UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science

COS 301 - Mini Project

Assignment 1

Author:	*Student number:*
Tshepo Malesela	u14211582
Kevin David Heritage	u13044924
Unarine Rambani	u14004489
Vukile Langa	u14035449
Wynand Hugo Meiring	u13230795
Nontokozo Hlastwayo	u14414555
Tim Kirker	u11152402

March 11, 2016

Requirements Specification

Team Lima github repository link

For further references see gitHub. March 11, 2016

1 Introduction

In this section we will explore an overview of everything that is included in the Software Requirements Specification (SRS) document.

1.1 Purpose

The purpose of this document is to give a detailed description of the features of the Mini-Project, to serve as a guide or reference for the software developers and primarily a proposal to the client approval.

1.2 Scope

The Mini-Project(Publication System) is a web and Android application which will assist research leaders as well as the heads of department to keep track of the progress of different research groups.
Users(UP Staff) will act as the administrators for the different research groups and provide information regarding research topics, group members, progress, etc using the web-portal.
The system will only keep track of metadata and not contain any of the actual publications. The information is maintained in a database that is located on a web-server. An internet connection will be required to fetch and display the information.

1.3 Overview

The remainder of this document includes four more sections and is organized as follows: Section 2 describes the Vision, which basically explains what the client wants to achieve with the project and what the average user would get out of the product.Section 3 continues to discuss the background, which includes the business/ research opportunity to simplify the administration and management of documents/publications in the world of research. This section also sheds light on the problems being faced that may have led to this project being started. Section 4 presents the Architecture requirements to the reader, including the integration and quality requirements as well as the architectural constraints. Section 5 describes the Functional Requirements and captures all the functionality which would be required by the users of the system. Section 6 brings up the Open Issues which entail anything that needs to be clarified regarding the requirements as well as any irregularities/inconsistencies found in the requirements.

2 Vision

2.1 Product Functions

The main aim of this system is to assist researchers(authors) and head of department with storing metadata of research papers(not the paper itself). In the system the users will be able to see how many papers a particular user is working on, identify which research papers are due at what time(the system will keep a daily reminder of the due date) and who are the other authors of the paper but will maintain privacy since a user won't be able to see other user's work(only going to see papers where the user co-author) unless it is the head of department or a supervisor/superuser. The system will keep a status of the paper(still working on it,submitted waiting for feedback,rejected, accepted, published or improving it after rejection).
The system should be able support 100 users concurrently without any difficulties. The head of department is allowed to see everything in the system meaning everyone's work. An author is not necessarily a user of the system meaning that a person can be added as one of the authors of a certain research paper, while they are not a user of the system, the primary author is responsible of adding and removing authors for a paper and the primary author should be a user of the system.
Every research paper on the system should have at least one user responsible for it. The system keeps the title of the paper, if a paper is accepted/published a link to where the actual paper is stored,if a paper is deleted in the system(by the primary author) a reason, should be provided for that termination, the type of a paper(journal, conference paper or book chapter) is also kept in the system, and the system should also keep report containing DoE Units, DoE Research Output Units, UPWeighted Research Outputs, Estimated DoE Outputs and Research Funding.

2.2 User classes and characteristics

Head of Department

- Must be a registered user of the user of the system.
- There is only one head of Department, since the system is for one department.
- Able to see everyone's work on the system.
- May be an author.
- May add or delete papers.

Primary Author/ Leading Researcher

- Must be a registered user of the user of the system.
- Any number less than a hundred.
- Able to see only the list papers of where is authoring or co-authoring.

- Can add or delete papers.
- Can add or remove authors.

Author

- May or may not be a user of the system.
- If a user able to see only the list papers of where is co-authoring.
- Otherwise not able to see anything in the system.

2.3 Operating Environment

The system will be a standalone system, it does not load any external module or library function. For the user to login to the system, they must have internet access. When a user modifies something, every other use who has access to the information needs access to its latest version. The system must run on any operating system, and it will have make use of a virtual interface for the different operating system and hardware platforms.

3 Background

The system gives different authors/researchers who are working on the same paper an opportunity to be able to stay updated about the progress of the paper. The system also gives the head of department an opportunity to better supervise the staff members on the research they are working on, since research is one of the crucial aspects of every department in the university. The system also gives the head of department an opportunity to evaluate the staff member using the report the system provides.

A correct usage of the system will yield good results in management of the department's research activity and will also provide a better way of co-authoring among different authors/users because when there is a change in the status of the paper every author and user will be made aware of the deadline.

The system will also improve the current way of storing research metadata the department uses, which is writing everything on a spreadsheet. The system will also add other things that are not in the spreadSheet like a full report of how the whole department is performing with regards to research papers and provide a more informed way of co-authoring, since everyone working on the paper will know what is going on.

4 Architecture requirements

4.1 Access channel requirements

There will be two access channels for the proposed system. An Android application client will be used to fetch data (such as the deadline dates) and integrate it into mobile devices for users to easily identify when their deadlines are. The mobile application will also allow a user to search for and identify the type of publication (journal, conference paper or book) that is being worked on as well as how many papers a user may be working on. The mobile application will also allow users to see the working status of a paper (still working on it, submitted and waiting for feedback, rejected, accepted, published or improving after rejection). Web servers will be used in order to store all the users and their details as well as all the papers that are being worked on by any particular user. These web servers will be set up so that they can be concurrently accessed by 100 users without any difficulties. The system will also make due with browser clients that will have all the same functionality as the mobile client and allow any user to access the system where ever they may be. Both the mobile application and browser client will interact with the web server in order to gain access to any information they might need.

4.1.1 Architectural Scope

- Access to the data from anywhere at any time ie. a database that is always running.
- A server to host the database that is always on.
- Persistent internet connection that has a backup to allow people to always be able to have access to the server.
- The ability to let users know via email about important notifications and so forth.
- A progress report on each project ie. how much work has been completed on the project.
- Setting of deadlines on research projects. (Perception of time)
- Concurrency: to allow 100 concurrent users to use the system at the same time.
- Authentication with University of Pretoria servers to validate the users staff/student number.
- Super user access for the Head Of Department
- Write permissions for files and log files
- Searching capabilities

4.1.2 Quality Requirements

In order to create a safe and legitimate system that will meet all the specifications of the project, some system requirements such as how the system performance, security, flexiblity, maintainability, auditablity, integrability, cost and usabilty have to be considered.

- **Performance**

  The system has to stable and responsive enough to handle up to one hundred users without lagging, that is the system's response time must prove to be exceptional no matter the work.

- **Security**

  The most important security measure that needs to be upheld in this system is that only U.P staff should be allowed to register as users, and may be the only members who can access the link to view papers. They will either set their own usernames or use their access card numbers in place of their usernames. They will also be required to set their own passwords. This password should be at least eight characters long to make it a bit more difficult for potential hackers to gain access.If a user forgets his/her password, there will need to be a way of retrieving it or re-setting a new one. My group and I decided that an email containing a link should be sent out to a user directing them to a page whereby they will be able to set a new password. If a user also fails to get the correct password for more than three tries, the user account should be blocked for safety reasons, and the user should be sent an email following the password re-setting procedure. If system fails to allow a user back into the system, the user needs to direct an enquiry to the administrative user. The user is also given the authority to remove or add authors to the a paper, they may also choose to terminate the paper and should also be allowed to revive that same paper that was terminated. Only users involved in the publication can edit a paper, this modification must be known by the other users involved in the same paper, so a notification method will be set up to alert the other members. The system will also have user profiles, a user can only edit their own profile. The system also has super-users that have more control than users, they can access all publications. There are also administrative users who have access to the the system.The administrative users can basically remove or add users to the system. Their main role is to fix any administrative problems that may be faced by the users. They can also view the activities of a particular user and because of this, all user activity must be logged. The hierarchy of this system in terms of privilege is that administrative users have the most priority, super-users have the more authority than users and finally authors have the least authority.

- **Reliability**

  Reliability for this application to work is crucial to its success. If a user wants to access a paper and it is urgent, the user must always be able to access what they have access to at all times without the system failing fetch the information the user seeks. Reliability includes keeping track of user authority, the system should not lose track of what actions each user can perform on the system. Super-users must be able to oversee all publications at all times, if that failed for some reason and inaccurate publication statuses were presented or incorrect co-authors were placed into a group super-users would not be able to keep track of user activity and the application's reliability would be flawed.

- **Flexibility**

  This application should be web-based and also have a mobile application that users can make use of when away from a desktop or laptop. The system needs to be able to run the same on all these platforms (with the exception of probably having different User Interfaces for the two platforms) and users must be able to do the same things on the web-based application and mobile-based application. For the mobile side of things, this application needs to be able to support older versions of android mobile system, so that no matter what device is used, a user can access the application and look up what they need at all times.

- **Maintainability**

  Maintenance of this application will be handled by the super-users. With just a little over one hundred users this system will have to be well maintained to be able to run things smoothly.

- **Monitorability**

  The super-users will be in charge of monitoring user activity. Since there will be a log for each users activity on the system, the super-user can look at the log to see what a specific user has been doing. Everything in the system will be logged to increase the level of monitorability.

- **Usability**

  This application should be simple enough for a user to use without the user asking for assistance more than once. The system must have basic instructions, that could perhaps be made available to a user on a pdf file so that they can read and understand how to navigate through the system. If users still do not have a clear perspective on how the system works they will just keep referring to the pdf.

- **Cost**

  The system being built should be complex enough to run smoothly on all devices but also it should be maintainable by students to cut costs and man hours.

- **Integrability**

  The system should be integrable to mobile devices and not just be web-based. There will be a mobile application for this system.

- **Integrability**

  The system should always be able running optimally no matter how many users ther are. With the system only being available from within the university grounds, the system should be abble to handle the workload and events that users make with no lagging.

4.1.3 Integration And Access Channel Requirements

4.1.4 Architecture constraints

The server should be available to allow all users to login at any time. There should be search functionality to allow users to navigate faster.The system will be web-based and will also built to to run on mobile devices.

- **Web-based development**

  We will be building the web-based system using HTML, CSS, and some javascript for some user side functionality. We will do all client validation using javascript to check if the user input is valid.

- **Server-side development**

  To run the database we will be using mySQL and PHP. We will store the user profile using a mySQL database and use, and all validation will be done in PHP.

- **Mobile application development**

  To build the mobile application, we will use androidStudio(java). The mobile application should only be able to access the server when you are within the premises of the University.

4.2 Architectural patterns or styles

For the architecture pattern to be used as a base we will be using the layering system.

Benefits

The reason for using the layering system is

- It is highly pluggable ie. easy to replace a layer. This is a crucial part of the program as it has to be available on multiple platforms such as:
  - Mobile application layer
  - Web page layer
- Easier to do unit tests on as the other layers are easier to mock
- Groups have members that have different programming skills. With this method of developing each member can focus on the task that they can perform the best and maybe even enjoy.
- With this scheme an API is essentially created and can be used on any platform.

Concerns

Drawbacks for using layering for this project

- When changing lower layers of the system it impacts all layers above it. Therefore we should create solid base layers that do not need to be updated regularly.
- The above point also raises the point of maintenance. Once a lower layer is updated it reflects all the way to the top layers.
- Performance is impacted because of communication between layers but this simplifies the process of communicating between different top layers as there will be 2 different display or interaction layers.

Breakdown of different layers

1. **Presentation Layer** - This is the layer that contains physical pages and widgets through which the user can interact with the system.
   - Also known as the Front-end Layer
   - In simple terms, it creates and displays the User Interface.
   - Data that is shown is fetched from the Business Layer.
2. **Service Layer** - This serves as a bridge between the Presentation Layer and the Business Layer

- Application layer communicates with the Business Layer via this layer
- Third party services can be added in this layer i.e. Can be used to expose a web service API and return results as JSON/XML
- Data is obtained from the Business Layer

3. **Business Layer** - This layer contains all the business logic
    - CRUD(Create, Retrieve, Update, Delete) operations go here.
    - Structure of how users are linked to projects and papers are done in this layer
    - Authentication is done on this layer as users should be authenticated before being able to make changes to anything.
    - Superuser logic is also added into this layer.

4. **Data Access Layer** - This layer contains functionality for database connectivity
    - Exposes stored data to the Business Layer.
    - Responsible for querying the database

4.3 Architecture tactics or strategies

An architectural tactic is a way of satisfying a quality requirement measure by manipulating some aspects of a quality attribute model. It is also a design decision that impacts the control of a quality attribute response.

Performance

Latency is affected by a various level of the demand for resources, in our system this includes event arrival rates, the execution time that comes from the event arrivals and also the variability level of each event.

Tactics for managing system demands:

- Manage and control the event rate, this includes how many times events are generated.
- Bound Execution time, in our system we need a way of limiting the execution time spent on each event so that resources can free up to allow other events to run.
- Increase the computational efficiency of the algorithm, this will reduce the demand for the processor time and improve latency.
- Determine the appropriate scheduling policy for our system, the most suited for this would be semantic-importance-based scheduling so that the system focuses it resources on the most significant work at the most significant time. We can also use fairness-based scheduling which functions on the basis of FIFO.

Flexibility

The responsibility of the functions that will be created will impact the cost of making a change in the system, some changes can affect other function.

The tactics used to improve flexibilty include:

- Raising the abstraction level, the more general functions are the easier it will be to use them on different platform of our system.
- Localize services that will be used by majority of the users

Usability

If a user is able to use the weba based version of the system they should be able to use the mobile application too. The tactics use to assist usablity are:

- Maintaining the same user interface across all boards.
- Keep a verysimple user interface that does not require a lot from the users.

Integrability

The system being implemented will be usable from both the internet and mobile application.The tactics used for this will be:

- Layering, this will be implemented so that if a platform is already running we can simply build on top of it.

Maintainability

For maintainablity of the system, we need to build the system such that when making changes to fuctions,we do not have to modify other functions that are not directly affected by these changes.

Tactics used for maintainability are:

- Breaking the dependancy chain between functions, we will have to create a intermedairy to keep functions as independant as possible.
- Make data on the system self identifying, this just means tagging the data with identification, like descriptions to allow other developers to work on the system too.

Security

- Hiding information, here we devide responsibities of functions into two main categories: public and private. Public responsibities will vary depending on the type of user, they will also be accessable to both developers and users. System administrator will able to see the private functions.
- Detecting attacks using event logging, to know where threat came from and also if it was a user.
- The system will treat all external requests as possible threats and not allow them onto the system, this will narrow th direction of where we recieve the threats from making it easier to detect the threats to the system.
- Maintain and check for attack signatures, so that we will know exactly where the threats are coming from, so basically we are to keep all trails in the system(trace actions of the attacker).
- To recover from attacts, we will drop all requests on the system with the intention of basically refrshing the system
- After dealing the threats on the system, the system will have to be restored, and resume basic functionality.
- Authenticating all users before they log in is another tactic, after this based on the kind of user logging into the system, user authorization will be performed to make sure user doesnt have access to what they arent meant to have access to. The users will have to enter they passwords to authenticate them, to maintain the systems integrity all passwords will be hashed and stored in the database and the system will need a way of encrypting all data and communication links.

Monitorability

- Manage the input and output on the system.
- Have a internal monitor that is built-in that is always running to keep track and log all user activitys. We will use a tactic called record and playback which is basically recieving infomation across an interface and using that infomation as input to test harness.

