Conference Management System

The Other Half

Members:

- Bratu Andrei (**Team leader**)
- Breje Mihai
- Catargiu Ecaterina
- Caterenciuc Corina
- Bogdan Cristina
- Capusan Cristian

Content

Short Presentation of the Brief

The objective of the conference Management System is to manage information about conferences. A conference is organized by a steering committee (chair, co-chair and other program committee members). When deciding to organize a conference, the PC members are required to upload information, such as:

- their name
- affiliation
- email address
- their personal web-page
- their username for the CMS
- the password for accessing the information about the conference

A conference has 3 phases. The deadline of each phase is decided by the PC members.

1. In the first phase, any CMS user that is interested in submitting a paper can upload the meta-information about their proposal:
   - the name of the proposal
   - the keywords
   - the topics,
   - the possible list of authors and their meta-information
   - the full paper (depending on conference)

   It is possible that the first phase has 2 deadlines: one for the abstract and one for the full paper.

2. The second phase has 2 deadlines: the bidding deadline and the reviewing deadline.

   In the bidding phase, the PC members do a brief analyze of the abstracts or papers and say which ones they want to review.

   In the reviewing phase, the chair or co-chair decides which PC member will review the remaining papers based on their biddings. A paper needs to have at least 2 reviewers and at most 4.

   A review is a justification and a grade.

   Papers are accepted or rejected based on the grade.

   The papers that are accepted can be modified if the reviewers suggest some improvements.

3. In the third phase the presentation of the papers occurs. A conference can be structured on sections based on the papers that are presented and the number of participants.

   Each section has its own chair, speakers and listeners. A speaker cannot be the chair of the section in which she/he present his paper.

Stages of implementation

In the beginning we came to the conclusion that a weekly meeting would help us a lot: at first because it helped us understand the requirement and then it led us to a better communication and teamwork. We decided to make a web application.

In the first half of the semester, we only worked on understanding the task, deciding which technologies we were going to use, making diagrams, choosing our side (back-end/front end).

Speaking of interfaces, it can be thought of as a contract between the system and the environment. In a computer program, the 'system' is the function or module in question, and the 'environment' is the rest of the project. Because we decided to work with Spring and Spring Boot, we certainly needed this concept in our work, and we chose the JPA repository interface.

Due to the fact that we chose a web application, we considered that the MVC(model view controller) design pattern would be the best fit for our ideas. The model is the central component of this pattern. It is the application's dynamic data structure, independent of the user interface which It directly manages the data, logic and rules of the application. The view is a representation of information and the controller accepts input and converts it to commands for the model or view. The choice of MVC and layered architecture is determined by a solution where the separation of concerns, ease of maintainability and extensibility of an application matters significantly. We also certainly need DTOs that carries data between processes.

JWT is an internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. We chose this for giving permissions to certain users to some related pages of the app.

Furthermore, for the security of our application we used hashing, which is essentially a process that translates information about the file into a code.

Service-oriented software engineering incorporates the best features of both the services and cloud computing paradigms, offering many advantages for software development and applications. However, the only kind of cloud that we have in our application is the database: PostgreSQL that is stored on the cloud, and which is an open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads.

In the second half of the semester, our meetings turned into coding sessions that helped a lot through the process of coming out with some good work. We developed the application based on the structure and decisions made in the first weeks and using the technologies described above.

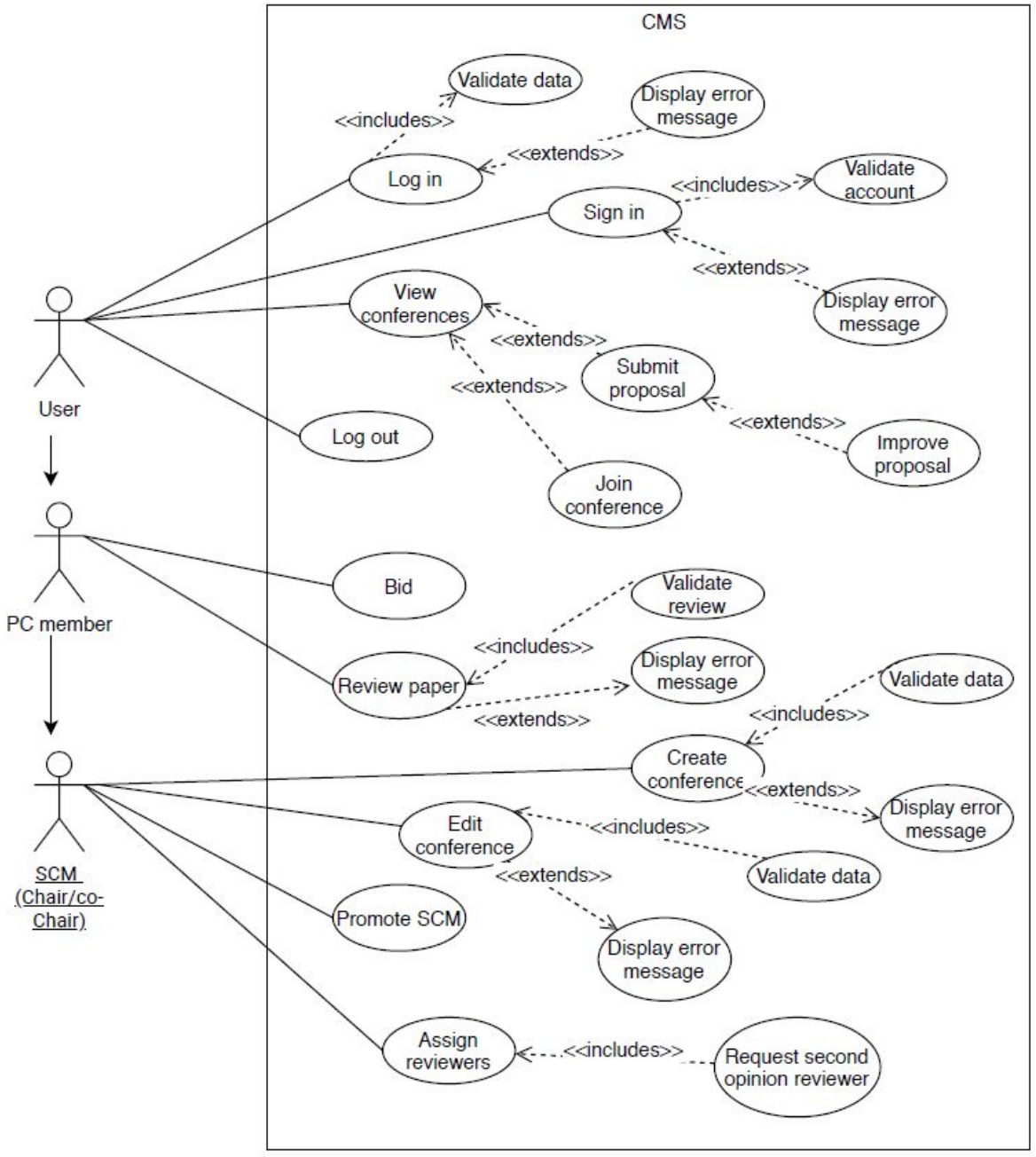
Used technologies

1. *Database Server:*
    - ➢ PostgreSQL
2. *Programming languages:*
    - ➢ *Front-end*: JavaScript (React), TypeScript, HTML, CSS
    - ➢ *Back-end*: Java, SpringBoot
3. *ORM*:
    - ➢ Hibernate
4. *Tools used for diagrams:*
    - ➢ draw.io
    - ➢ StarUML
    - ➢ Visual Paradigm
5. *Version control:*
    - ➢ BitBucket (Git)
6. *Task management:*
    - ➢ Trello
7. *GUI prototyping*:
    - ➢ JavaScript
    - ➢ React - base UI
    - ➢ HTML
    - ➢ CSS
8. *Client-server communication:*
    - ➢ Redux

Diagrams

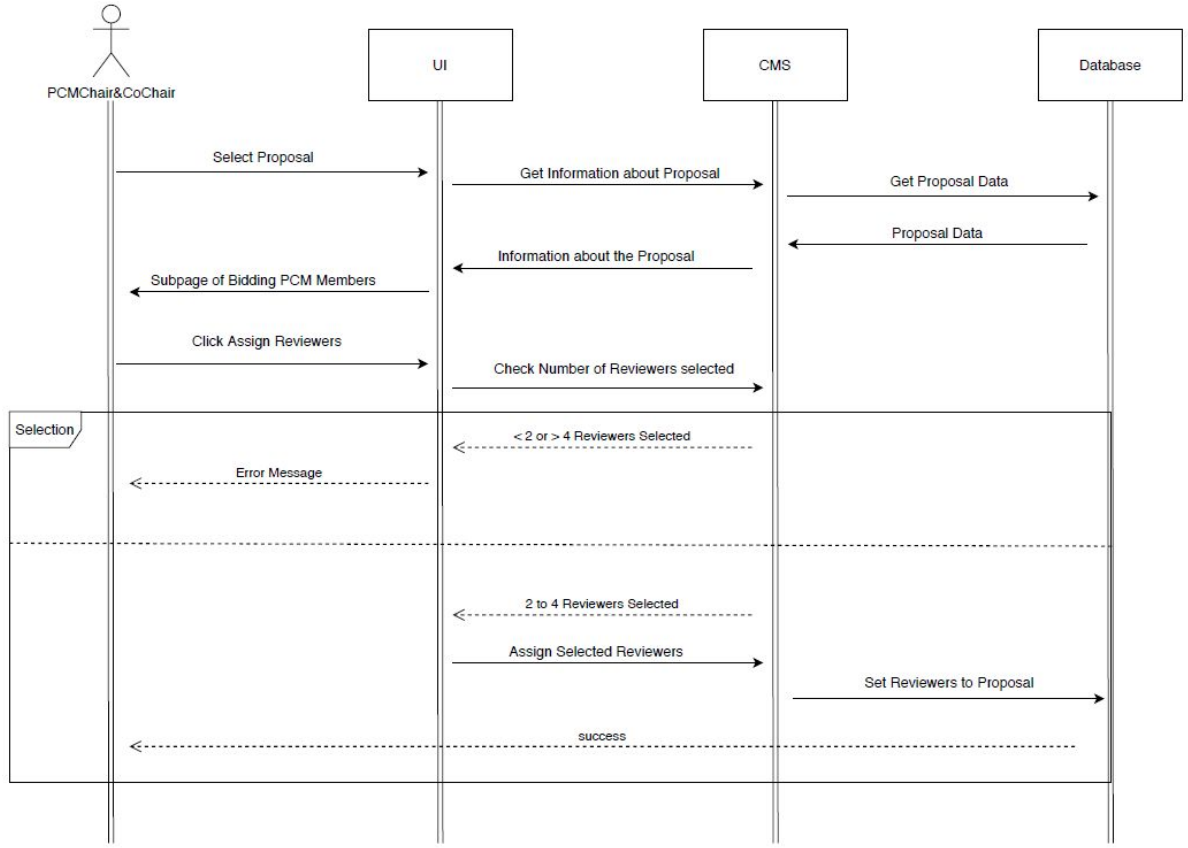
- *Use case diagram*
- *Sequence diagrams*
- *Communication diagrams*
- *Architecture diagrams*
- *State Machine Diagrams*

