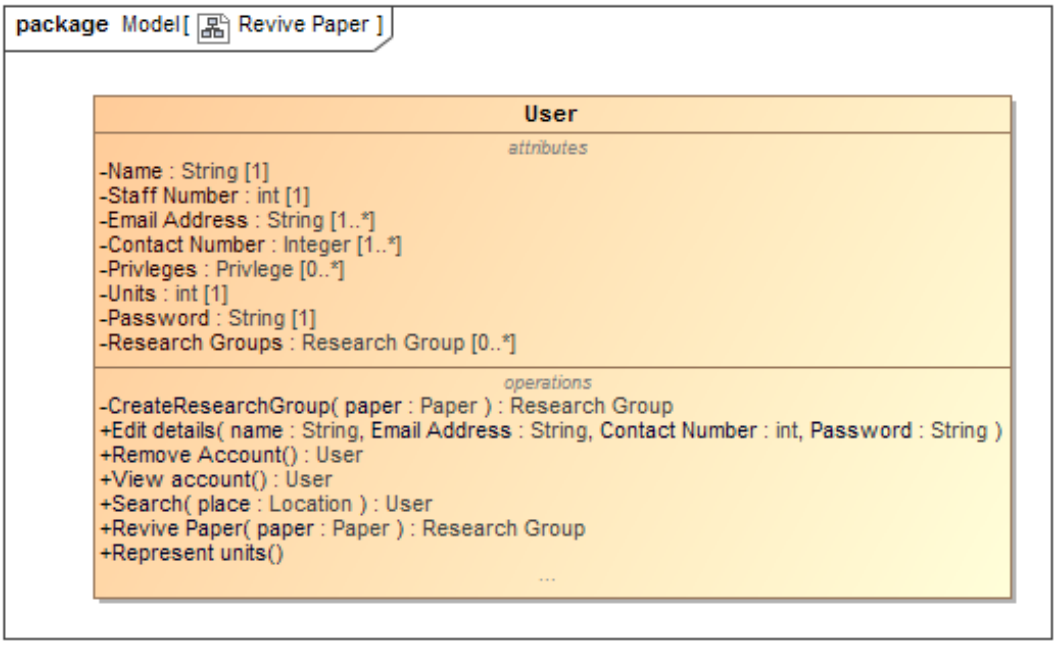
5.2.14 Set notification

- **Description:** The ability to set who receives a notification as well as when the notification should be issued
- **Pre-Condition:** The user setting the notification must be a Primary Author, Admin, Superuser or Head of Department
- **Post-Condition:** The notification is set and will email the specified user at the specified time/s.



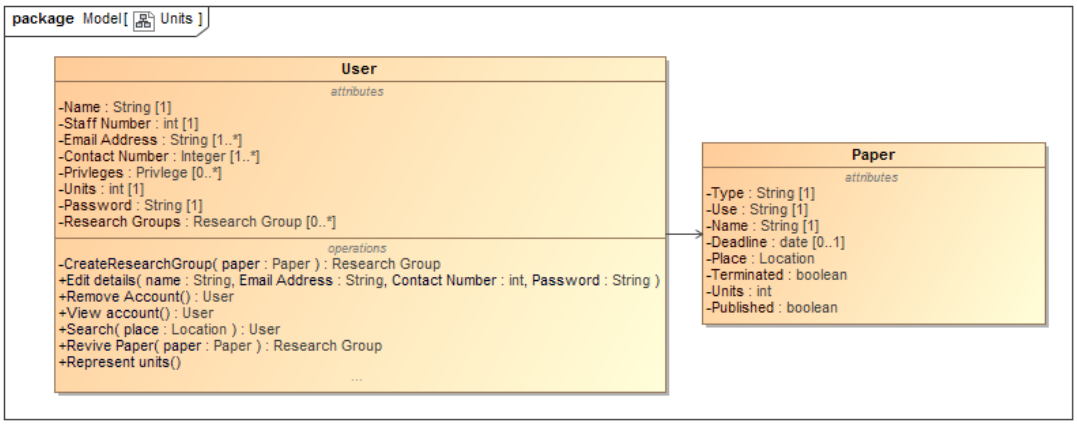
5.2.15 Revive paper

- **Description:** Assign a terminated paper to a new research group
- **Pre-Condition:** The terminated paper must exist within the system
- **Post-Condition:** A new research group is created with the requesting user as the primary-author



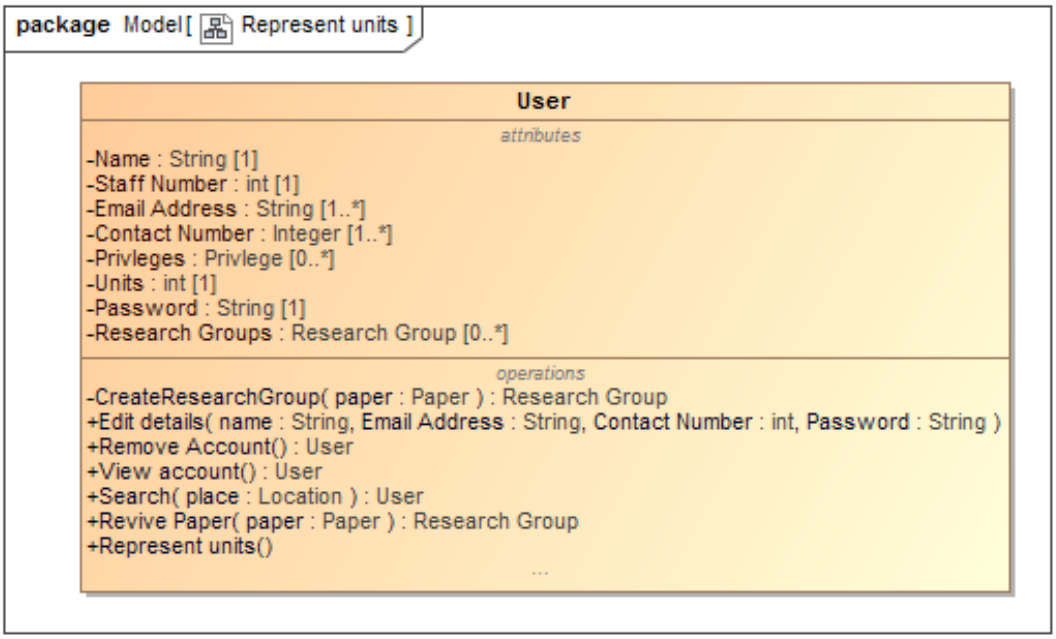
5.2.16 Units accumulated from work on papers is reflected on ther user's profile