Scalability

- When the system is under heavy workload, and cannot process all requests, to restore the speed old request that have been running for some time will be terminated and user will have to make another request.
- Increase disc space, this will help the system to handle more requests

4.4 Use Of Reference Architectures And Frameworks

Java-EE Reference Architecture

The aim of the java-EE platform is to provide developers with a powerful set of APIs while shortening development time,reducing application complexity, and improving application performance.

Application Server Component at first level of granularity.

Pattern: Layering

- Access layer: web container,message queues
- Business processes layer: EJB container
- Persistence layer: Persistence context

Tactics: Clustering for scalability,availability and reliability

- All layer elements
- JNDI repositories
- Load balancers, firewalls,DBs

JMX for management and monitoring

Application concepts

- None at this level of granularity

Application development concepts and constraints

- Pattern constraining application logic
- Concepts for application development
- Constraints for application development
- Strategies for applicaion logic

Quality Attributes of Java-EE

- Good Scalability
- Good Reliability
- Average Flexibility

- Average Performance
- Auditability is not directly supported
- Good Security
- Quite good Integrability

CodeIgniter

As we are using PHP for server work, we will be using CodeIgniter as a framework to handle the server side of the work. This is to empower the back-end of the model.

Some advantages of choosing this:

- The framework allows a model-view-controller system of handling the interaction for users
- It can be integrated into the lower layers of the Java EE reference system, allowing the mobile application and web pages
- It allows easier to work with mySQL for handling the users and their documents

MongoDB

Another possibility, on top of mySQL, would be to use MongoDB. There are a few reasons for this, such as:

- It handles documents better, which could be used for the research papers
- PHP does support MongoDB, so integrating it would be simple

4.5 Access and integration channels

Integration requirements

The main integration channel that will be used in the system is the database which will allow users to access to external systems either via the mobile application or the browser client. An EAI system can be used to participate in concurrent integration operations at any one time. Security plays a vital role in the integration channels as unauthorised authors may not see, contribute or change documents while heads of departments or supervisors have the authority to see the papers being worked on. The integration channel has to be reliable as papers that are being worked on are of utmost importance and would be detrimental to all those involved in the papers if something were to happen to them. The system must keep all documents secure and accounted for.

Access channels

There will be multiple access channels for the proposed system. Mobile/android application clients will be used to fetch data(such as the deadline dates) and integrate them into mobile device for users to easily be able to identify when their deadlines are for. The mobile application will also allow a user to search for and identify the type of paper(journal, conference paper or book) that is being worked on as well as how many papers a user may be working on. The mobile application will also allow users to see the status of a paper(still working on it, submitted and waiting for feedback, rejected, accepted, published or improving after rejection). Web services will be used in order to store all the users and their details as well as all the papers that are being worked on by any particular user. These web services will be set up such that 100 users can concurrently access the server without any difficulties. The system will also make due with browser clients that will have all the same functionality as the mobile client and allow any user to access the system where ever they may be. Both the mobile application and browser client will interact with the web server in order to gain access to any information it might need.

4.6 Technologies

Deciding on the technologies to be used with this project is vital, though there are many ways in which to perform the same task, choosing the right tool for the job makes it that much easier to achieve one's goal. The project's technologies can be subdivided into 3 major categories, namely: server-side back end, website front end and finally the Android mobile application.

Website Front End Technologies

- HTML 5 - Defines the structure and content of the website.
- CSS3 - Styles the website as well as provide some of the animations to be used.
- SCSS - A flavour of SASS, though optional since CSS3 is present but should be utilized and converted to CSS3 when deploying the project.
- Javascript - Provides functionality for the website, will be used with JSON using AJAJ to dynamically update the page while fetching from the server.
- jQuery library - Will be utilized to simplify the Javascript being used.

These form the standard modern day HTML5 web technologies used for the front end.

Server-side Back End Technologies

- Linux - Operating system that will be used to host the website and back end of the service.
- Apache - The web server.

- MySQL - SQL database used to store data and will be queried for information.
- PHP - Server scripting language used to interact.
- MongoDB - NoSQL database which might be used on top of MySQL.
- CodeIgniter - Lightweight MVC framework for rapid development of web applications.

LAMP (Linux, Apache, MySQL, PHP) will be the back end of choice. We require reliability, performance and security. LAMP is a tried and tested technologies with fewer vulnerabilities compared to newer back end stacks. This is of great interest when security is key. For the system to reduce cost, students will be used to maintain it. Since most students are familiar with LAMP technologies, it is thus even better suited for the job.

Android Mobile Application Technologies

- Java EE - Programming language with APIs to be used.
- GSON - Library for converting between Java Objects and JSON representations
- RoboGuice - Dependency Injection framework for Android
- Gradle - Build system to allow for more robust, well defined builds

There are 3 possible types of Android applications which can be built. For this project a native Android application will be built. Native applications offer the best performance and the best user interface compared to the other types. This is especially important when scalability is a factor and presenting the content correctly. As this will be a native application, Java will be used as the programming language.

The GSON library will be used to convert between Java objects and JSON representations and visa versa which will be useful when communicating with the web server on campus.

5 Functional requirements and application design

This section discusses the application functionality required by users (and other stakeholders).

5.1 Use case prioritization

Critical

- The system needs to be able to handle 100 users concurrently and at the same time.
- Only University of Pretoria staff members may register to become users.
- Users are only allowed to see publications that they are involved in.
- There should be a a superuser or root account for the Head of Department, the superuser should be able to see all research papers/projects that is on the system.
- Users and authors should be able to upload their documents to the research project.

Important

- Users(UP staff) can add and remove authors.
- Users(UP staff) decide when the deadline of a research paper is.
- Users(UP staff) can terminate a research paper and revive a terminated research paper.
- A user can edit the details about a research paper that they are involved in.
- User profiles should show the total accumulated units.
- The system has to log everything.
- Users must be able to enter historical data (previously completed research papers).
- Every research paper only has one primary author.
- Users(UP staff) decide when the deadline of a research project is.
- Users(UP staff) can terminate a research project and revive a terminated research project.
- A user can edit the details about a research project that they are involved in.
- User profiles should show the total accumulated units.
- The system has to log everything.
- Users must be able to enter historical data (previously completed research projects).

- Every research project only has one primary author.
- Co-authors are added and sequenced.

Nice-To-Have

- There should be a checkbox that defaults users as authors for research papers that they administrate, if they are just administrating they should explicitly uncheck that they are not an author.
- There should be an indication of progress on research papers (progress bar/percentage indication of work done)
- Users should be able to search for authors if they have been stored in the system.

5.2 Use case/Service Contracts

Critical

Handle 100 users concurrently

- Pre-conditions
  - System needs to be online.
  - Users need to connect to the system.
  - System needs to have concurrency controls to help aid 100 users at the maximum.
- Post-conditions
  - System can successfully maintain the concurrent use of 100 users.

Only UP staff members may register to become users

- Pre-conditions
  - User needs to be registered at the University of Pretoria.
  - User needs a valid staff number from the University of Pretoria.
- Post-conditions
  - User that is now registered on the system may now add research projects and so forth.

The superuser account

- Pre-conditions

  - User needs to be a staff member at the University Of Pretoria.
  - User needs to be the head of the department.
- Post-conditions
  - Super user can see all research projects on the system.
  - Super user has control of the whole system.

Users and authors should be able to upload their documents to the research project

- Pre-conditions
  - User should be registered on the system
  - User should be involved on the research project they want to upload a document to.
- Post-conditions
  - User is able to upload a document to the research project for the rest of the users/authors to see.

Important

Users can add and remove authors

- Pre-condition
  - User has to be registered on the system.
  - User has to be the primary author on a research project that they want to add and remove other authors to and from.
- Post-condition
  - User can add and remove authors from the research project that they are the primary author on.

Users decide when the deadline of a research project is.

- Pre-conditions
  - Users need to be registered on the system.
  - Users need to be a primary author on a research project for which they want to set a deadline.
  - Deadline cannot be in the past.
- Post-conditions
  - Deadline is set on the research project.

  - All co-authors are notified via email what the deadline on that research project is.

Users can terminate and revive a research project

- Pre-conditions
  - User needs to be registered on the system.
  - User needs to be the primary author on the research project which needs to be terminated or revived.
- Post-conditions
  - The selected research project is terminated or revived.
  - All co-authors are notified via email that the research project is terminated or revived.
- Exceptions
  - Superuser also has control over any users research project.

Users can edit details about a research paper they are involved in

- Pre-conditions
  - User has to be registered on the system.
  - User has to be added to the research project or be the primary author.
  - There has to be a research paper uploaded to the system on which the details need to be changed.
- Post-conditions
  - The details of the research paper can be edited by the author.
  - All co-authors can see the edited details of the research paper.

User profiles should show the total accumulated units

- Pre-conditions
  - User has to be registered on the system.
  - User needs to be involved on a research project.
- Post-conditions
  - Users profile will show the total accumulated units of contributions.

The system has to log everything

- Pre-conditions

  - System has to be online.

- Post-conditions
  - Everything that any user does on the system is added to a log.

Users must be able to enter historical data

- Pre-conditions
  - User must be registered on the system.
  - User must be involved with a research project.
  - There must be a previously completed research paper/project on which historical data needs to be entered.
- Post-conditions
  - Historical data is added to the previously completed research paper/project.
  - All co-authors can see this historical data that has been added about the previously completed research paper/project.

Every research project has only one primary author

- Pre-conditions
  - User needs to be registered on the system.
  - User has to be the creator of the research project or superuser.
- Post-conditions
  - User becomes the primary author on the research project.
- Exceptions
  - User that used to be the primary author hands the research project over to one of the co-authors to become the primary author.

Co-authors are added and sequenced

- Pre-conditions
  - Co-authors need to be registered on the system.
  - The primary author needs to specify which authors to add to the research project.
- Post-conditions
  - Co-authors are added to the research project and sequenced in a hierarchy.

Nice-to-Have

Users are authors by default for research projects that they administrate. If the user is only an administrator they should explicitly mark them self as administrator only

- Pre-conditions
  - User needs to be registered on the system.
  - User needs to be an administrator of a research project or
  - User needs to be the primary author of a research project.
- Post-conditions
  - User can remove themselves as an author and only be an administrator of the research project.
- Exceptions
  - Superuser is an administrator on all research project.

Indication of progress on research projects

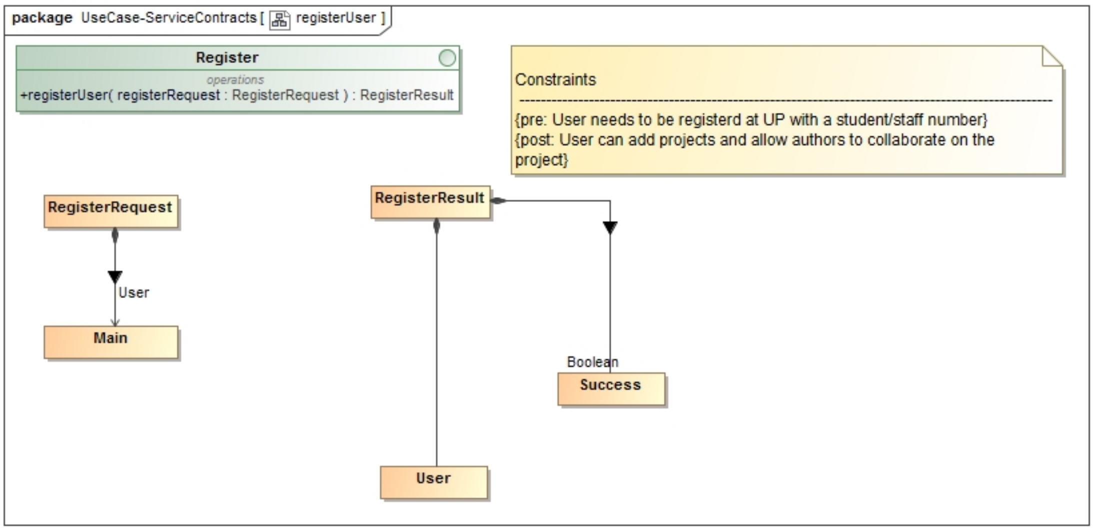
- Pre-conditions
  - The system needs to have a research project that has active authors.
- Post-conditions
  - There is a progress bar of how much work is done on that research project before the research project is completed.

Users should be able to search for other authors

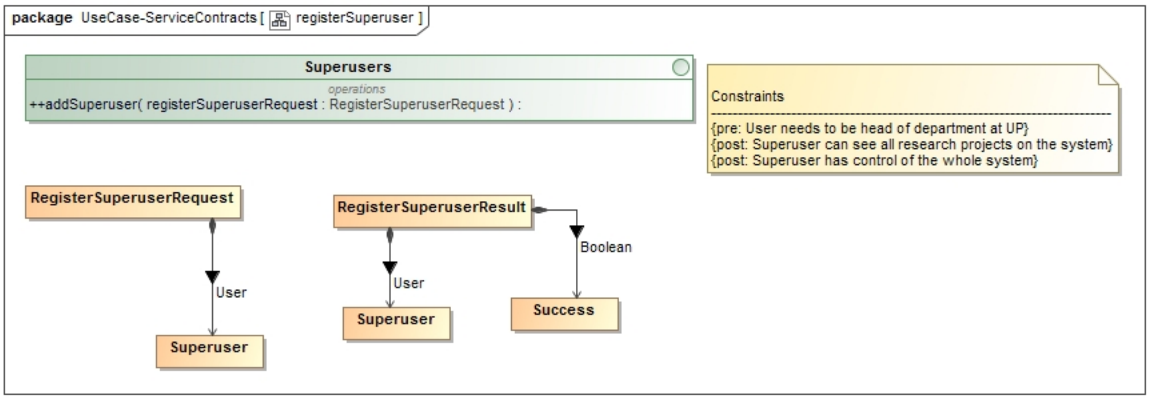
- Pre-conditions
  - User should be registered on the system
  - Other users should be registered on the system
- Post-conditions
  - User has the ability to search in the system for other users/authors.