Use case diagram

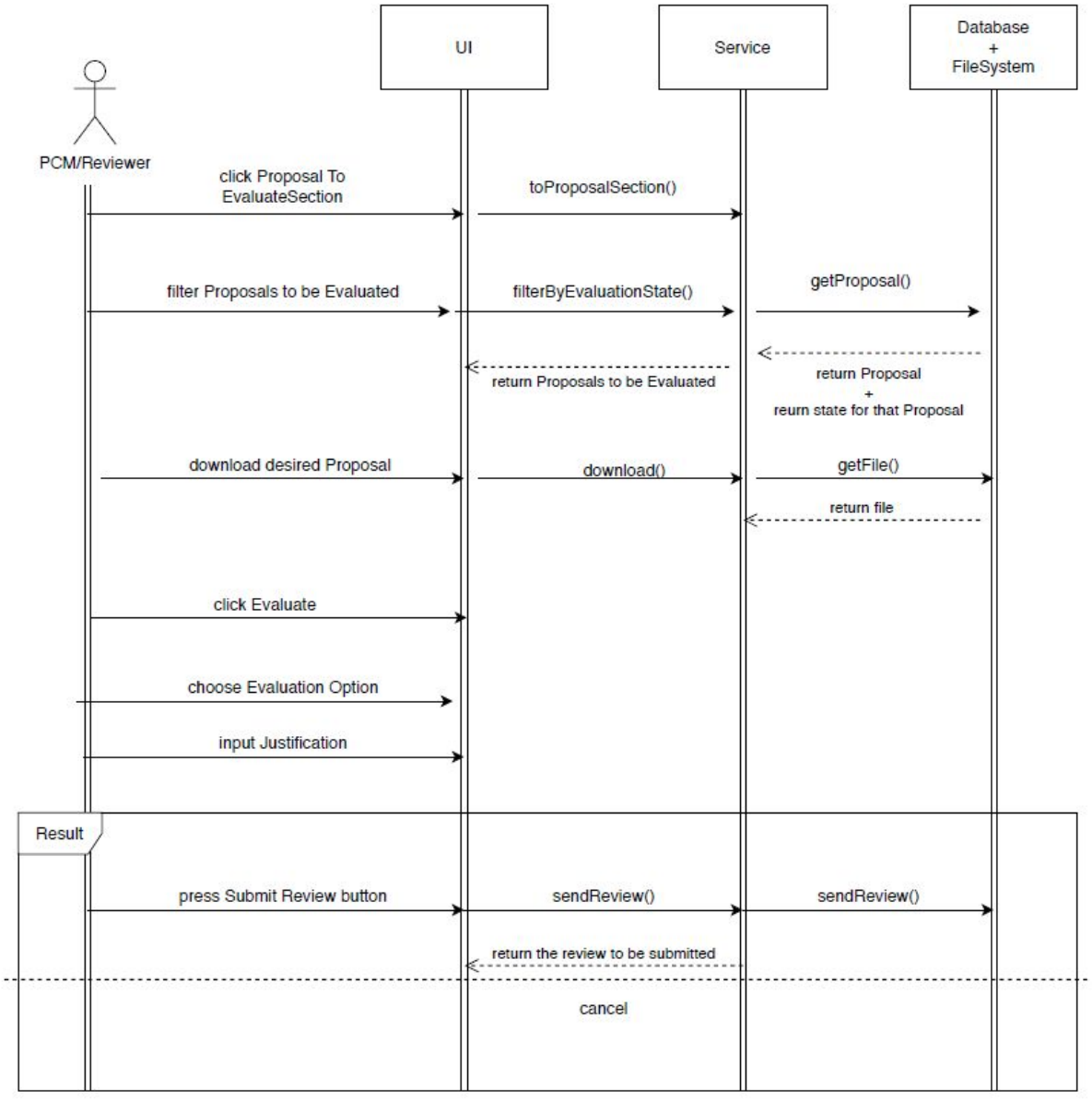


Sequence diagrams

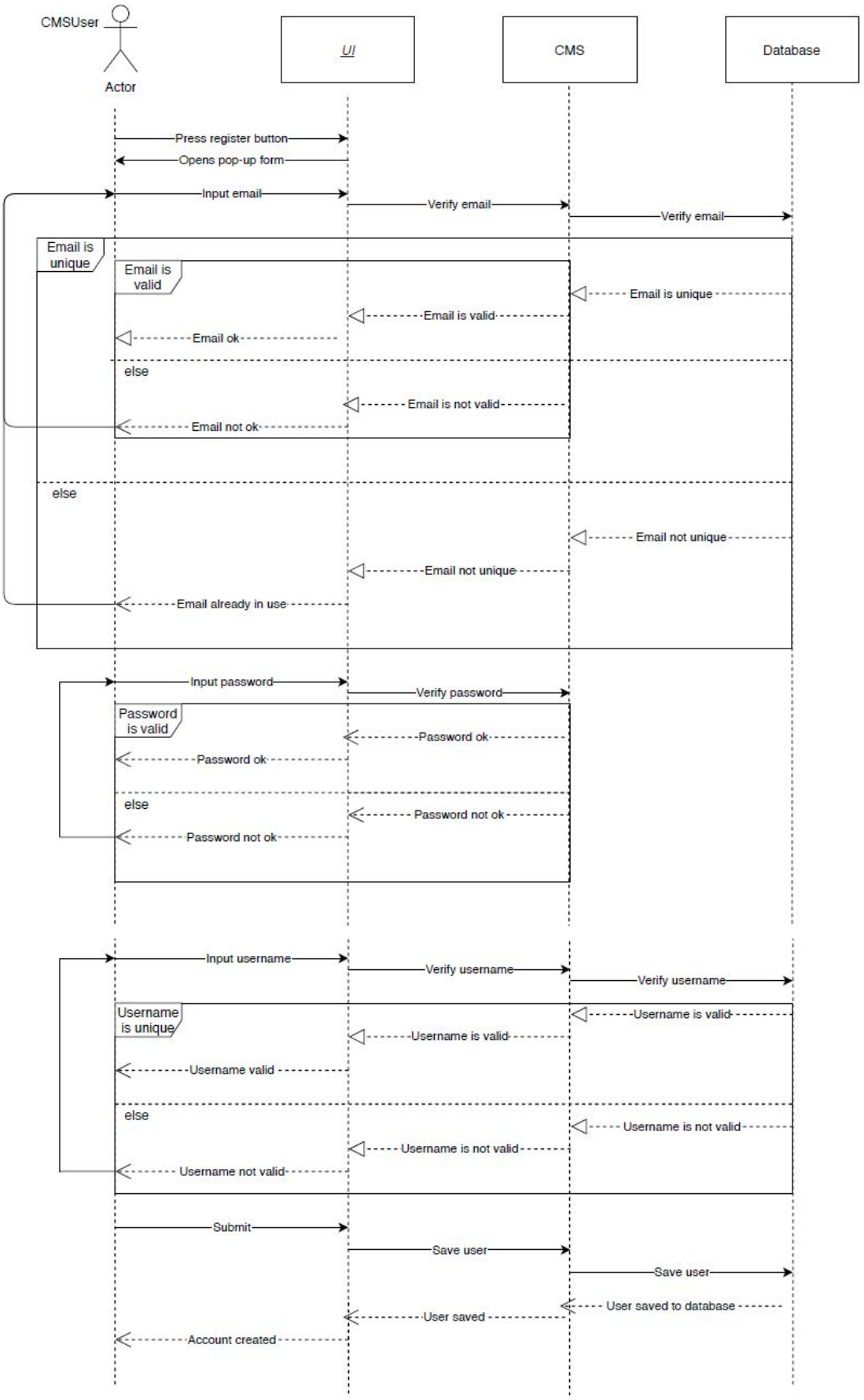
Assign Reviewer to Proposal Sequence Diagram