- **Description:** Units accumulated from work on papers is reflected on ther user's profile
- **Pre-Condition:** A paper worth a certain number of units must be published and the user needs to be an author of the paper.
- **Post-Condition:** The units are added to the user's profile and displayed when the profile is viewed



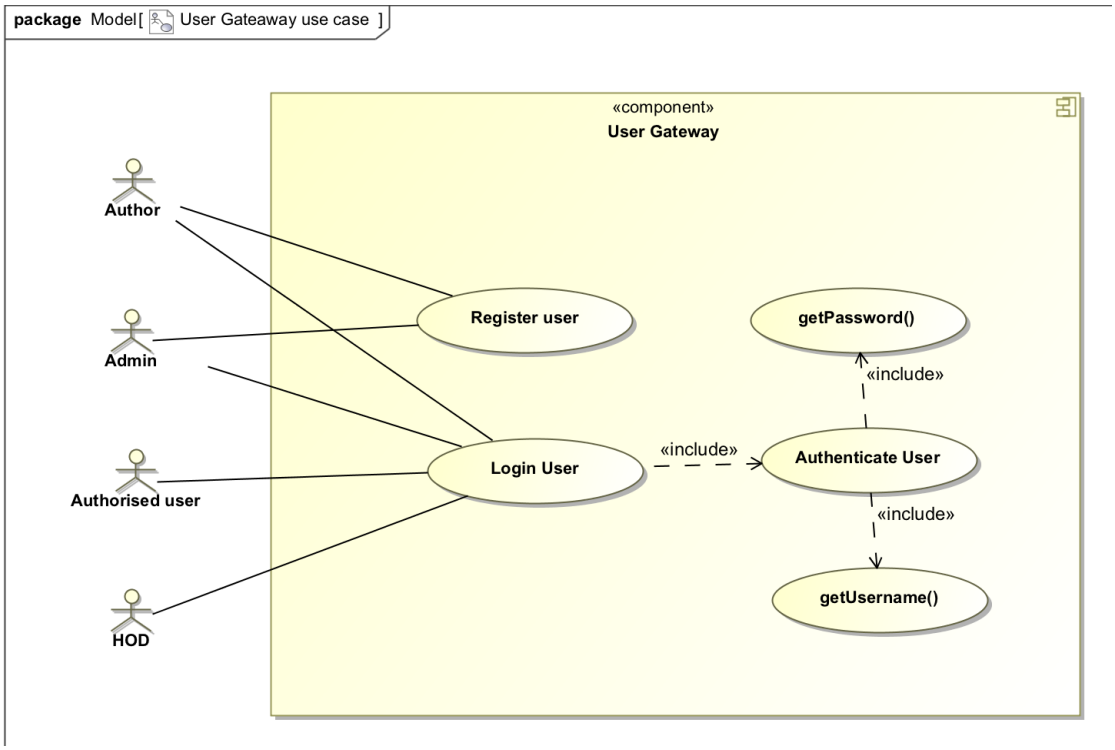
5.2.17 Data representation of units accumulated

- **Description:** A bar chart representation of the units a user has accumulated over time
- **Pre-Condition:** The user must exist and the user requesting to view the representation must be the owner of the account, an Admin, Superuser or HoD.
- **Post-Condition:** The bar chart is presented to the requesting user.