Request and Results Data Structures

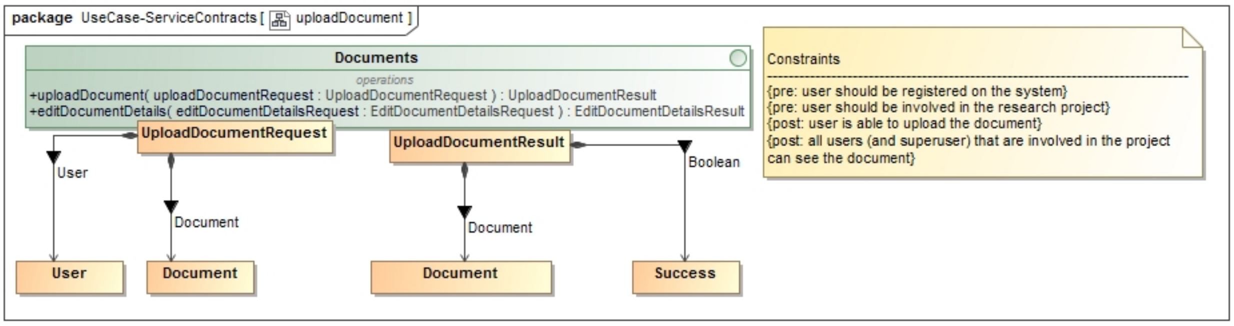
Service Contracts


package UseCase-ServiceContracts [registerUser]
Register
operations
+registerUser(registerRequest : RegisterRequest) : RegisterResult
Constraints
{pre: User needs to be registerd at UP with a student/staff number}
{post: User can add projects and allow authors to collaborate on the project}
RegisterRequest
User
Main
RegisterResult
Boolean
Success
User

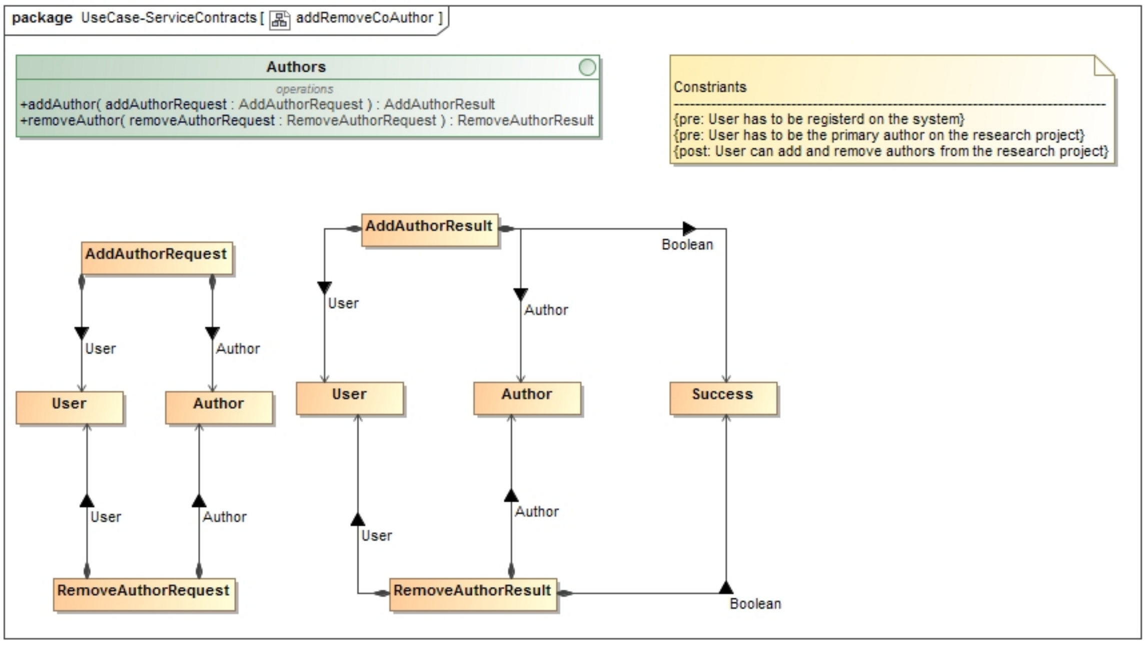
Only UP staff members may register to become users


package UseCase-ServiceContracts [registerSuperuser]
Superusers
operations
++addSuperuser(registerSuperuserRequest : RegisterSuperuserRequest) :
Constraints
{pre: User needs to be head of department at UP}
{post: Superuser can see all research projects on the system}
{post: Superuser has control of the whole system}
RegisterSuperuserRequest
User
Superuser
RegisterSuperuserResult
User
Superuser
Boolean
Success

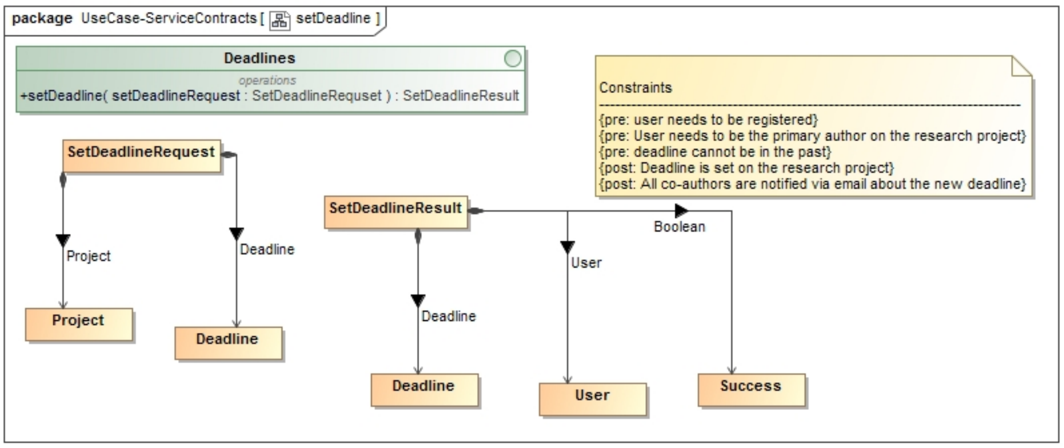
Creating the super user account


package UseCase-ServiceContracts [uploadDocument]
Documents
operations
+uploadDocument(uploadDocumentRequest : UploadDocumentRequest) : UploadDocumentResult
+editDocumentDetails(editDocumentDetailsRequest : EditDocumentDetailsRequest) : EditDocumentDetailsResult
UploadDocumentRequest
UploadDocumentResult
User
Document
Document
Boolean
User
Document
Document
Success
Constraints
{pre: user should be registered on the system}
{pre: user should be involved in the research project}
{post: user is able to upload the document}
{post: all users (and superuser) that are involved in the project can see the document}

Users should be able to upload their documents


package UseCase-ServiceContracts [addRemoveCoAuthor]
Authors
operations
+addAuthor(addAuthorRequest : AddAuthorRequest) : AddAuthorResult
+removeAuthor(removeAuthorRequest : RemoveAuthorRequest) : RemoveAuthorResult
Constriants
{pre: User has to be registerd on the system}
{pre: User has to be the primary author on the research project}
{post: User can add and remove authors from the research project}
AddAuthorResult
Boolean
AddAuthorRequest
User
Author
User
Author
User
Author
User
Author
Success
User
Author
User
Author
RemoveAuthorRequest
RemoveAuthorResult
Boolean

Users can add and remove authors

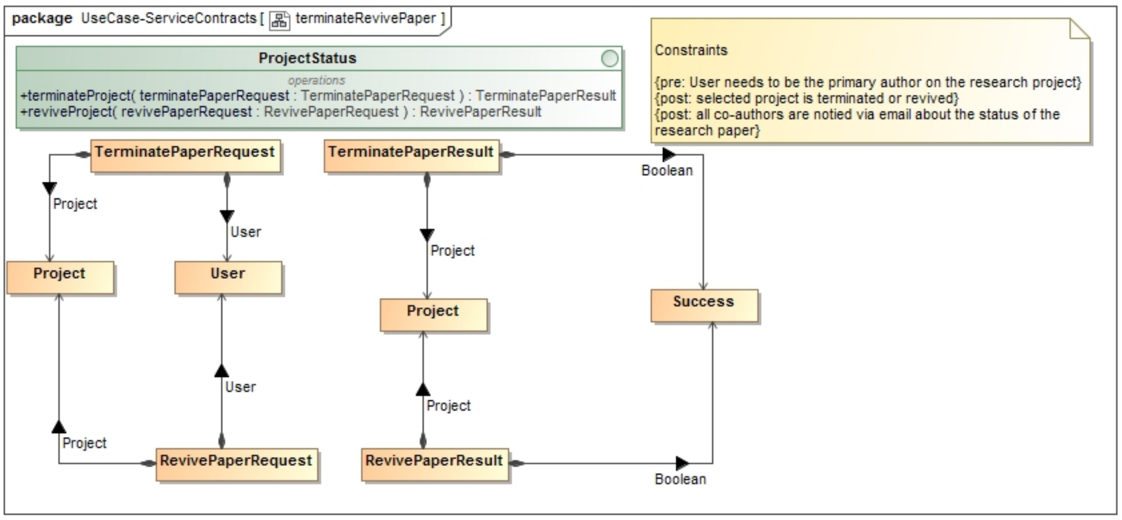


+setDeadline(setDeadlineRequest : SetDeadlineRequset) : SetDeadlineResult

Constraints

{pre: user needs to be registered}
{pre: User needs to be the primary author on the research project}
{pre: deadline cannot be in the past}
{post: Deadline is set on the research project}
{post: All co-authors are notified via email about the new deadline}

Users decide the deadline of a research project

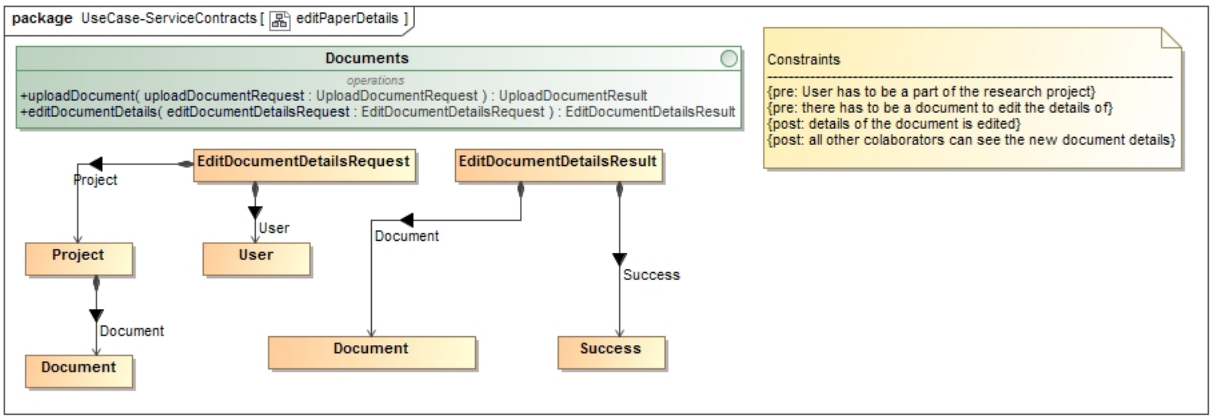


+terminateProject(terminatePaperRequest : TerminatePaperRequest) : TerminatePaperResult
+reviveProject(revivePaperRequest : RevivePaperRequest) : RevivePaperResult

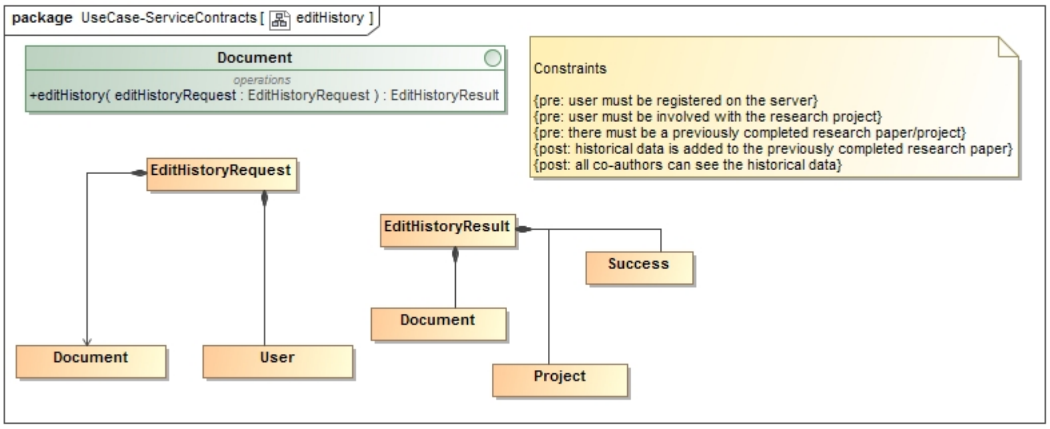
Constraints

{pre: User needs to be the primary author on the research project}
{post: selected project is terminated or revived}
{post: all co-authors are notied via email about the status of the research paper}

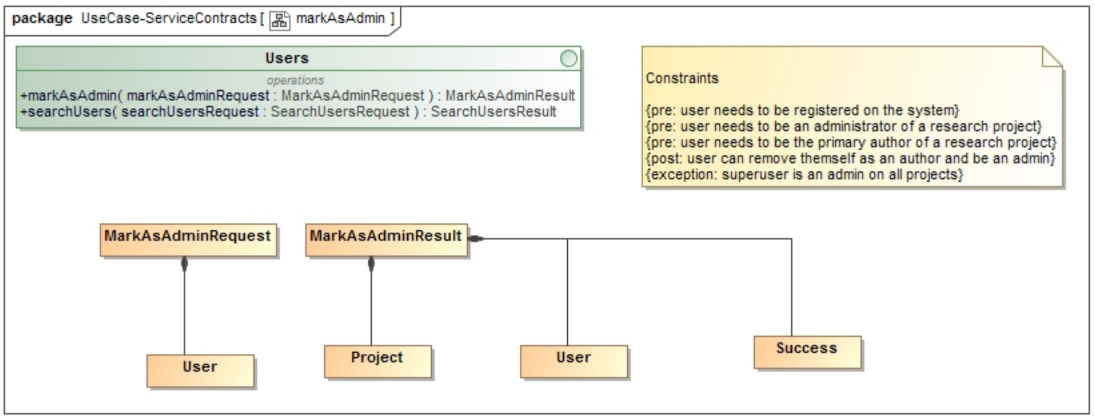
Users can terminate and revive a research project


package UseCase-ServiceContracts [editPaperDetails]
Documents
operations
+uploadDocument(uploadDocumentRequest : UploadDocumentRequest) : UploadDocumentResult
+editDocumentDetails(editDocumentDetailsRequest : EditDocumentDetailsRequest) : EditDocumentDetailsResult
Constraints
{pre: User has to be a part of the research project}
{pre: there has to be a document to edit the details of}
{post: details of the document is edited}
{post: all other colaborators can see the new document details}
EditDocumentDetailsRequest
EditDocumentDetailsResult
Project
User
Document
Success
Project
User
Document
Document
Success
Document

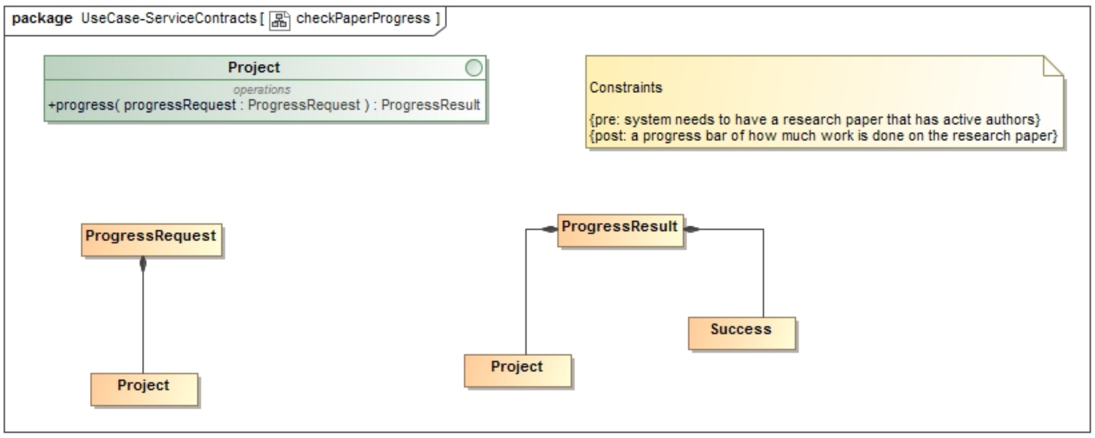
Users can edit details about a research paper they are involved in


package UseCase-ServiceContracts [editHistory]
Document
operations
+editHistory(editHistoryRequest : EditHistoryRequest) : EditHistoryResult
Constraints
{pre: user must be registered on the server}
{pre: user must be involved with the research project}
{pre: there must be a previously completed research paper/project}
{post: historical data is added to the previously completed research paper}
{post: all co-authors can see the historical data}
EditHistoryRequest
EditHistoryResult
Success
Document
Document
User
Project