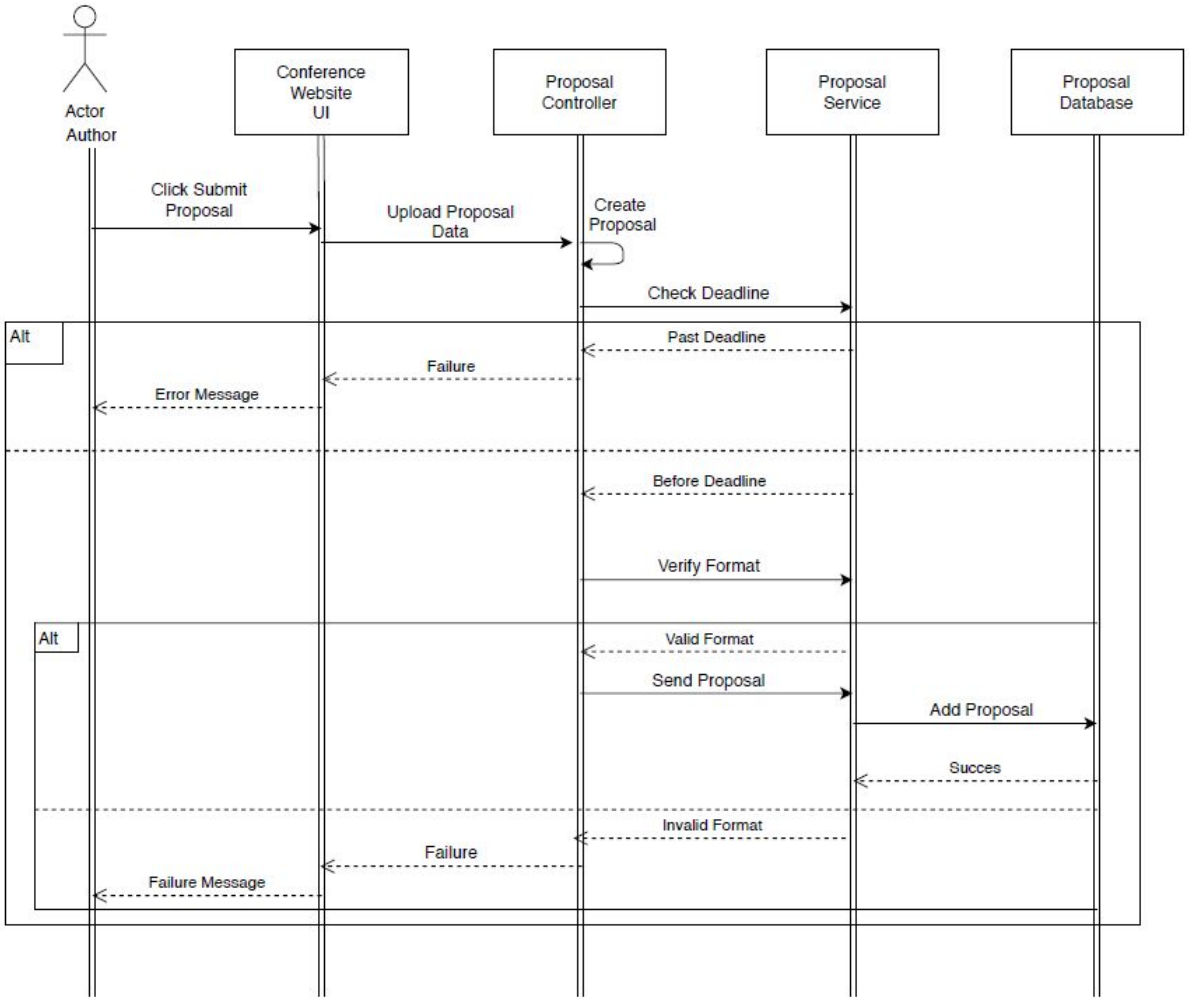
Evaluate Proposal Sequence Diagram



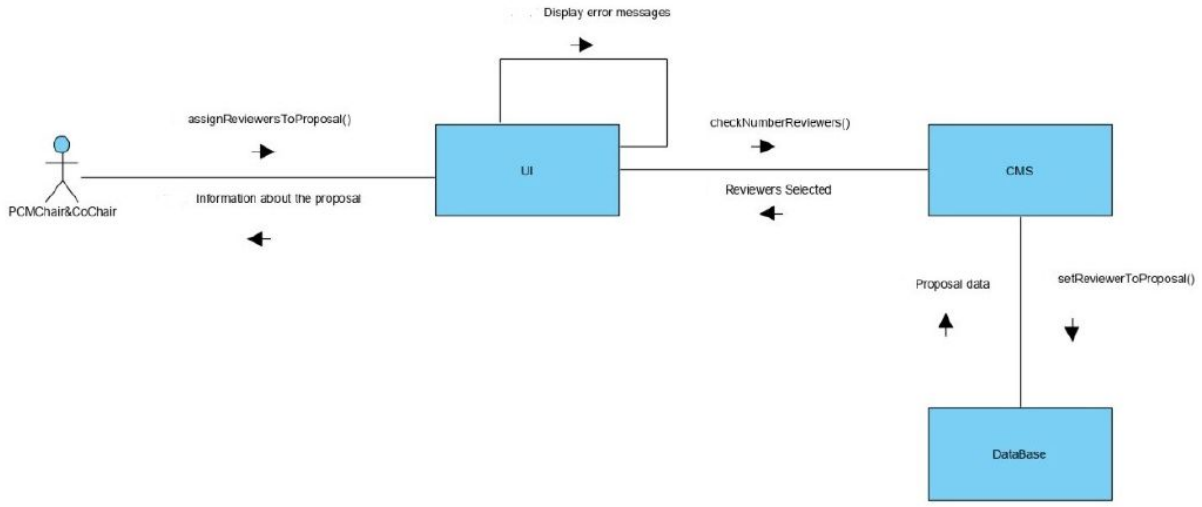
Register Sequence Diagram

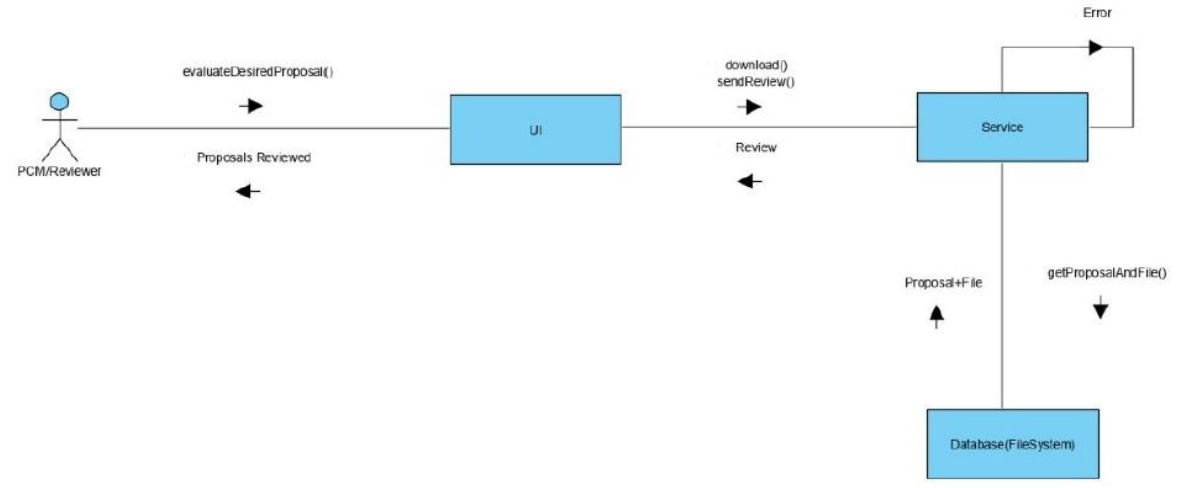


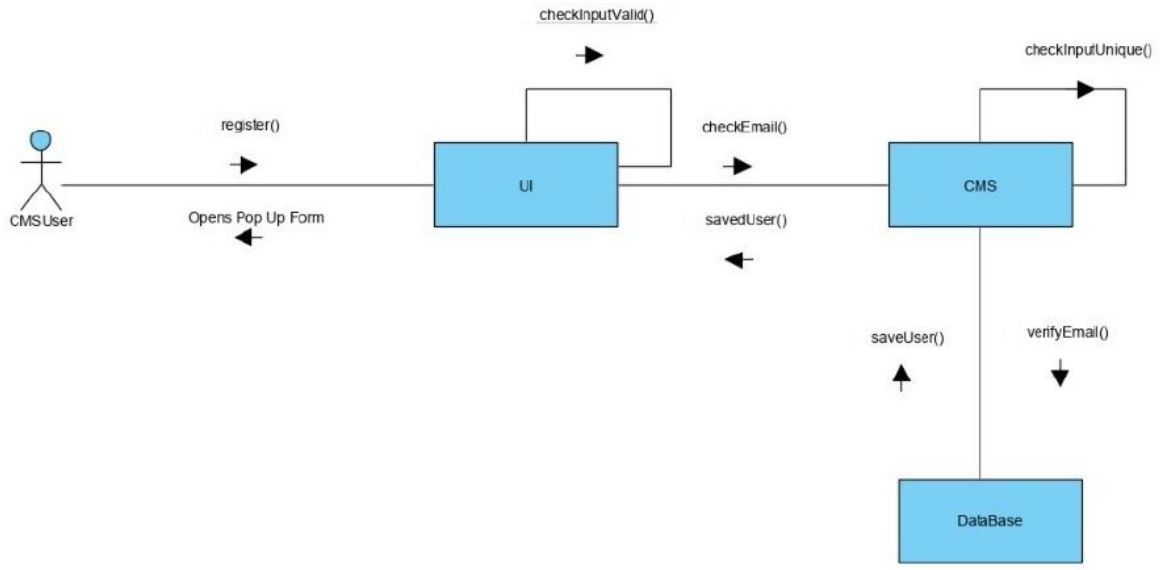
Submit Proposal Sequence Diagram

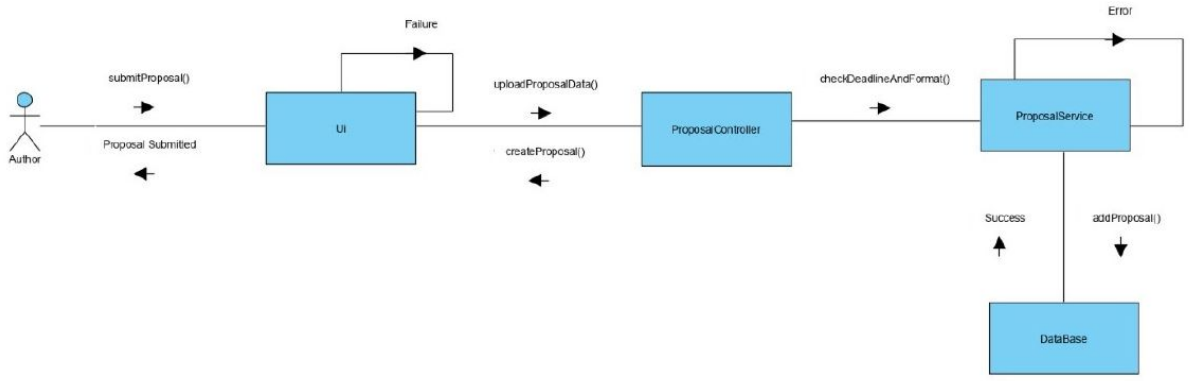


Communication diagrams



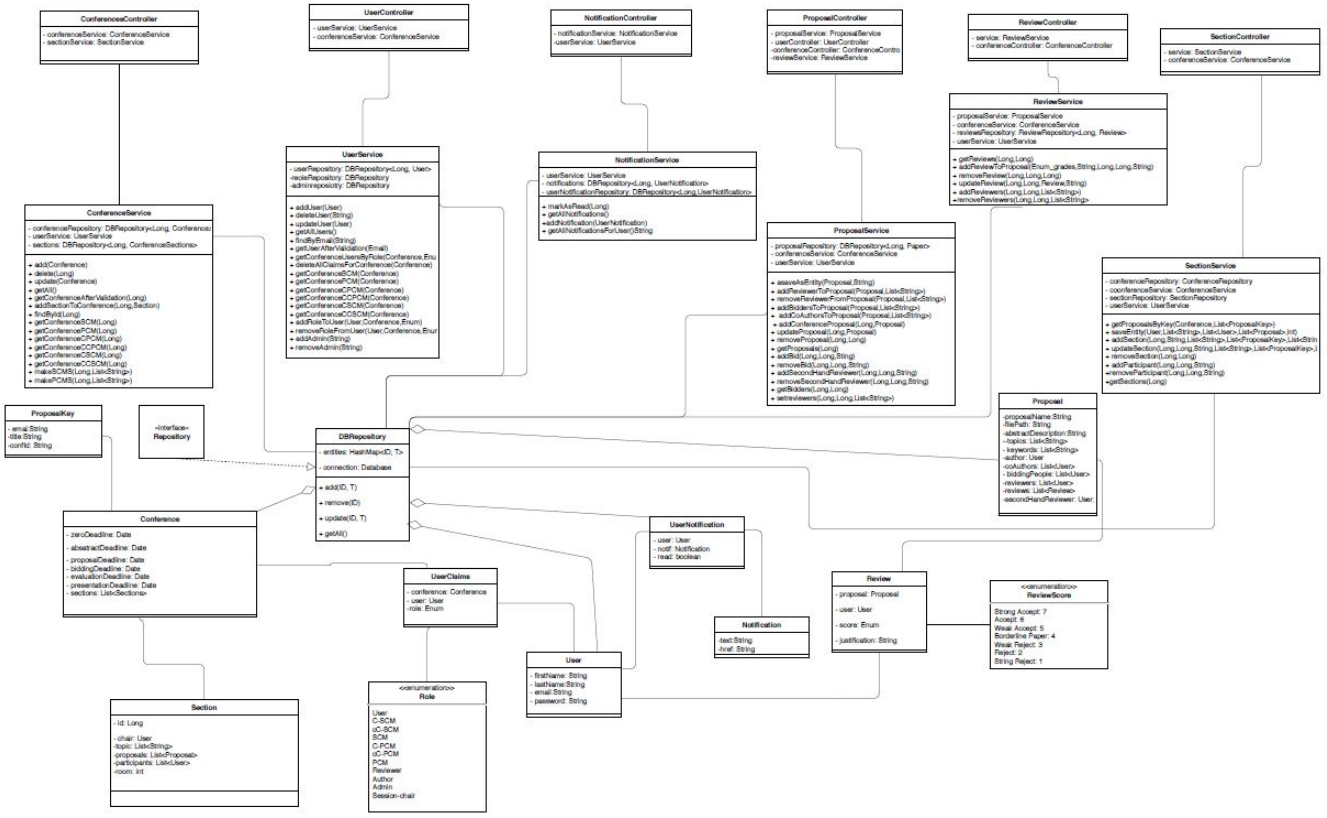



checkInputValid()
checkInputUnique()
register()