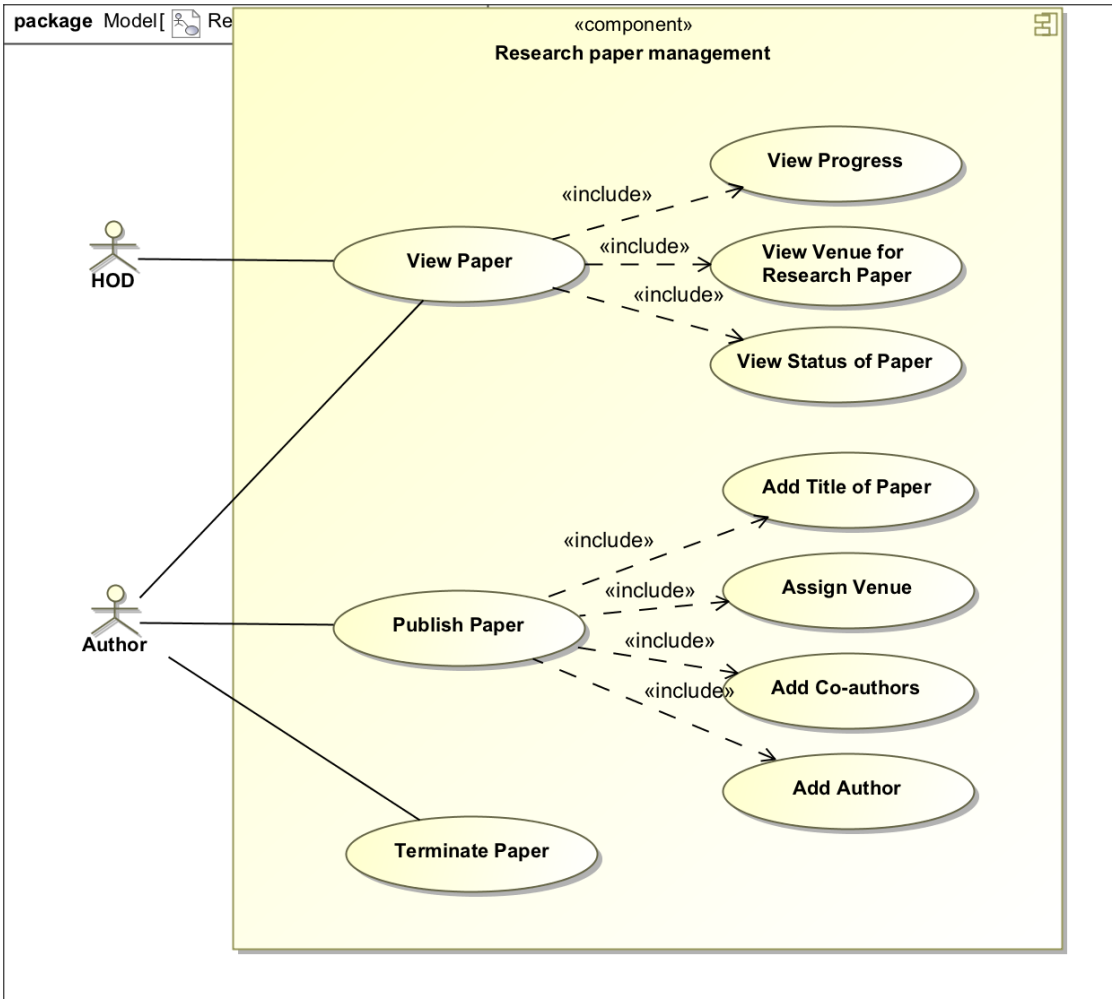


5.3 Required functionality

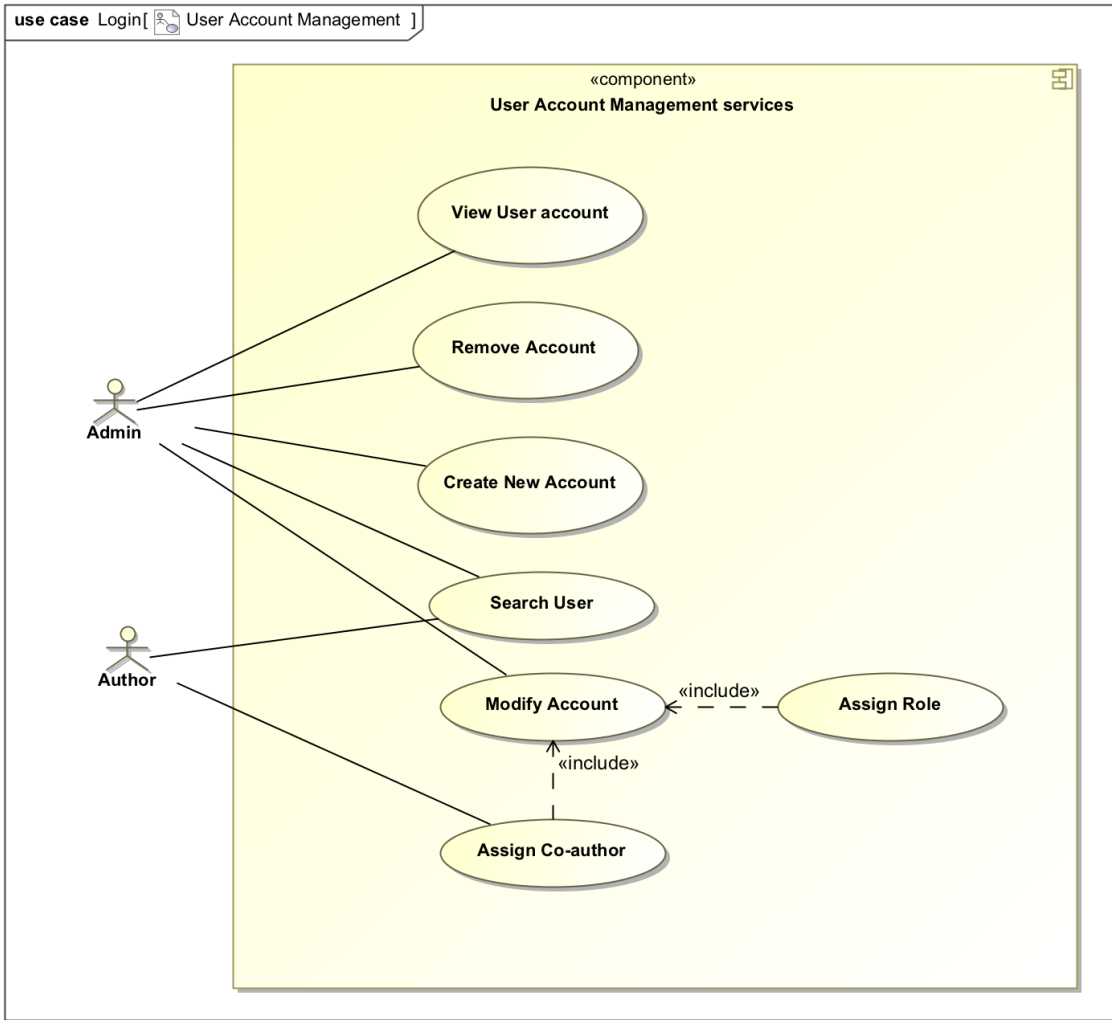
5.3.1 User Gateaway



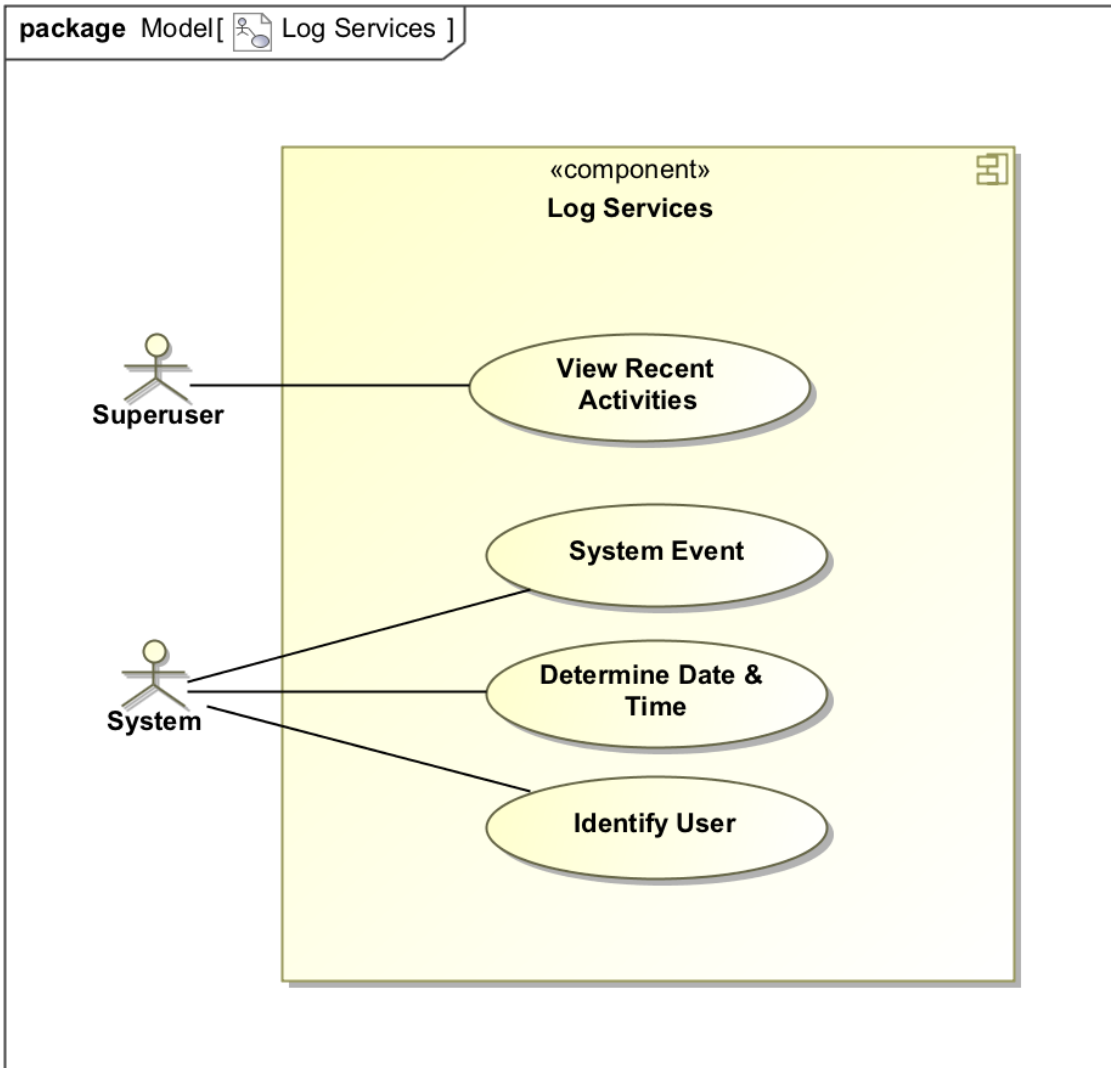
5.3.2 Research Paper Management



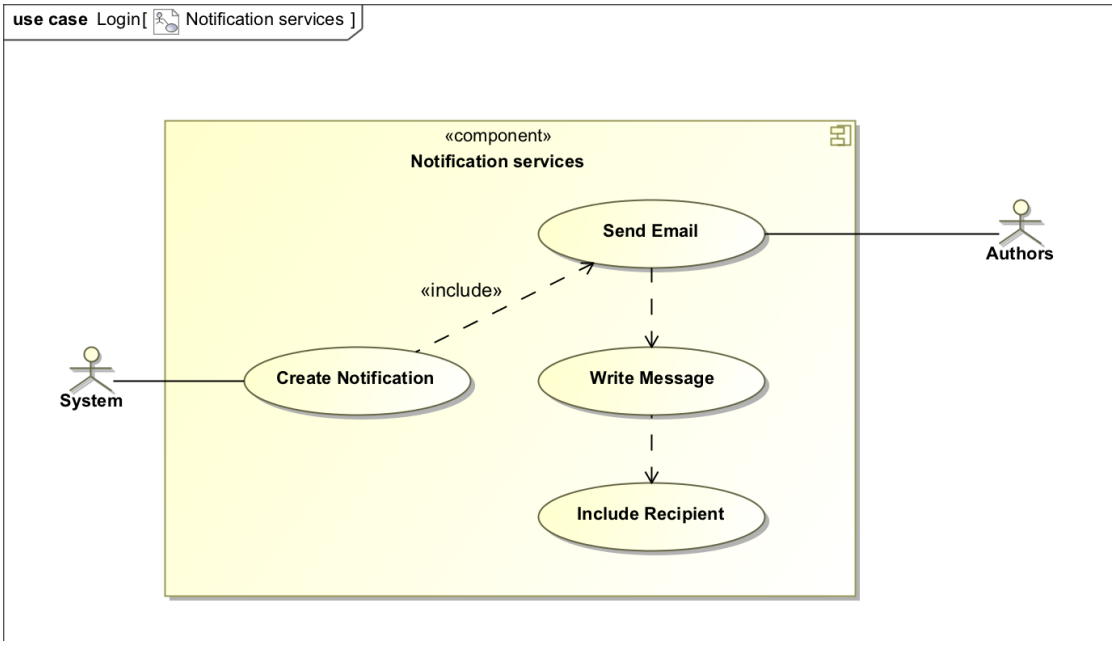