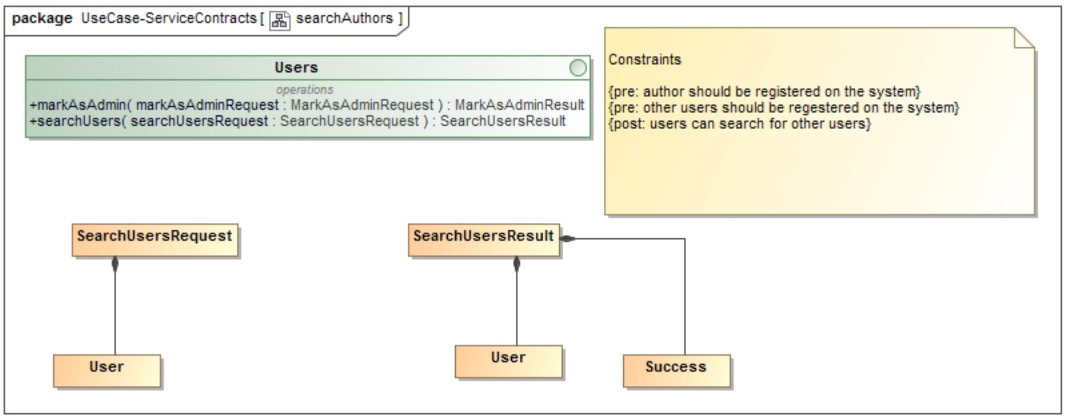
Users must be able to enter historical data


package UseCase-ServiceContracts [markAsAdmin]
Users
operations
+markAsAdmin(markAsAdminRequest : MarkAsAdminRequest) : MarkAsAdminResult
+searchUsers(searchUsersRequest : SearchUsersRequest) : SearchUsersResult
Constraints
{pre: user needs to be registered on the system}
{pre: user needs to be an administrator of a research project}
{pre: user needs to be the primary author of a research project}
{post: user can remove themself as an author and be an admin}
{exception: superuser is an admin on all projects}
MarkAsAdminRequest
MarkAsAdminResult
User
Project
User
Success

Users can be marked as administrators only


package UseCase-ServiceContracts [checkPaperProgress]
Project
operations
+progress(progressRequest : ProgressRequest) : ProgressResult
Constraints
{pre: system needs to have a research paper that has active authors}
{post: a progress bar of how much work is done on the research paper}
ProgressRequest
ProgressResult
Project
Project
Success

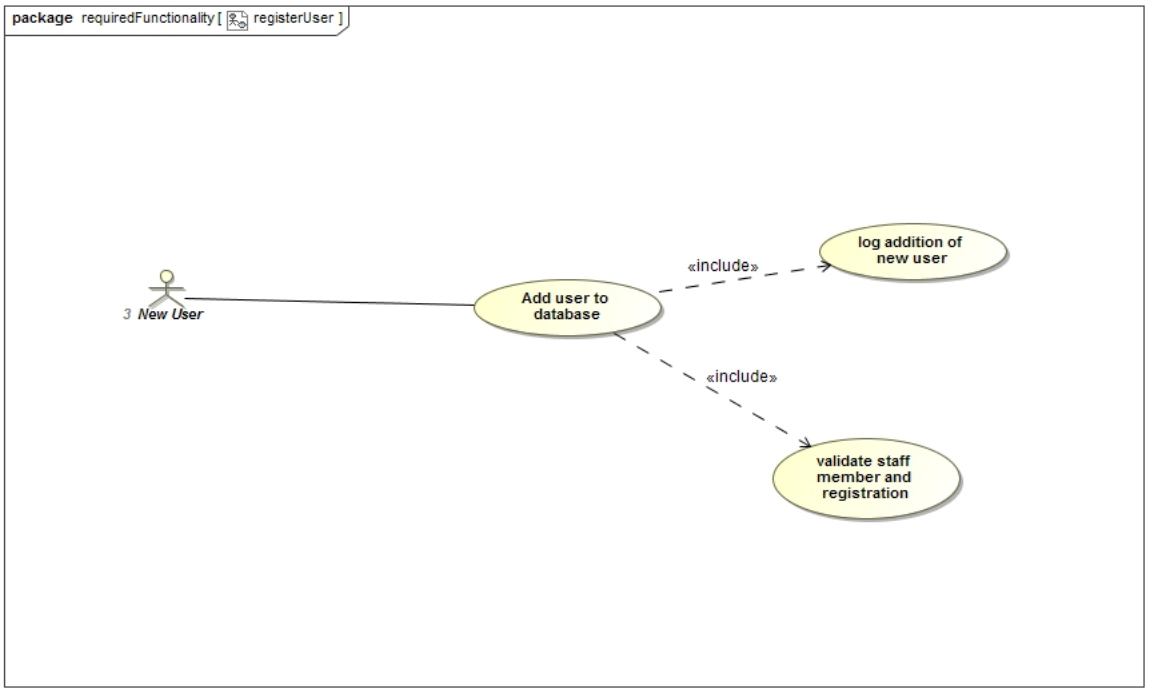
Indication of progress on research project


package UseCase-ServiceContracts [searchAuthors]
Users
operations
+markAsAdmin(markAsAdminRequest : MarkAsAdminRequest) : MarkAsAdminResult
+searchUsers(searchUsersRequest : SearchUsersRequest) : SearchUsersResult
Constraints
{pre: author should be registered on the system}
{pre: other users should be regestered on the system}
{post: users can search for other users}
SearchUsersRequest
SearchUsersResult
User
User
Success

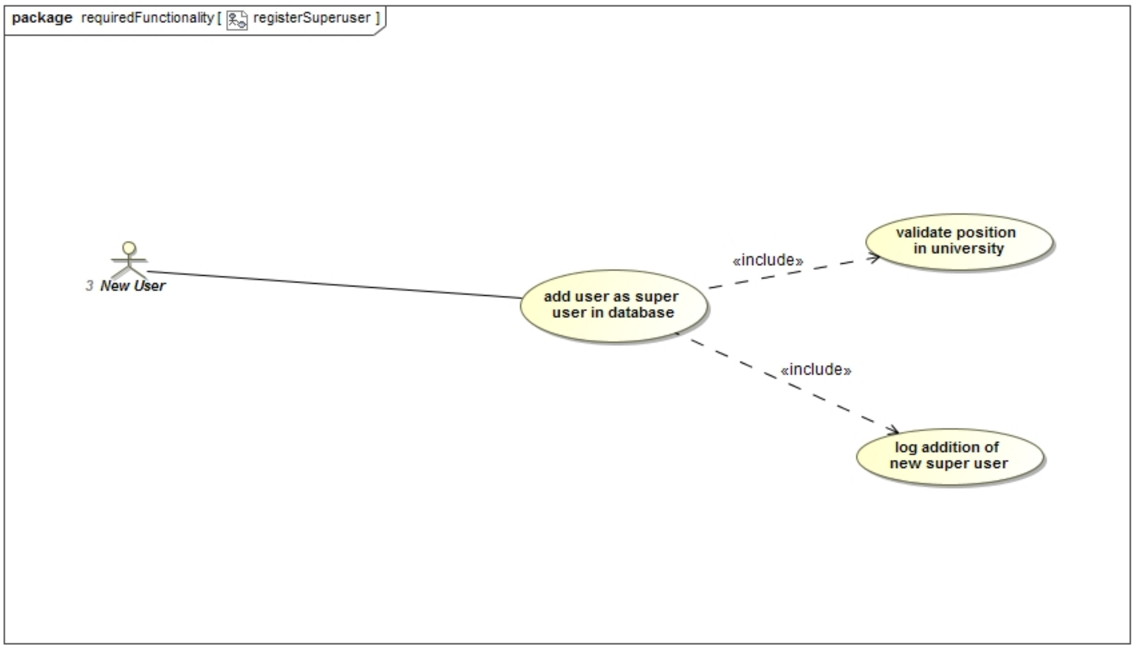
Users should be able to search for other authors

5.3 Required functionality

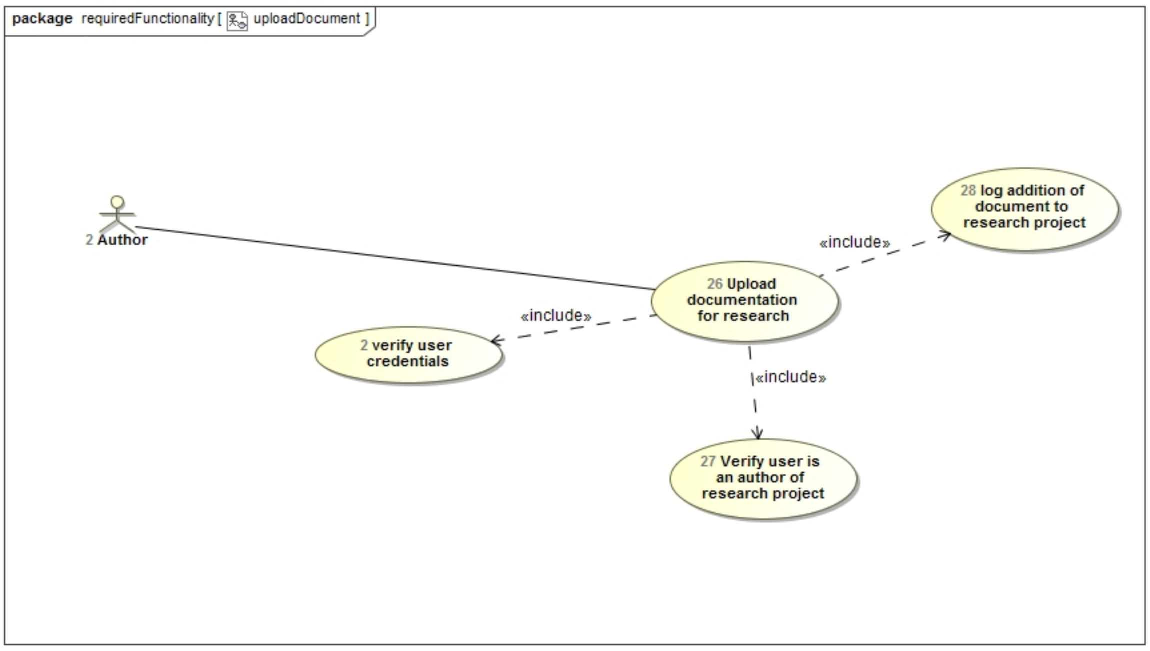
Critical use cases


package requiredFunctionality [registerUser]
3 New User
Add user to database
«include»
log addition of new user
«include»
validate staff member and registration

Only UP staff members may register to become users

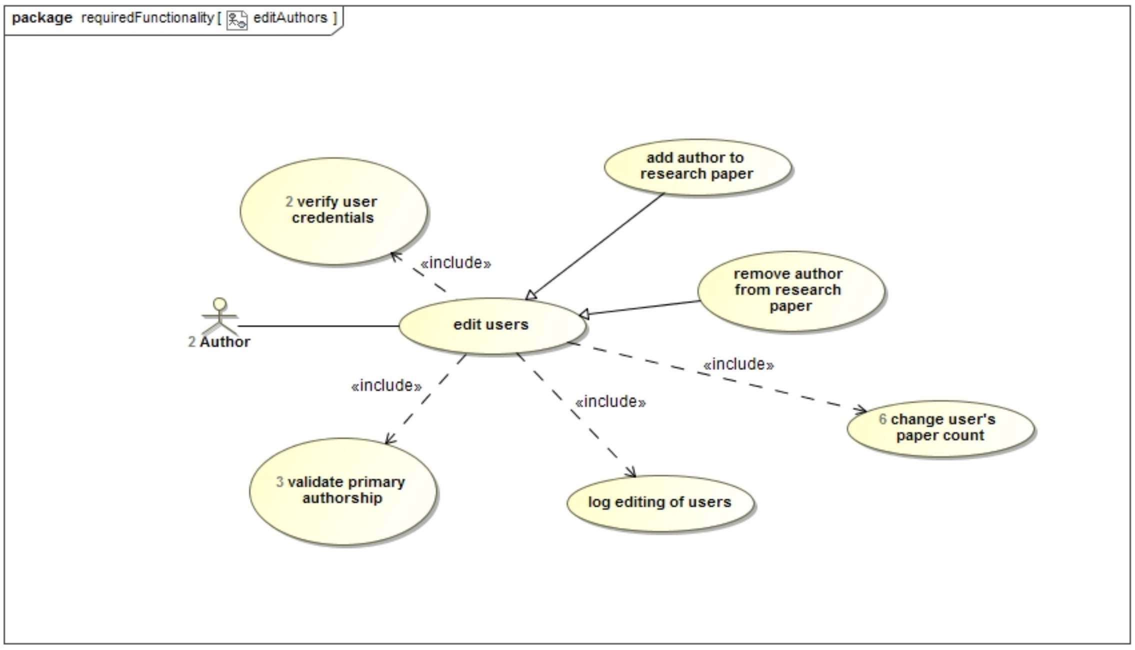

package requiredFunctionality [registerSuperuser]
3 New User
add user as super user in database
«include»
validate position in university
«include»
log addition of new super user

Creating the super user account

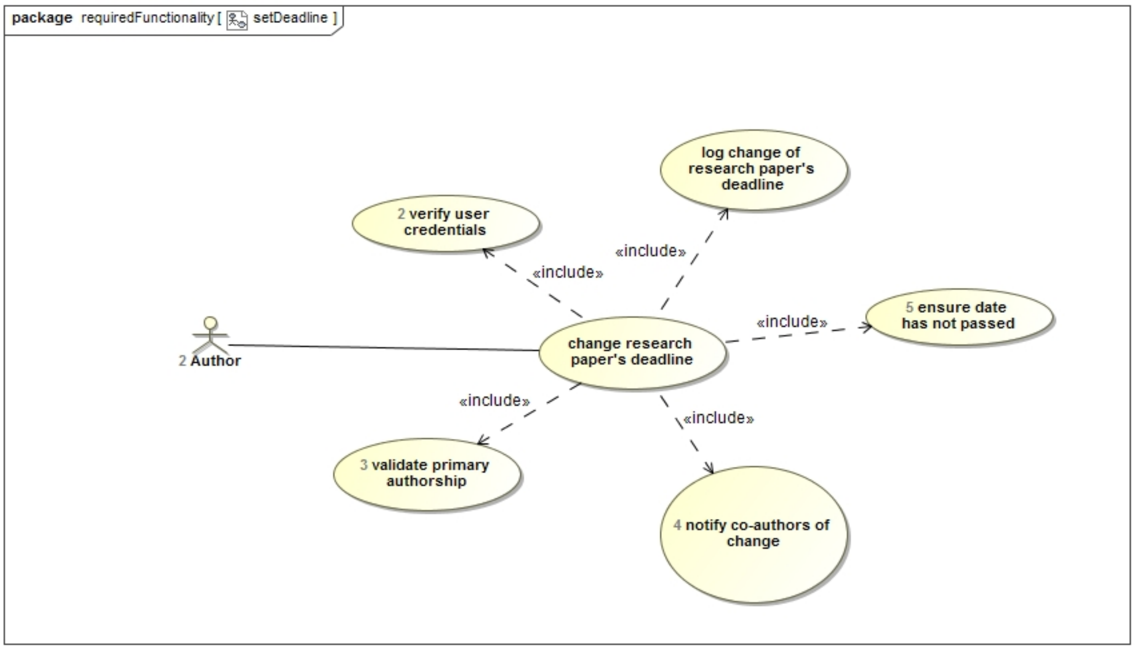

package requiredFunctionality [uploadDocument]
2 Author
26 Upload documentation for research
28 log addition of document to research project
2 verify user credentials
27 Verify user is an author of research project
«include»
«include»
«include»