checkEmail()
CMSUser
UI
CMS
Opens Pop Up Form
savedUser()
saveUser()
verifyEmail()
DataBase

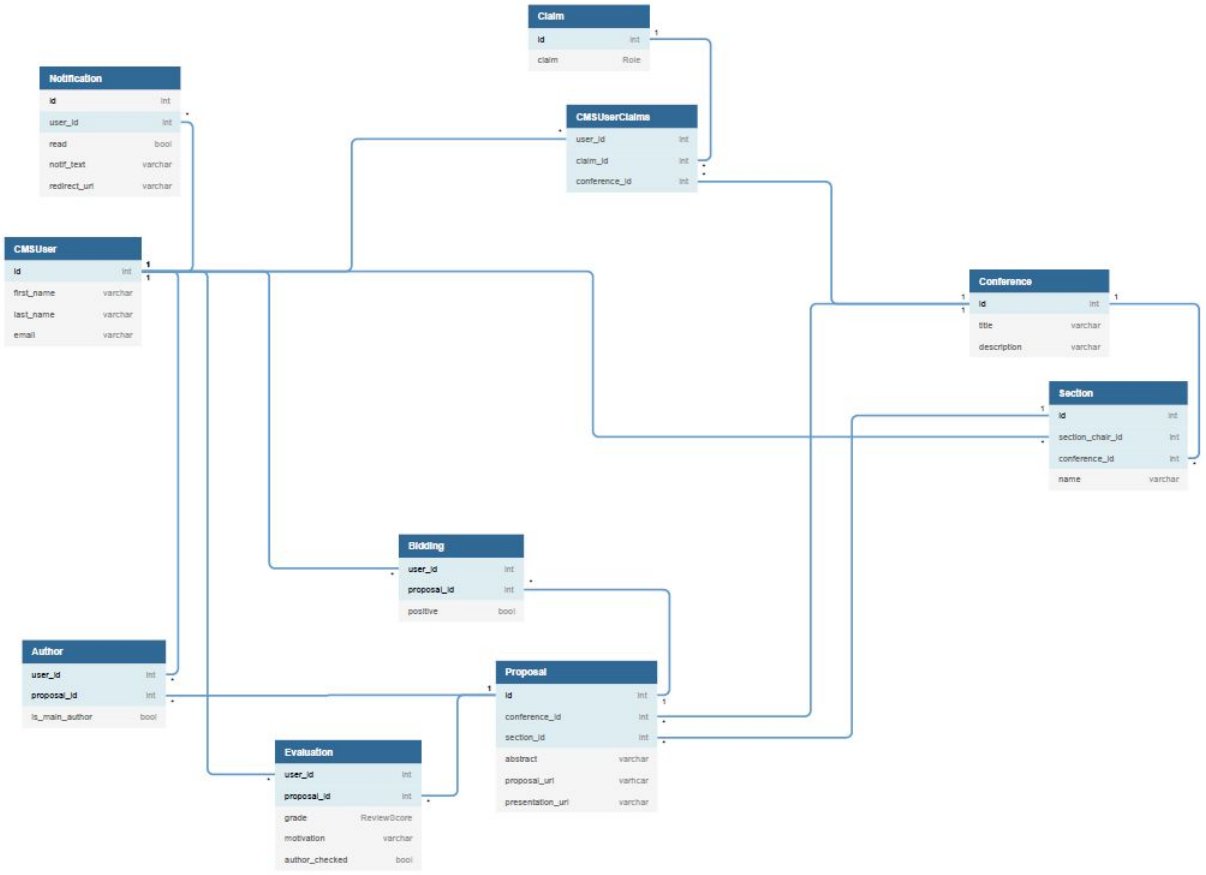

Failure
Error
submitProposal()
uploadProposalData()
checkDeadlineAndFormat()
Author
Proposal Submitted
UI
ProposalController
ProposalService
createProposal()
Success
addProposal()
DataBase

Architecture diagrams

[Class Diagram]



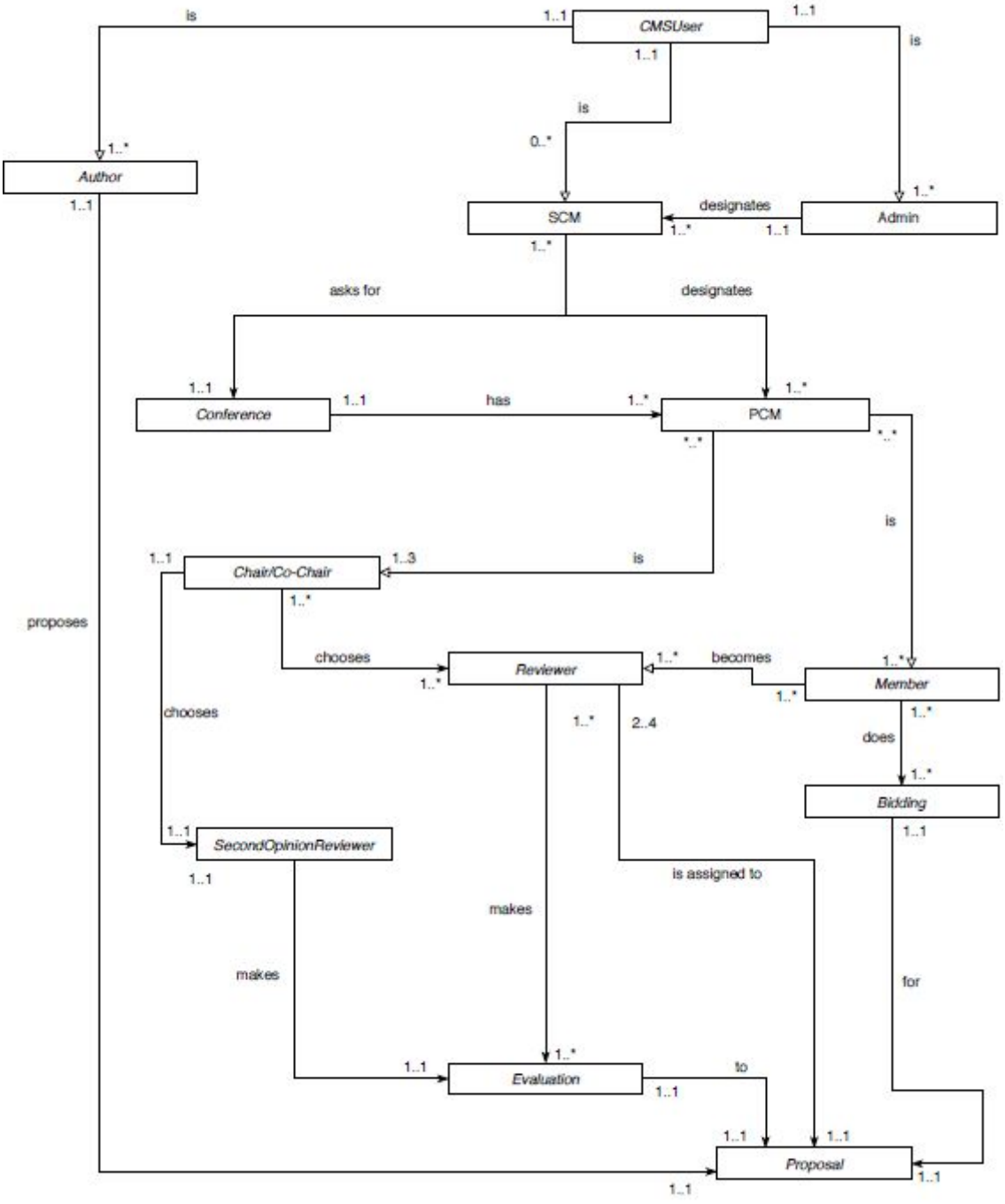
[Database Diagram]