5.3.3 User Account Management Services



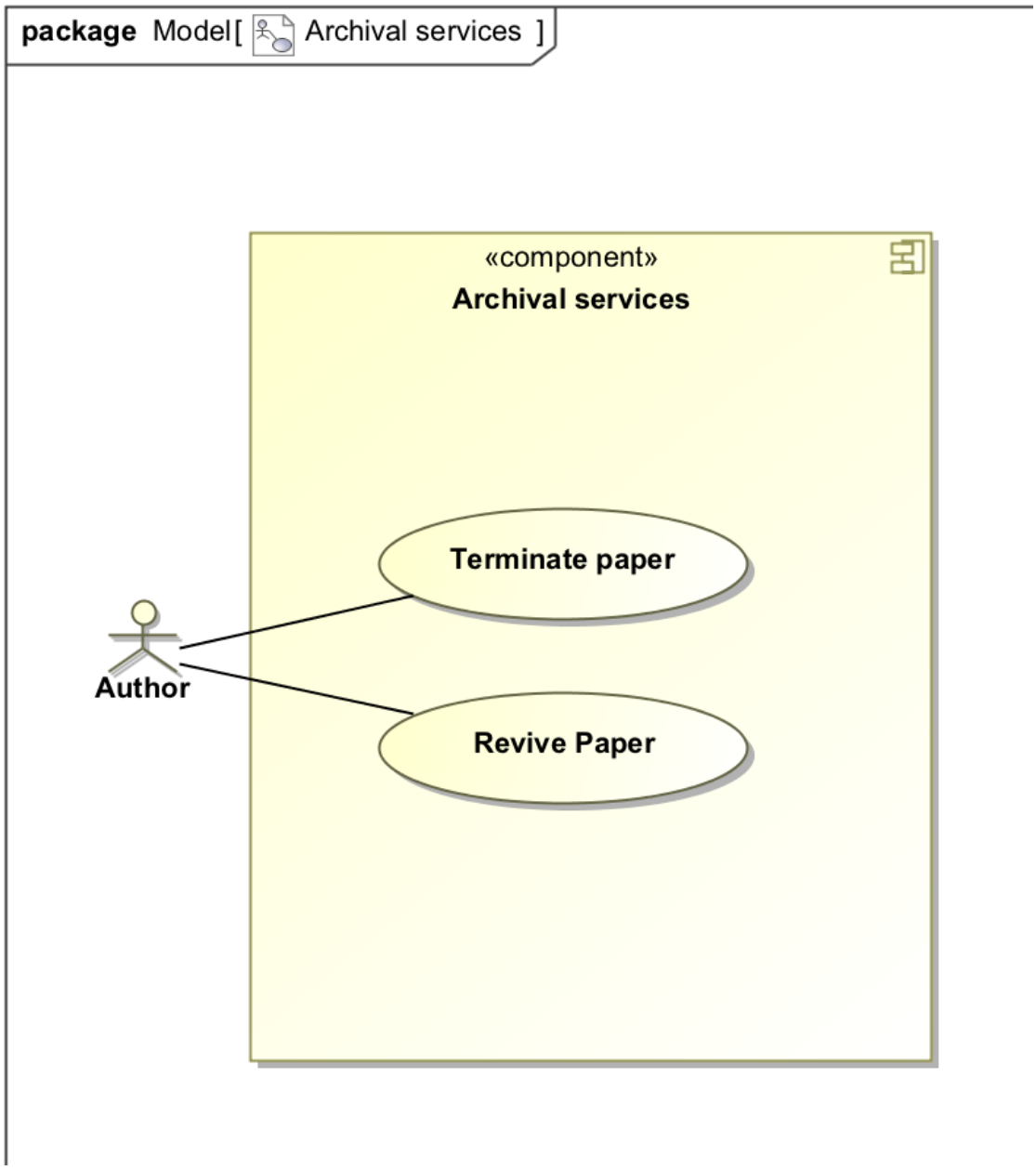
5.3.4 Log Services



5.3.5 Notification Services

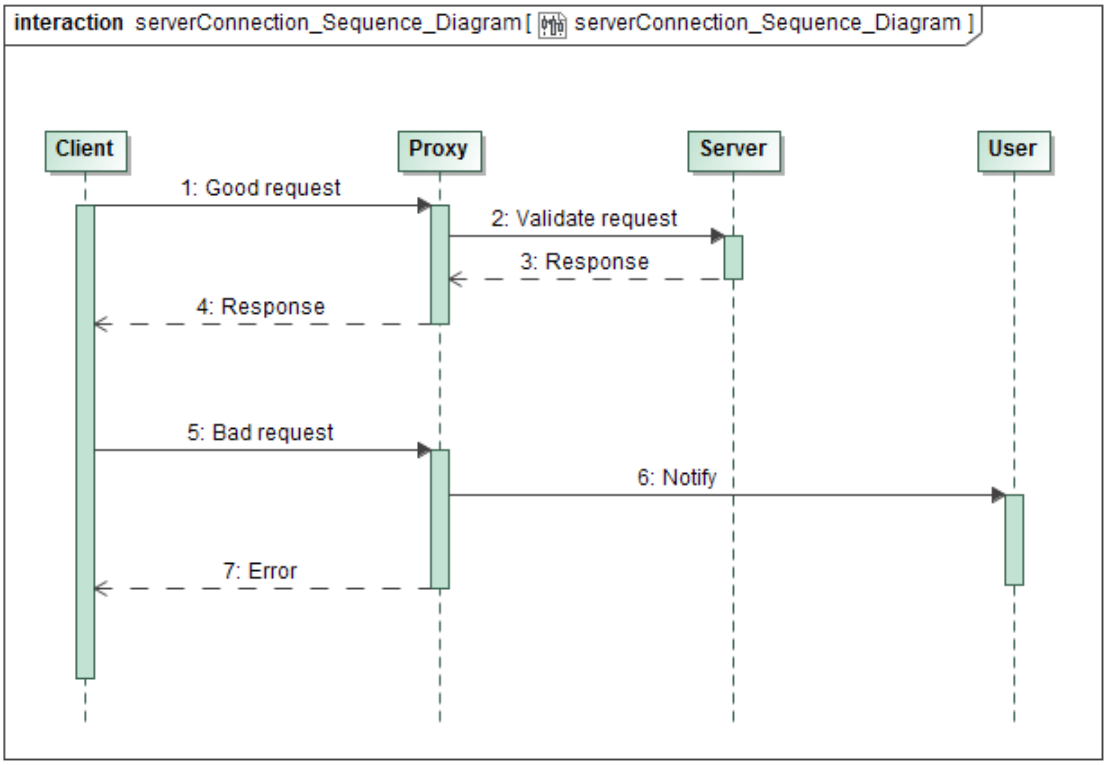


5.3.6 Archival Services

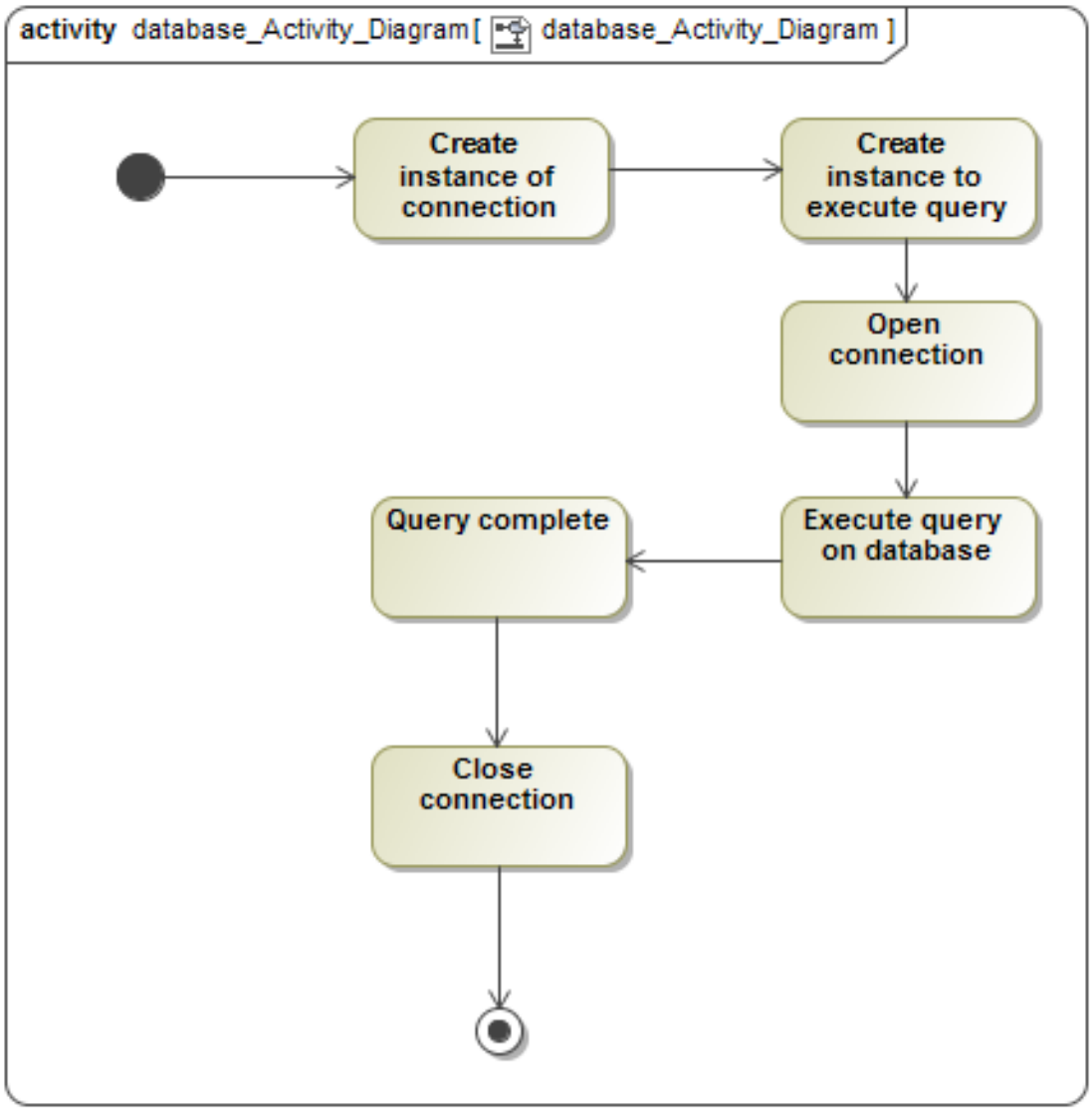


5.4 Process specifications

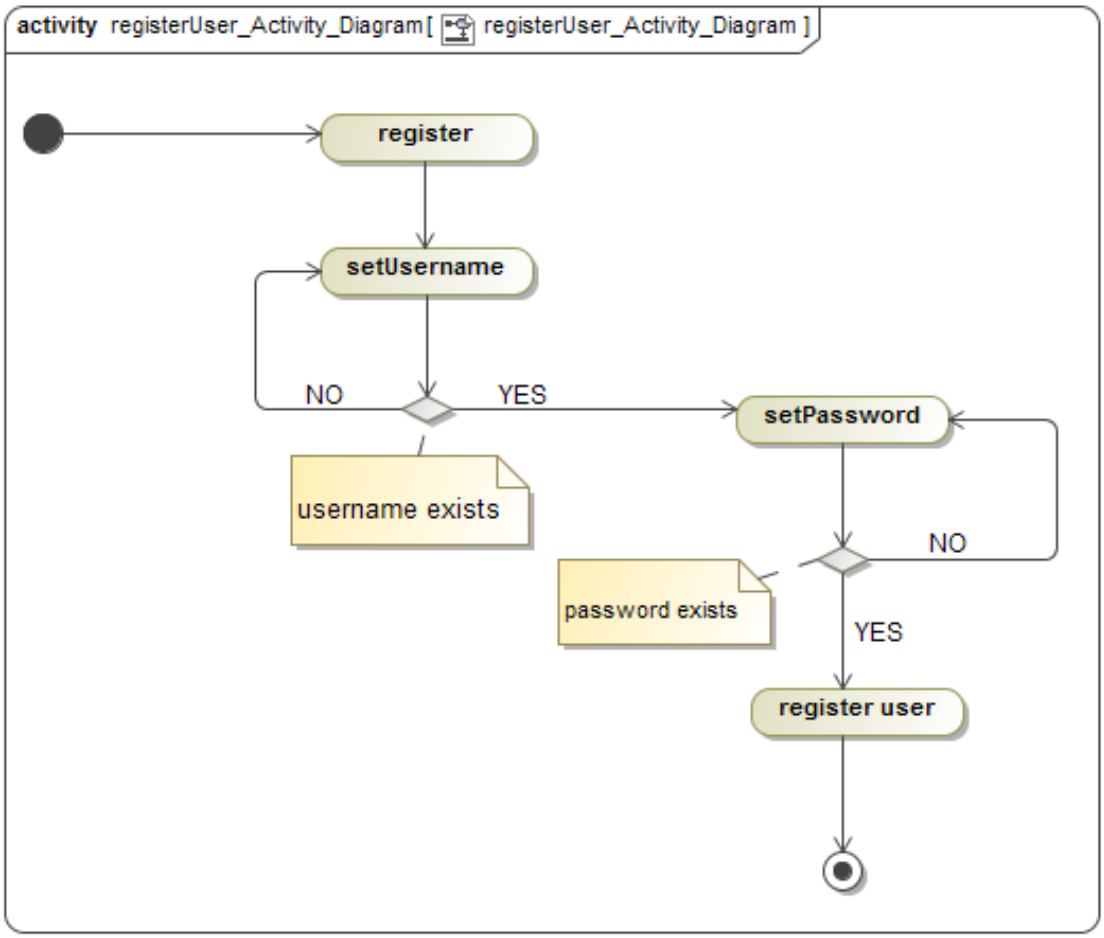
5.4.1 Server Connection



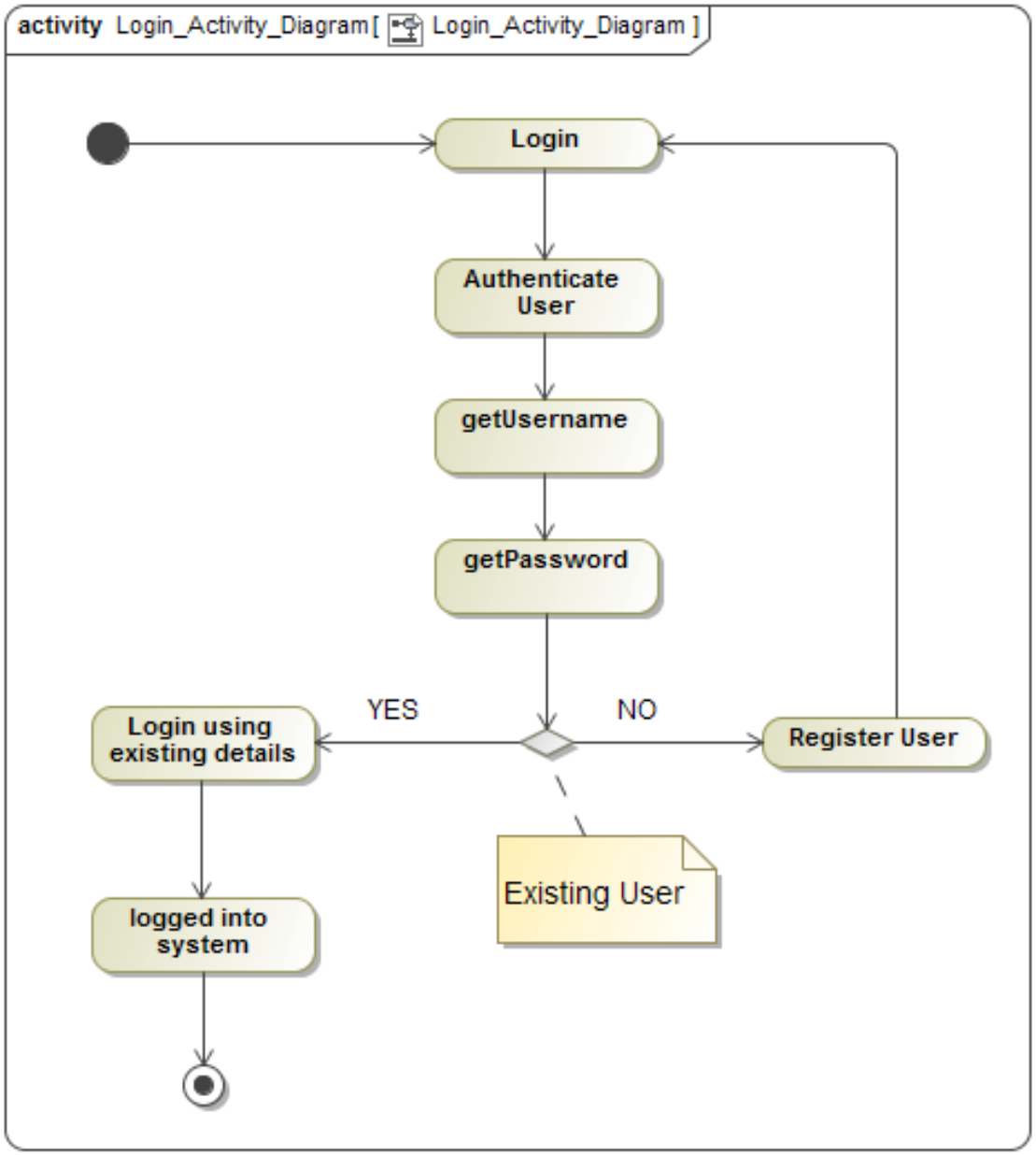
5.4.2 Database Connection



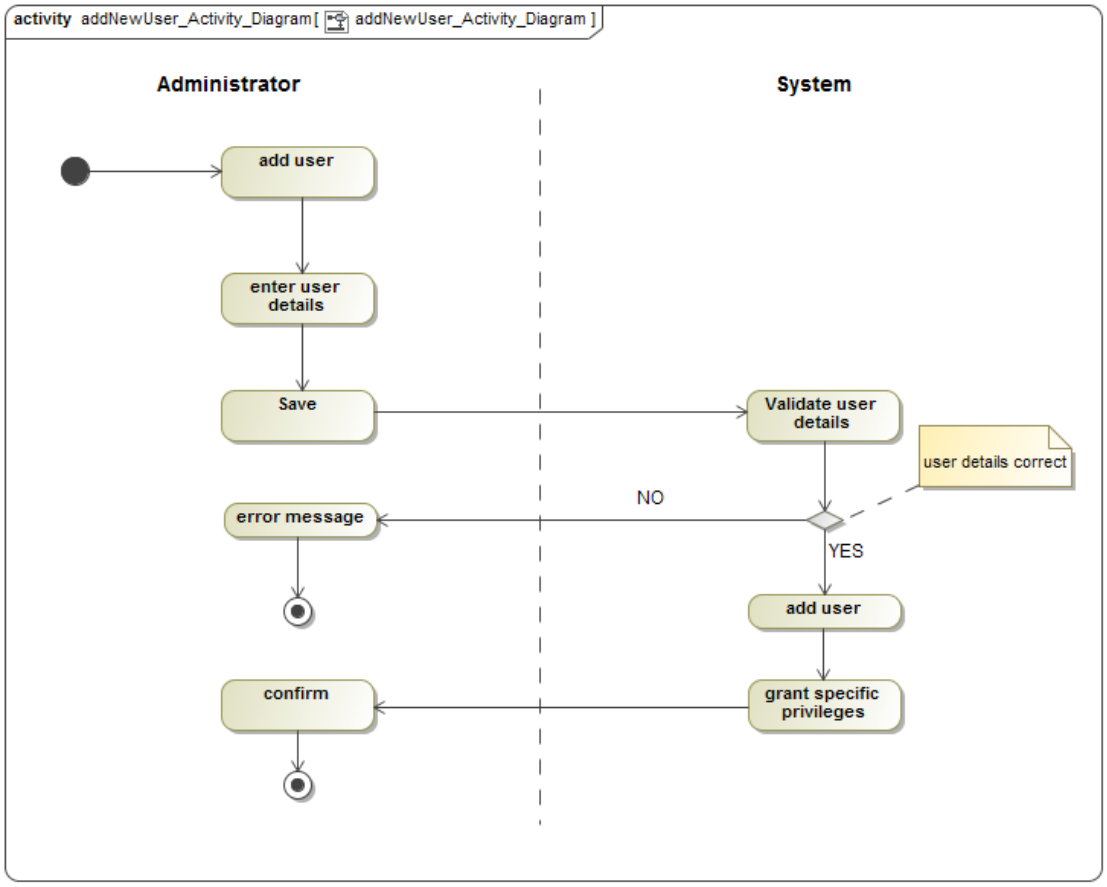
5.4.3 Register User



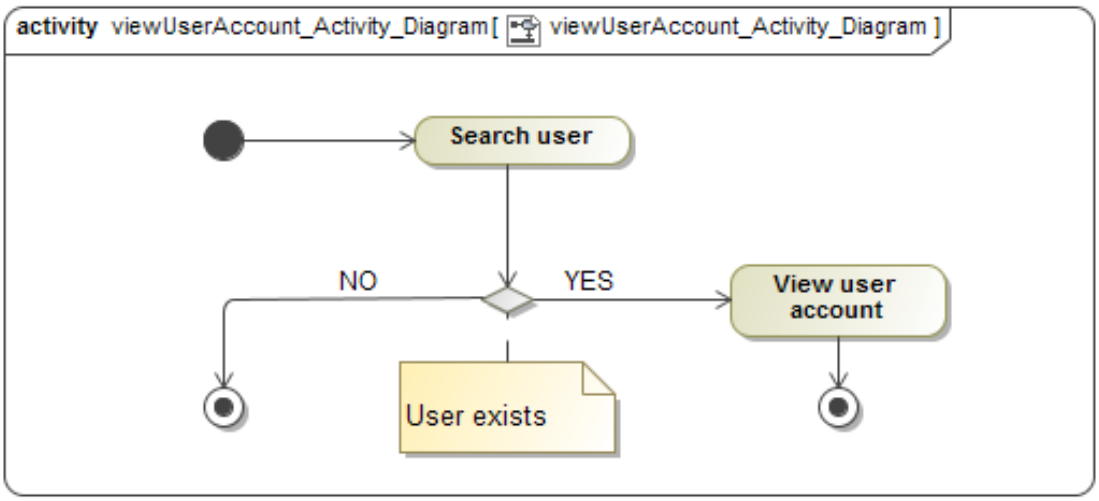
5.4.4 Login


activity Login_Activity_Diagram [Login_Activity_Diagram]
Login
Authenticate User
getUsername
getPassword
YES
NO
Login using existing details
Register User
Existing User
logged into system

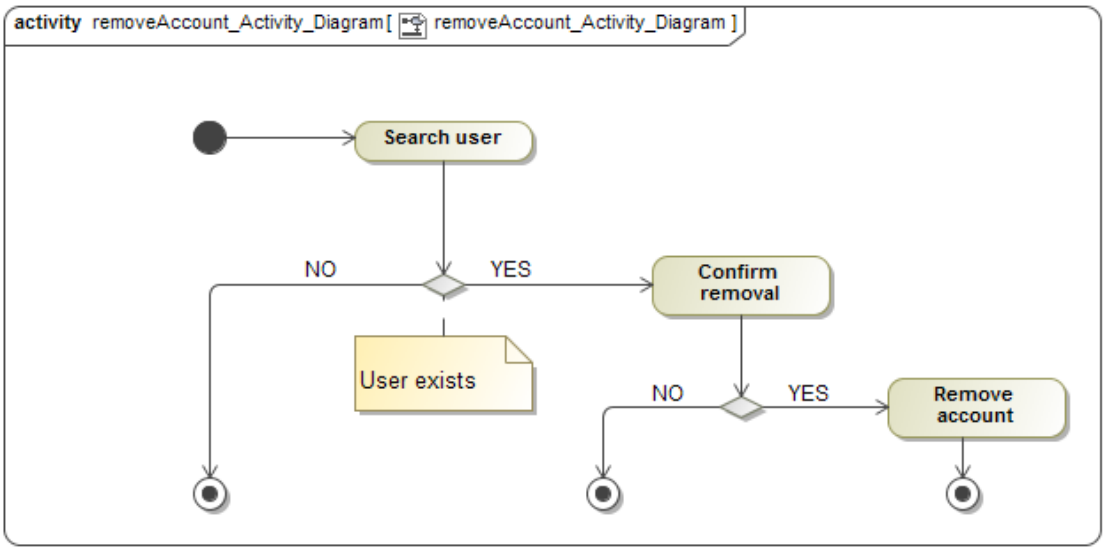
5.4.5 Add new User



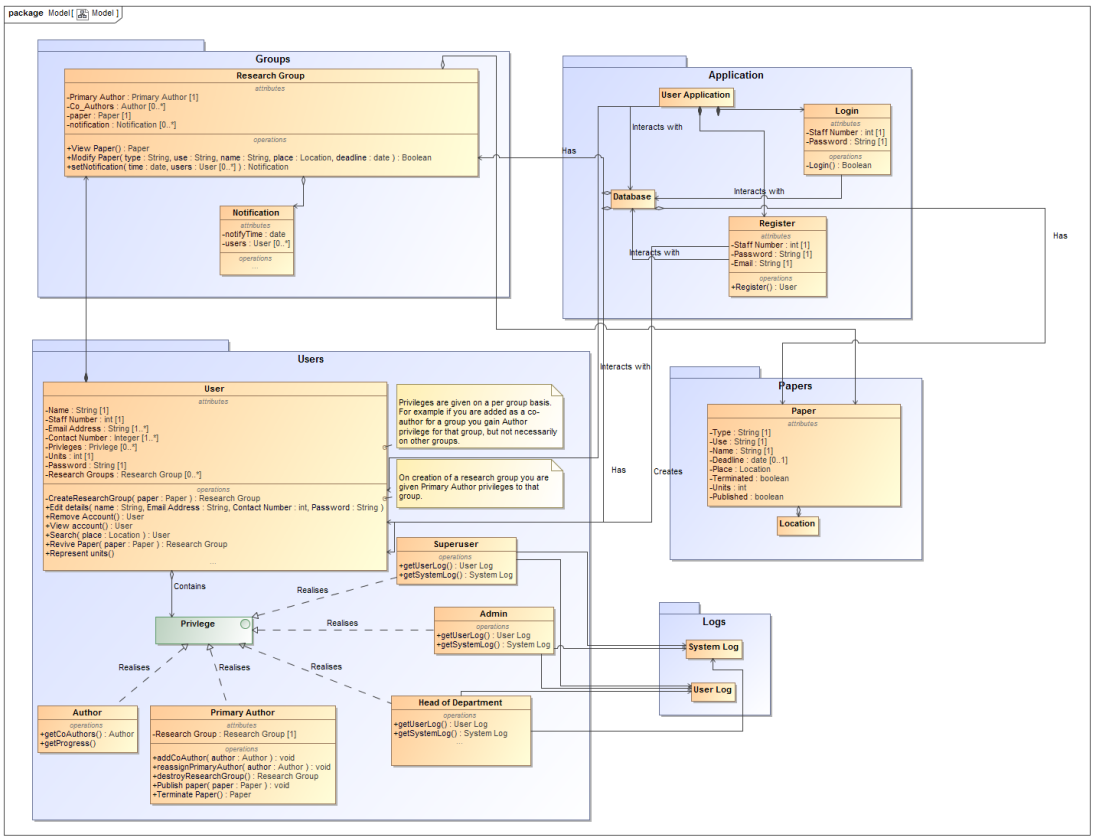
5.4.6 View User Account



5.4.7 Remove Account



5.5 Domain Model



6 Open Issues

- For further references and to access Team Braovo's github repository, please click on this link.

References

[1] Mrs Vreda Pieterse. Cos 301 mini project: Client specification. COS 301 lecture notes and template documents via the Computer Science website, February 2016. Specifications also provided during COS 301 lectures at the University Of Pretoria.