Users should be able to upload their documents

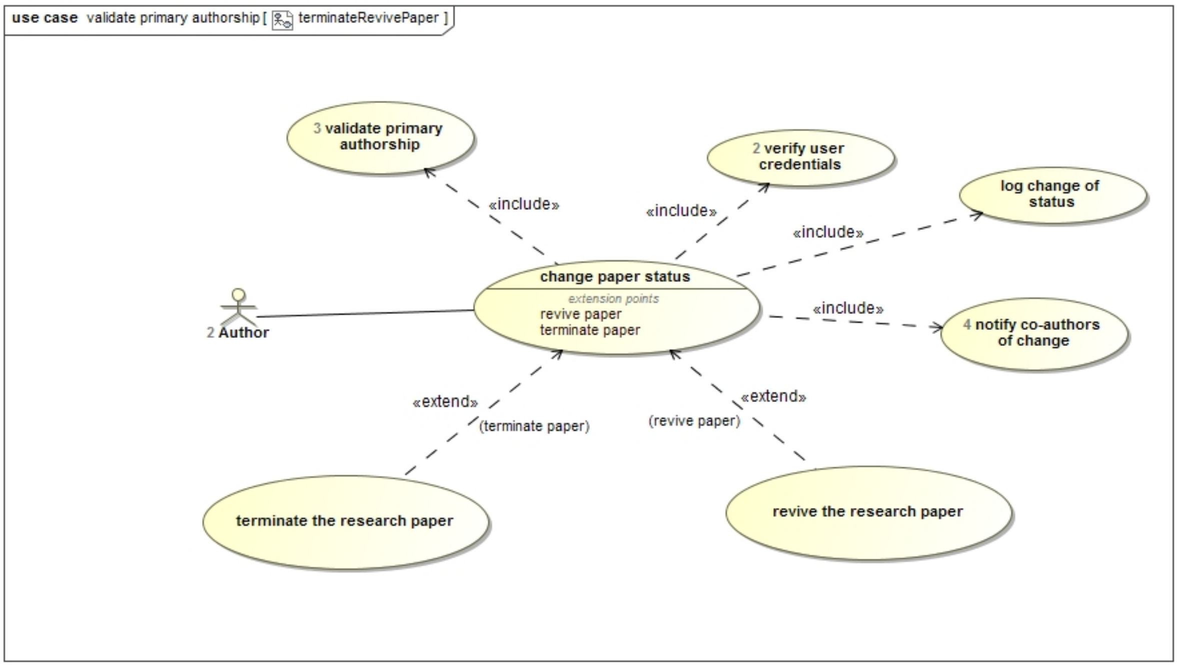
Important use cases


package requiredFunctionality [editAuthors]
add author to research paper
2 verify user credentials
remove author from research paper
«include»
2 Author
edit users
«include»
«include»
«include»
6 change user's paper count
3 validate primary authorship
log editing of users

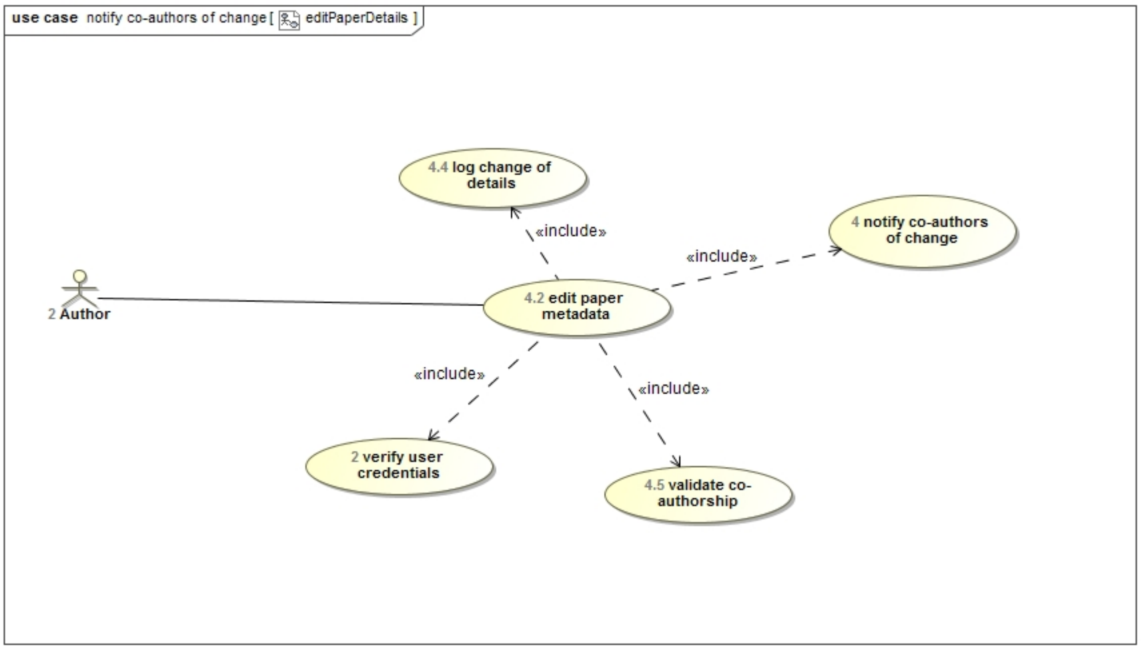
Users can add and remove authors


package requiredFunctionality [setDeadline]
log change of research paper's deadline
2 verify user credentials
«include»
«include»
«include»
5 ensure date has not passed
2 Author
change research paper's deadline
«include»
«include»
3 validate primary authorship
4 notify co-authors of change

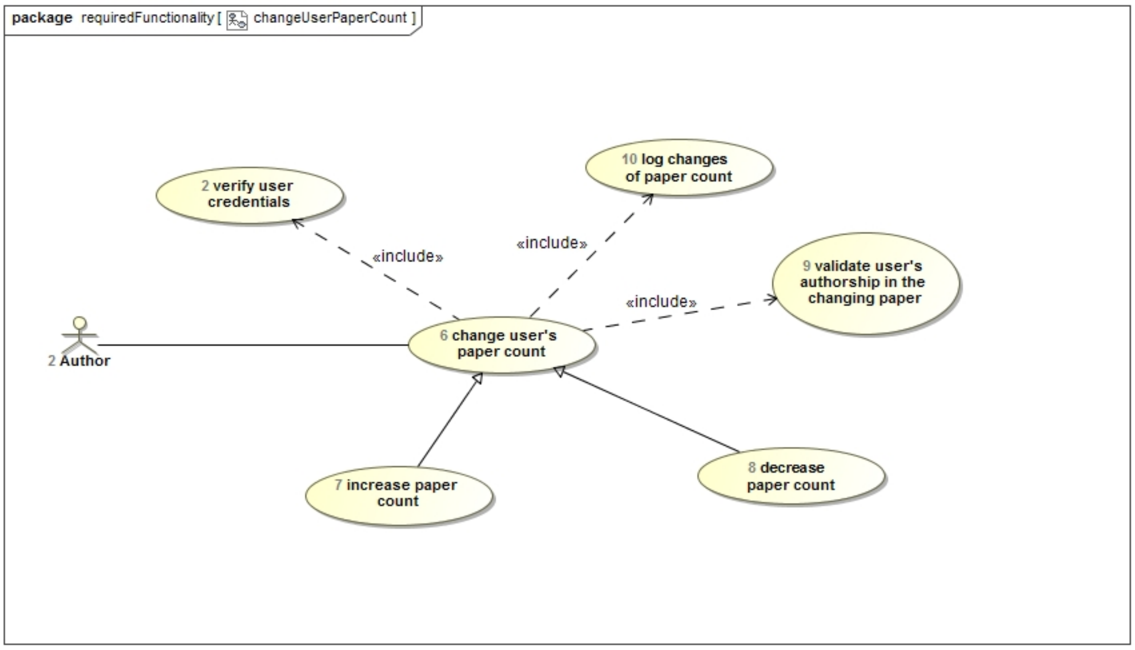
Users decide the deadline of a research project


use case validate primary authorship [terminateRevivePaper]
3 validate primary authorship
2 verify user credentials
log change of status
«include»
«include»
«include»
«include»
2 Author
change paper status
extension points
revive paper
terminate paper
4 notify co-authors of change
«extend»
(terminate paper)
«extend»
(revive paper)
terminate the research paper
revive the research paper

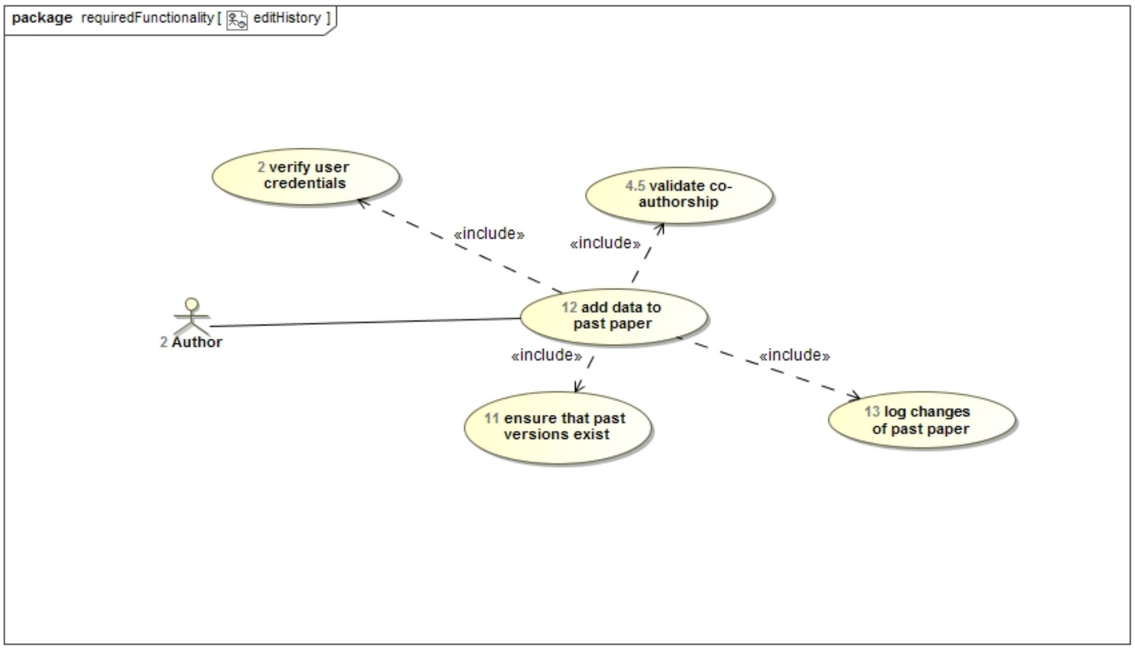
Users can terminate and revive a research project


use case notify co-authors of change [editPaperDetails]
4.4 log change of details
«include»
4 notify co-authors of change
«include»
2 Author
4.2 edit paper metadata
«include»
«include»
2 verify user credentials
4.5 validate co-authorship

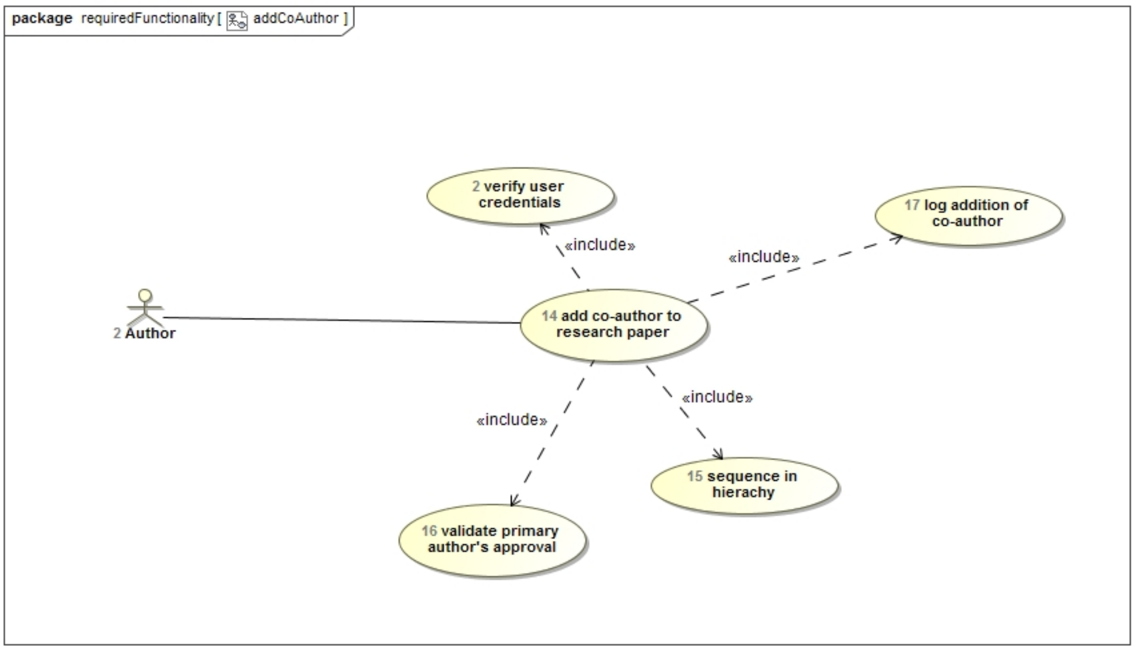
Users can edit details about a research paper they are involved in


package requiredFunctionality [changeUserPaperCount]
2 verify user credentials
10 log changes of paper count
«include»
«include»
9 validate user's authorship in the changing paper
«include»
2 Author
6 change user's paper count
7 increase paper count
8 decrease paper count

User profiles should show the total accumulated units

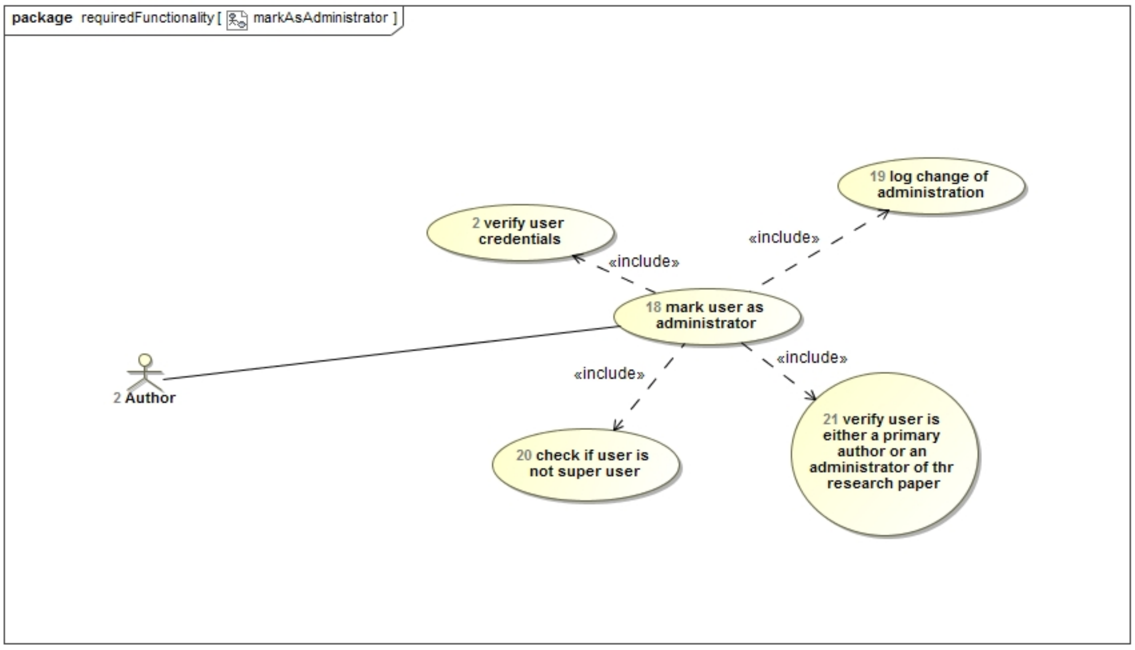

package requiredFunctionality [editHistory]
2 verify user credentials
4.5 validate co-authorship
«include»
«include»
2 Author
12 add data to past paper
«include»
«include»
11 ensure that past versions exist
13 log changes of past paper

Users must be able to enter historical data

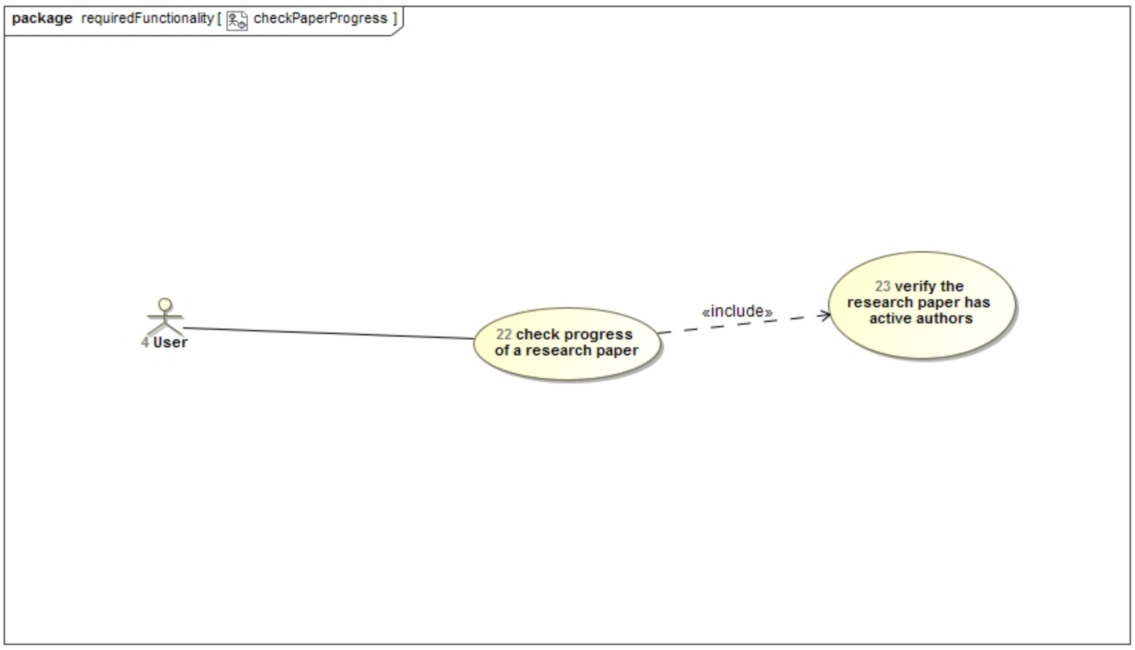

package requiredFunctionality [addCoAuthor]
2 Author
14 add co-author to research paper
2 verify user credentials
17 log addition of co-author
15 sequence in hierachy
16 validate primary author's approval
«include»
«include»
«include»
«include»

Co-authors are added and sequenced

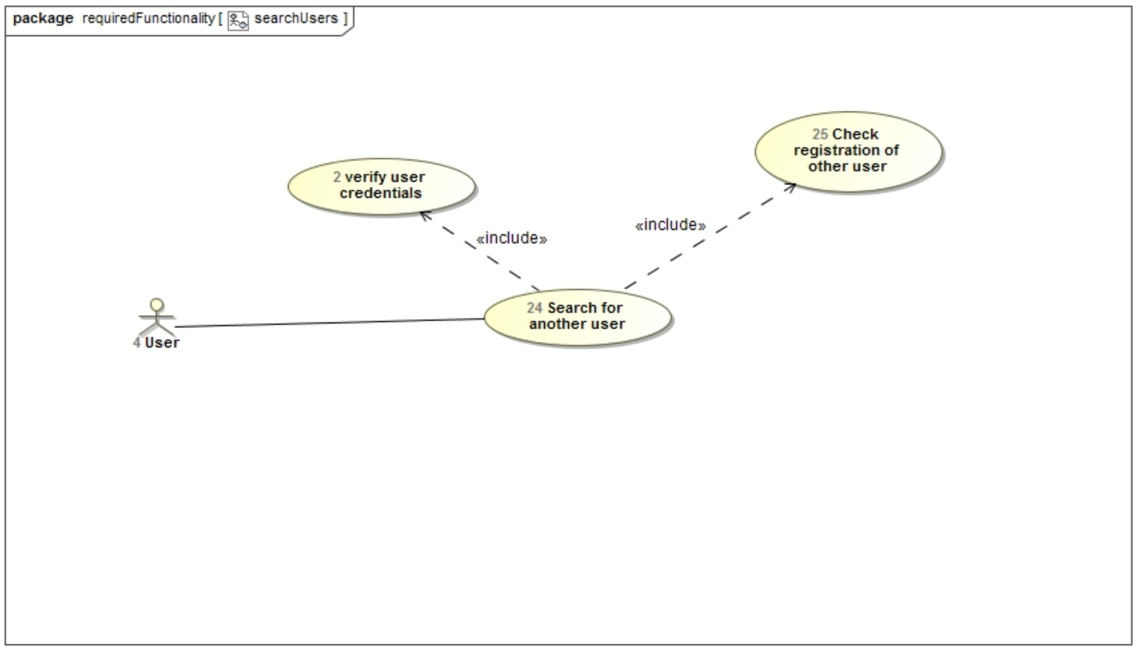
Nice-to-have use cases


package requiredFunctionality [markAsAdministrator]
2 Author
18 mark user as administrator
2 verify user credentials
19 log change of administration
20 check if user is not super user
21 verify user is either a primary author or an administrator of thr research paper
«include»
«include»
«include»
«include»

Users can be marked as administrators only


package requiredFunctionality [checkPaperProgress]
4 User
22 check progress of a research paper
«include»
23 verify the research paper has active authors

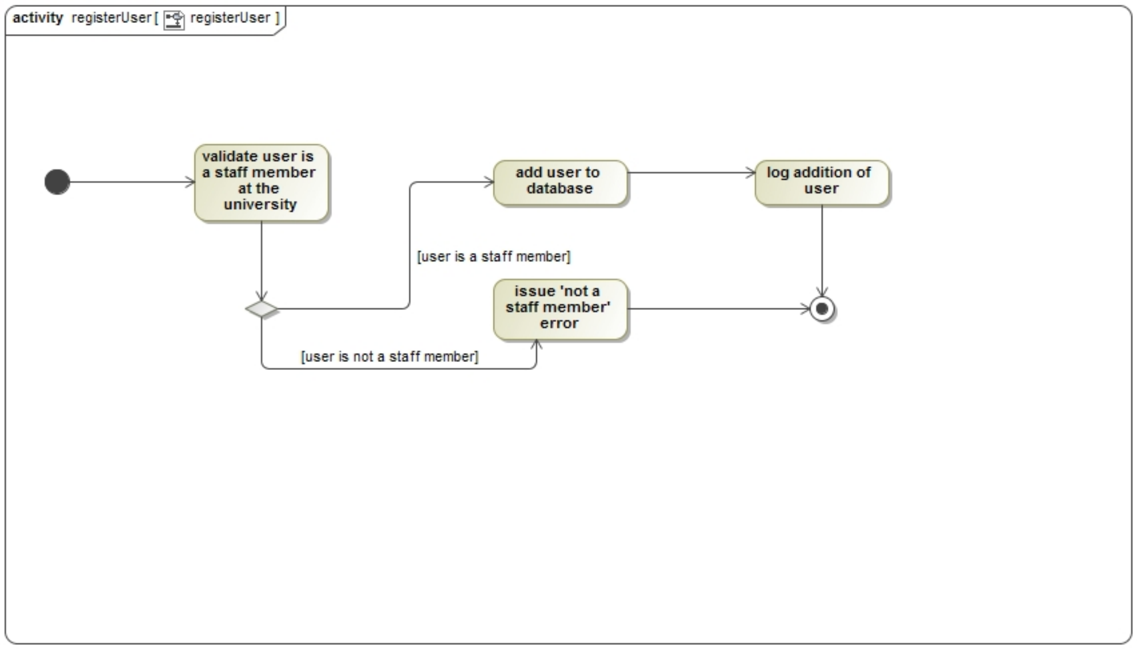
Indication of progress on research project


package requiredFunctionality [searchUsers]
2 verify user credentials
25 Check registration of other user
«include»
«include»
24 Search for another user
4 User

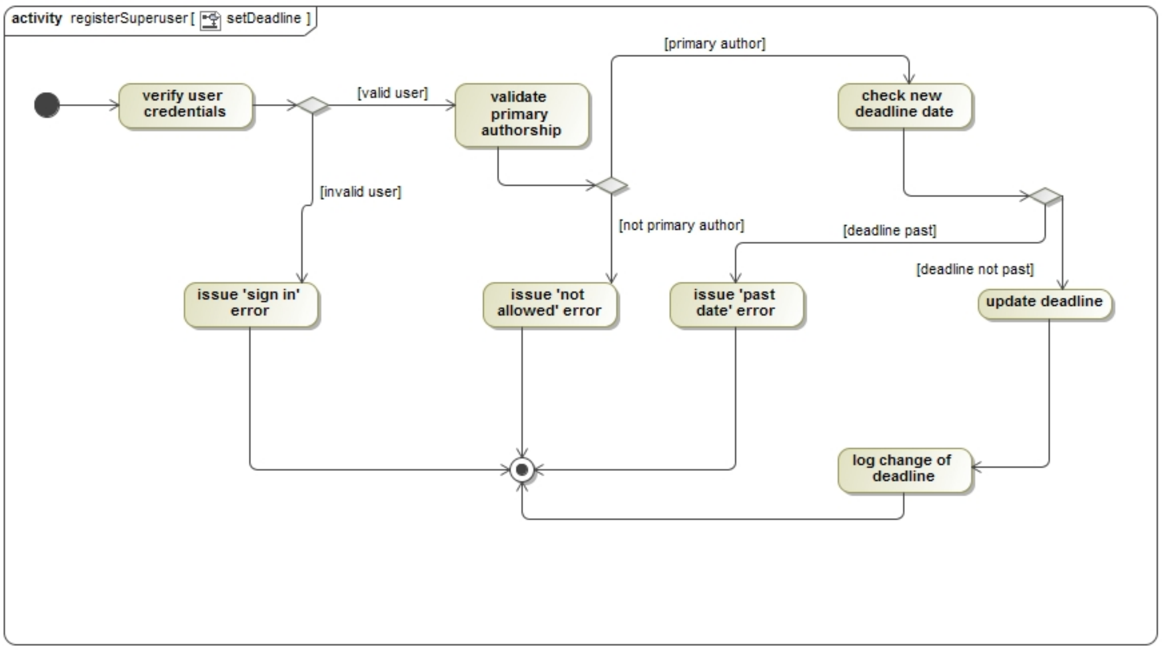
Users should be able to search for other authors

5.4 Process specifications

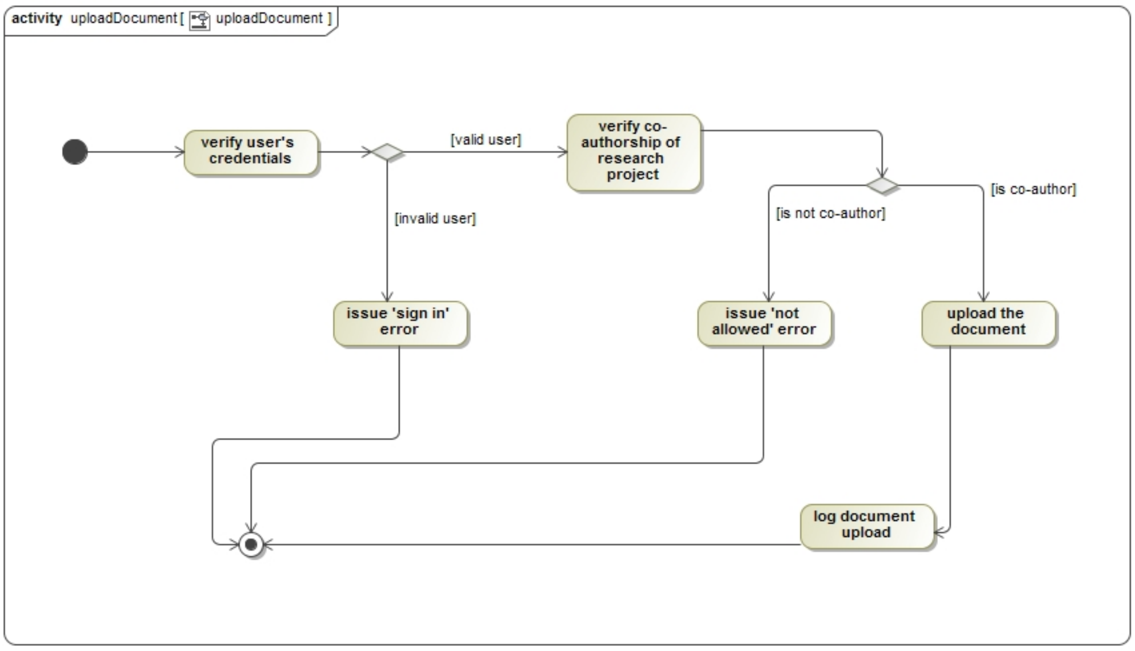
Critical use cases


activity registerUser [registerUser]
validate user is a staff member at the university
add user to database
log addition of user
[user is a staff member]
issue 'not a staff member' error
[user is not a staff member]

Only UP staff members may register to become users

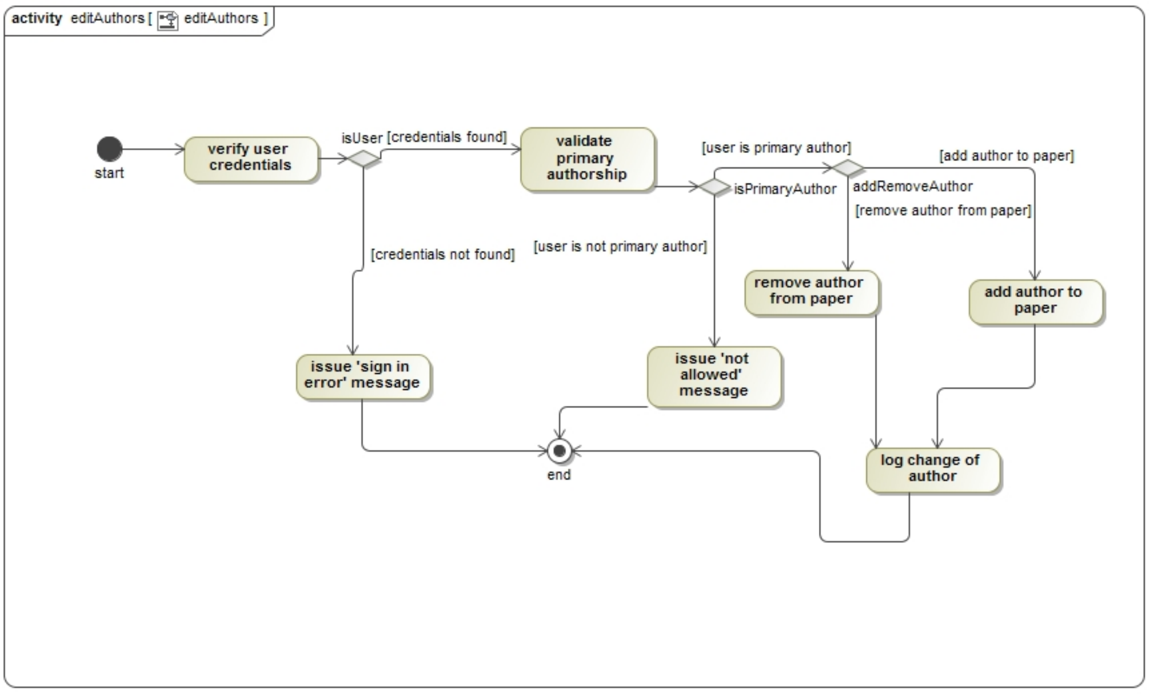


Creating the super user account

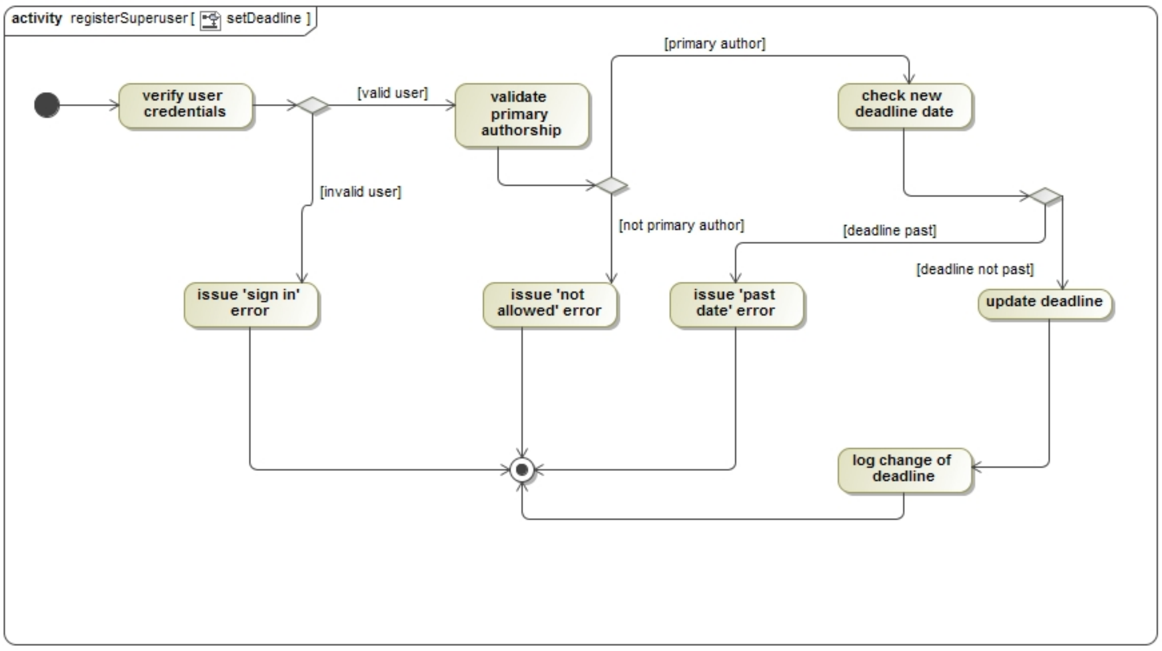

activity uploadDocument [uploadDocument]
verify user's credentials
[valid user]
verify co-authorship of research project
[invalid user]
[is not co-author]
[is co-author]
issue 'sign in' error
issue 'not allowed' error
upload the document
log document upload

Users should be able to upload their documents

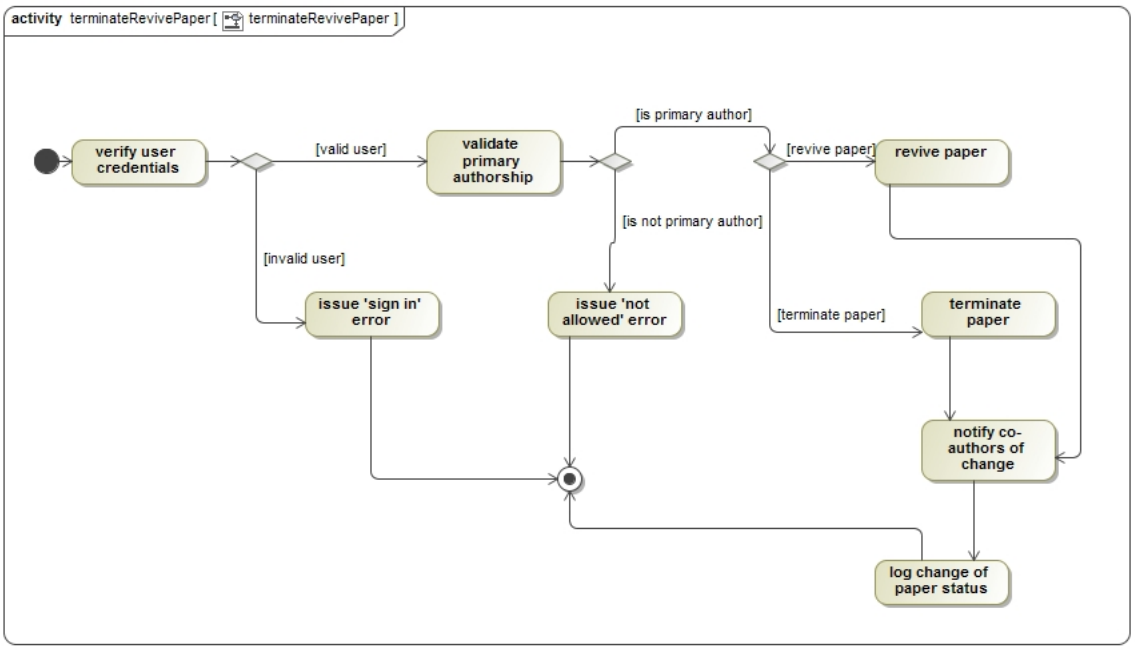
Important use cases


activity editAuthors [editAuthors]
start
verify user credentials
isUser [credentials found]
validate primary authorship
[user is primary author]
isPrimaryAuthor
[add author to paper]
addRemoveAuthor
[remove author from paper]
[credentials not found]
[user is not primary author]
remove author from paper
add author to paper
issue 'sign in error' message
issue 'not allowed' message
end
log change of author

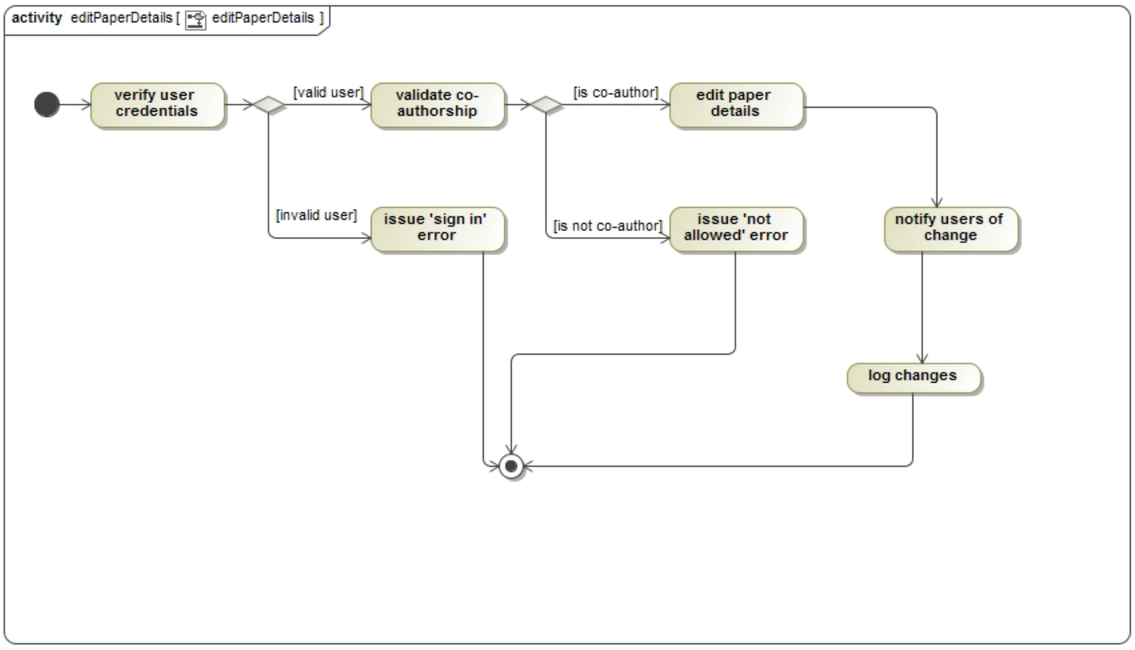
Users can add and remove authors



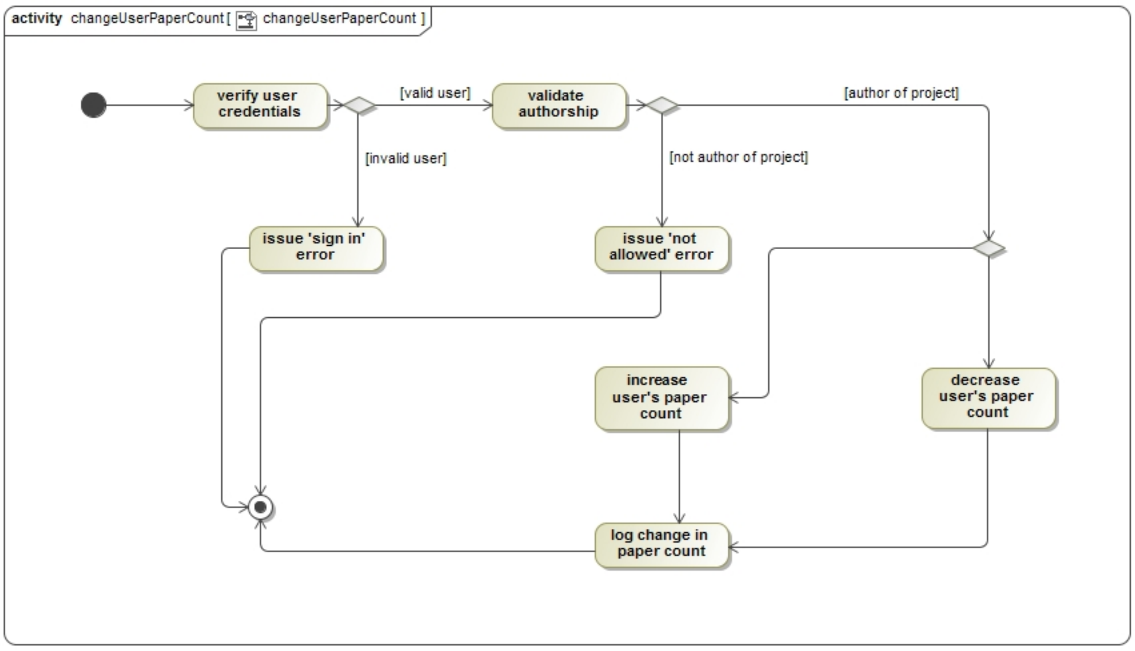
Users decide the deadline of a research project



Users can terminate and revive a research project

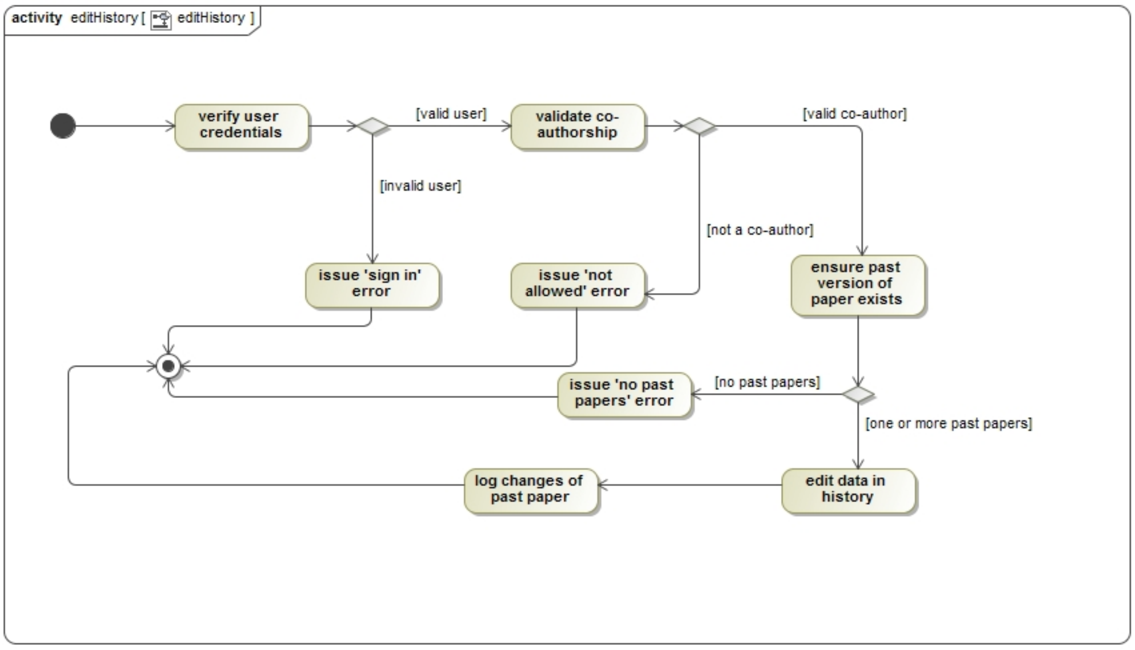


Users can edit details about a research paper they are involved in



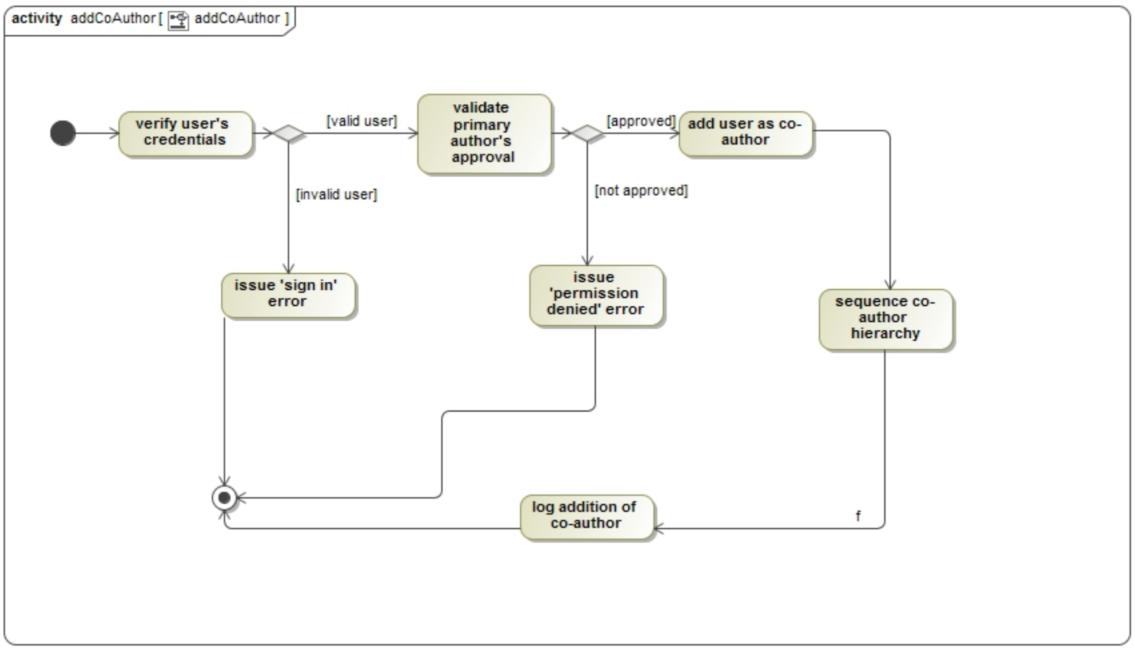
activity changeUserPaperCount [changeUserPaperCount]
verify user credentials
[valid user]
validate authorship
[author of project]
[invalid user]
[not author of project]
issue 'sign in' error
issue 'not allowed' error
increase user's paper count
decrease user's paper count
log change in paper count

User profiles should show the total accumulated units



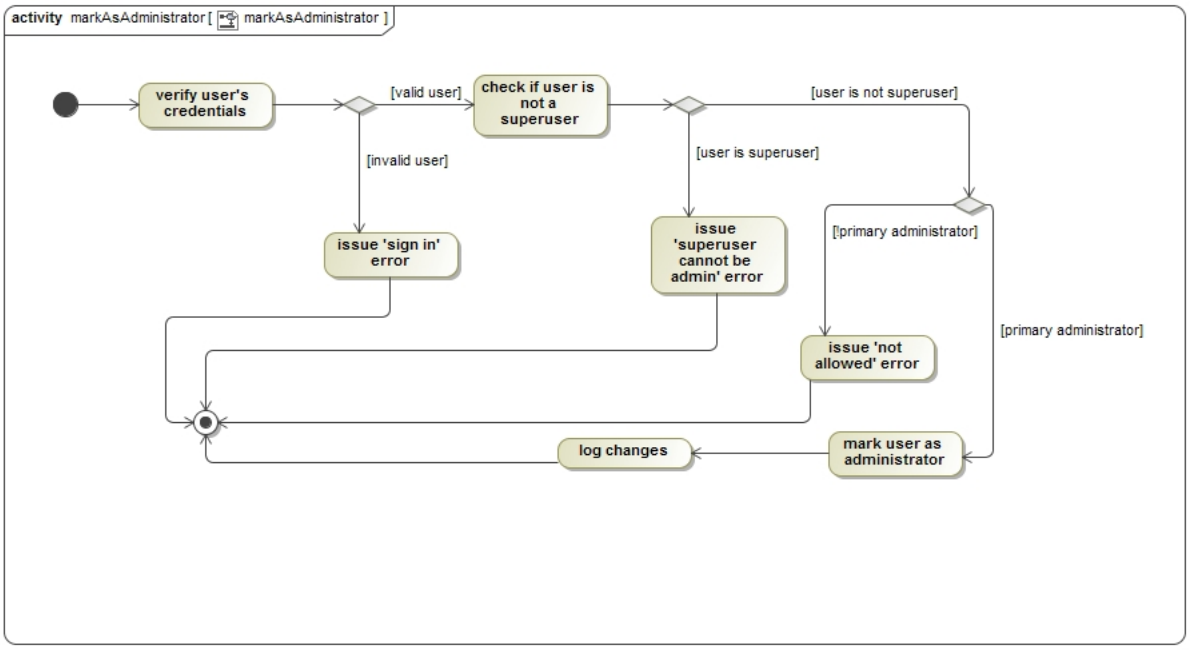
activity editHistory [editHistory]
verify user credentials
[valid user]
validate co-authorship
[valid co-author]
[invalid user]
[not a co-author]
issue 'sign in' error
issue 'not allowed' error
ensure past version of paper exists
issue 'no past papers' error
[no past papers]
[one or more past papers]
log changes of past paper
edit data in history

Users must be able to enter historical data

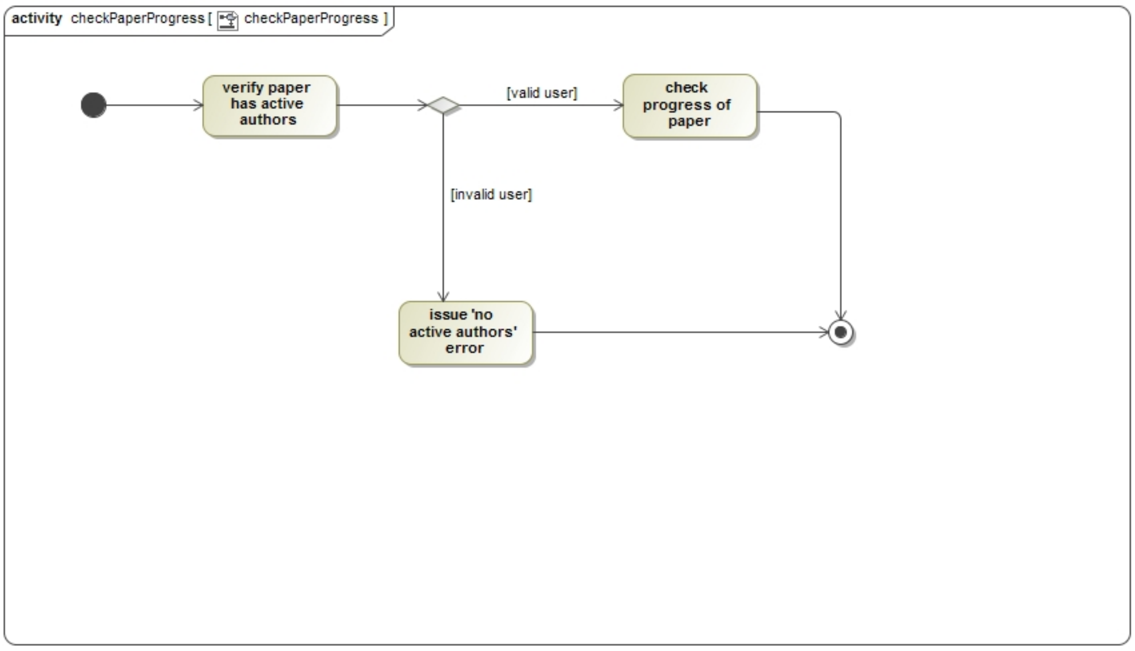

activity addCoAuthor [addCoAuthor]
verify user's credentials
[valid user]
[invalid user]
validate primary author's approval
[approved]
[not approved]
add user as co-author
issue 'sign in' error
issue 'permission denied' error
sequence co-author hierarchy
log addition of co-author
f

Co-authors are added and sequenced

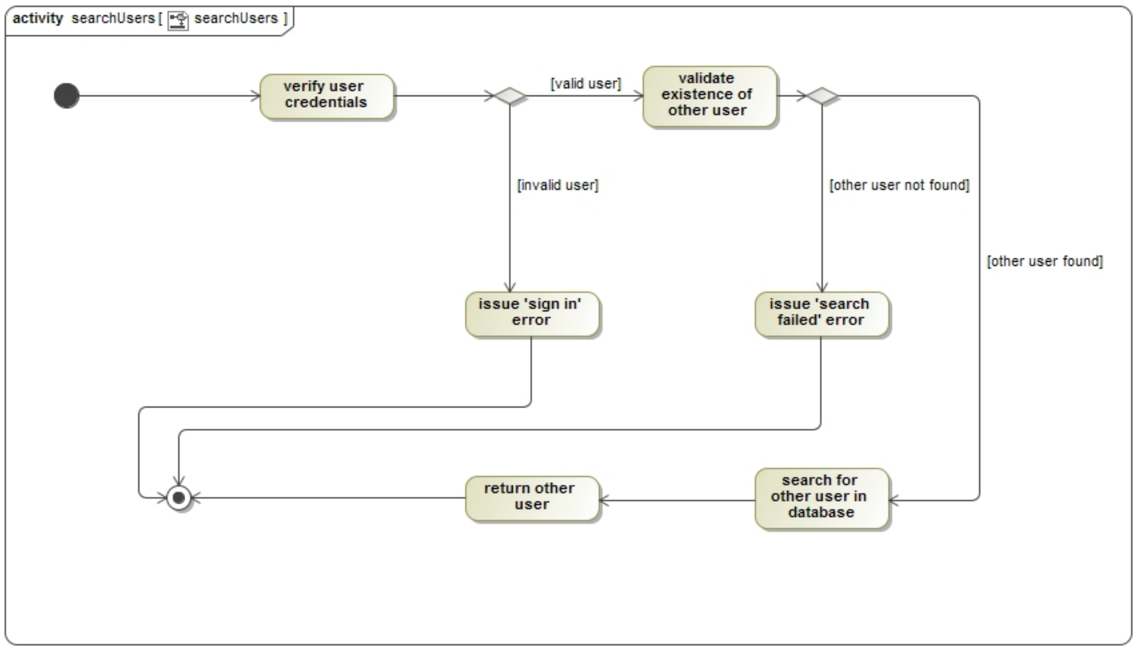
Nice-to-have use cases


activity markAsAdministrator [markAsAdministrator]
verify user's credentials
[valid user]
[invalid user]
check if user is not a superuser
[user is not superuser]
[user is superuser]
issue 'sign in' error
issue 'superuser cannot be admin' error
[!primary administrator]
[primary administrator]
issue 'not allowed' error
mark user as administrator
log changes

Users can be marked as administrators only

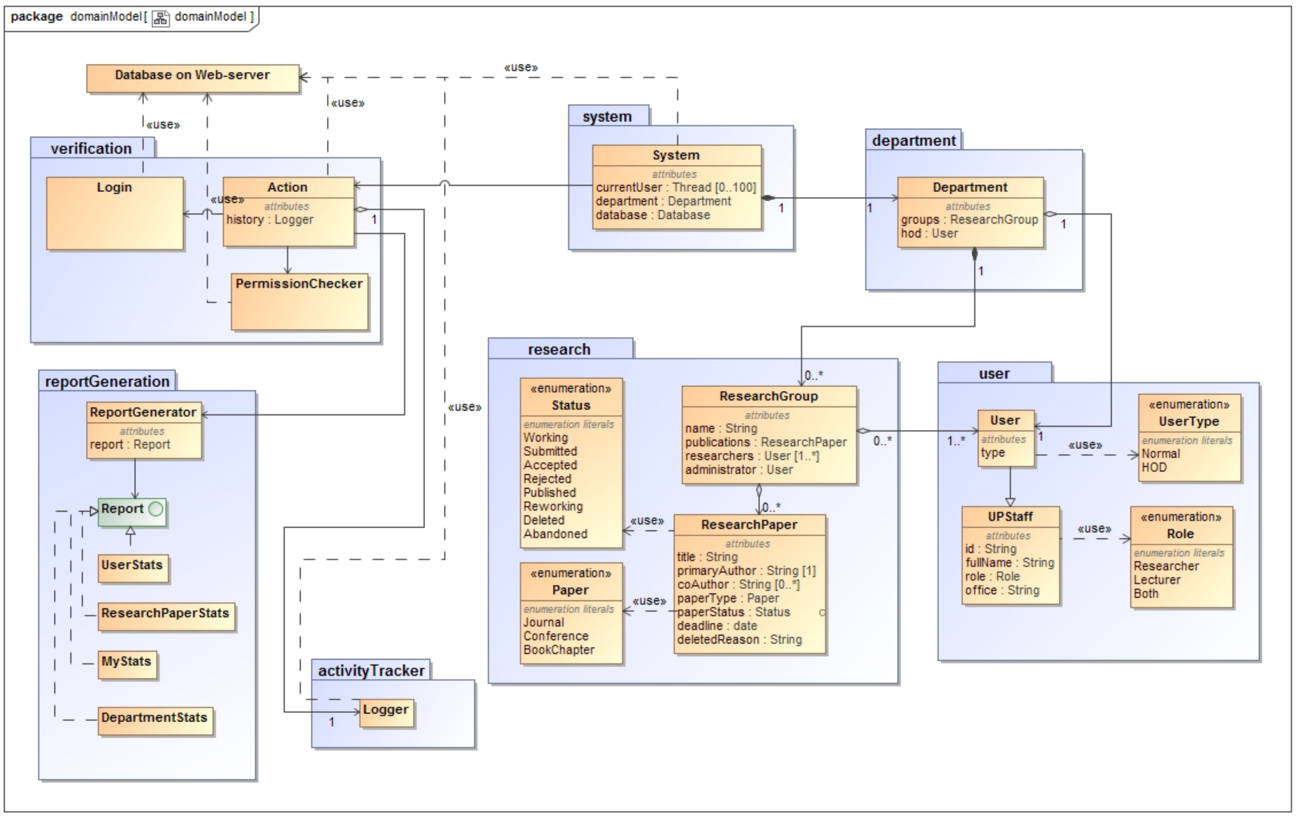

activity checkPaperProgress [checkPaperProgress]
verify paper has active authors
[valid user]
check progress of paper
[invalid user]
issue 'no active authors' error

Indication of progress on research project


activity searchUsers [searchUsers]
verify user credentials
[valid user]
validate existence of other user
[invalid user]
[other user not found]
[other user found]
issue 'sign in' error
issue 'search failed' error
return other user
search for other user in database

Users should be able to search for other authors

5.5 Domain Model



Domain Model representation of the System

6 Open Issues

- It was never specified whether researchers(non-users) would be able to see the information about the publications they were involved in
- Still unclear whether researchers(both users and non-users) will be notified if they are removed from a research group
- We are not sure if a user can have multiple sessions open concurrently - if a user is already logged in on the web application, will they be able to log in using their mobile device?
- Will the progress be represented by a percentage / a progress bar?
- Does the user set the order of the authors or is it automatically set when the user inputs these into a specific document
- Is there an option that will be provided for how long before a deadline the user will want to receive a notification?
- Won't privacy be an issue if researcher(non-user) information is put up and every user - even those not in the same research group - is able to view it?
- Will there be different tabs for different types of publications a user is busy with? ie. screens they can switch between to check on articles, Case reports, terminated publications, etc
- Will there be a way that research group members can access each other's information so that they can make contact?
- What other applications or platforms must this system be able to integrate with?