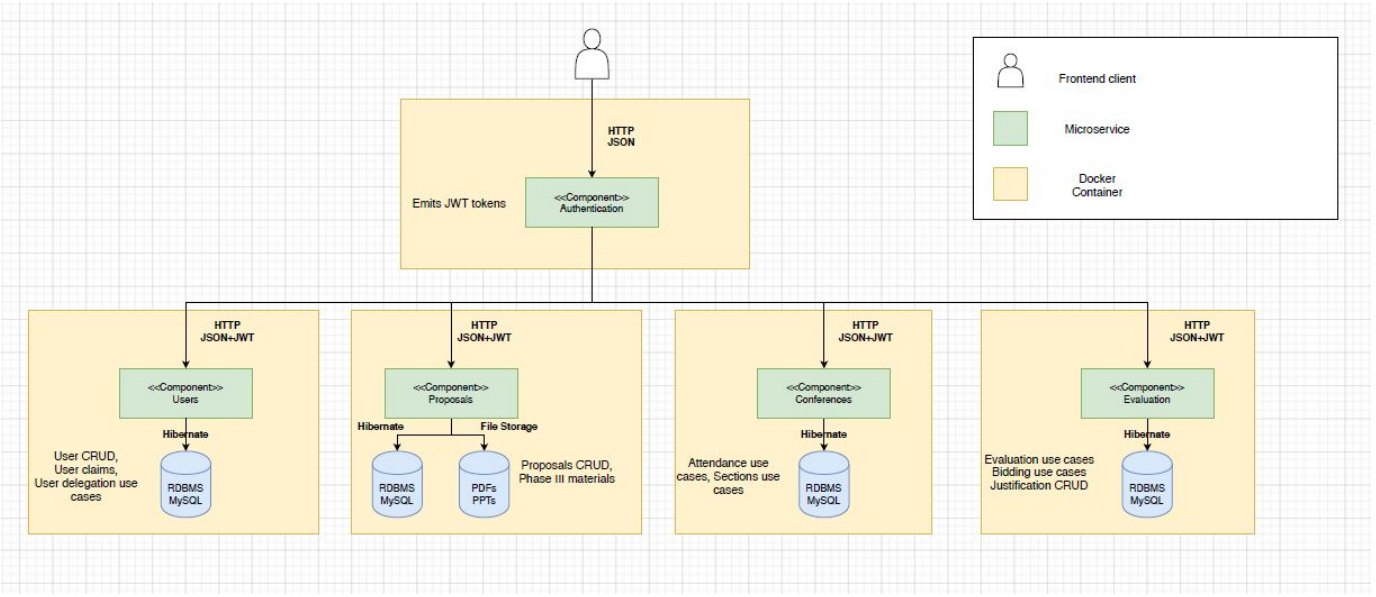

```
Enum Role {
  User
  C_SCM
  cC_SCM
  SCM
  C_PCM
  cC_PCM
  PCM
  Reviewer
  Author
  Speaker
  Admin
  C_Session
}
```

```
Enum ReviewScore{
  Strong_Accept
  Accept
  Weak_Accept
  Borderline_Paper
  Weak_Reject
  Reject
  Strong_Reject
}
```

Entity Diagram

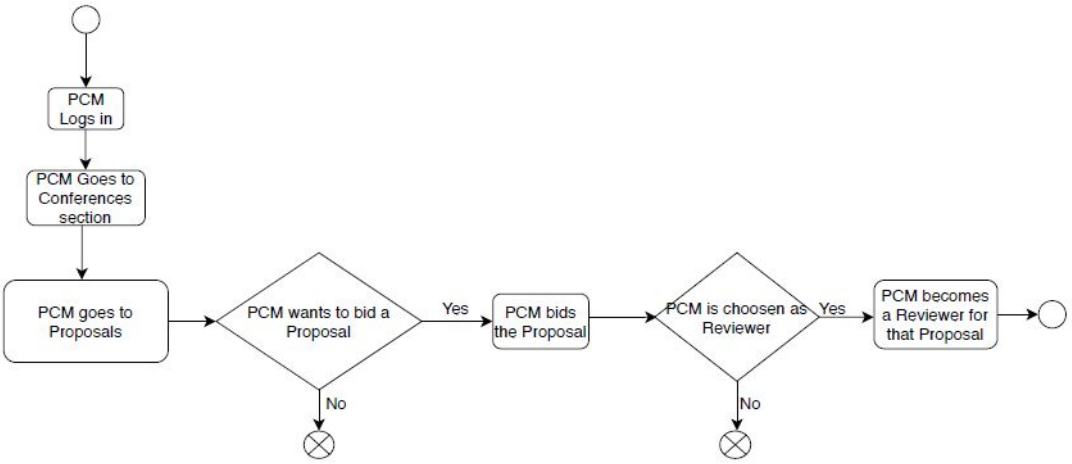


Microservices Architecture

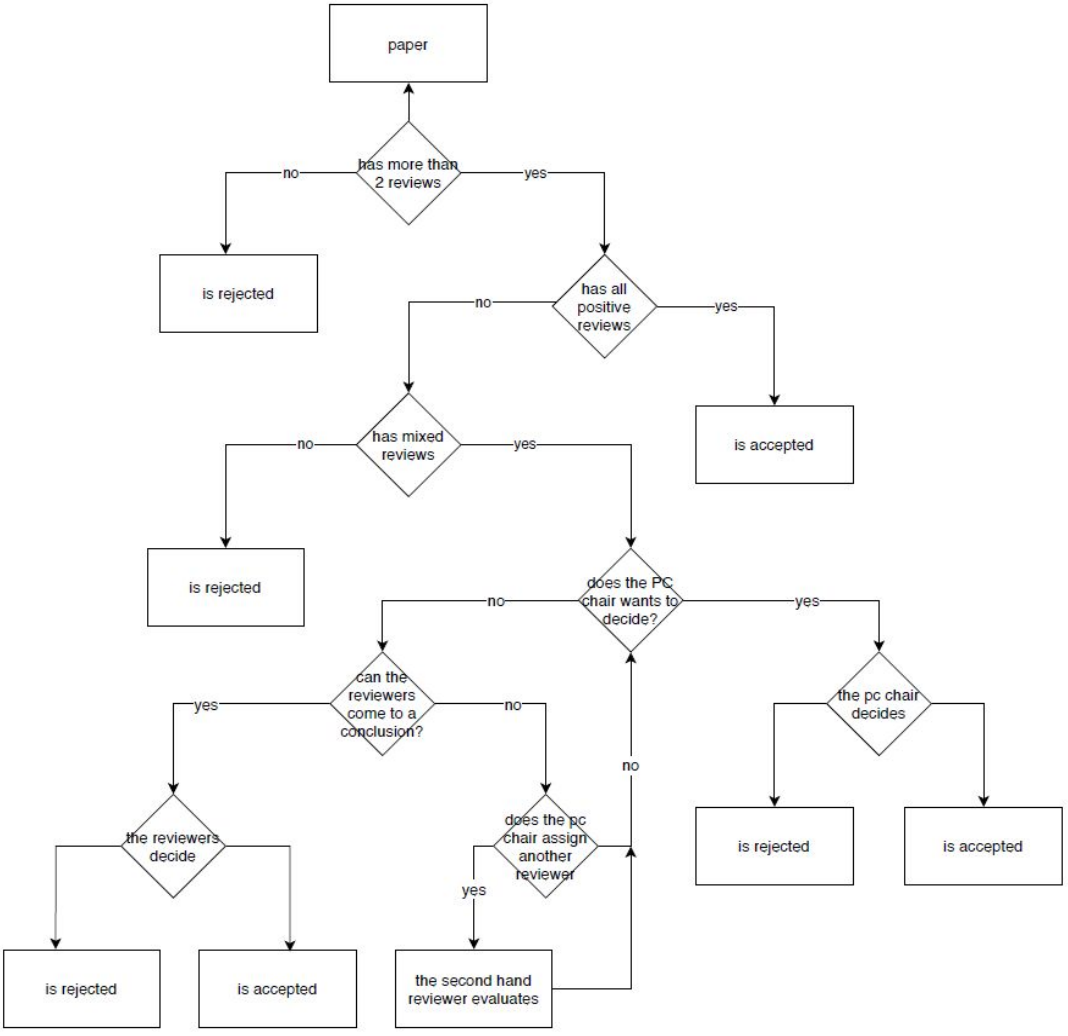


State Machine diagrams

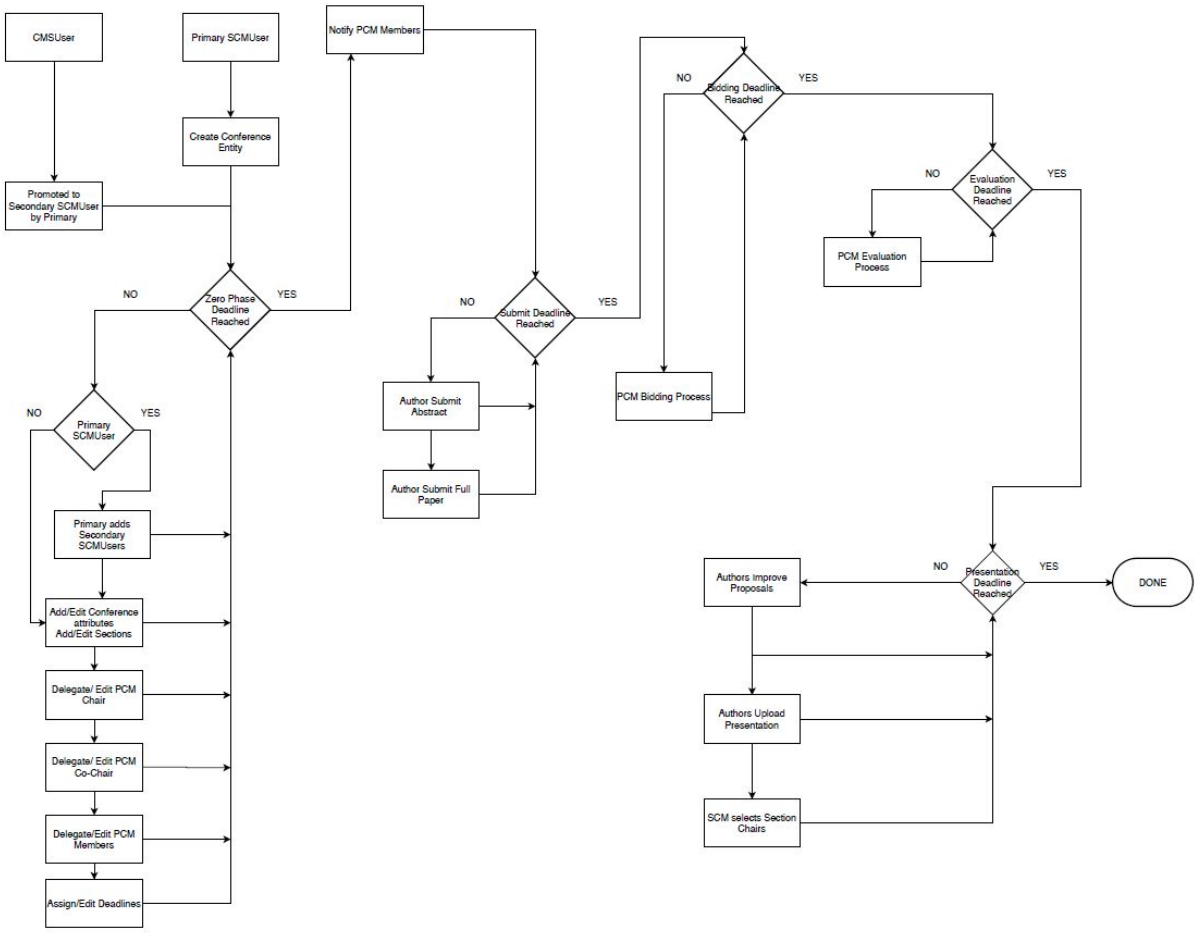
PCM Bid State Machine



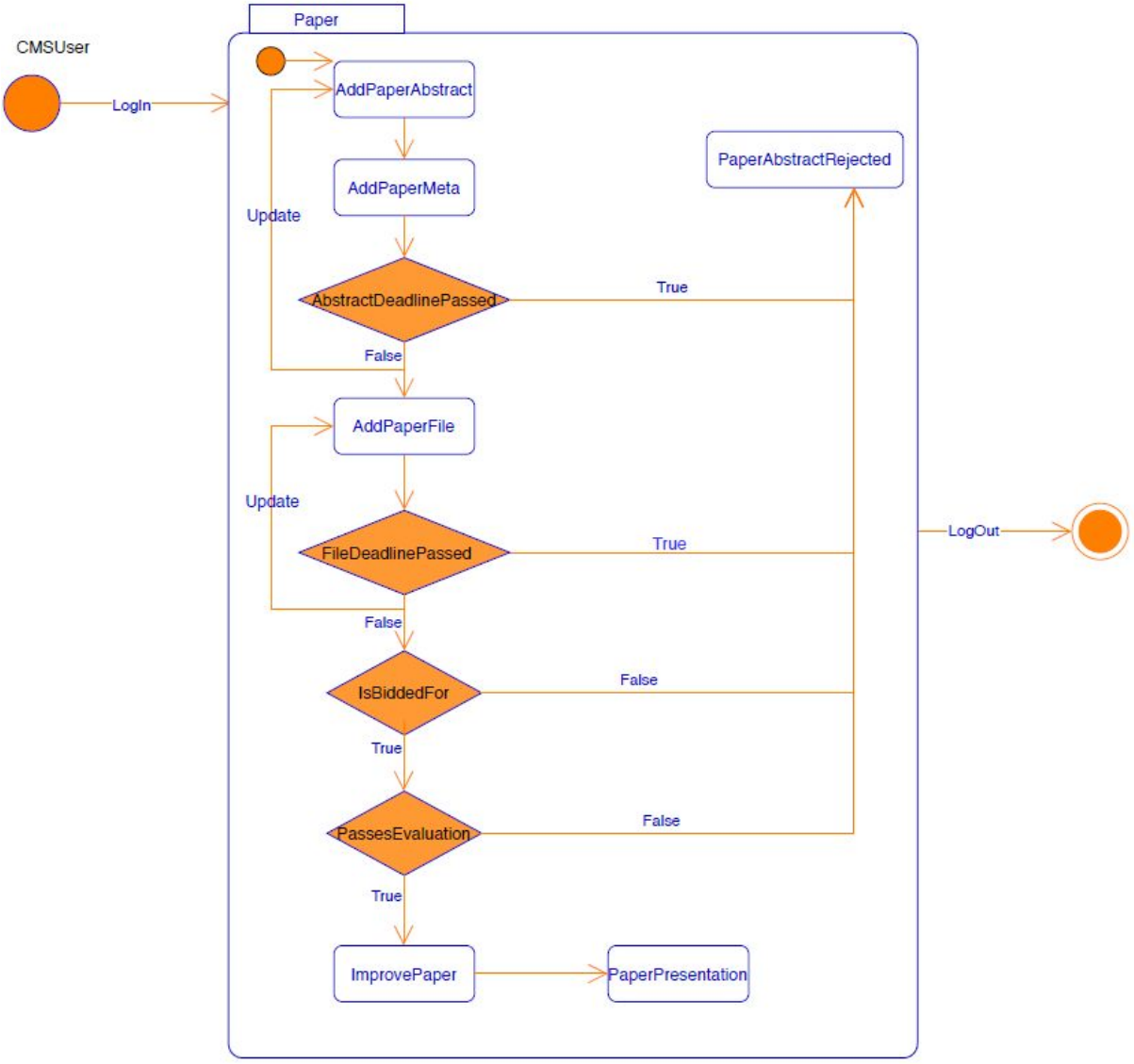
Evaluate Paper State Machine



Conference State Machine



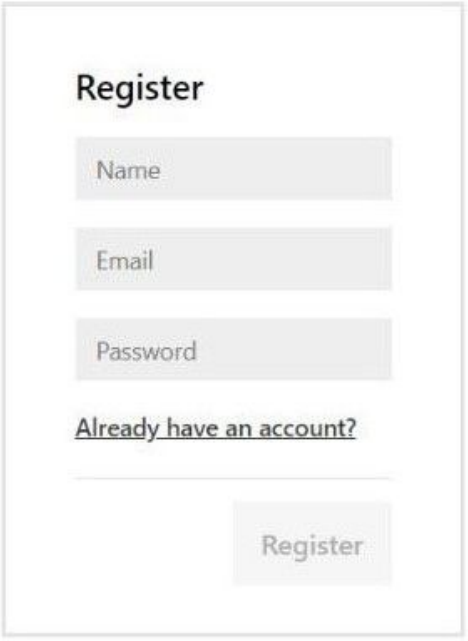
Paper State Machine



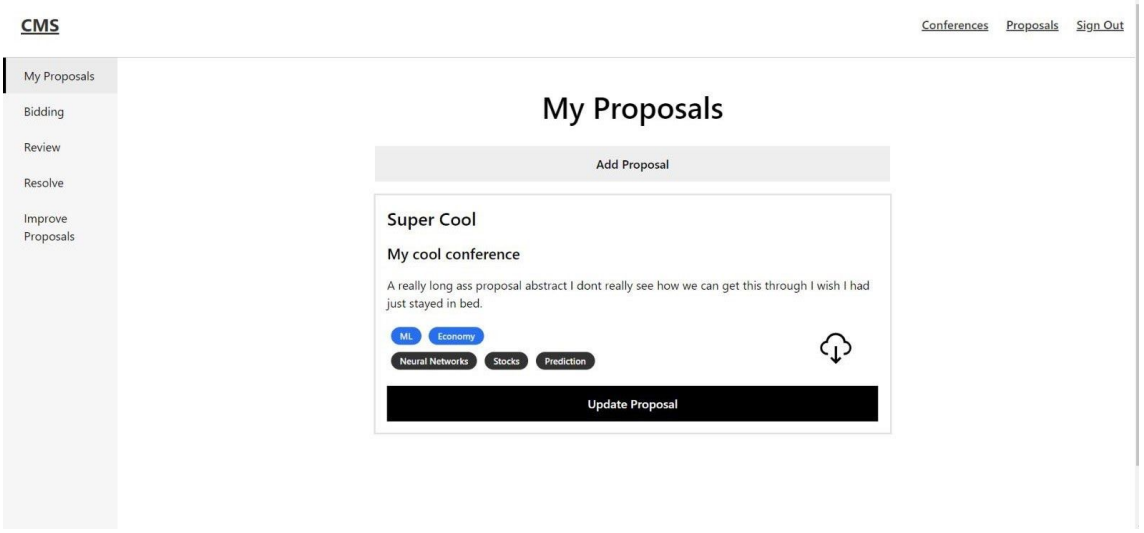
Help - User Instruction

Before getting started, we want to give you an insight into the journey of using the application:

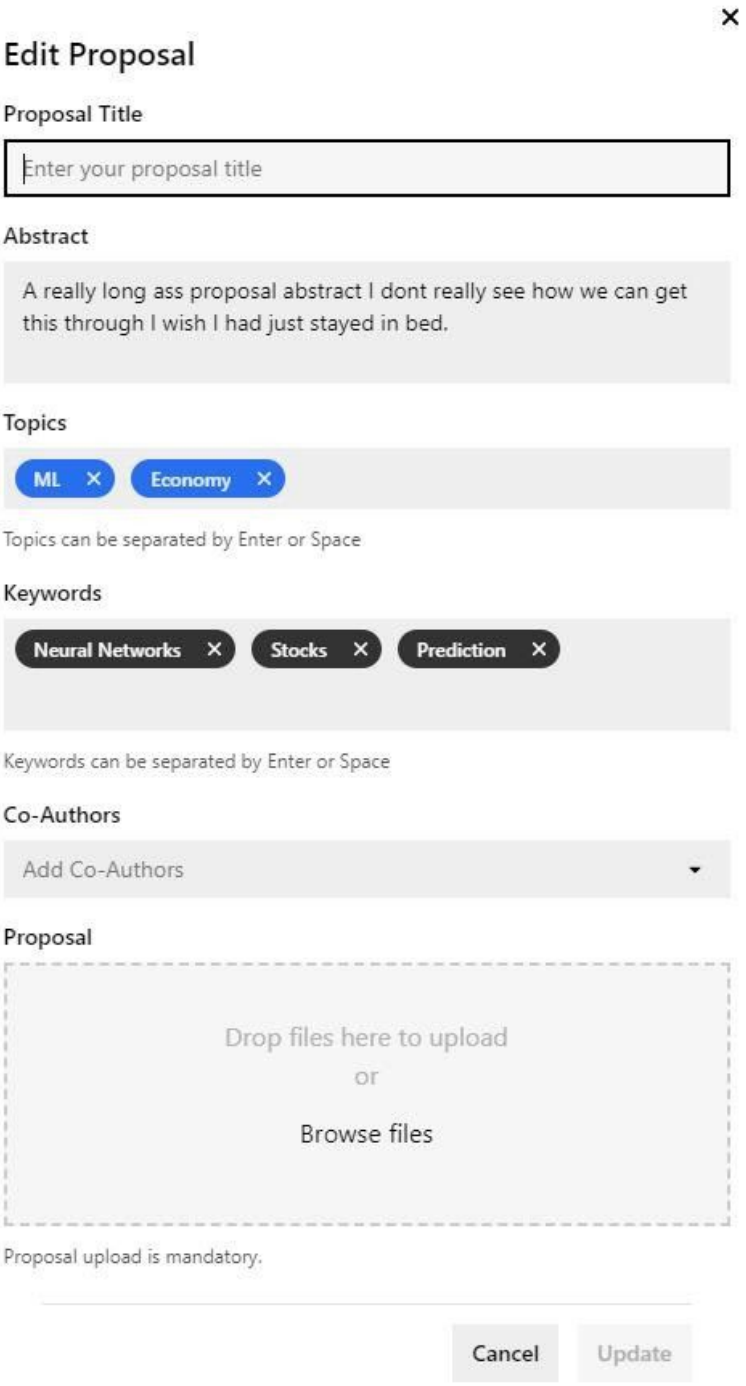
Login/Register forms:



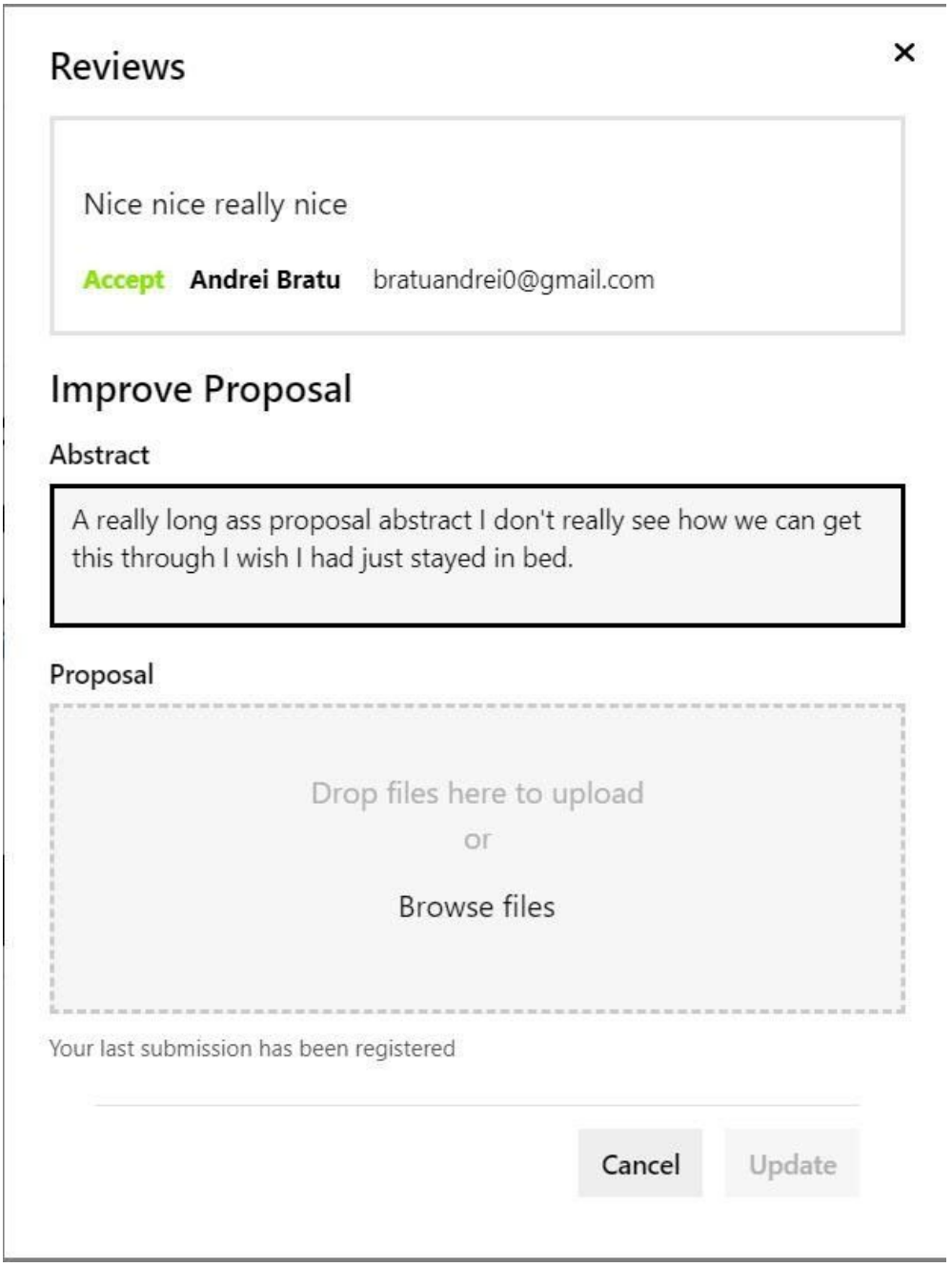
Adding, Seeing and Updating your proposal:



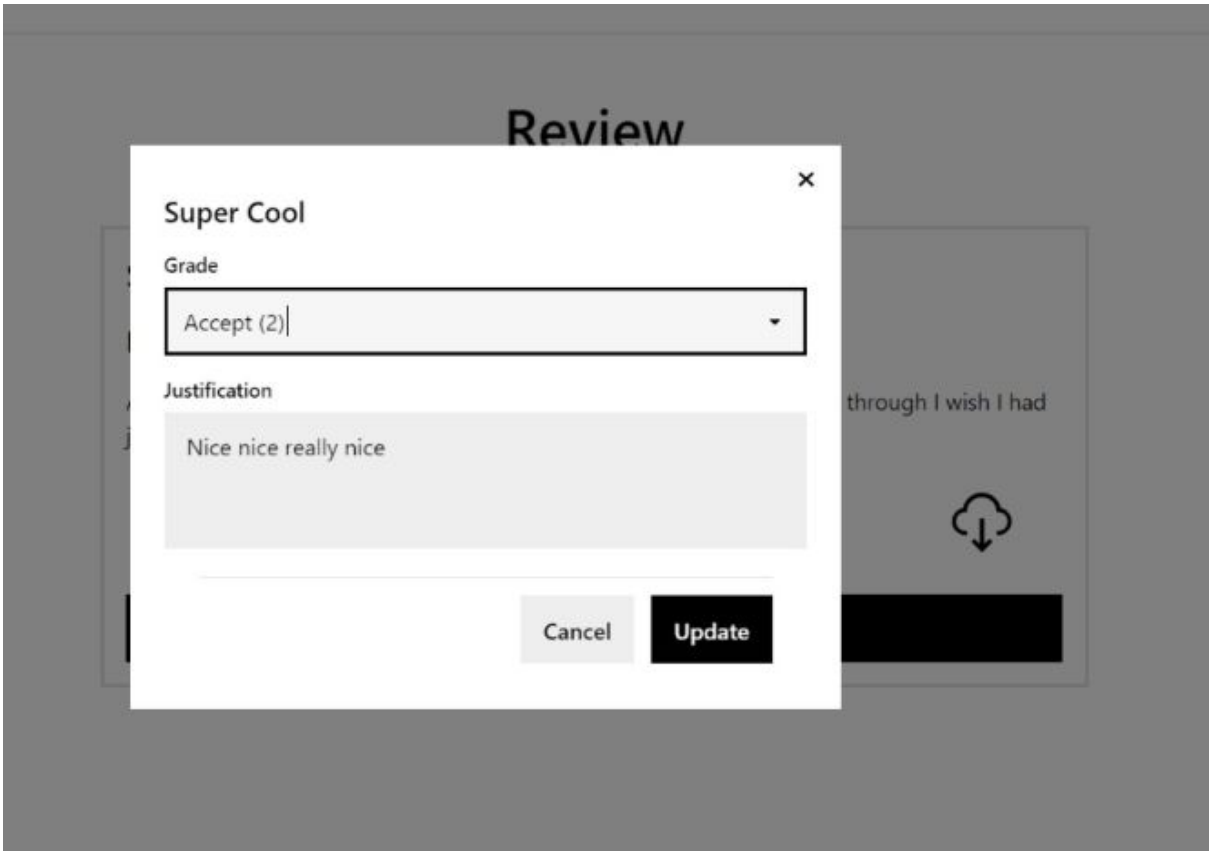
Editing your proposal :



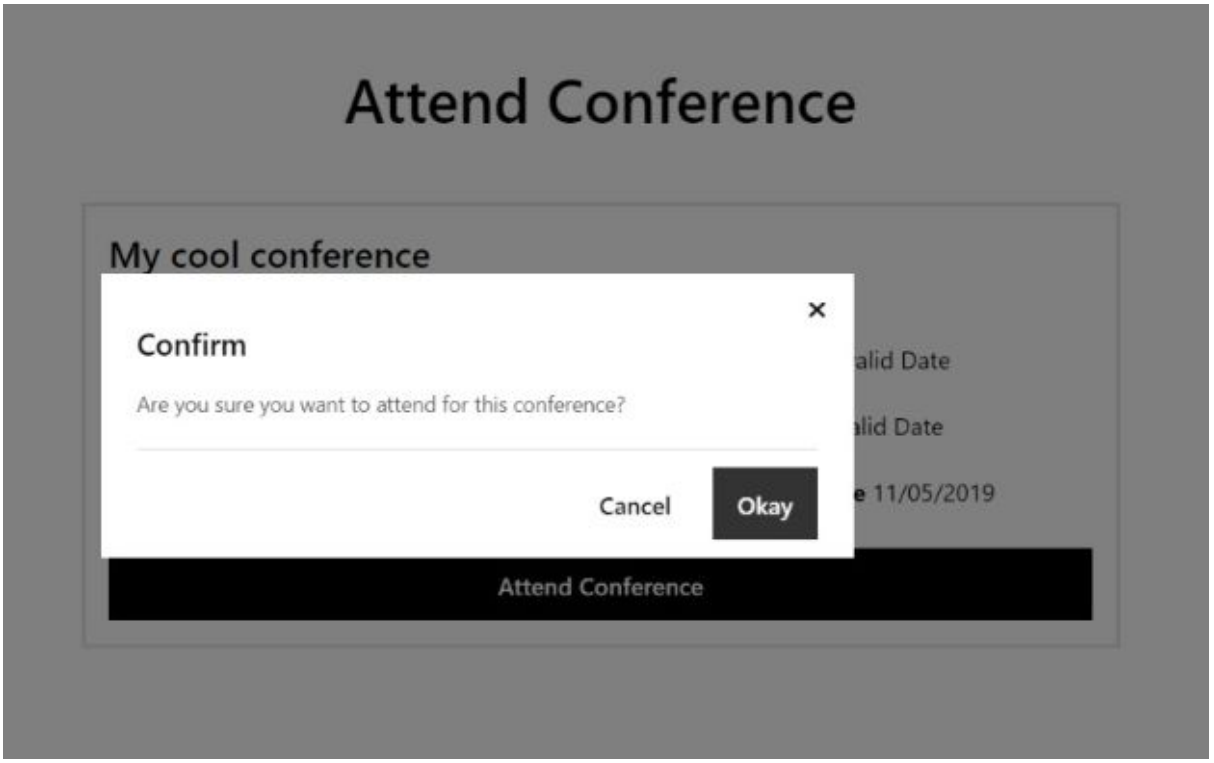
View the Reviews of your Proposal:



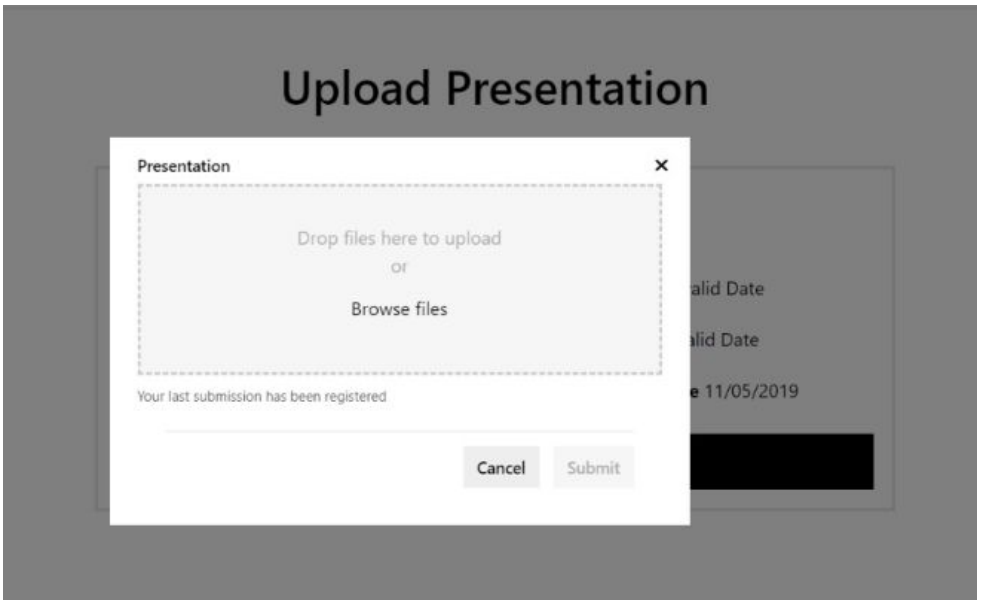
Giving a review to a proposal :



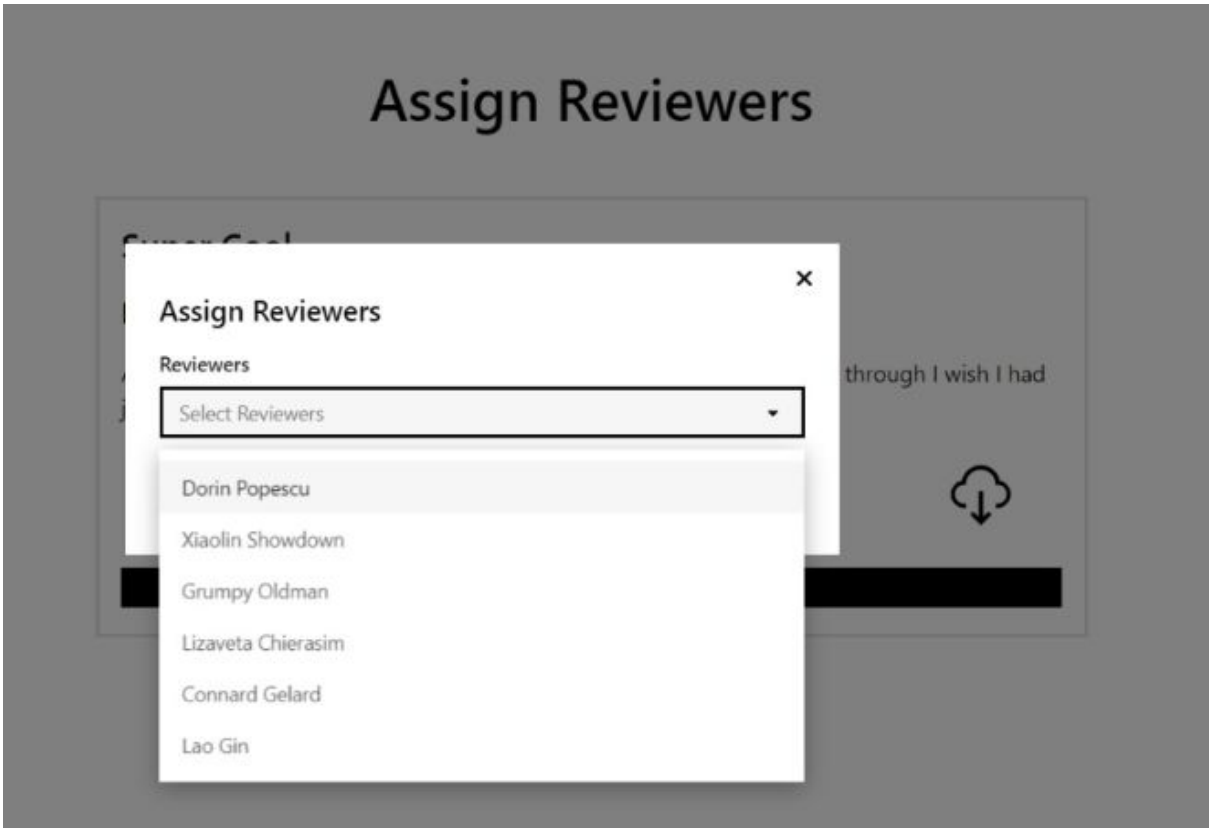
Confirmation of attending a certain conference :



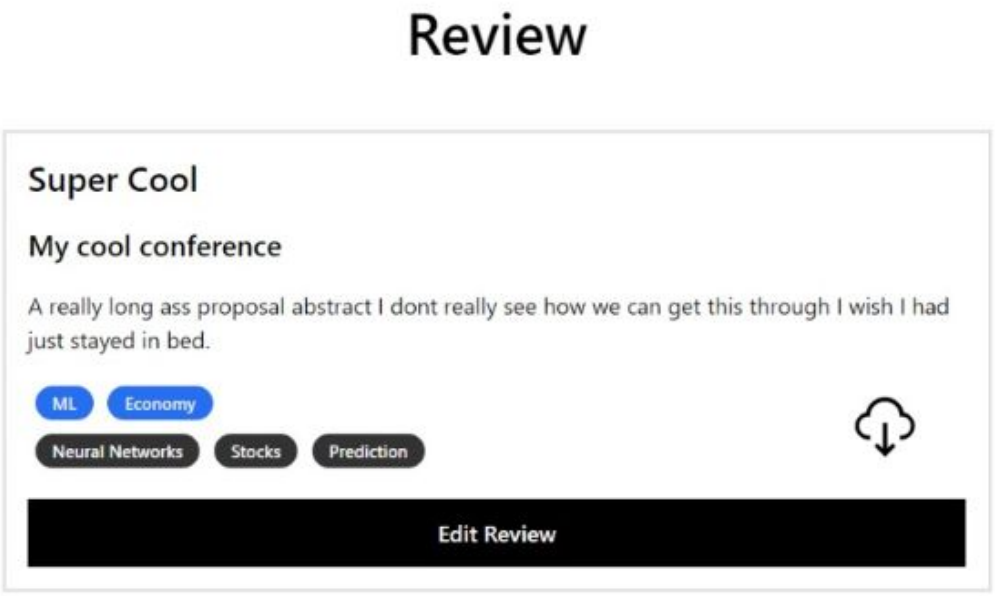
Upload presentation :

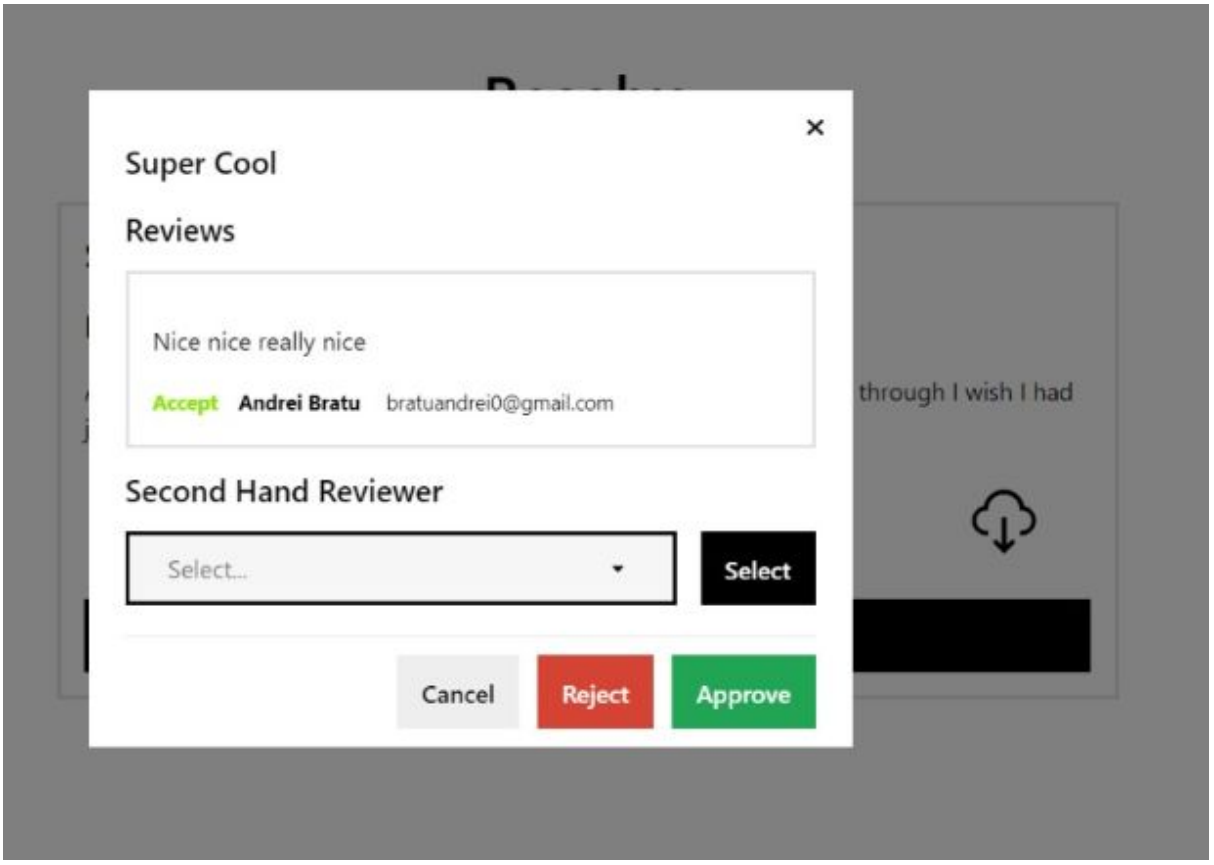


Assign reviewers to proposals :



Writing a review with the possibility of editing it later :





Bidding process: