Table of Contents

1. Software Engineering Design

1.1. Architecture Design

1.1.1. Candidate Architectures

Below are a series of architectures that were considered for our project, along with the justification for using each architecture.

1.1.1.1. Three-Tier Architecture

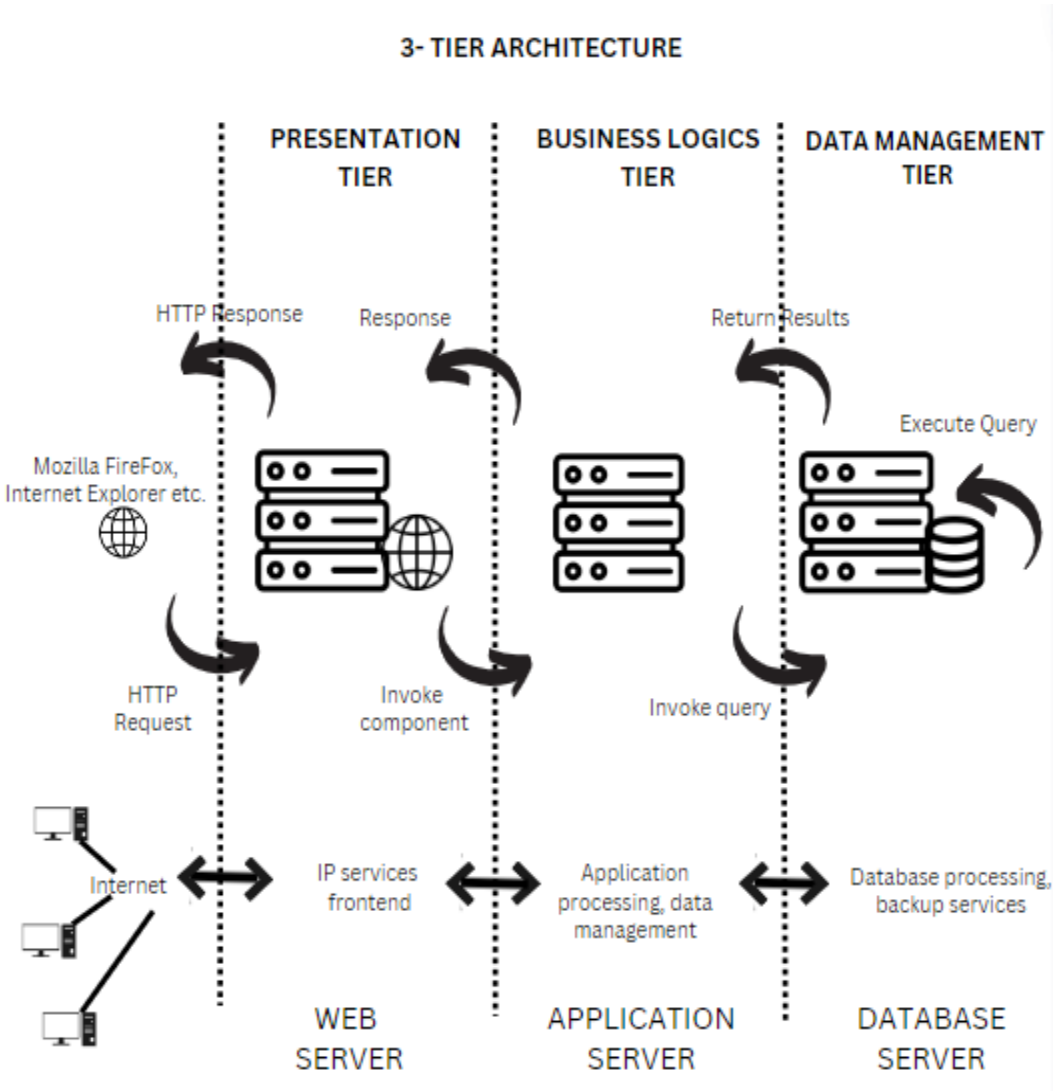


Figure 1: Diagram for Three-Tier Architecture

Justification:

The 3 tier architecture is a client-server architecture, in which tier represents physical separation and layer represents logical separation. The presentation tier, or user interface, the application tier, where data is processed, and the data tier, where the application's associated data is stored and managed, are the three logical and physical computing tiers that make up the three-tier architecture.

The tiers in web development have many names but serve the same purposes:

The presentation tier and user interface are provided by the web server. Typically, this is a web page or website where a user enters payment information, or sets up an account. HTML, CSS, and Javascript are typically used in the development of the content, which might be static or dynamic.

The business logic used to process user inputs is housed on the application server, which is equivalent to the middle tier. In keeping with the previous e-commerce example, this layer is responsible for adding information to customer profiles or querying the inventory database to determine whether a product is available. This layer typically runs a framework and was created using Python, Ruby, or PHP.

The data, or backend, tier of a web application is the database server. It utilizes database management systems like MySQL, Oracle, DB2, or PostgreSQL etc.

The main advantage of a three-tier architecture is that each tier may be built concurrently by a different development team and can be updated or scaled as necessary without affecting the other tiers because each tier runs on its own infrastructure. Furthermore, it has increased reliability as the availability or performance of the other tiers is less likely to be affected by a failure in one tier. A well-designed application tier can act as a kind of internal firewall, preventing SQL injections and other dangerous vulnerabilities because the presentation tier and data tier cannot connect with each other directly. However, the cost of installation can be high and the software is more complex than 1 or 2 tiered architectures.

1.1.1.2. Thin Client-Server Architecture

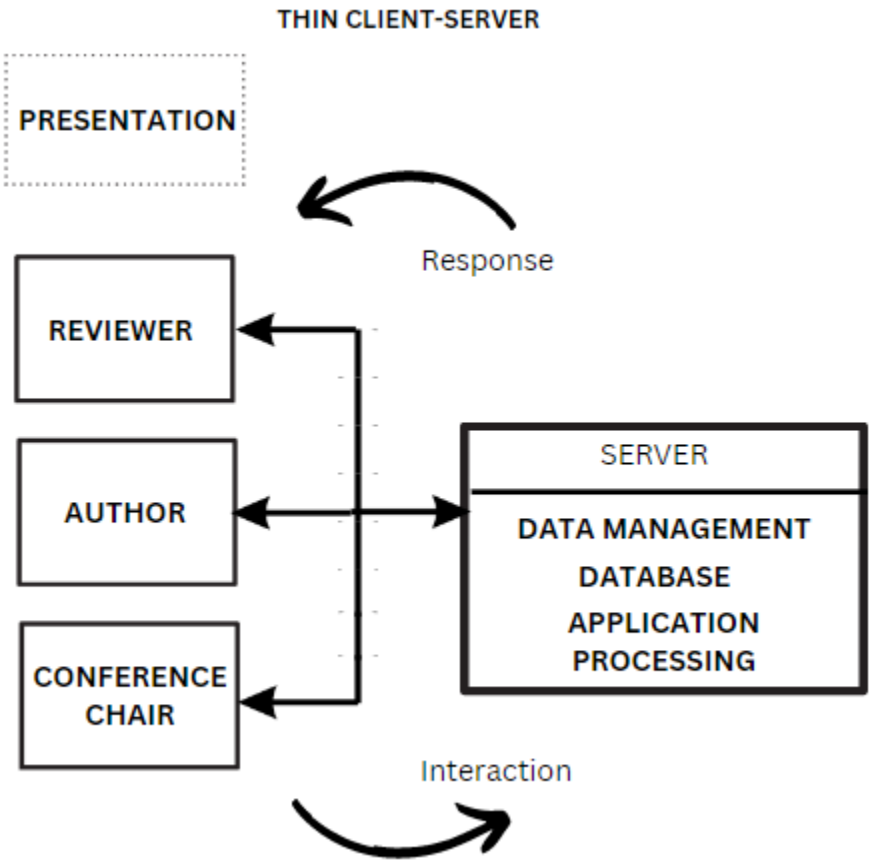


Figure 2: Diagram for Thin Client-Server Architecture

Justification:

A thin client-server architecture uses "thin clients" rather than typical clients and is a two tiered client-server architecture. A thin client, as the name suggests, has significantly fewer computational resources than a standard client (such as RAM and CPU), but the thin client server enables these thin clients to operate exactly like their traditional counterparts. Users can interact with the virtualized desktops and programs on the server just like they would with a traditional client by downloading them to each individual thin client. Utilizing a thin client server has the advantage of being largely centralized in terms of administration, maintenance, and security. Installation of new software or updates can be done on the server rather than on each individual client.

The presentation layer is client-side, whereas the server handles application processing, data storage, and data management. The system is more accessible because clients connected to the server don't need to have much processing capacity. Because a user only has to be logged in to access the system, it can be scaled quickly. As the client is controlled by the server and cannot access information without permission, the system will be more secure.

1.1.1.3. Multi-Layered Client-Server Architecture

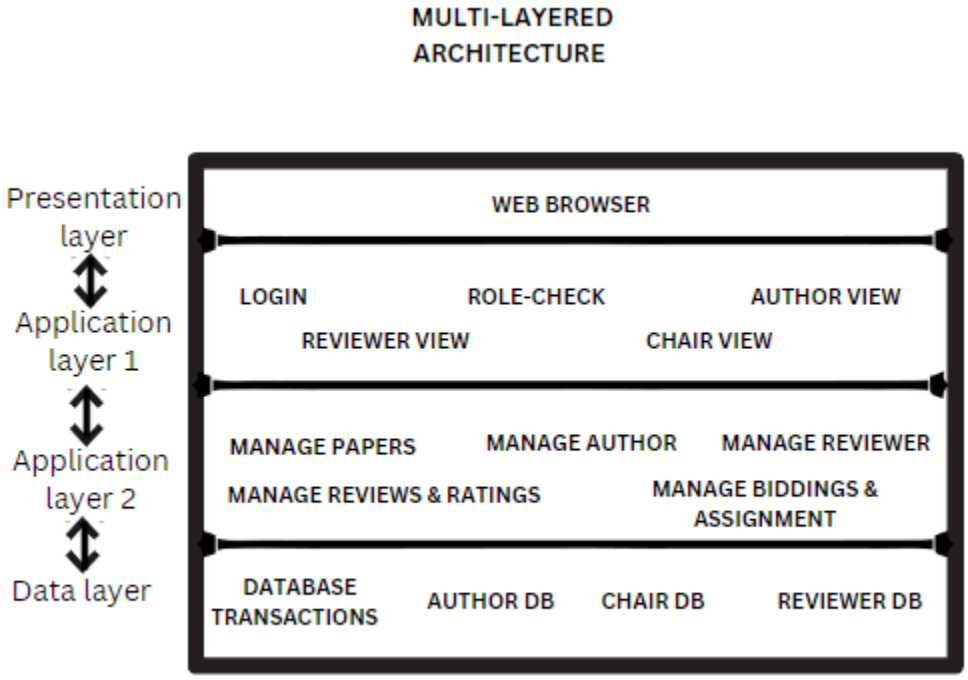


Figure 3: Diagram for Multi-Layered Client-Server Architecture

Justification:

In multi-layered architecture, the system can be divided into layers, and as long as each layer still communicates directly with the adjacent layers, it can be altered easily. It enables the management of software infrastructure in a simpler manner. The presentation layer and data layer are still present in a multi-layered software architecture. Simply put, the application layer is divided and expanded. These additional components are essentially various services that are included in the application layer. The presentation layer offers presentation services, i.e., the GUI-based presentation of content to the user which can be accessed by any client device, such as a desktop, laptop, tablet, smartphone, thin client, etc. The application tier is where the application's business logic is implemented. The storage and retrieval of application data are the primary concerns of

the data layer. This implies that the software should now be more scalable and have more functional dimensions.

The multi-layered technique is particularly effective for reasonably risk-free development and quick hosting of web-scale, production-grade, and cloud-hosted apps. Additionally, layering the architecture makes updating any old systems easier because the necessary changes should be less involved than they otherwise would be. As layers are unable to access resources they do not need to, layering the system can potentially reduce availability and performance while increasing security.

1.1.1.4. Model-View-Controller

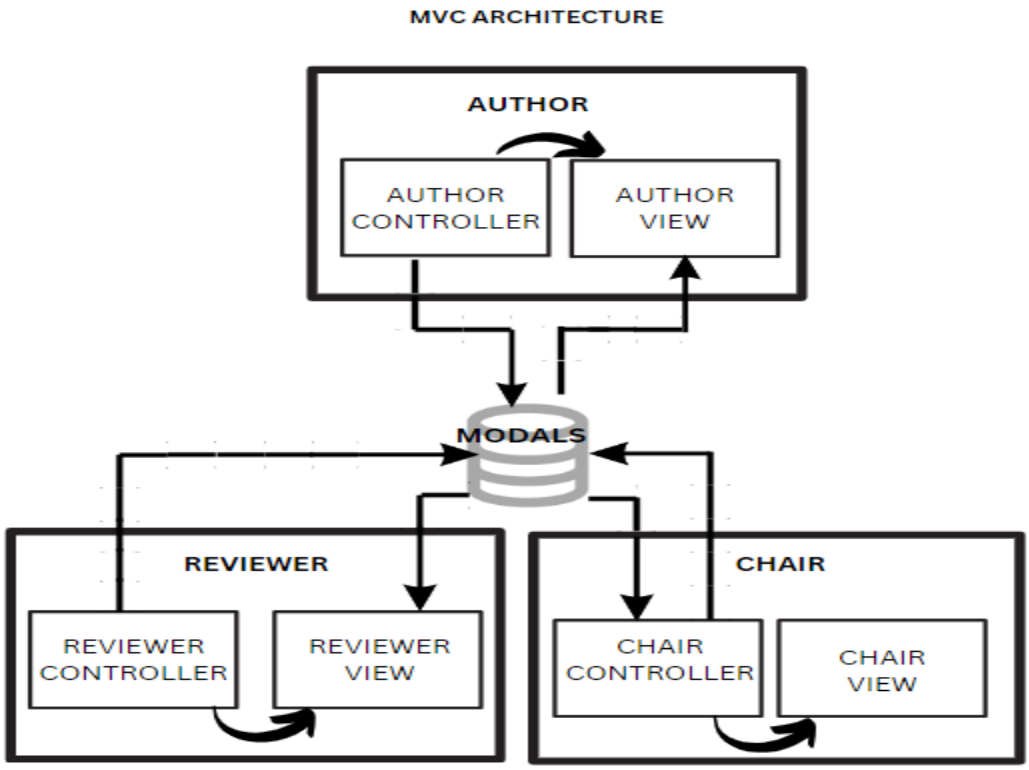


Figure 4: Diagram for Model-View-Controller

Justification:

MVC is an architecture invented by Trygve Reenskau that divides software into smaller components. Model, View, and Controller make up the three components of the MVC architectural pattern. The Model deals with data logic, and the View gives the user access to the model's data.

Every time data changes, the view is updated by the controller, which manages the data flow into a model object.

The three primary user types in the system are Author, Reviewer and Conference Chair. Each module has the potential to be modified, allowing for future maintainability. This "division" promotes readability and modularity while also making the testing process simpler. Although it has advantages like reusability of components and easy maintenance, the architecture can also be complex.

1.1.1.5. Repository Architecture

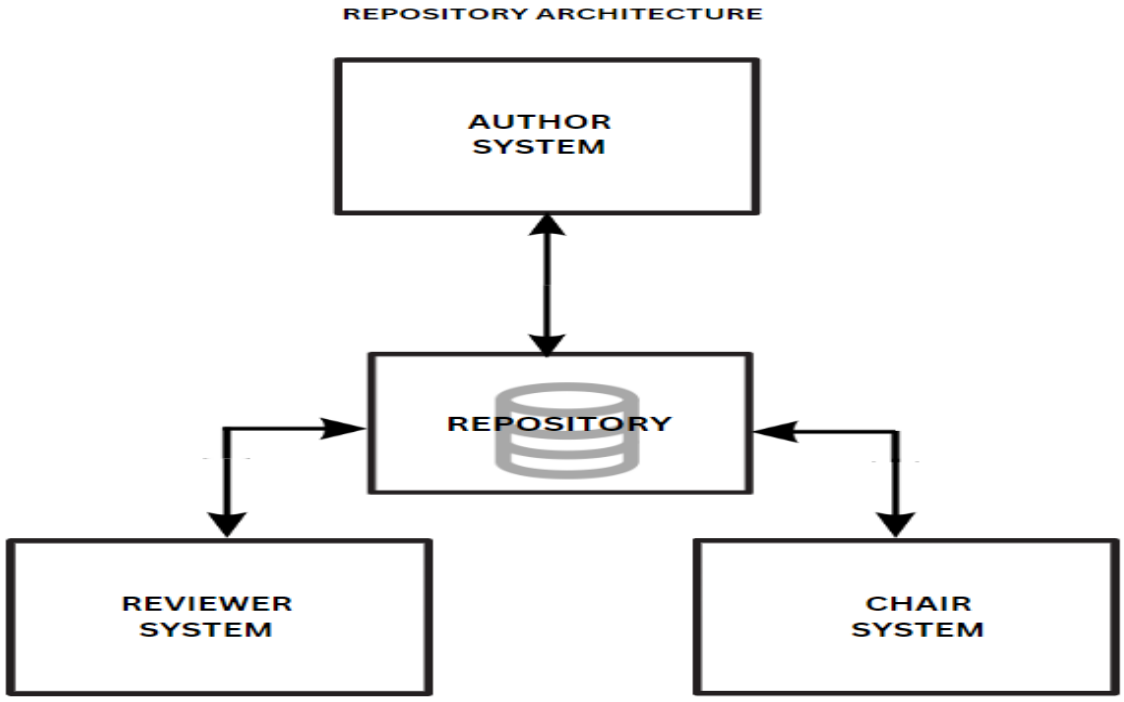


Figure 5: Diagram for Repository Architecture

Justification:

Data access logic is divided by the repository pattern, which then maps it to the business entities used in the business logic. The business logic and data access logic can communicate using interfaces. The clients (Author, Reviewer, and Chair) of the data store are active and govern the logic flow while the data store or repository itself is passive. The involved components periodically examine the data store for modifications. To conduct activities (such as insert data), the client

sends a request to the system. The independent computational processes are triggered by incoming requests.

Repository pattern, to put it simply, is a type of container where data access logic is kept. Business logic is shielded from the specifics of data access logic. Business logic can access the data object without requiring it to be aware of the underlying data access architecture. It is crucial that the data in this system be accurate and reliable. It has many benefits, including easy maintenance of coding as the data access mechanism is centralized. Features for data integrity, backup, and restoration are provided. Because the agents are not in direct contact with one another, this allows for scaling and reuse.

However, if the repository fails, the entire system will also fail, which will reduce the system's dependability. The possibility of data replication or duplication exists, and it is more prone to failure.

1.1.2. Final Architecture Design

The final selected design is three-tier architecture where the client is a “thin-client”.

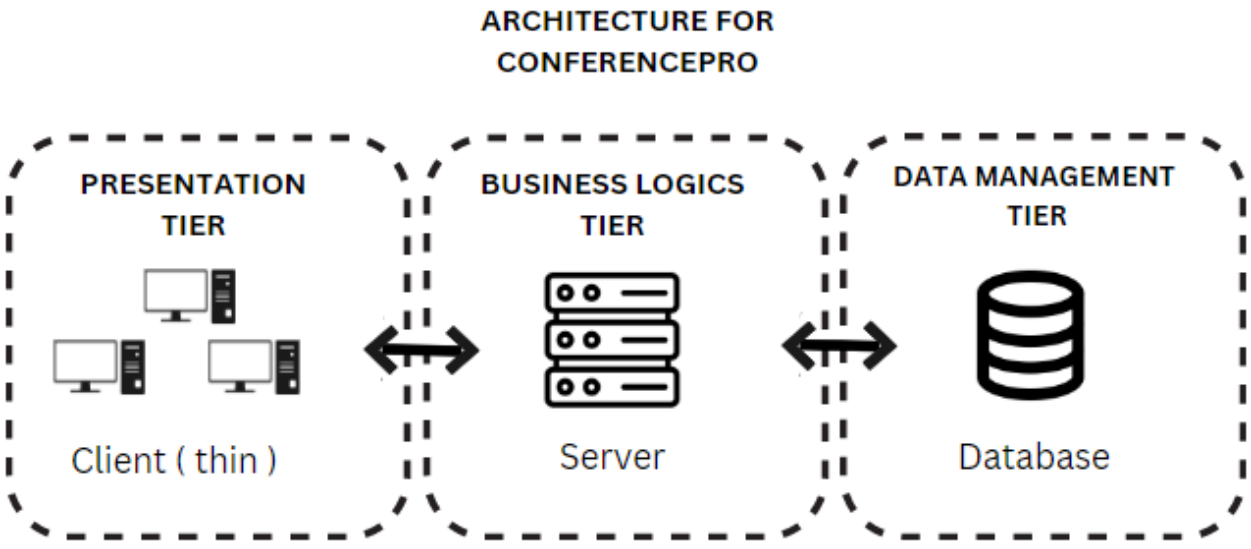


Figure 6: Diagram for Final Architecture Design

Justification:

The client-side is where the view is located. For the three various user categories who will use the system, there are three separate views for Author, Reviewer and Conference Chair. The thin client is a streamlined computer that uses a different server to run programmes and carry out user operations. This is suitable for the proposed research conference system as any of the client functionality doesn't involve much processing. Furthermore, it offers simplified management as the updates and plugins can be done at the server level and scaling of additional workstations can be done within minutes.The security is also enhanced as there are restrictions on who can add new programmes to the network system at the server level.

REST requests are used by the back-end, which is a Spring Boot application, to connect with the front-end application. To update or retrieve data to deliver to the front-end, the back-end accesses the database. The database is a noSQL solution hosted in the cloud by MongoDB.

1.2. Finalized User Stories

1.2.1. User Management Subsystem

- **Create account:** As an author/reviewer, I want to create an account so that I can access services offered by the system.
- **Login:** As a user, I want to login to my account so that I can access services offered by the system.

1.2.2. Resources Management Subsystem

1.2.2.1. Papers Management

- **Submit paper:** As an author, I want to submit a paper to the conference so that it can be assessed.
- **View paper submissions :** As an author, I want to view all the papers I submitted so that I can check their status.
- **View all papers:** As a conference chair, I want to view all papers so that I can assign them to bidders.
- **View assigned papers:** As a reviewer, I want to view all papers assigned to me so that I can review their content.

1.2.2.2. Reviewers Management

- **Bid for paper:** As a reviewer, I want to bid for a paper so that I can be assigned to rate and review the paper.
- **Add review preferences:** As a reviewer, I want to select my topics of expertise and workload preference so that my bid to review the paper is considered.
- **View paper biddings:** As a conference chair, I want to be able to see a list of reviewers who bidded for a paper along with their topic and workload preferences, to see which suitable reviewers are available to be assigned a paper.
- **Assign reviewer:** As a conference chair, I want to be able to assign a set of reviewers to a paper so they can review it.

1.2.3. Conference Activities Management Subsystem

1.2.3.1. Reviewing & Rating Processes Management

- **Rate paper:** As a reviewer, I want to rate a paper so the paper can be assessed for a conference.
- **Review paper:** As a reviewer, I want to review a paper so the paper can be assessed for a conference.
- **Edit rating:** As a reviewer, I want to be able to edit the rating I saved for a paper to make changes.
- **Edit review:** As a reviewer, I want to be able to edit the review I saved for a paper to make changes.
- **View previous reviews:** As a reviewer, I want to be able to view past reviews for a paper so that I can analyze them.
- **Give review feedback:** As a reviewer, I want to be able to give feedback on a past review so that I can discuss the review.

1.2.3.2. Decision Making & Notification Management

- **View paper ratings & reviews:** As a conference chair, I want to see the ratings and reviews for a paper so that I can make a decision to accept or reject it.
- **Accept/Reject paper:** As a conference chair, I want to accept or reject a paper based on the ratings and reviews for the paper so that I can inform the author of the decision.
- **View paper outcome:** As an author, I want to receive an email regarding the final decision for my paper so that I can know if it has been accepted or rejected.
- **View paper ratings & reviews:** As an author, I want to see the ratings and reviews for my paper so that I use the feedback to improve my paper.
- **Rate Review:** As an author, I want to rate the review provided by a reviewer for my paper so that I can inform the reviewer if I am satisfied or dissatisfied with the review.

1.3. Finalized Use Cases

1.3.1. User Management Subsystem

Use Case 1.1 - Create Account			
Actors		Author, Reviewer	
Stakeholders		Author, Reviewer	
Preconditions		• An account for the user doesn't already exist • No accounts are currently logged in on the user's device	
Postconditions		• A new account is created for the user.	
Trigger		User chooses option to create an account	
Normal Flow		**Actor**	**System**
	1.	User clicks 'Sign Up' button	Displays registration form
	2.	Fills form with personal information	
	3.	Clicks 'Submit' button	Shows preview of the user's information
	4.	Confirms details	Confirms successful registration
Exceptions		3. Form is incomplete 3. Form field is invalid 3. User already exists	

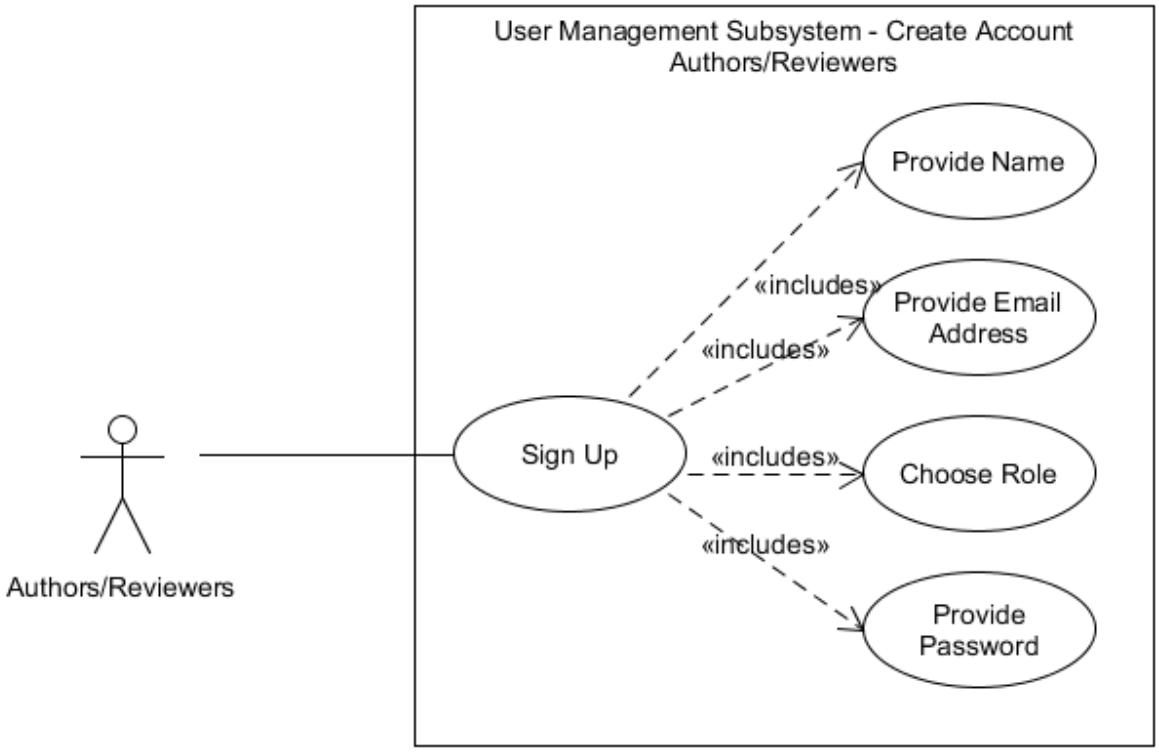


Figure 7: Use Case Diagram 1 - Create Account

Use Case 1.2 - Login			
Actors		Author, Reviewer, Conference chair	
Stakeholders		Author, Reviewer, Conference chair	
Preconditions		• A user must have an active account. • A user must not already be logged in.	
Postconditions		• A user will no longer be given the option to log in or create an account. • A user will be given the option to log out.	
Trigger		User chooses 'Login' option	
Normal Flow		**Actor**	**System**
	1.	User clicks 'Login' button	Displays login form
	2.	Enters email and password	

	3.	Clicks 'Submit' button	Confirms successful login
Exceptions	3. Incorrect email ID 3. Incorrect user password		

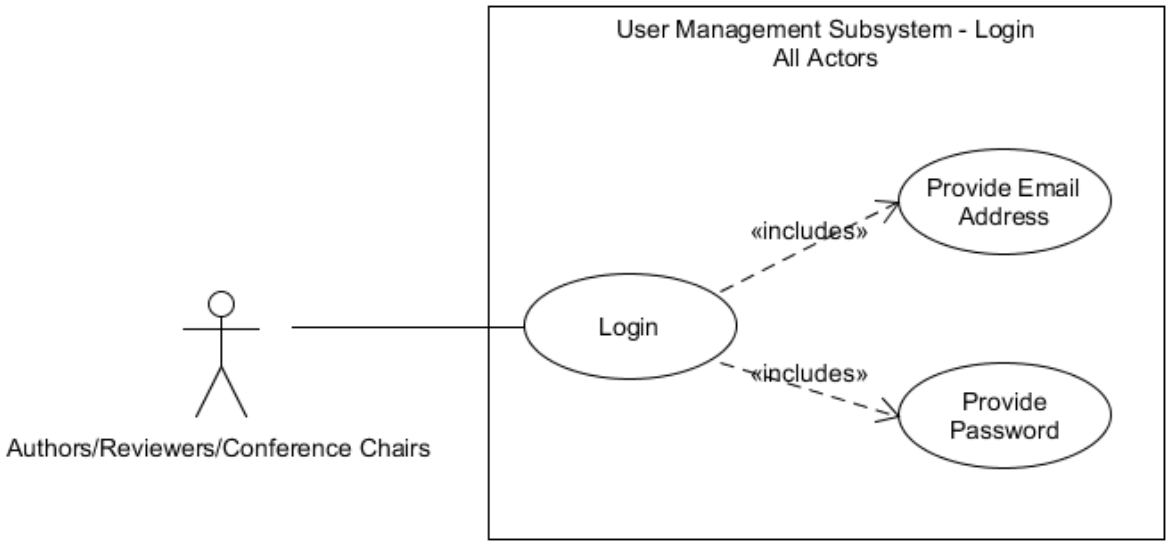


Figure 8: Use Case Diagram 2 - Login

1.3.2. Resources Management Subsystem

1.3.2.1. Papers Management

Use Case 1.3 - Submit Paper			
Actors	Author		
Stakeholders	Author		
Preconditions	• User is logged in as an author. • User has a submission to make.		
Postconditions	• User's paper submission is accepted and added to the database.		
Trigger	An author wants to submit a paper to a conference.		
Normal Flow		**Actor**	**System**
	1.	Author clicks 'New Submission'	Displays submission Terms & Conditions

		button	
	2.	Accepts the Terms & Conditions.	Displays submission form
	3.	Enters title, abstract and authors(s) details	Displays list of topics
	4.	Selects relevant topics of the paper.	Shows preview of entered information
	5.	Clicks 'Confirm submission'	Submission is successful
Exceptions		2. Author does not accept Terms & Conditions 3. Author does not enter all required information 4. Author does not select topics	

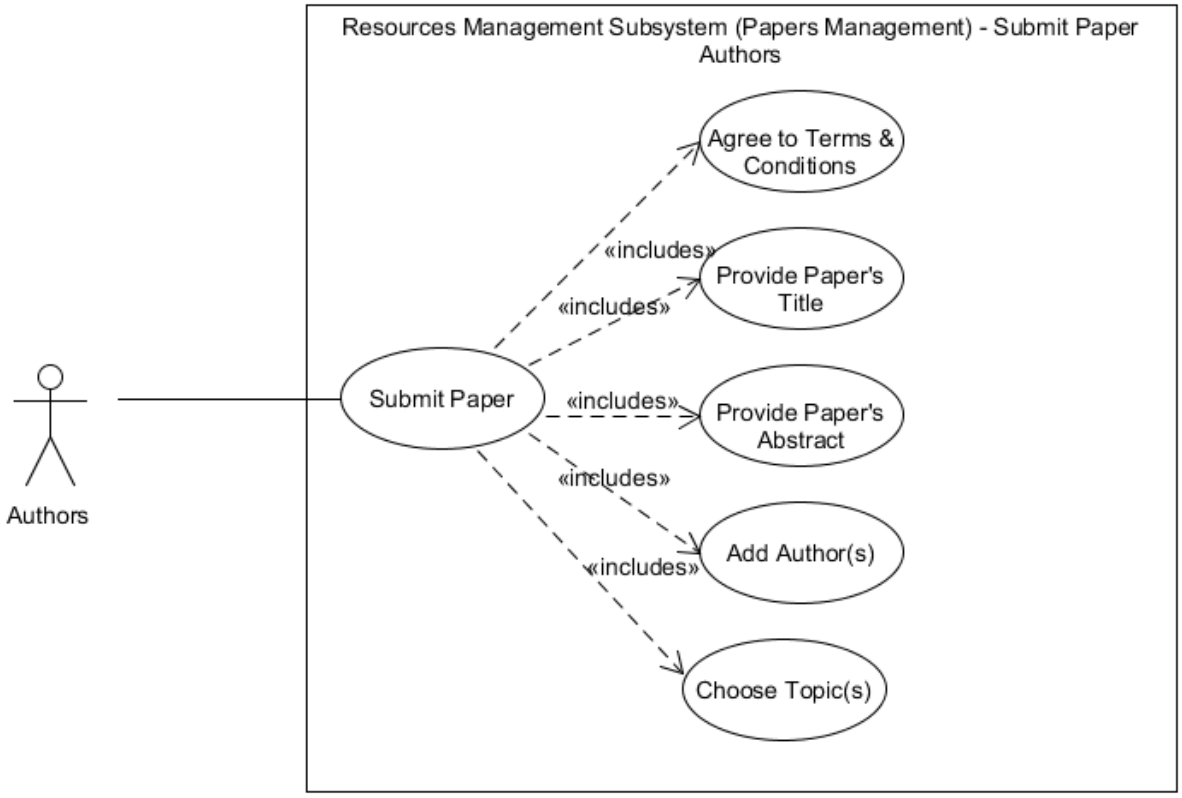


Figure 9: Use Case Diagram 3 - Submit Paper

Use Case 1.4 - View All Submissions	
Actors	Author

Stakeholders		Author	
Preconditions		• User is logged in as an author. • User has made at least one submission.	
Postconditions		None.	
Trigger		An author wants to view all papers they have submitted.	
Normal Flow		**Actor**	**System**
	1.	Author clicks "My Submissions" button	Retrieves the reviewed and pending review papers that have been submitted by the author
	2.		Displays the reviewed and pending review papers that have been submitted by the author on screen
Exceptions		1. Author has not made any submissions.	

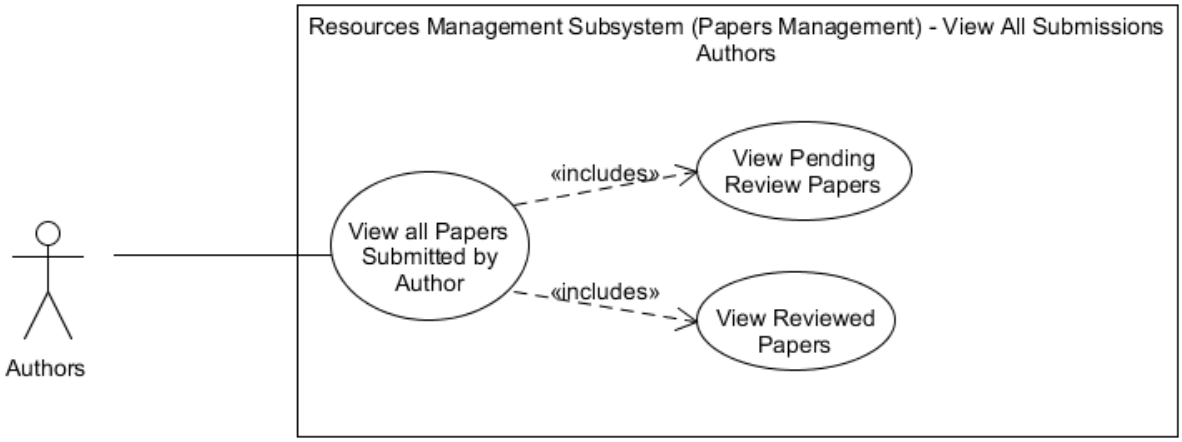


Figure 10: Use Case Diagram 4 - View All Submissions

Use Case 1.5 - View All Papers	
Actors	Conference Chair
Stakeholders	Conference Chair
Preconditions	• User is logged in as a chair.

		• At-least one paper must be submitted to the conference led by the chair	
Postconditions		None.	
Trigger		A chair wants to see a list of all papers submitted to a conference.	
Normal Flow		**Actor**	**System**
	1.	Chair clicks “Papers” button	Retrieves assigned and unassigned papers
	2.		Displays a list of assigned and unassigned papers on the screen
Exceptions		1. No paper has been submitted to the conference.	

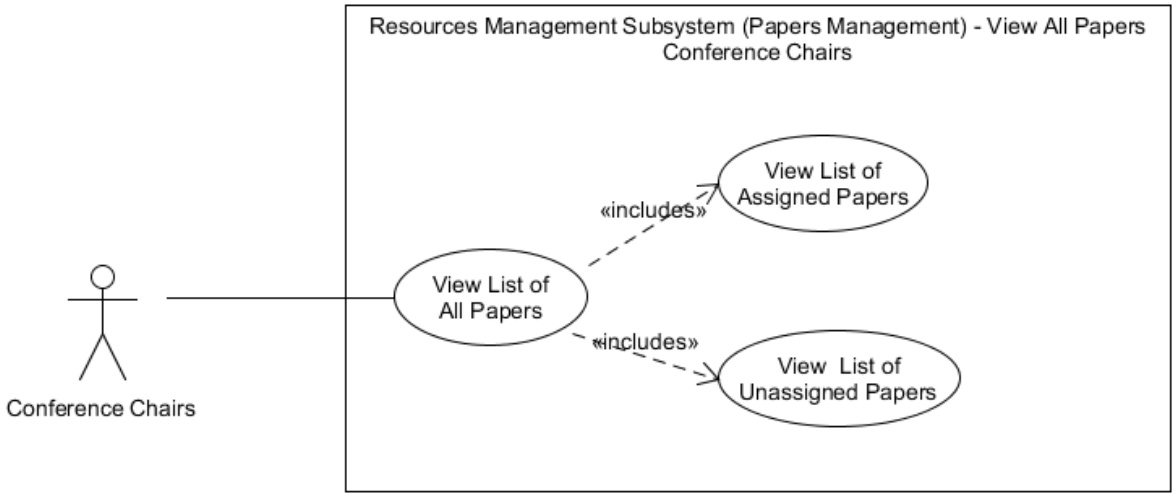


Figure 11: Use Case Diagram 5 - View All Papers

Use Case 1.6 - View Assigned Papers	
Actors	Reviewer
Stakeholders	Reviewer
Preconditions	• User is logged in as a reviewer. • At-least one paper is assigned to the reviewer.
Postconditions	None.

Trigger		A reviewer wants to see all papers that were assigned to them.	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks "My Reviews" button	Retrieves and displays a list of all papers assigned to the reviewer and their review status.
	2.	Clicks "View Paper" button	Retrieves and displays all paper information
Exceptions		1. No paper has been assigned to the reviewer	

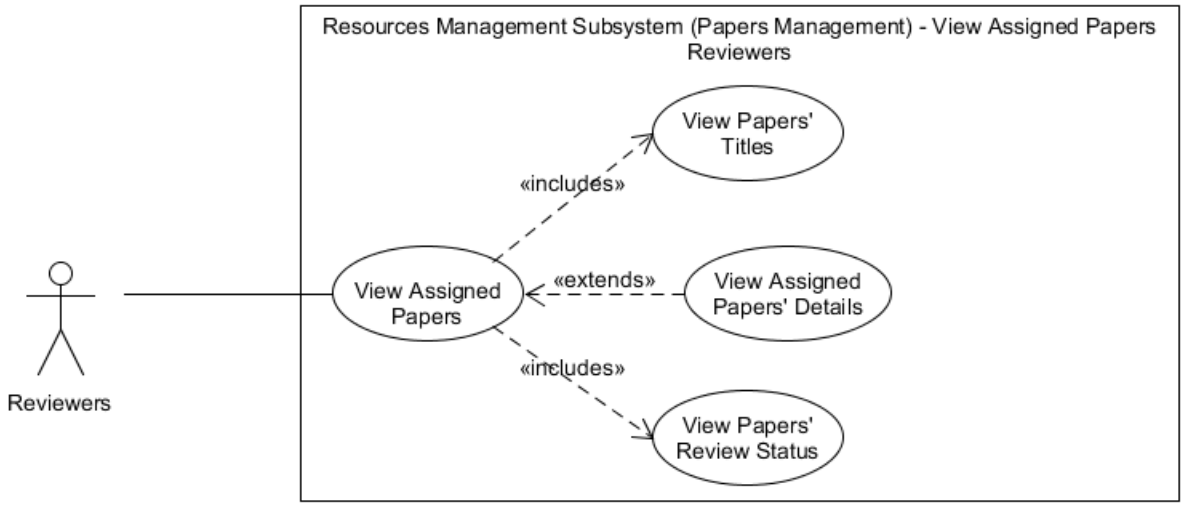


Figure 12: Use Case Diagram 6 - View Assigned Papers

1.3.2.2. Reviewers Management

Use Case 1.7 - Bid for Paper	
Actors	Reviewer
Stakeholders	Reviewer
Preconditions	• User is logged in as a reviewer. • Papers filtered according to reviewers workload and topic preferences are displayed.

Postconditions		None.	
Trigger		A reviewer wants to bid to review a specific paper.	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks “Papers” button	Retrieves and displays a list of all papers matching reviewers preferences
	2.	Clicks “View Abstract” button	Retrieves and displays paper abstract
	3.	Clicks “Bid for paper” button	Sends request reviewers request to bid to chair
Exceptions		1. No paper is available that matches the reviewer’s workload and topic preferences.	

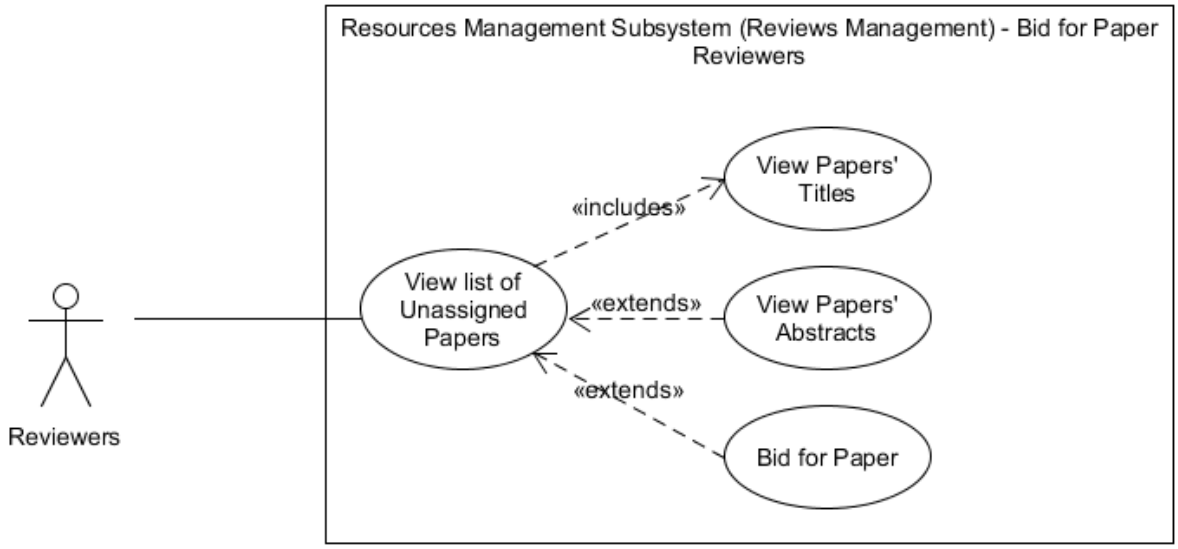


Figure 13: Use Case Diagram 7 - Bid for Paper

Use Case 1.8 - Edit Review Preferences	
Actors	Reviewer
Stakeholders	Reviewer
Preconditions	• User is logged in as a reviewer.

Postconditions		• Updated user preferences are saved in the database.	
Trigger		A reviewer wants to add their workload and topics preferences.	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks on the 'Edit Preferences' button.	Displays form to update preferences
	2.	Fills form with workload and topic preferences	
	3.	Clicks "Save Preferences" button	Saves updated preferences
Exceptions		2. Workload entered is greater than 10 2. No topics of interest are selected	

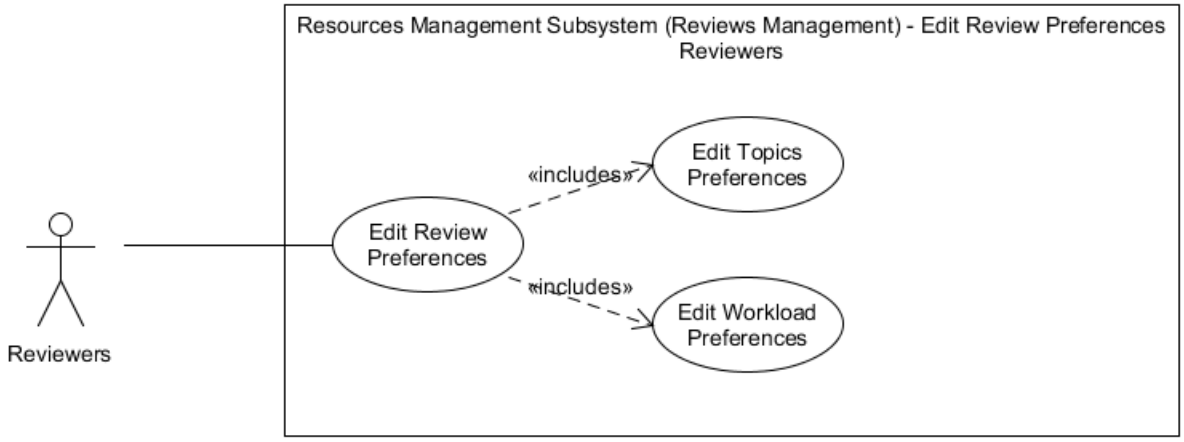


Figure 14: Use Case Diagram 8 - Edit Review Preferences

Use Case 1.9 - View Paper Biddings	
Actors	Conference Chair
Stakeholders	Conference Chair
Preconditions	• User is logged in as a chair.

Postconditions		None	
Trigger		A wants to view all reviewers that have bidded for a paper.	
Normal Flow		**Actor**	**System**
	1.	Chair clicks the "All Papers" button.	Displays a list of assigned and unassigned papers
	2.	Clicks on "Unassigned papers" tab	Retrieves and displays a list of papers pending assignment
	3.	Clicks "Pending assignment" button	Displays details of the paper
	4.	Clicks "View Reviewers"	Displays a list of all reviewers that bidded for the paper along with their workload status and topic of interest
Exceptions		2. All papers have been assigned 4. No reviewer has bidded for the paper	

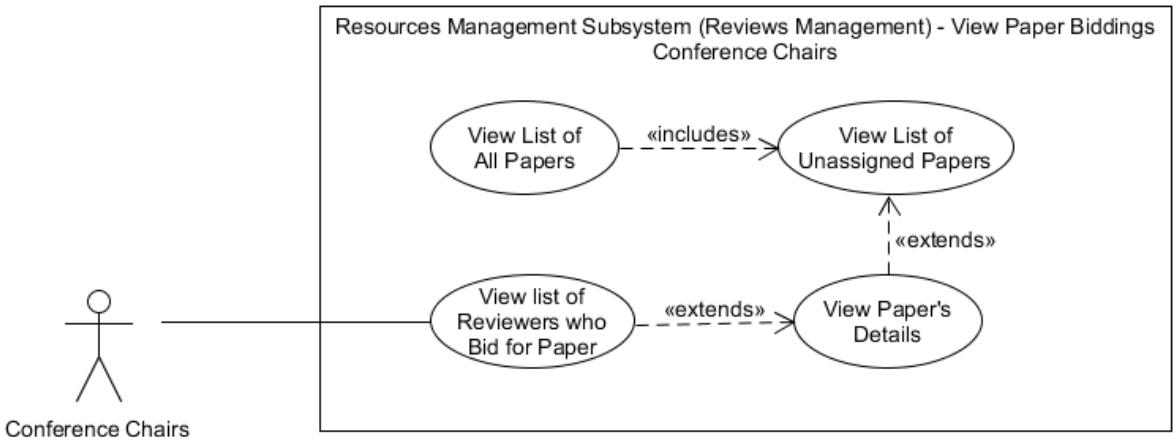


Figure 15: Use Case Diagram 9 - View Paper Biddings

Use Case 1.10 - Assign Reviewer	
Actors	Conference Chair
Stakeholders	Conference Chair

Preconditions		• User is logged in as a chair.	
Postconditions		None	
Trigger		A chair wants to assign a paper to one or more reviewers who have bidded for it.	
Normal Flow		**Actor**	**System**
	1.	Chair clicks the "All Papers" button.	Displays a list of assigned and unassigned papers
	2.	Clicks on "Unassigned papers" tab	Retrieves and displays a list of papers pending assignment
	3.	Clicks "Pending assignment" button	Displays details of the paper
	4.	Clicks "View Reviewers"	Displays a list of all reviewers that bidded for the paper along with their workload status and topic of interest
	5.	Selects suitable reviewer(s) by clicking the checkbox	
	6.	Clicks "Confirm" button	Notifies the reviewer(s) of a successful paper assignment.
Exceptions		2. All papers have been assigned 4. No reviewer has bidded for the paper	

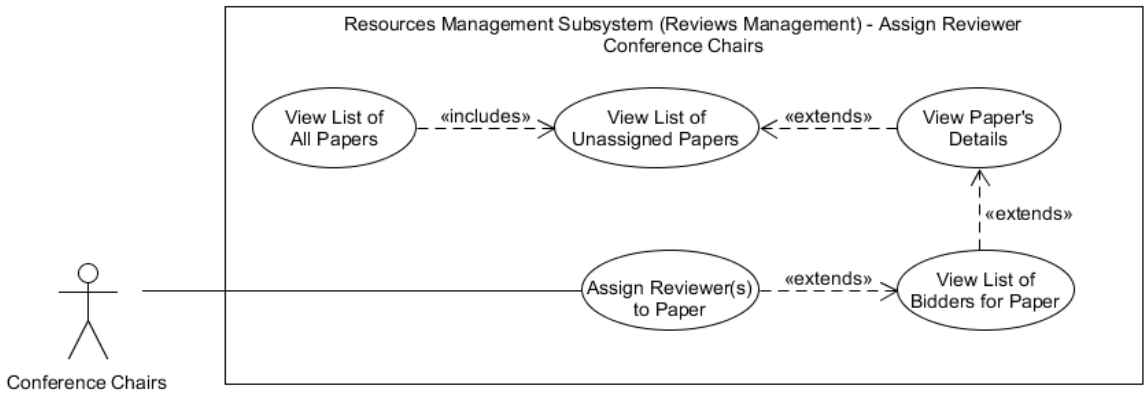


Figure 16: Use Case Diagram 10 - Assign Reviewer

1.3.3. Conference Activities Management Subsystem

1.3.3.1. Reviewing & Rating Processes Management

Use Case 1.11 - Rate & Review Paper			
Actors		Reviewer	
Stakeholders		Reviewer	
Preconditions		• User is logged in as a reviewer.	
Postconditions		• Review for paper is added to past reviews for the paper	
Trigger		A reviewer wants to rate & review a paper assigned to them.	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks the "My reviews" button.	Displays a list of papers assigned to the reviewer
	2.	Clicks "View Paper" button	Displays details of the paper
	3.	Clicks "Start Review" button	Displays a form with a scale of ratings between -3 and 3
	4.	Selects suitable rating for overall merit	Displays a textbox to add review comments
	5.	Types out the review for paper	
	6.	Clicks "Complete Review" button	Displays a message that informs the use that no changes can be made after submission
	7.	Clicks "Confirm Submission"	Review for the paper is submitted and paper is marked as "Reviewed"
Exceptions		1. No paper has been allocated to the reviewer. 5. No information is entered in the required field for comments.	

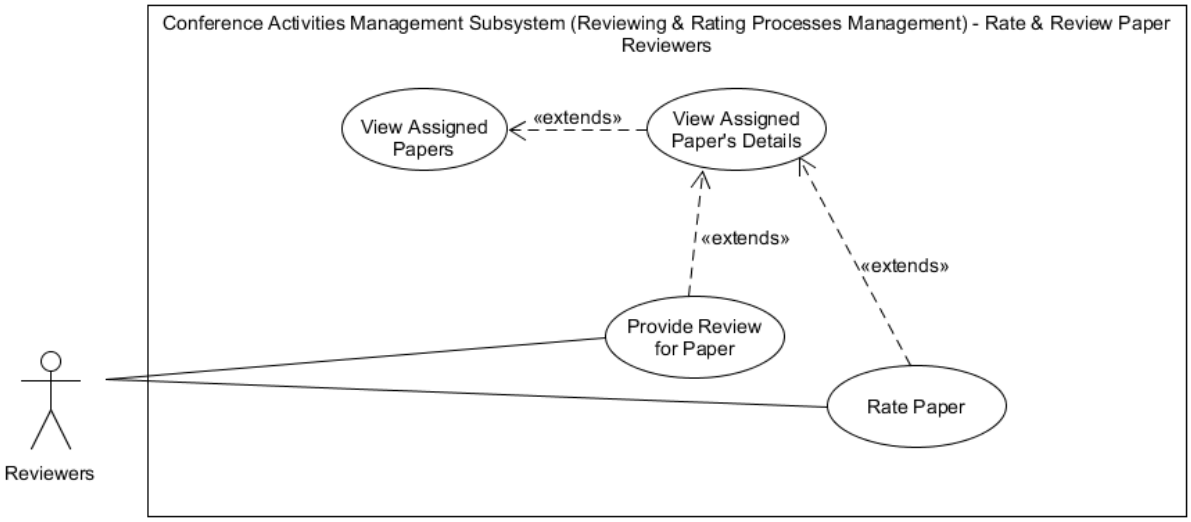


Figure 17: Use Case Diagram 11 - Rate & Review Paper

Use Case 1.12 - Edit Rating			
Actors		Reviewer	
Stakeholders		Reviewer	
Preconditions		• User is logged in as a reviewer.	
Postconditions		• Updated rating for the paper is saved.	
Trigger		A reviewer wants to edit a rating they had saved for a paper	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks the "My reviews" button.	Displays a list of papers assigned to the reviewer
	2.	Clicks "View Paper" button	Displays details of the paper
	3.	Clicks "Continue Review" button	Displays a form with a scale of ratings between -3 and 3
	4.	Selects a new rating for paper	

	5.	Clicks “Save & Close”	Saves the updated rating.
Exceptions		3. A review is not in progress for the paper.	

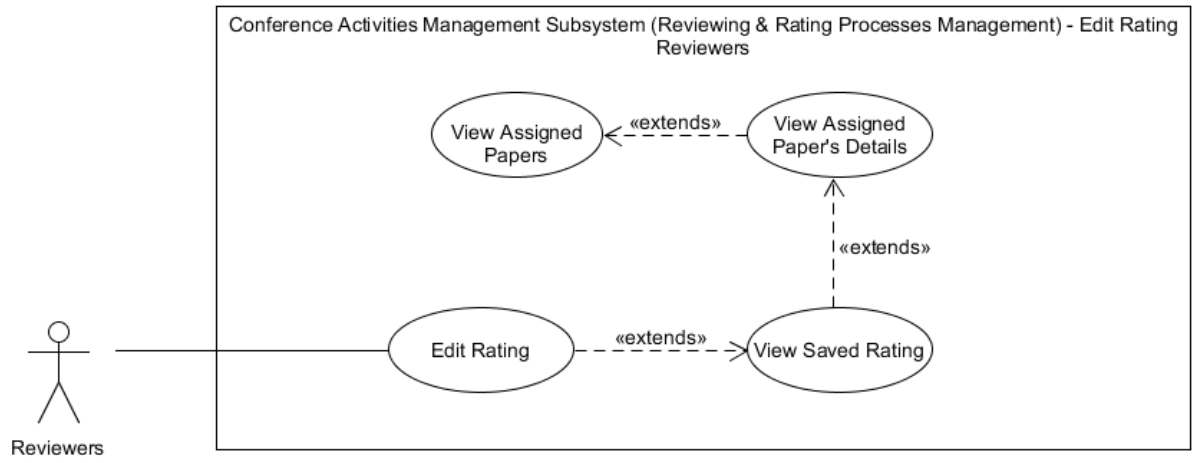


Figure 18: Use Case Diagram 12 - Edit Rating

Use Case 1.13 - Edit Review			
Actors		Reviewer	
Stakeholders		Reviewer	
Preconditions		• User is logged in as a reviewer.	
Postconditions		• Updated review for the paper is saved.	
Trigger		A reviewer wants to edit a review they had saved for a paper	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks the “My reviews” button.	Displays a list of papers assigned to the reviewer
	2.	Clicks “View Paper” button	Displays details of the paper
	3.	Clicks “Continue Review” button	Displays a textbox to add updated review comments

	4.	Enters updated review comments	
	5.	Clicks “Save & Close”	Saves the updated review.
Exceptions		3. A review is not in progress for the paper.	

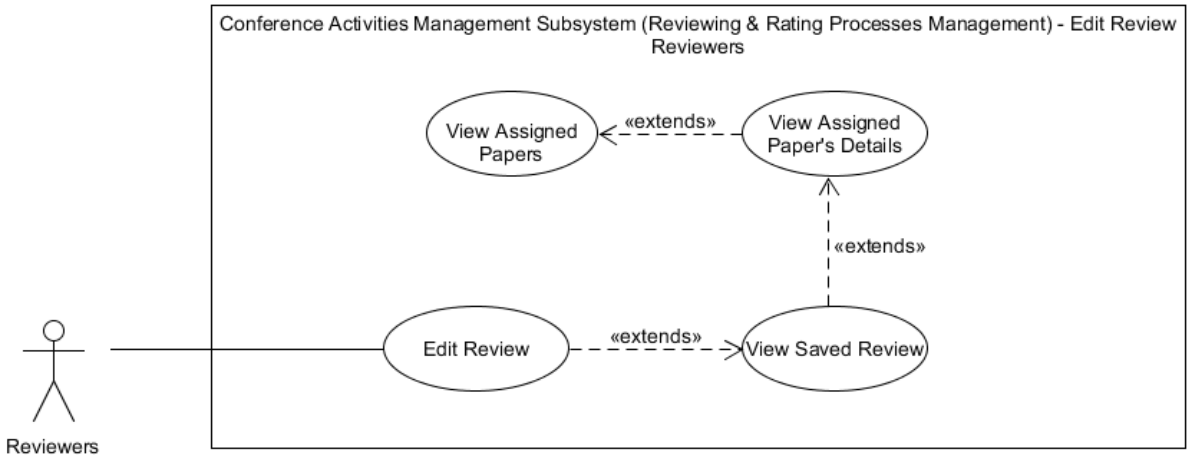


Figure 19: Use Case Diagram 13 - Edit Review

Use Case 1.14 - View Previous Reviews			
Actors		Reviewer	
Stakeholders		Reviewer	
Preconditions		• User is logged in as a reviewer.	
Postconditions		None	
Trigger		A reviewer wants to view past reviews for a paper after they have completed the paper review.	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks on “View Paper” for a paper that they have completed review for	Displays paper details and a list of all previous ratings and reviews

Exceptions	1. No past reviews for the paper exist

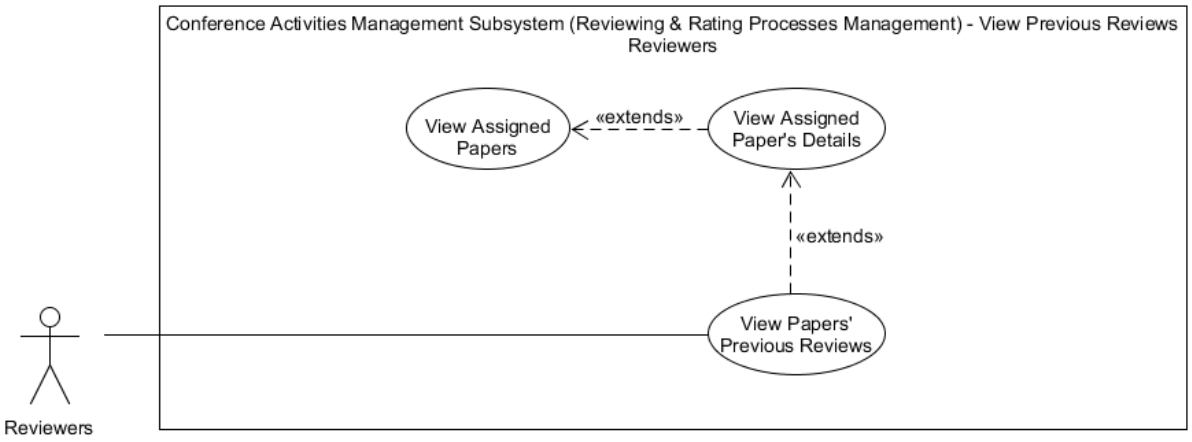


Figure 20: Use Case Diagram 14 - View Previous Reviews

Use Case 1.15 - Add Feedback for Past Review			
Actors		Reviewer	
Stakeholders		Reviewer	
Preconditions		• User is logged in as a reviewer.	
Postconditions		None	
Trigger		A reviewer wants to add a feedback comment to a previous review by a different author.	
Normal Flow		**Actor**	**System**
	1.	Reviewer clicks on "View Paper" for a paper that they have completed review for	Displays paper details and a list of all previous ratings and reviews
	2.	Clicks "Add Feedback"	Displays a form to enter feedback details
	3.	Clicks "Save"	Saves and displays the feedback added by the author
Exceptions		1. No past reviews for the paper exist	

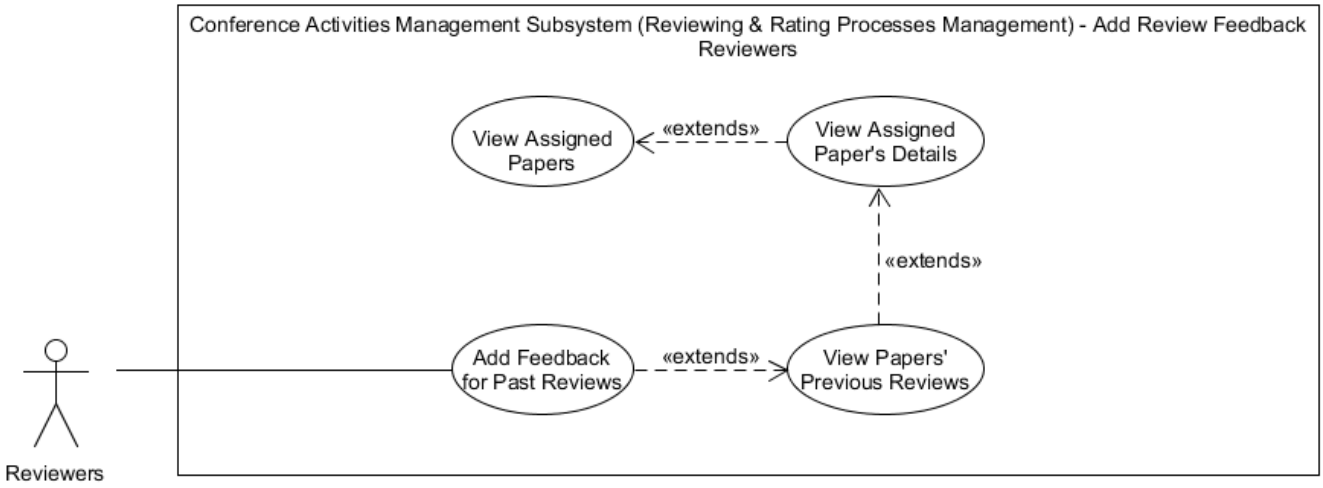


Figure 21: Use Case Diagram 15 - Add Review Feedback

1.3.3.2. Decision Making & Notification Management

Use Case 1.16 - View Paper Ratings & Reviews			
Actors		Conference Chair	
Stakeholders		Conference Chair	
Preconditions		• User is logged in as a chair.	
Postconditions		None	
Trigger		A chair wants to review all ratings and reviews for a paper provided by reviewers.	
Normal Flow		**Actor**	**System**
	1.	Chair clicks “Papers” tab	Displays a list of assigned papers
	2.	Clicks “Pending Decision” button	Displays details of the paper and a list of all ratings and reviews for the paper
Exceptions		2. The paper is pending a review.	

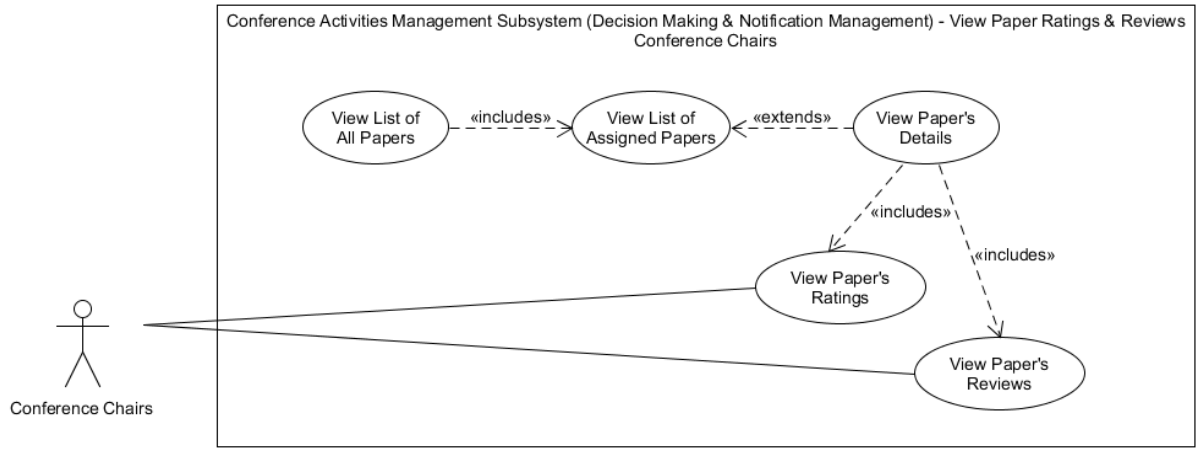


Figure 22: Use Case Diagram 16 - View Paper Ratings & Reviews

Use Case 1.17 -Accept/Reject Paper			
Actors		Conference Chair	
Stakeholders		Conference Chair	
Preconditions		• User is logged in as a chair.	
Postconditions		None	
Trigger		A chair wants to review all ratings and reviews for a paper provided by reviewers.	
Normal Flow		**Actor**	**System**
	1.	Chair clicks “Papers” tab	Displays a list of assigned papers
	2.	Clicks “Pending Decision” button	Displays details of the paper and a list of all ratings and reviews for the paper
	3.	Clicks “Accept” or “Reject” button	Displayed confirmation message and notifies author of decision
Exceptions		2. The paper is pending a review.	

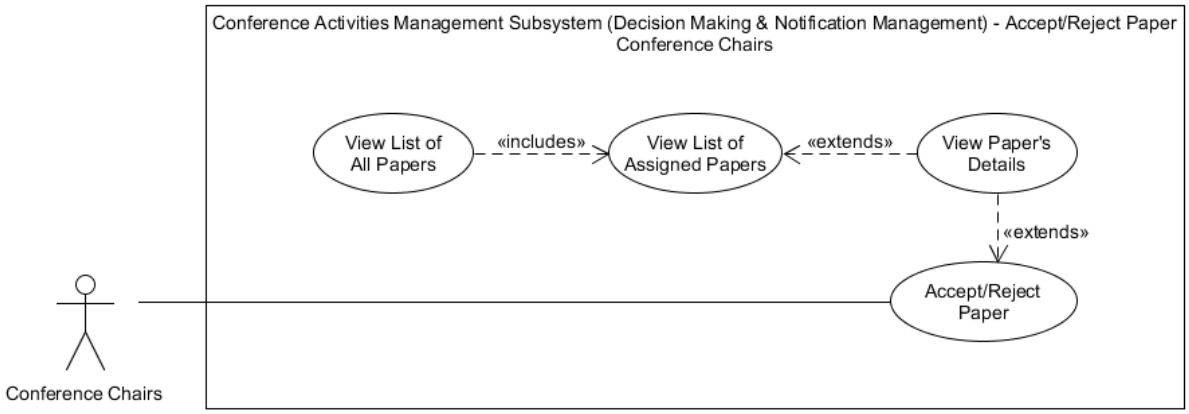


Figure 23: Use Case Diagram 17 - Accept/Reject Paper

Use Case 1.18 - View Submission Outcome			
Actors		Author	
Stakeholders		Author	
Preconditions		• User is logged in as an author.	
Postconditions		None	
Trigger		An author wants to know whether their submission was accepted or rejected.	
Normal Flow		**Actor**	**System**
	1.	Author clicks "My Submissions" tab	Retrieves and displays a list of all submissions along with their status (Pending Review/Reviewed)
	2.	Clicks "View Paper" button for a paper whose review is complete	Displays all Reviews and Rating given to the paper along with information about paper acceptance/rejection.
Exceptions		2. The paper is pending a decision.	

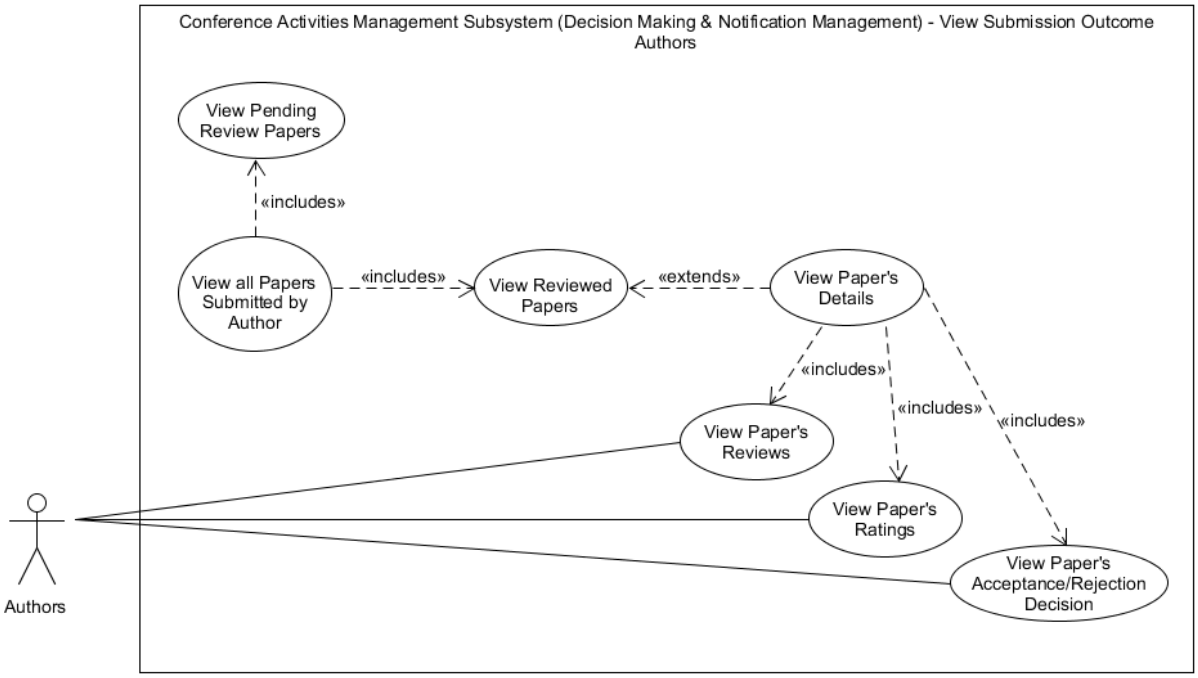


Figure 24: Use Case Diagram 18 - View Submission Outcome

Use Case 1.19 - Rate Review			
Actors		Author	
Stakeholders		Author	
Preconditions		• User is logged in as an author.	
Postconditions		None	
Trigger		An author wants to rate the review they have received for their submission	
Normal Flow		**Actor**	**System**
	1.	Author clicks on "My Submissions"	Retrieves and displays a list of all submissions along with their status (Pending Review/Reviewed)
	2.	Author clicks on "View paper" button for a paper that has been reviewed	Retrieves and displays a list of all ratings and reviews for the submission

	3.	Author rates the review as "Satisfactory" or "Unsatisfactory:	Saves the rating for the review.
Exceptions	None		

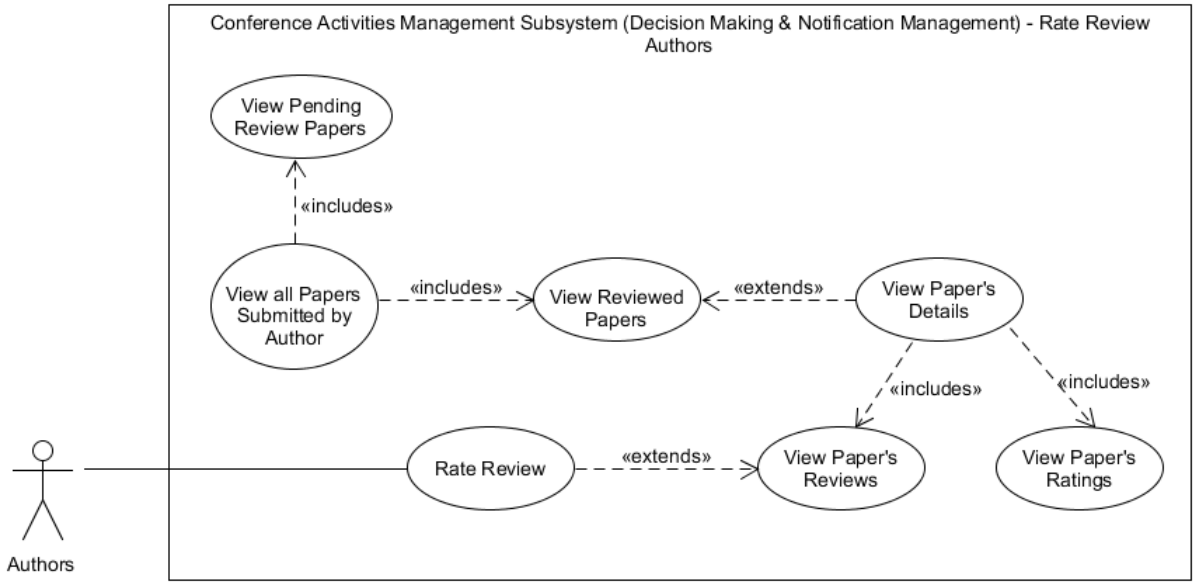


Figure 25: Use Case Diagram 19 - Rate Review

1.4. UML Class Diagram(s)

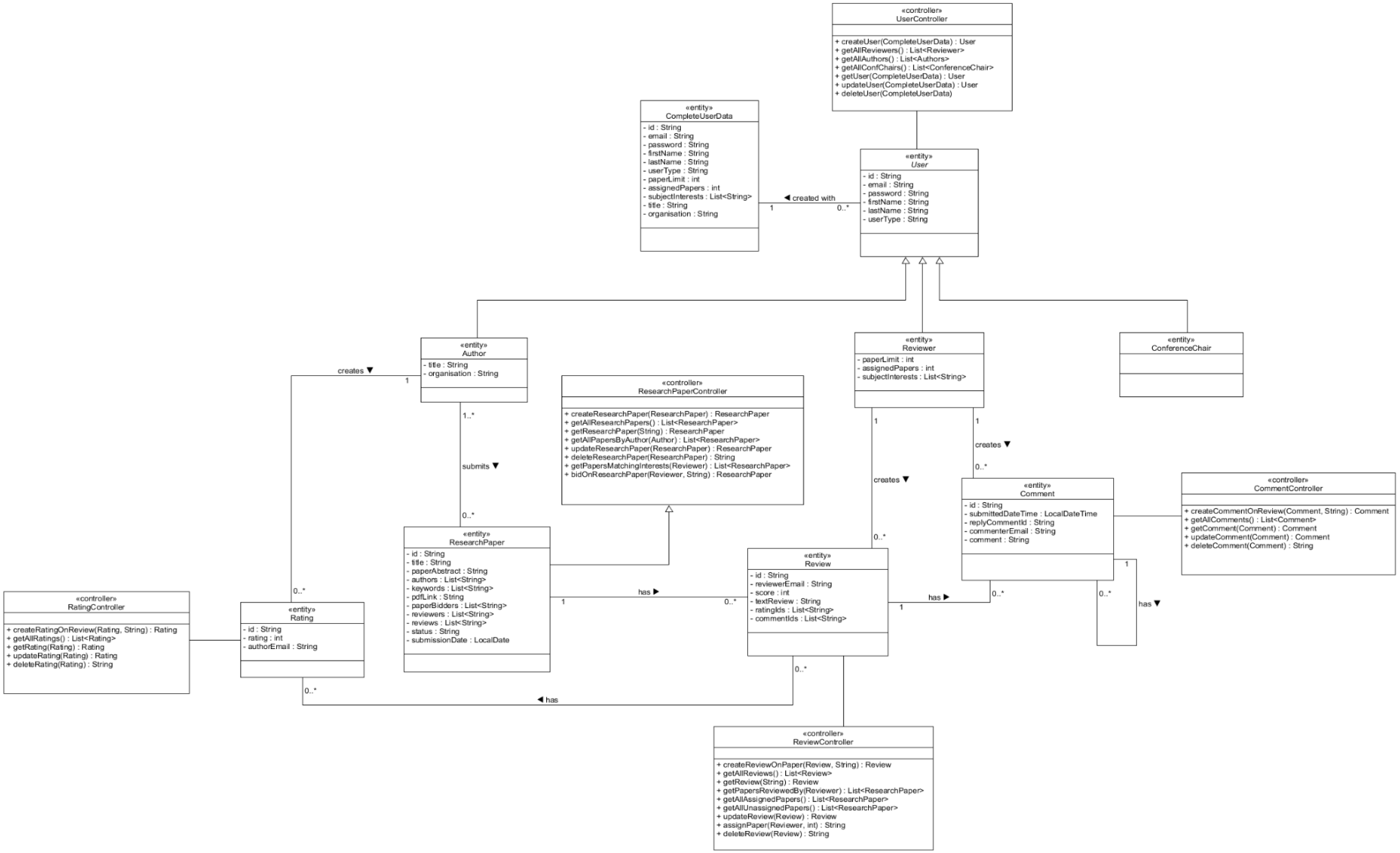


Figure 26: Class Diagram

1.5. UML Sequence Diagrams

1.5.1. User Management Subsystem

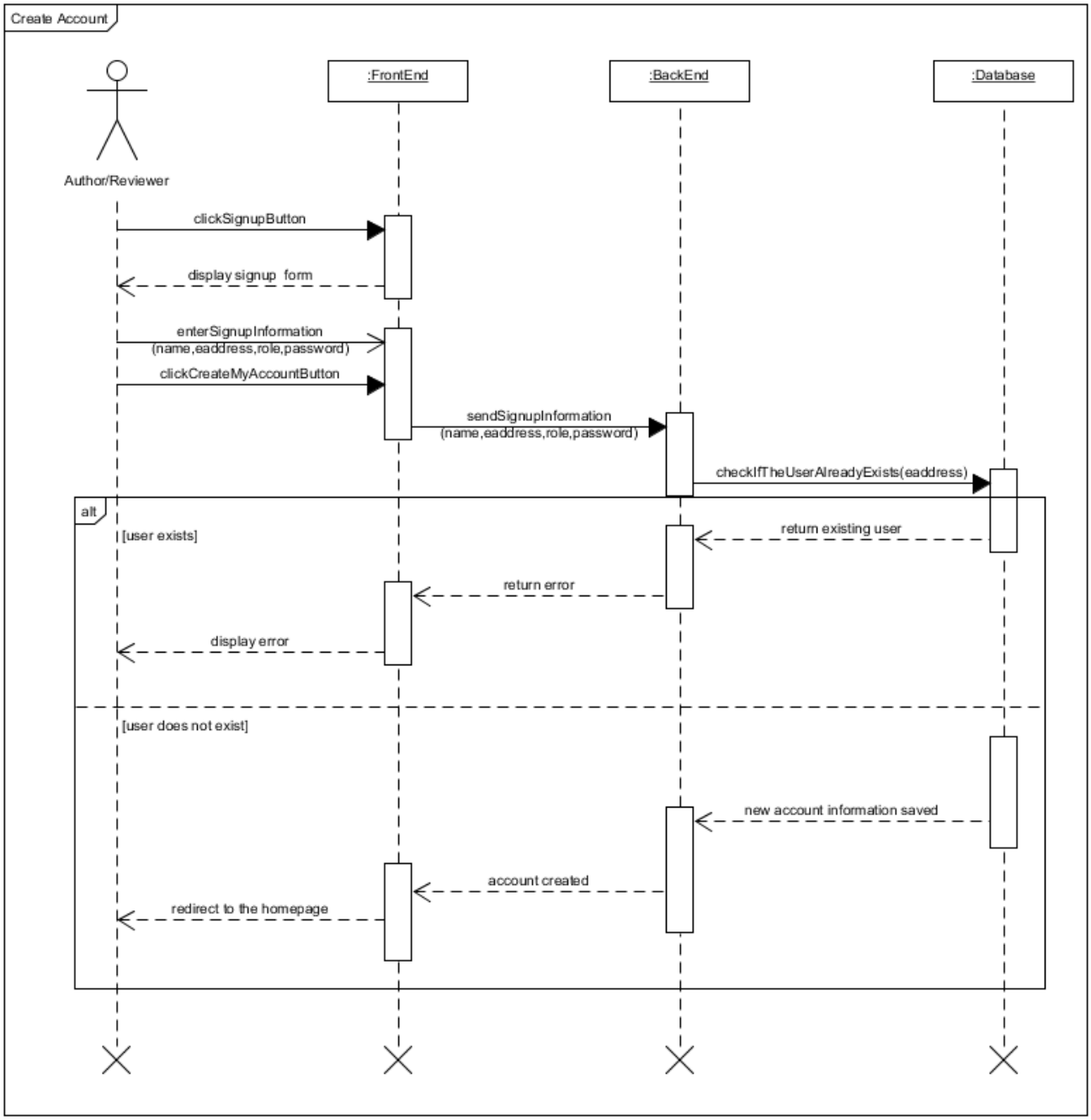


Figure 27: Sequence Diagram 1 - Create Account

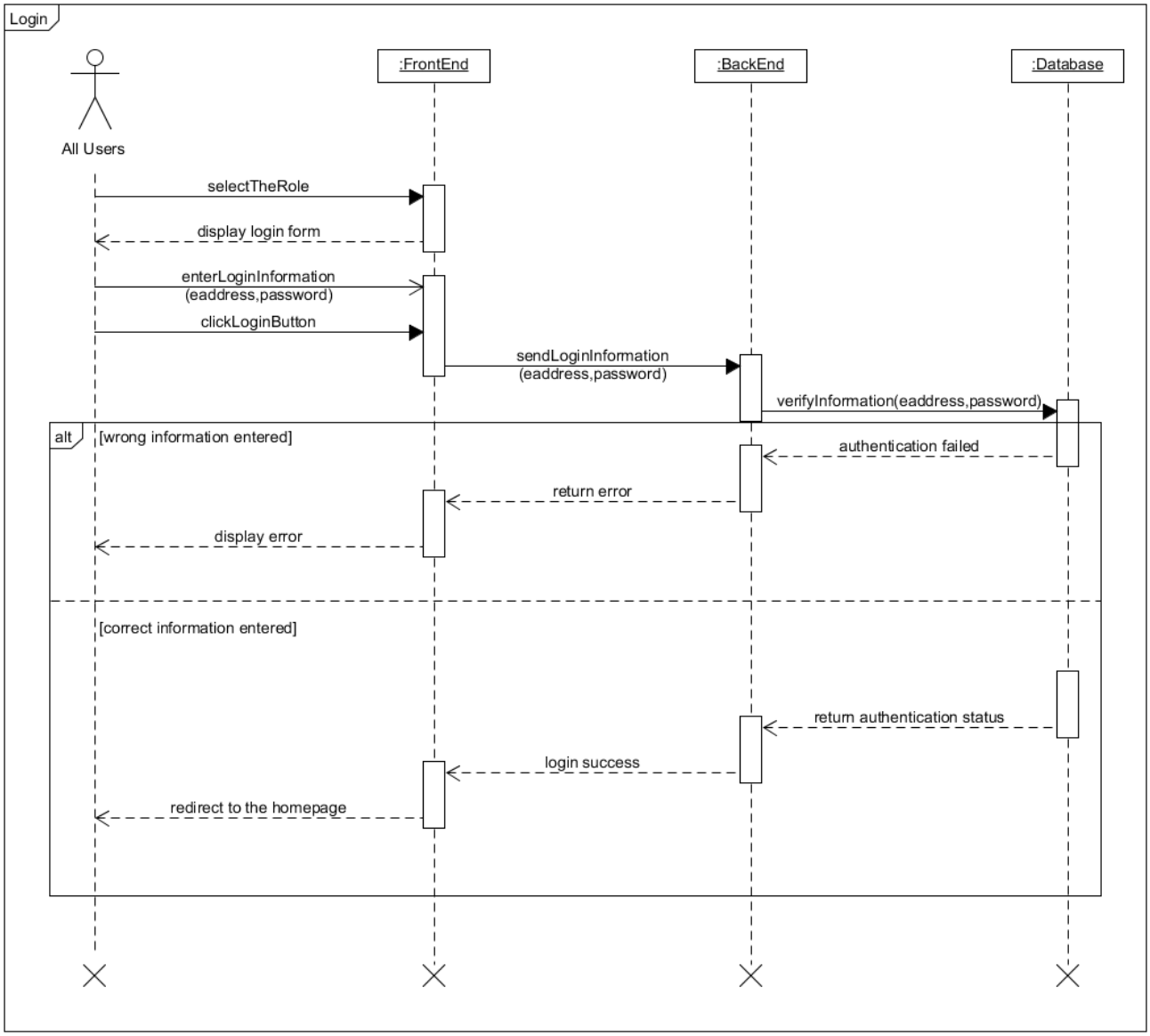


Figure 28: Sequence Diagram 2 - Login

1.5.2. Resources Management Subsystem

1.5.2.1. Papers Management

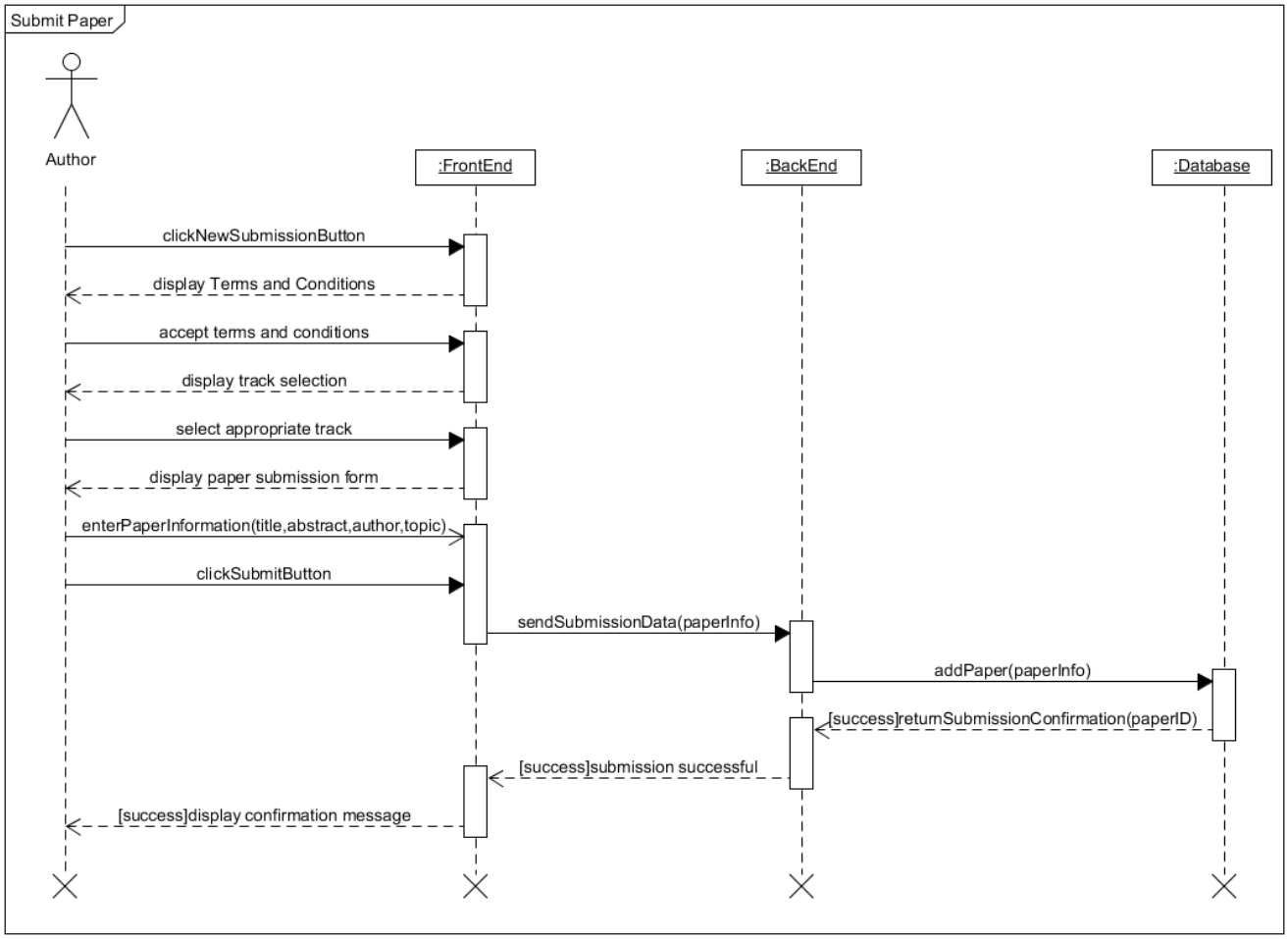


Figure 29: Sequence Diagram 3 - Submit Paper

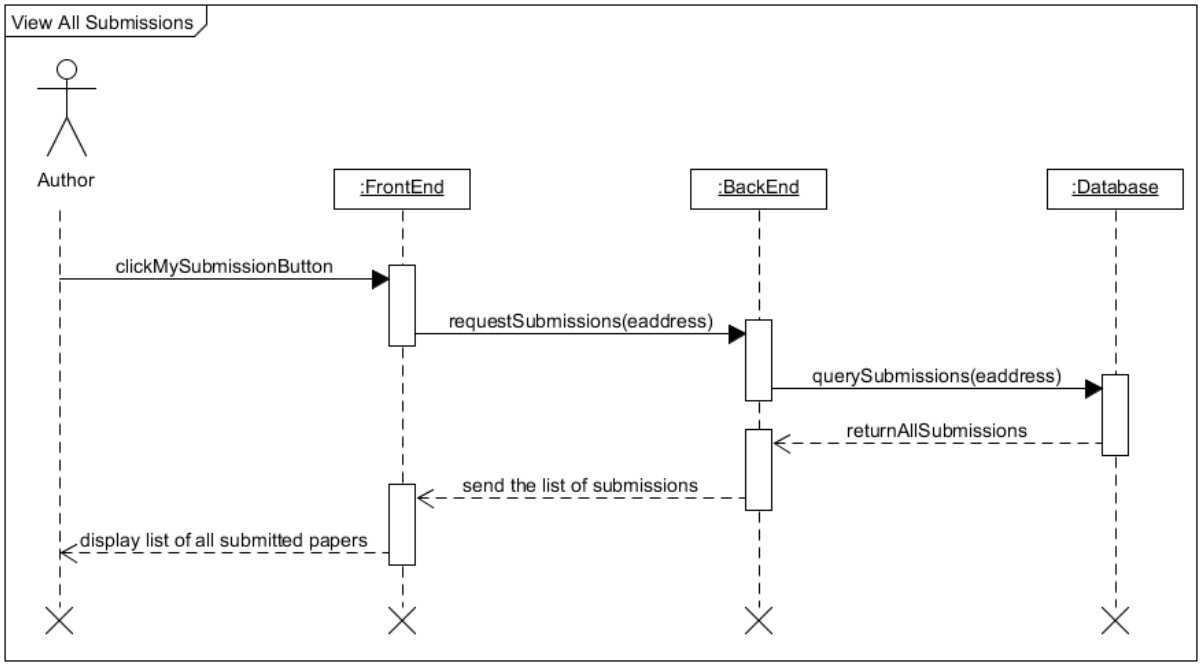


Figure 30: Sequence Diagram 4 - View All Submissions

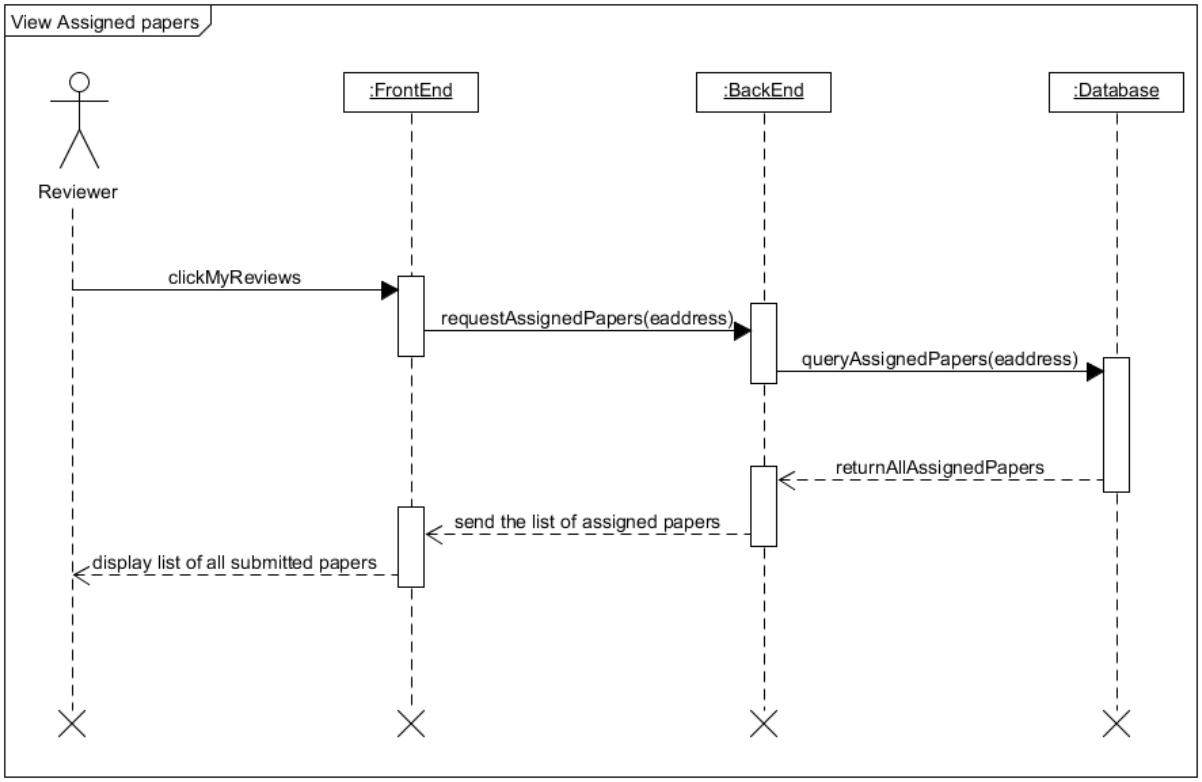


Figure 31: Sequence Diagram 5 - View Assigned Papers

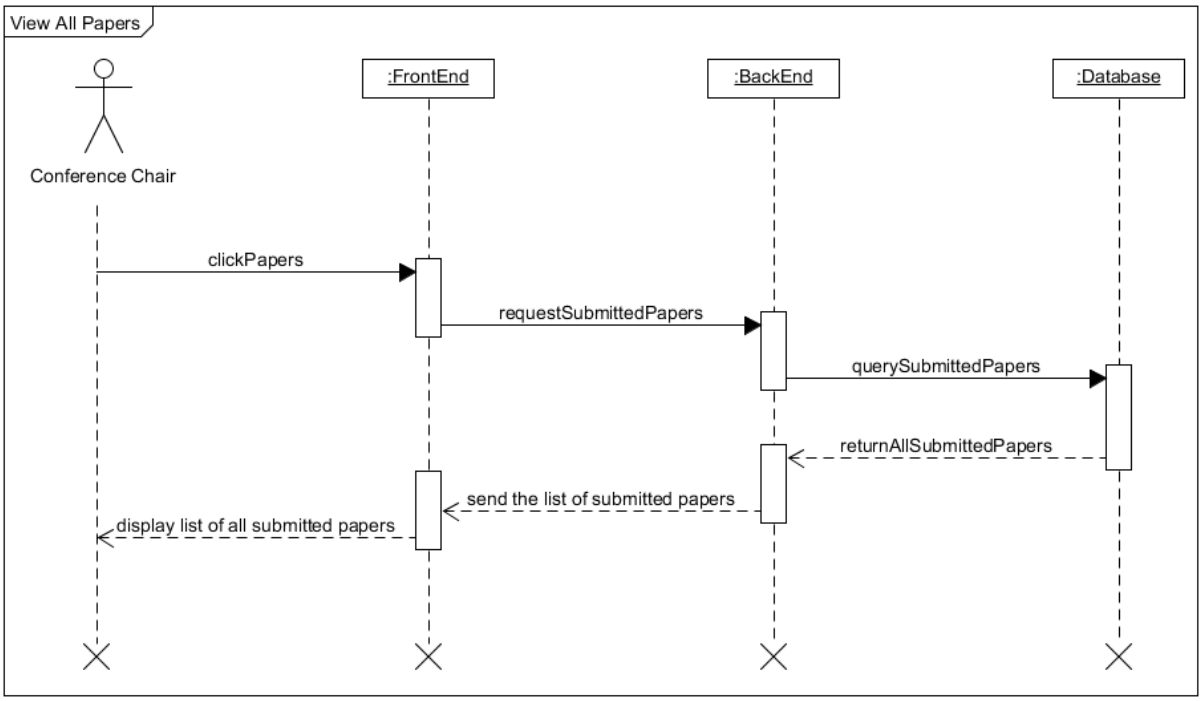


Figure 32: Sequence Diagram 6 - View All Papers

1.5.2.2. Reviewers Management

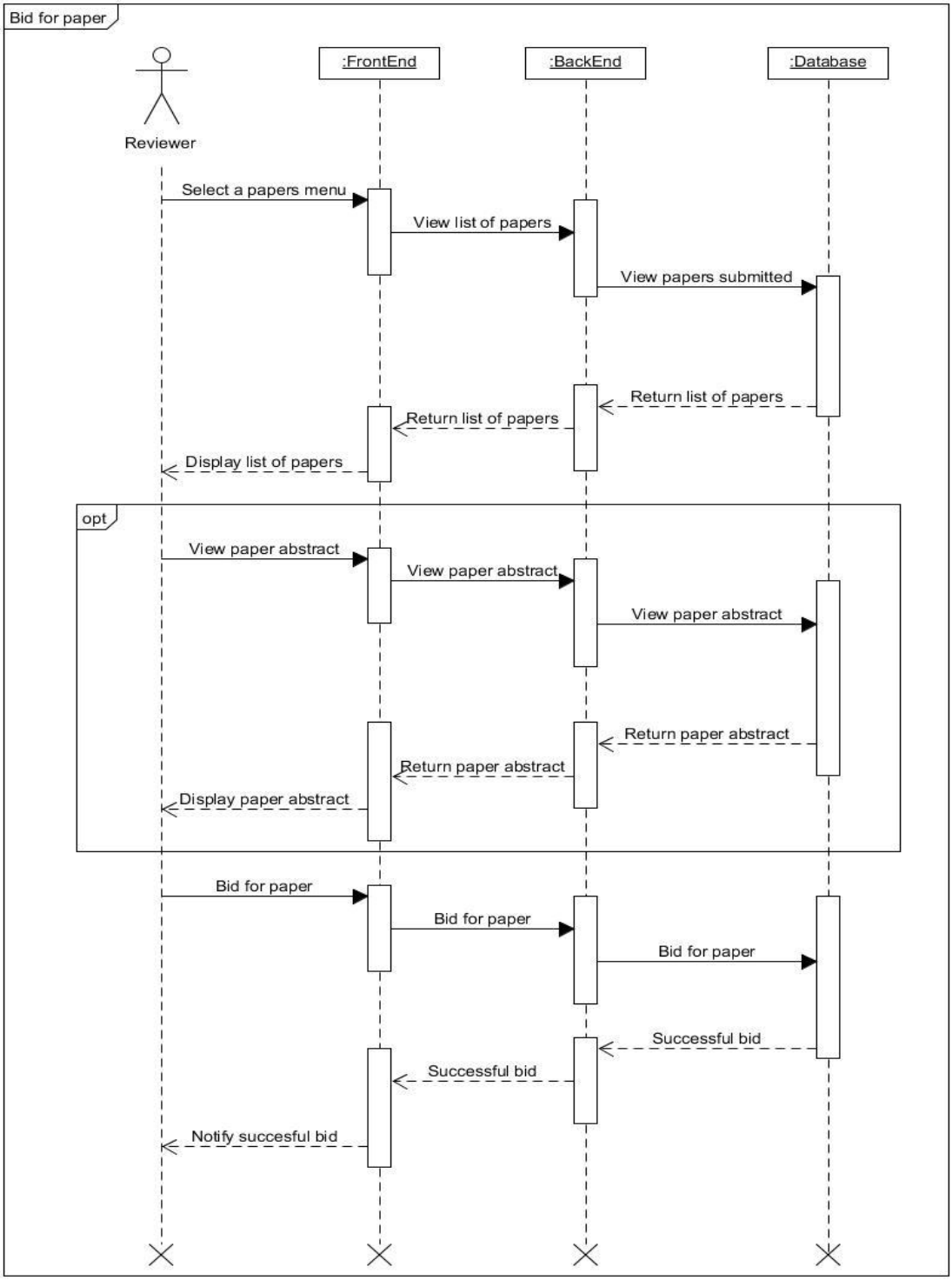


Figure 33: Sequence Diagram 7 - Bid For Paper

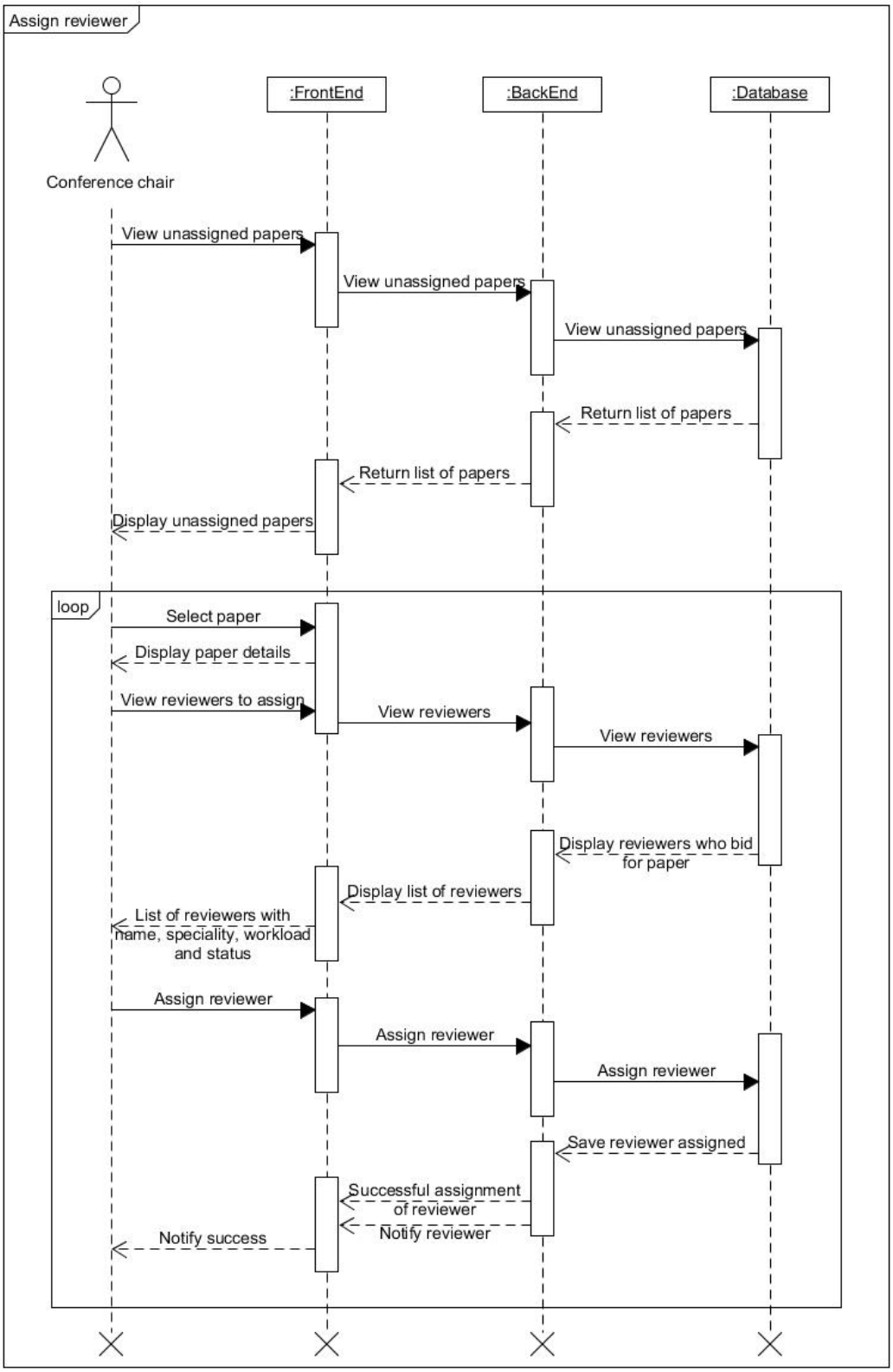


Figure 34: Sequence Diagram 8 - Assign Reviewer

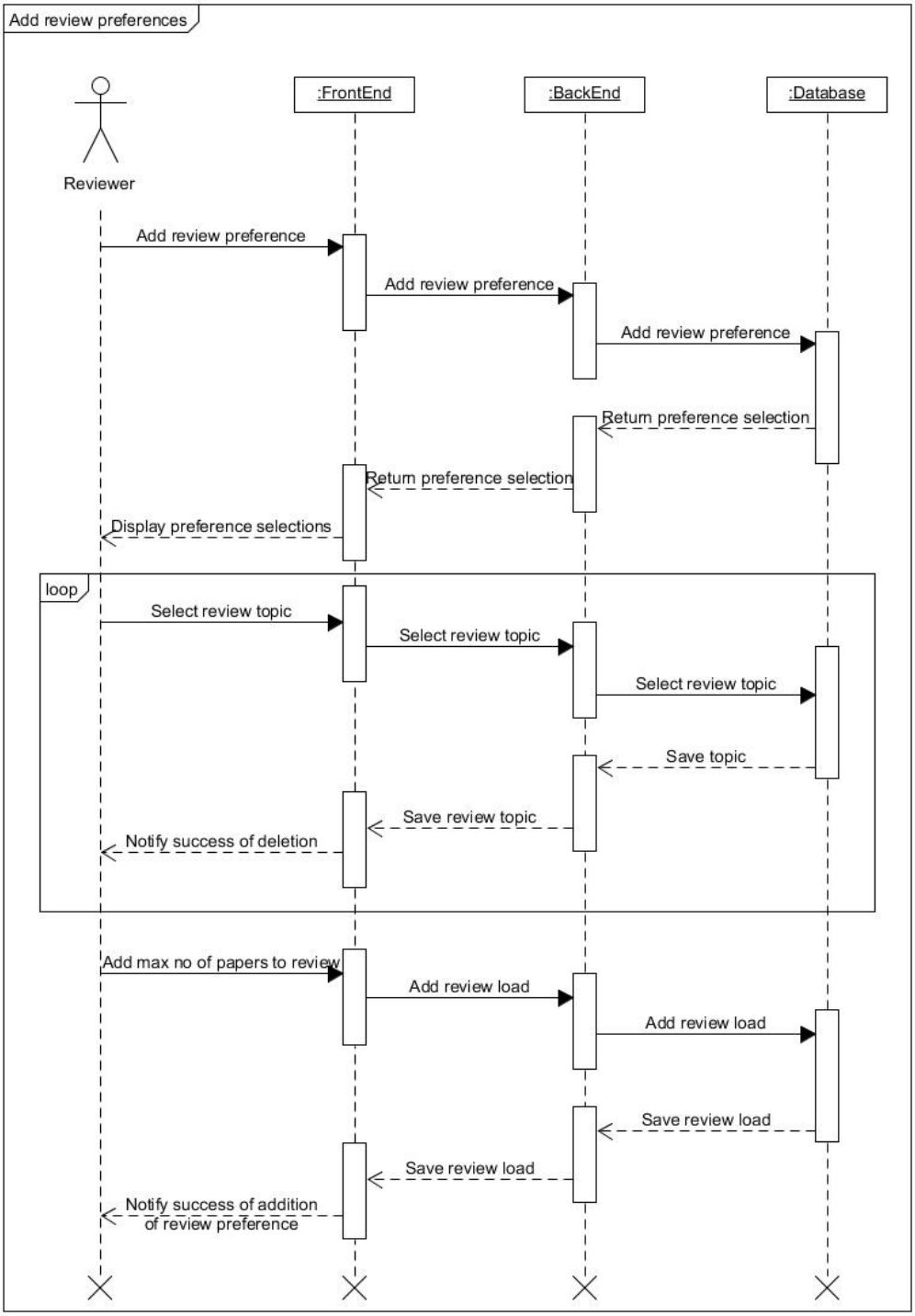


Figure 35: Sequence Diagram 9 - Add Review Preferences

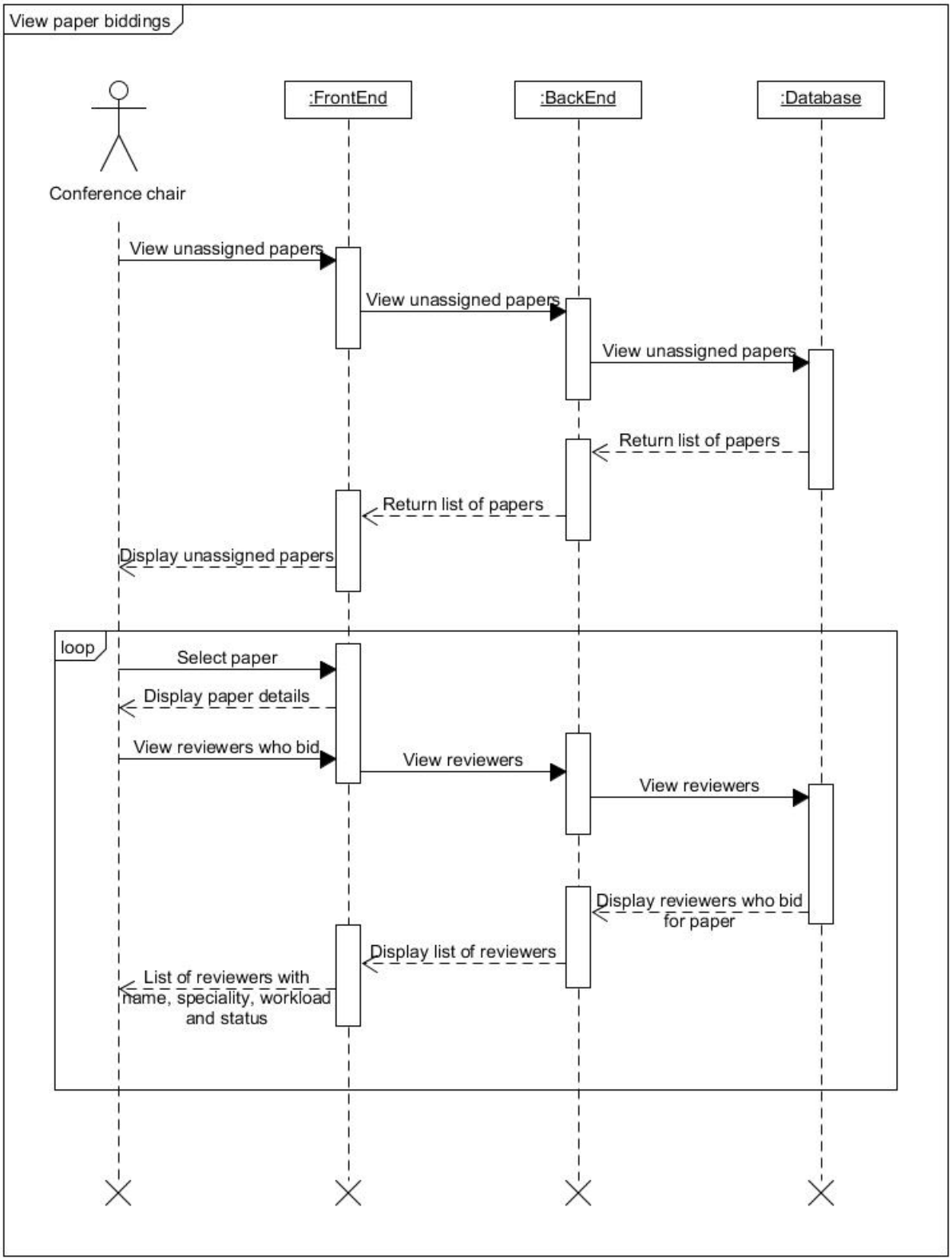


Figure 36: Sequence Diagram 10 - View Paper Biddings

1.5.3. Conference Activities Management Subsystem

1.5.3.1. Reviewing & Rating Processes Management

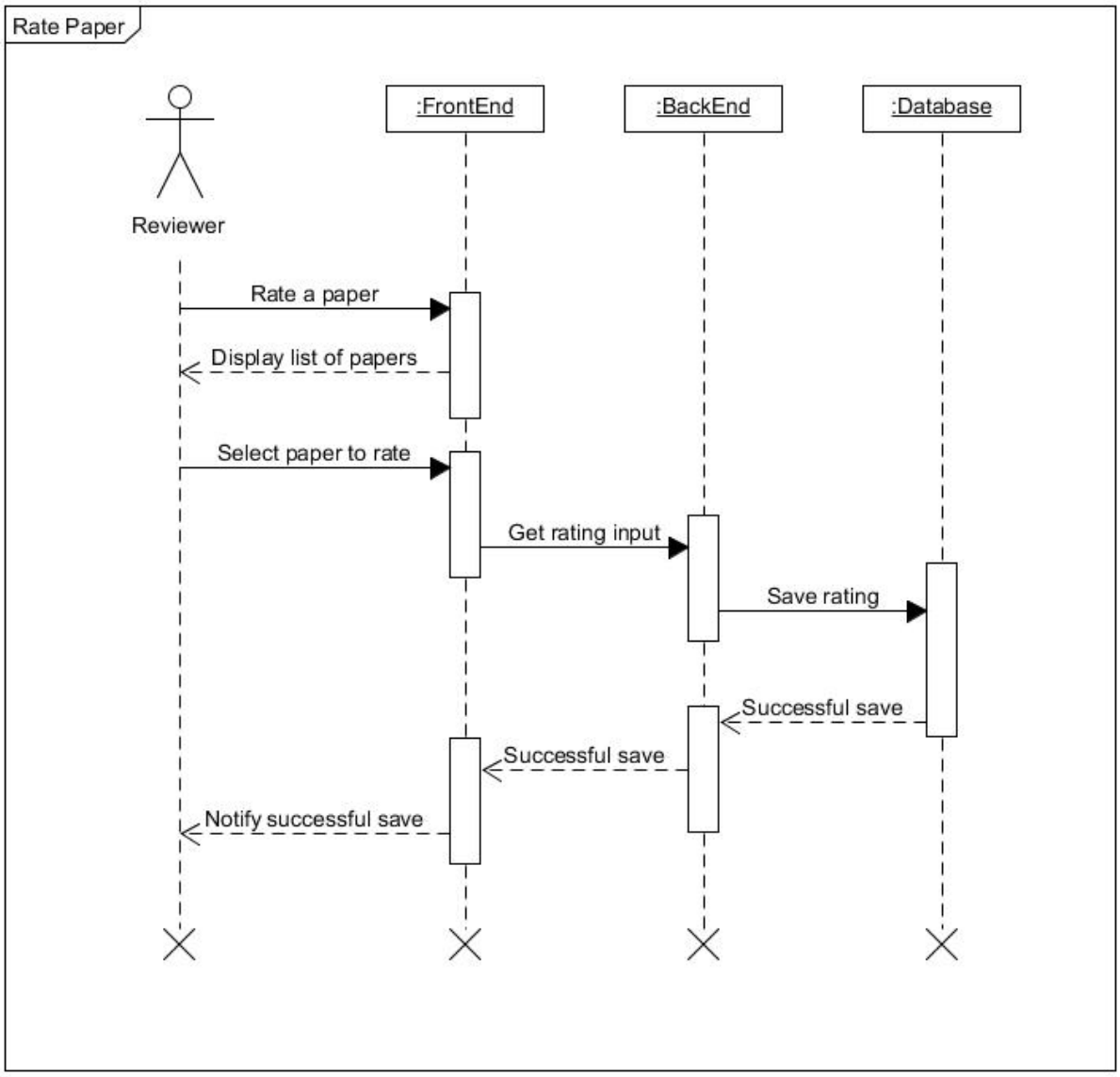


Figure 37: Sequence Diagram 11 - Rate Paper

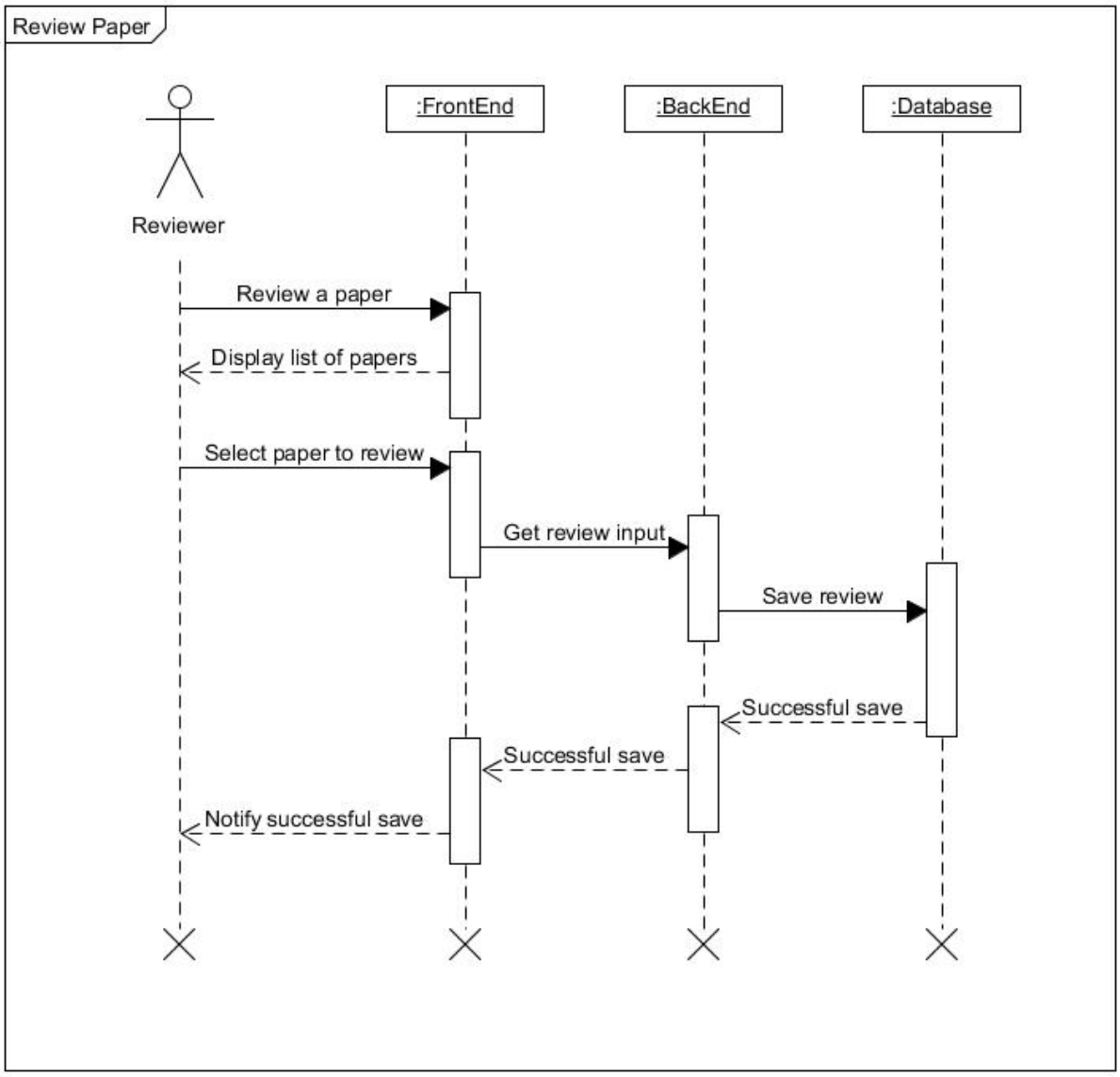


Figure 38: Sequence Diagram 12 - Review Paper

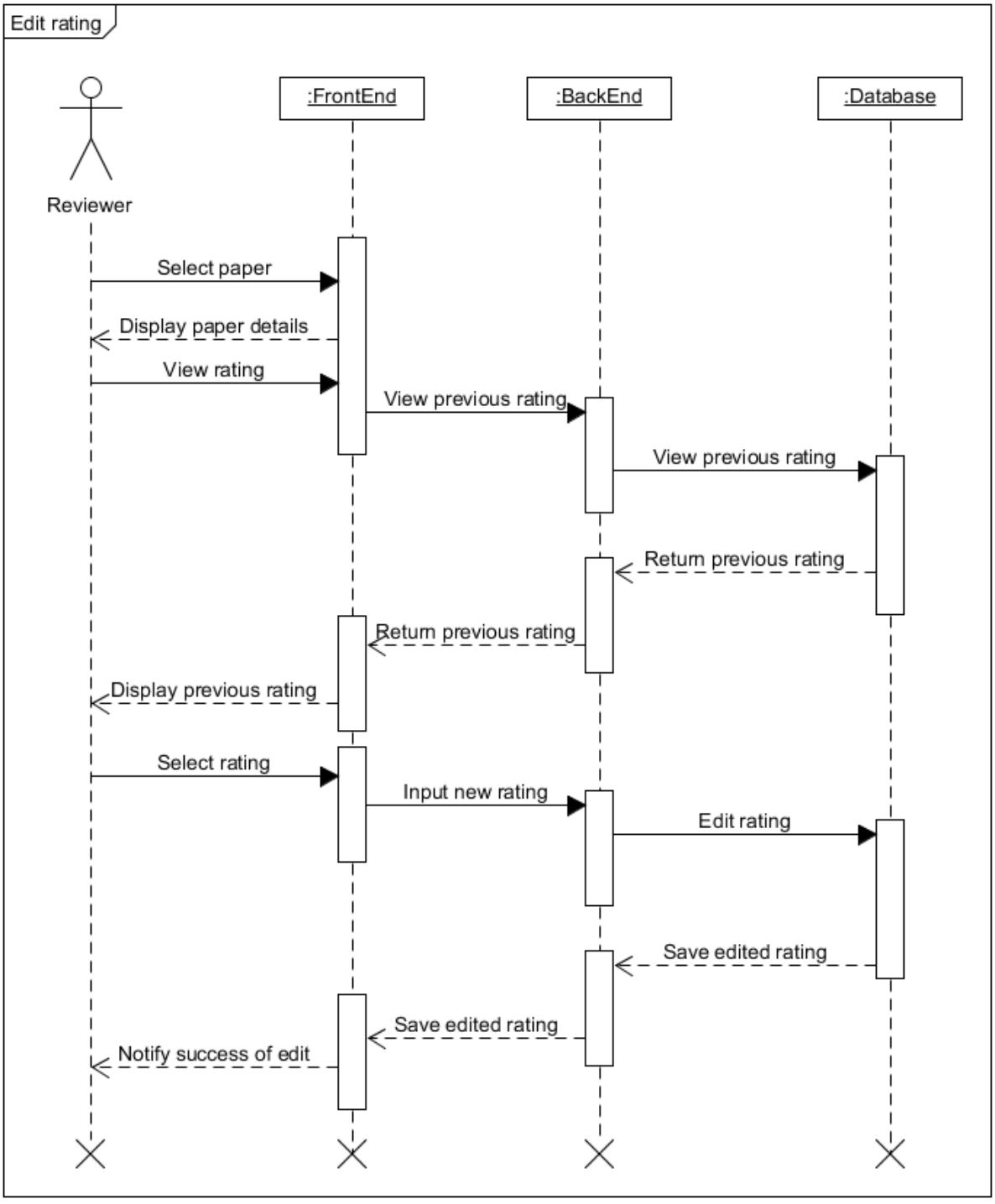


Figure 39: Sequence Diagram 13 - Edit Rating

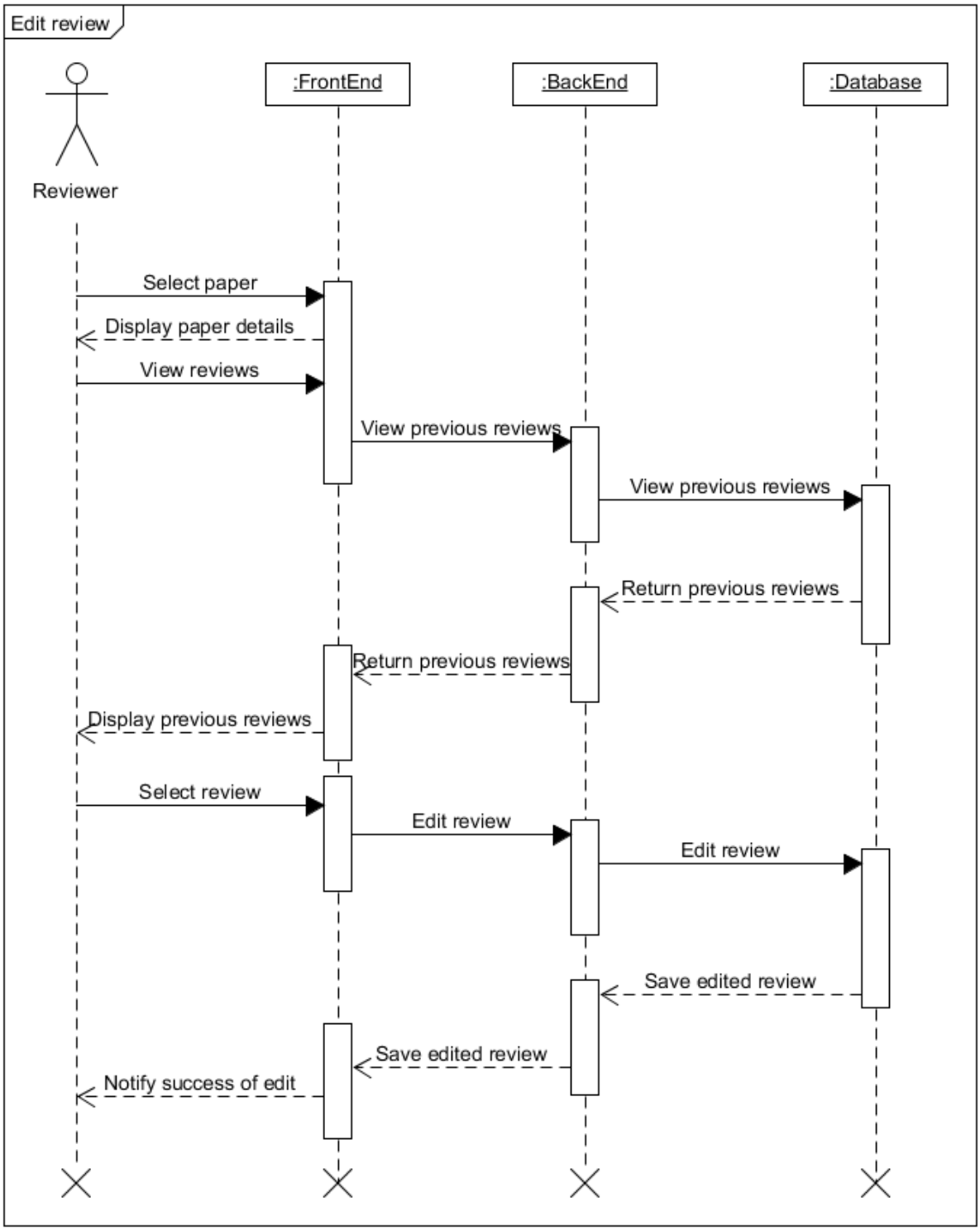


Figure 40: Sequence Diagram 14 - Edit Review

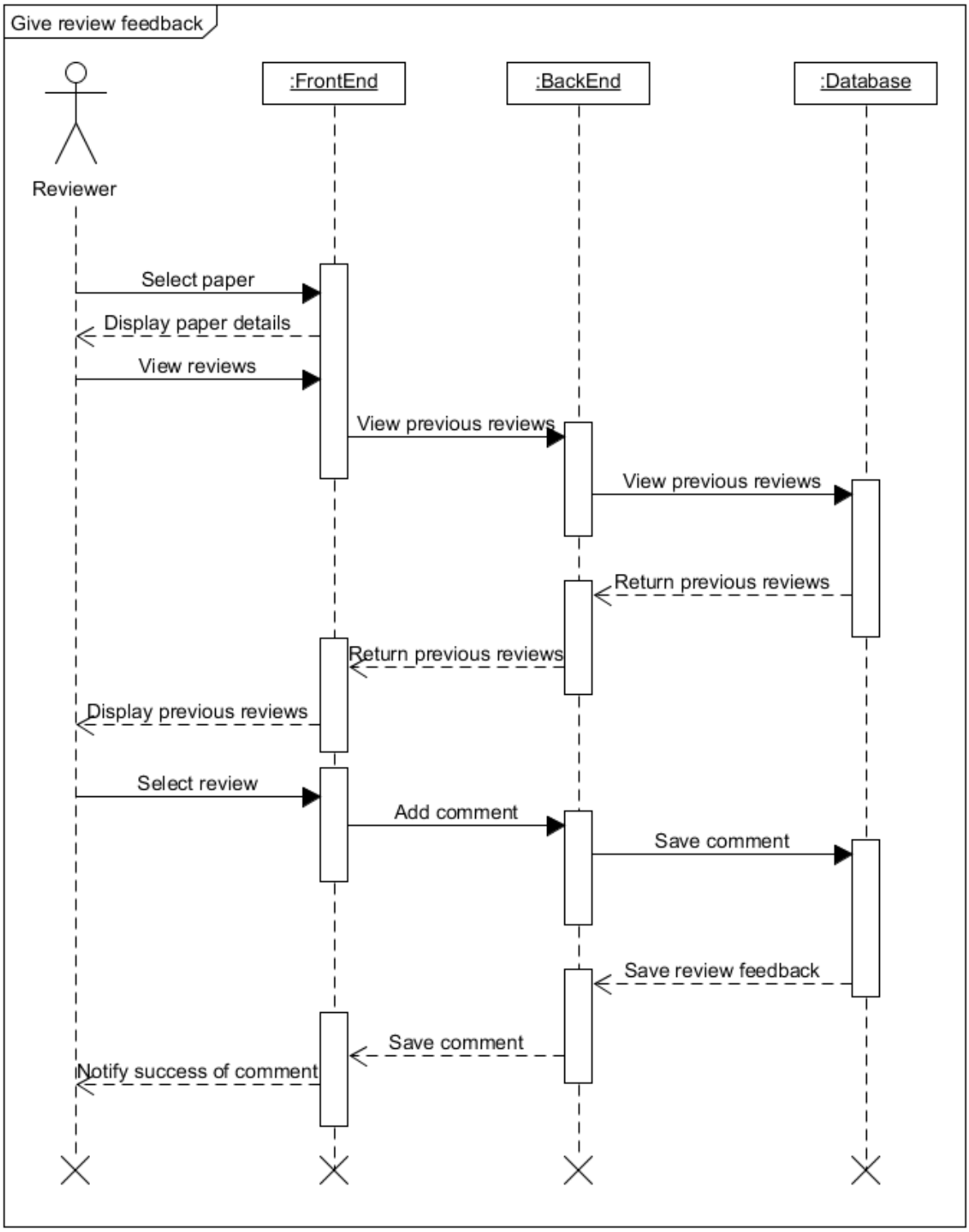


Figure 41: Sequence Diagram 15 - Give Review Feedback

1.5.3.2. Decision Making & Notification Management

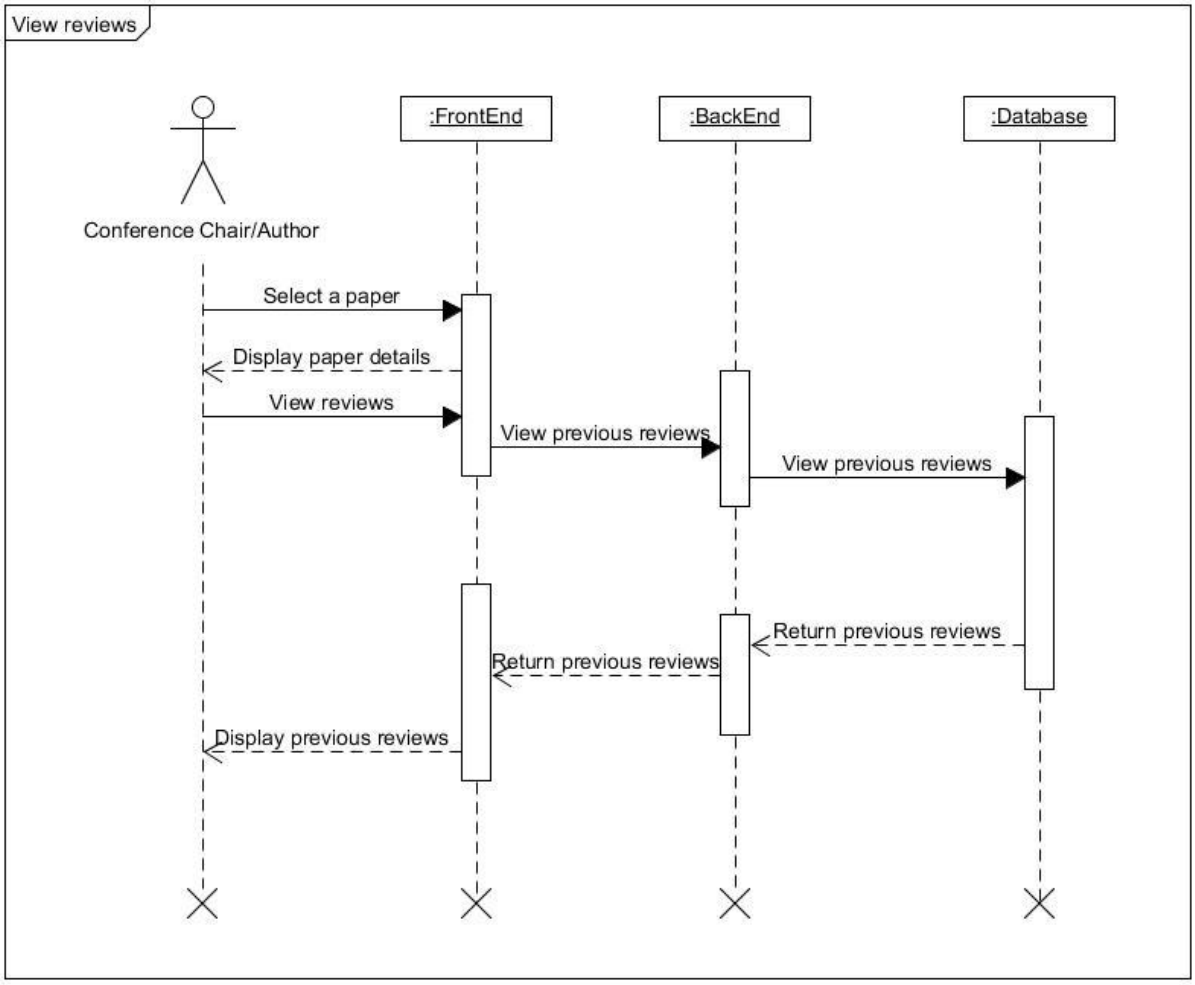


Figure 42: Sequence Diagram 16 - View Reviews

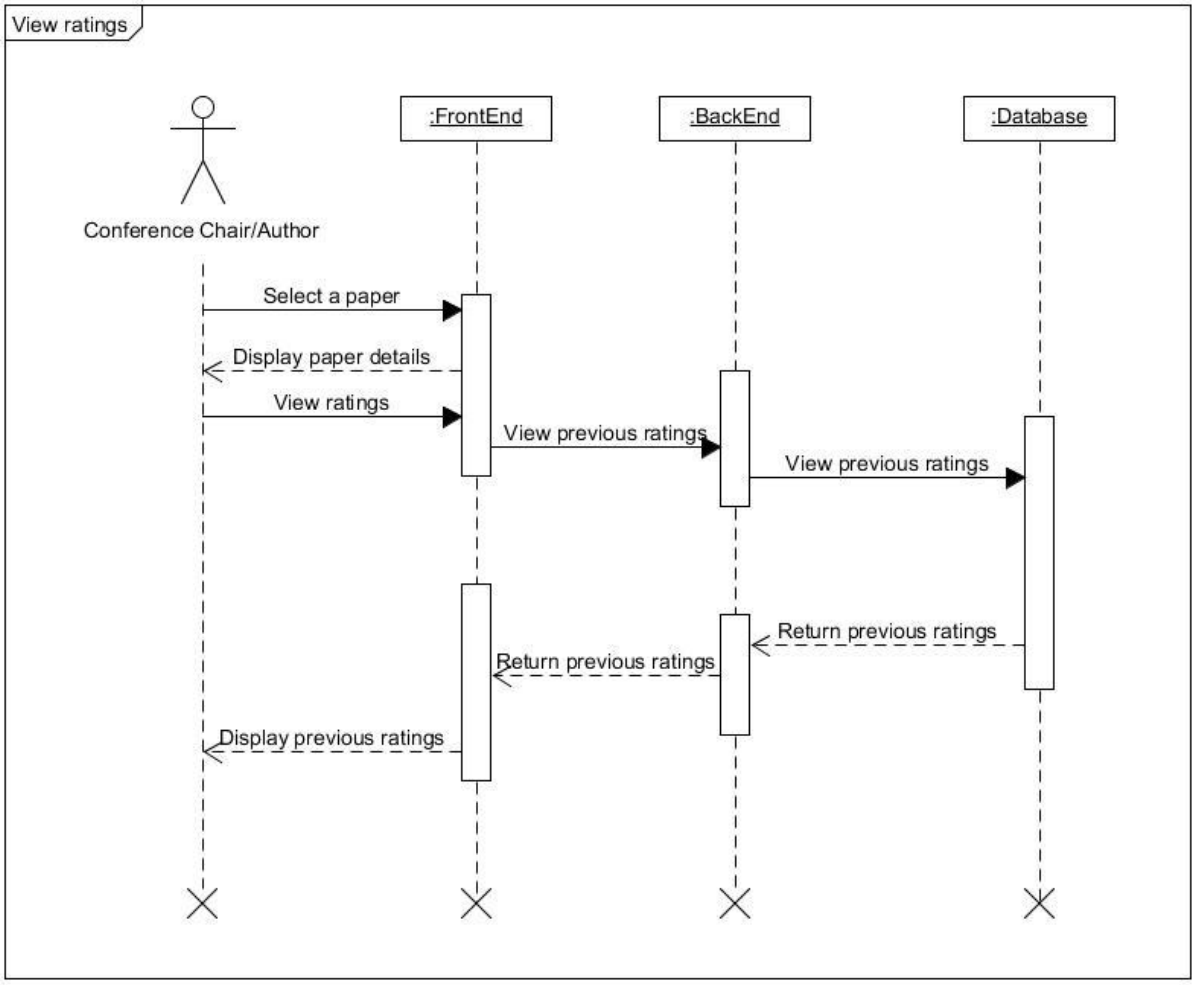


Figure 43: Sequence Diagram 17 - View Ratings

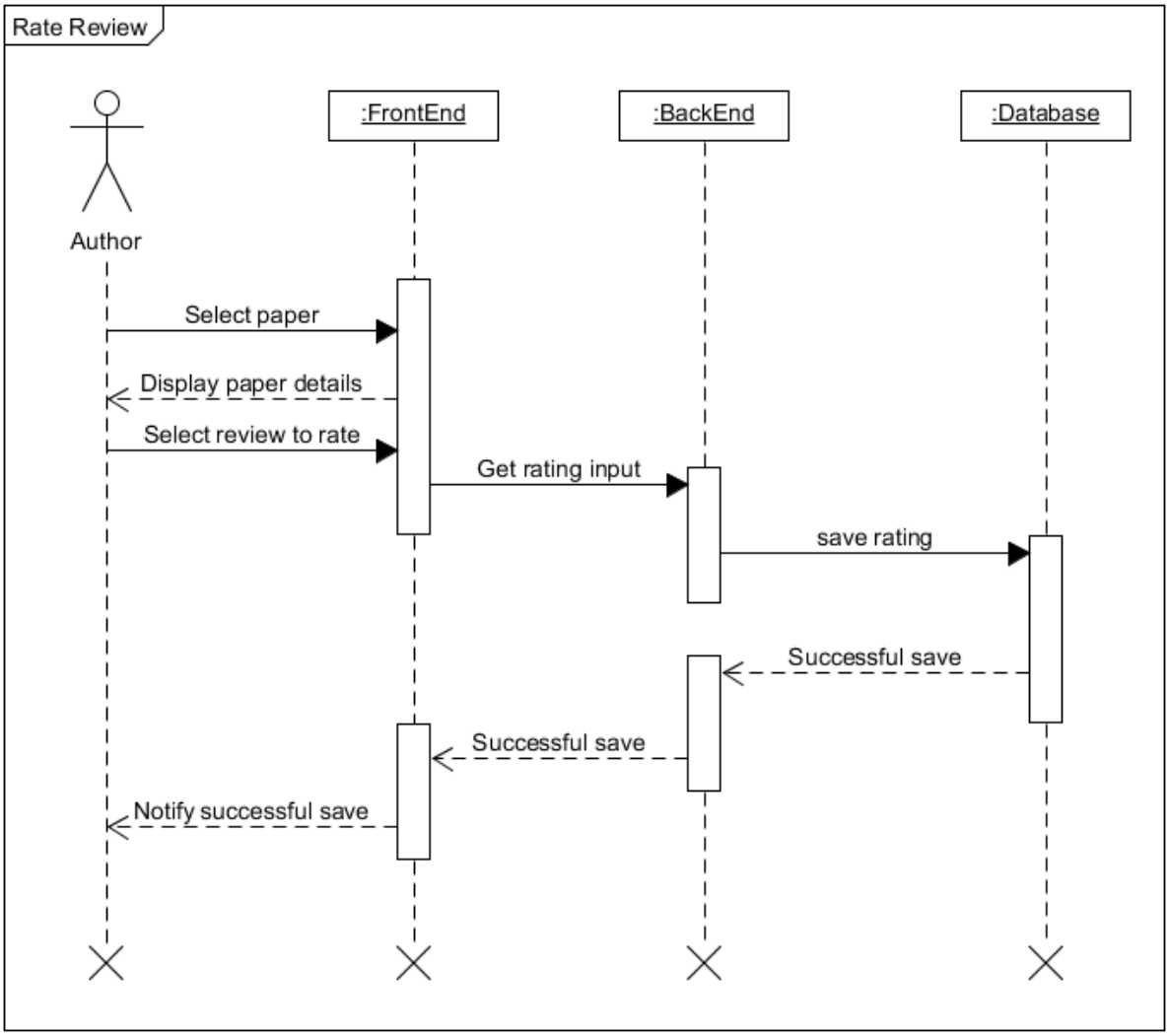


Figure 44: Sequence Diagram 18 - Rate Review

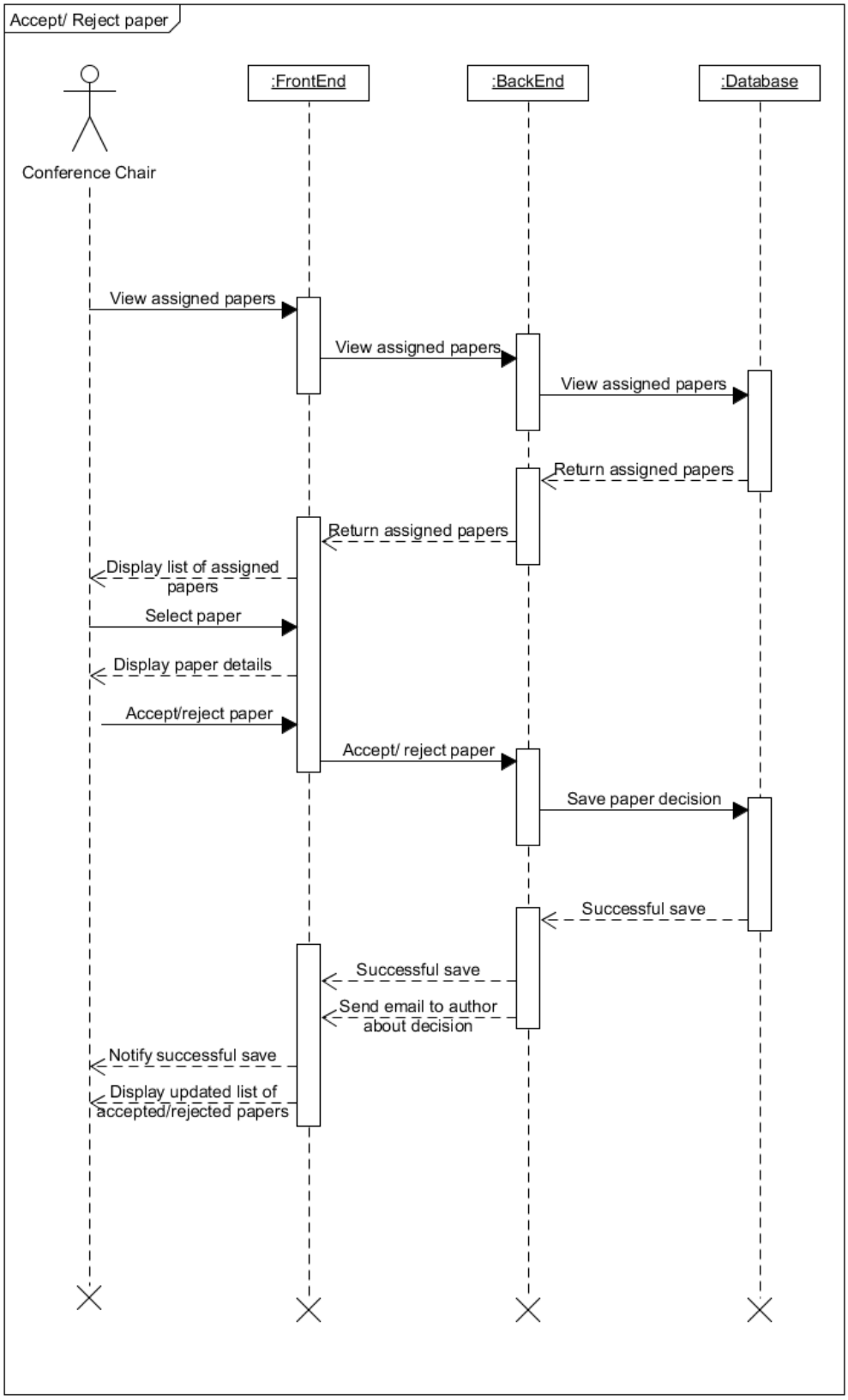


Figure 45: Sequence Diagram 19 - Accept/Reject Paper

1.6. UML State Diagrams

1.6.1. User Management Subsystem

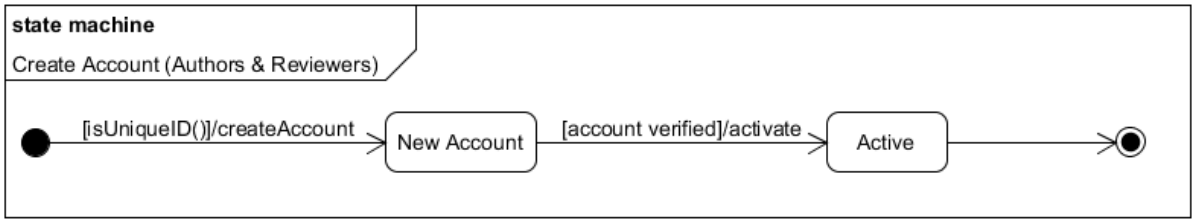


Figure 46: State Diagram 1 - Create Account(Authors & Reviewers)

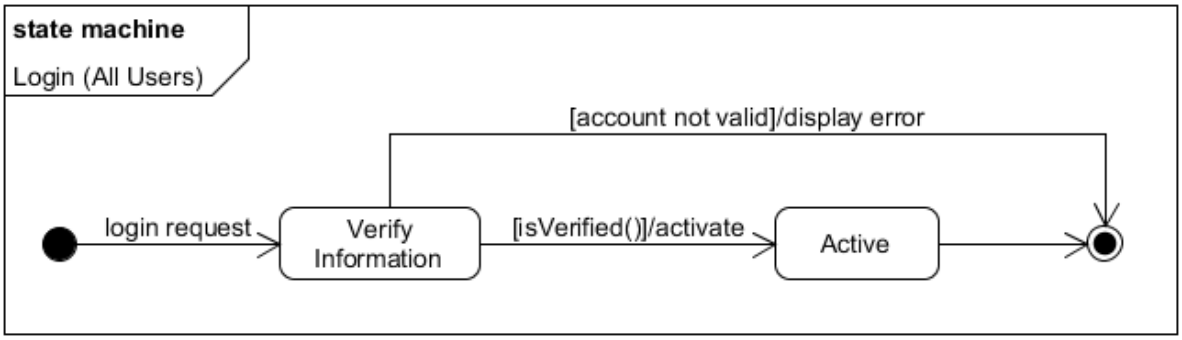


Figure 47: State Diagram 2 - Login(All Users)

1.6.2. Resources Management Subsystem

1.6.2.1. Papers Management

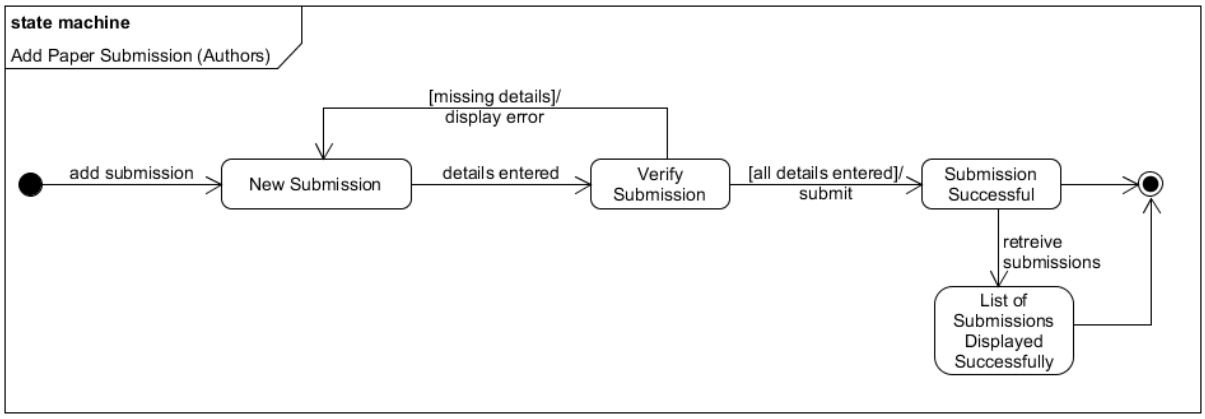


Figure 48: State Diagram 3 - Add paper Submission(Authors)

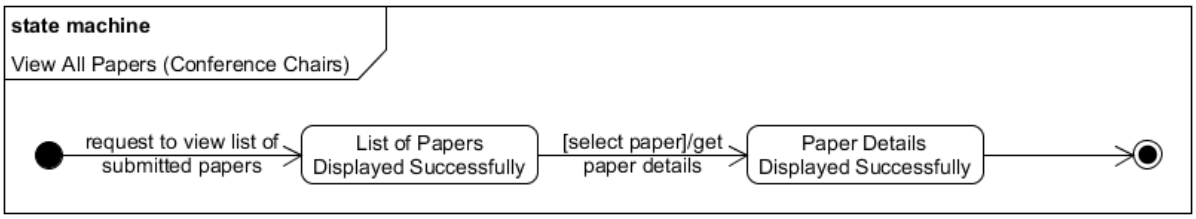


Figure 49: State Diagram 4 - View All Papers(Conference Chairs)

1.6.2.2. Reviewers Management

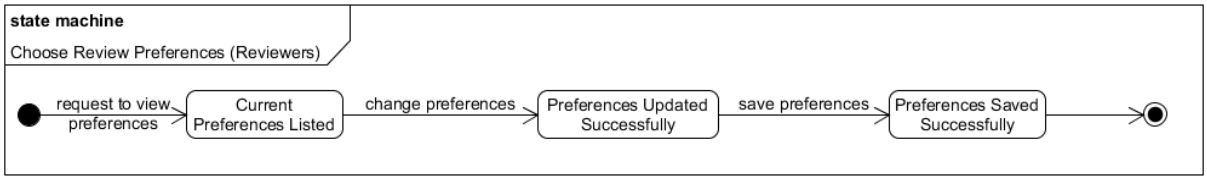


Figure 50: State Diagram 5 - Choose Review Preferences(Reviewers)

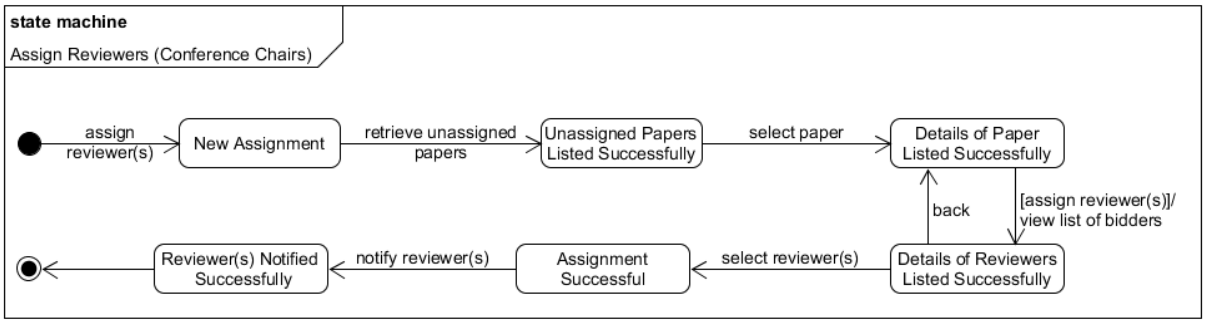


Figure 51: State Diagram 6 - Assign Reviewers(Conference Chairs)

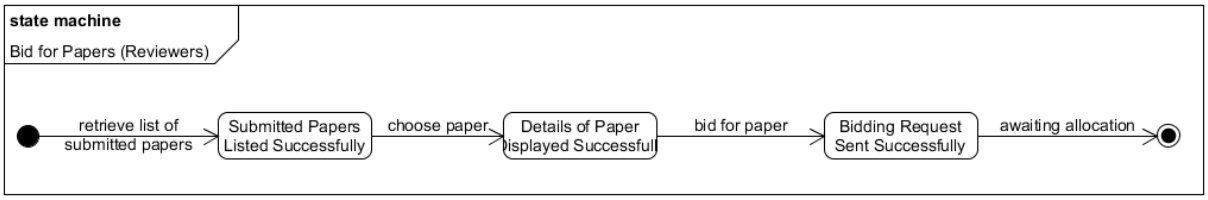


Figure 52: State Diagram 7 - Bid for Papers(Reviewers)

1.6.3. Conference Activities Management Subsystem

1.6.3.1. Reviewing & Rating Processes Management

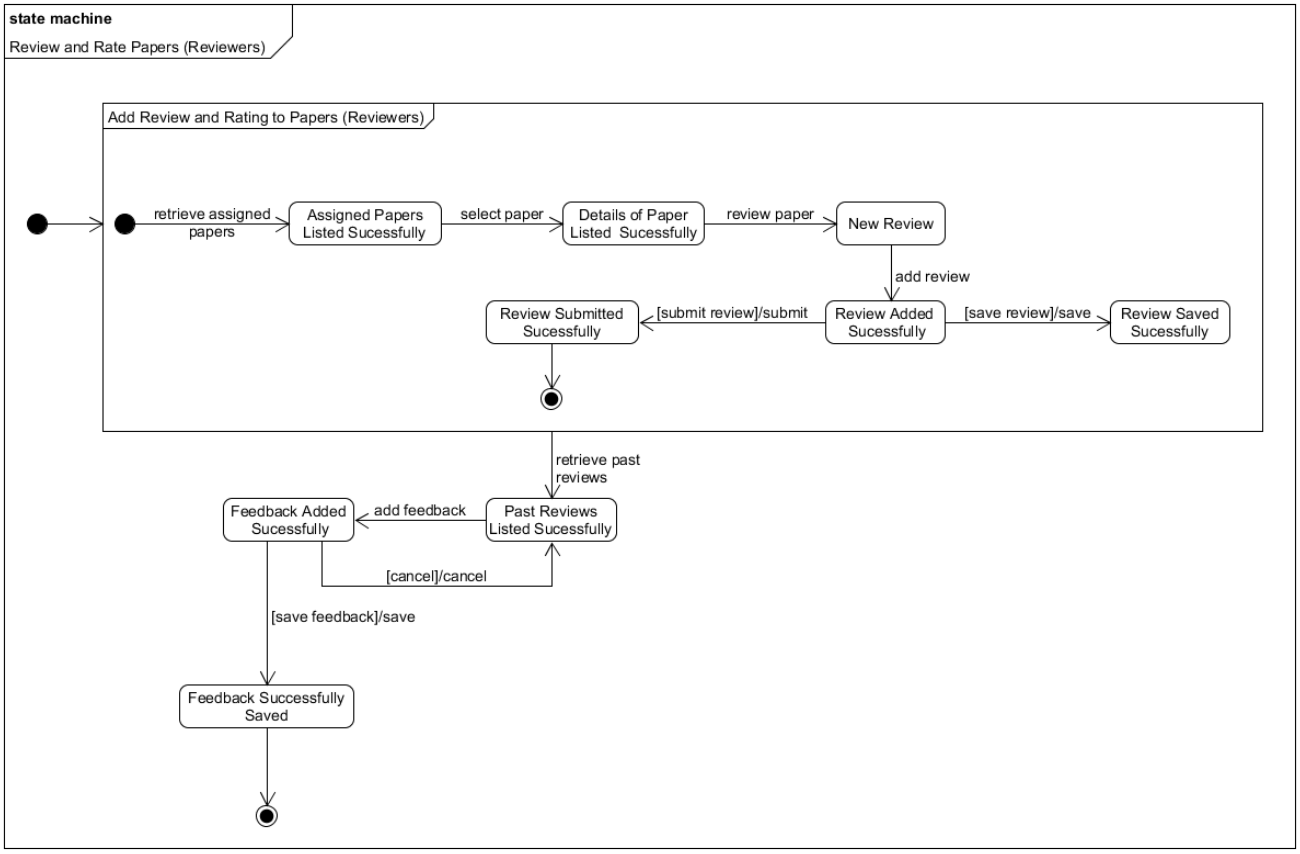


Figure 53: State Diagram 8 - Review and Raye Papers(Reviewers)

1.6.3.2. Decision Making & Notification Management

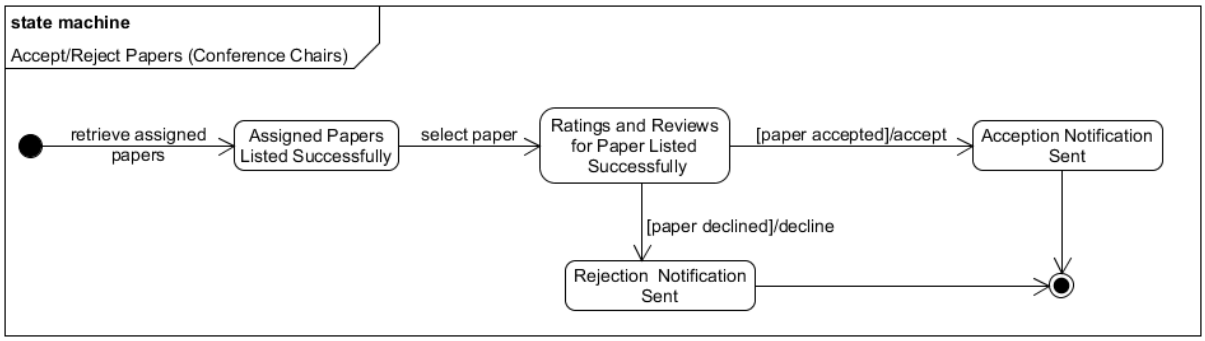


Figure 54: State Diagram 9 - Accept/Reject Papers(Conference Chairs)

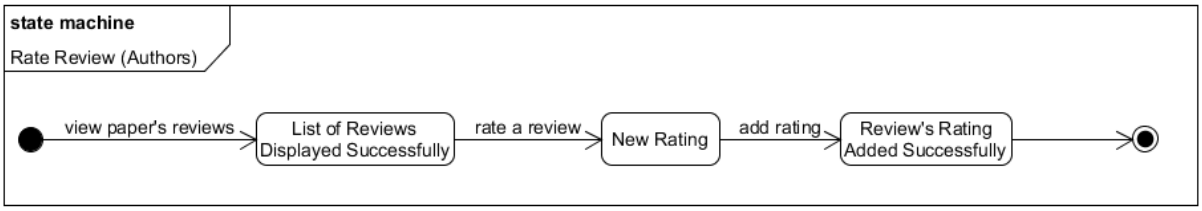


Figure 55: State Diagram 10 - Rate Review(Authors)

1.7. Database Design Description

The database for ConfrencePro was built using MongoDB Atlas Database, a very popular noSQL backend solution, that handles all the complexity of deploying and managing deployments on the cloud. MongoDB Atlas provides authorization access that allows the data to be stored securely. Hence, allowing us to achieve the non-functional requirements that we identified during the planning stage of the system. The MongoDB REST API uses Springboot to communicate to the frontend. MongoDB was ideal for the project as it documents with flexible schemas and is widely accepted and native-code data access, is an approachable design, has powerful analytics and querying, sharding allows for an easy horizontal scale-out, simple to setup, as well as cost-effective. In addition high performance and scalability are offered. As Agile development allows for the possibility of requirement changes, and MongoDB supports schema changes it was suitable for the project.

We employed a NoSQL solution for the project because it allows rapid development and flexible changes, as there is no need to define strict schemas and relations before using the database, facilitating rapid development of the system. Thus, considering the 7 weeks time frame to undertake development, the unstructured layout of NoSQL solution is ideal to achieve rapid installation and IDE integration,

A "User" entity is used to manage the system's users. Each user's account information is stored in this class UserFactory.java. Following that, it returns a type of user for each sort of User, including Reviewer, Chair, and Author. These items are all contained in the “User” entity and are created to provide more information to the "User" JSON entity. To avoid spending too much time creating separate entities for each type of user, which would complicate the system's architecture and have no positive effects on performance, this design element was created that allows for the addition of additional information that is unique to each user under the same object. The factory method pattern gives subclasses the option of selecting the kind of objects to produce. By removing the requirement to tie application-specific classes into the code, it encourages loose coupling.

The “Author”, “Reviewer” and “ConferenceChair” are types of “User” objects. This interaction maintains the data flow between system users simple and enables the implementation of the same logic for all system users, the system adds on to the information according to the type of user. The “Author” interacts on a “One-to-Many” to “Zero-Many” relationship with the

“ResearchPaper”. This indicates that an author can be listed as the owner of zero or more papers and a paper can be submitted with one or more authors. This data is stored in arrays in the database to allow for ease of access and simplifies the flow of data in the system.

“Review” has a “Zero-Many-One” relationship with the “ResearchPaper”, which means that there can be zero to many reviews for exactly one research paper. “Review” object also has “Zero-Many-One” relationship with the “Reviewer”. This means that a particular review can only be created by exactly one reviewer but a reviewer can create zero or more reviews. The “Comment” object also interacts with the “Review” in a “Zero-Many-One” relationship. Similarly a comment can be commented exactly to one review, whereas a review can have zero or many comments. The “Comment” object also interacts directly onto the “Reviewer” object and has a “Zero-Many-One” relationship. The reviewers can create zero to many comments, but one comment is created exactly by one reviewer. In addition, “Review” object has a “Many-to-Many”relationship with the “Rating” object. The “Rating” has a “Zero-Many-One” relationship to the “Author” meaning that a rating should be created exactly by one author whereas the author can create zero or many ratings.

1.8. User Interface Design Description

Each user's flow can be viewed through the links provided, otherwise, screenshots are provided in the Appendix.

1.8.1 Landing Page - Sign In / Sign Up

Below is the landing page that is shown on launch of the application on a desktop. It features four different buttons to start the sign in process for the different roles.

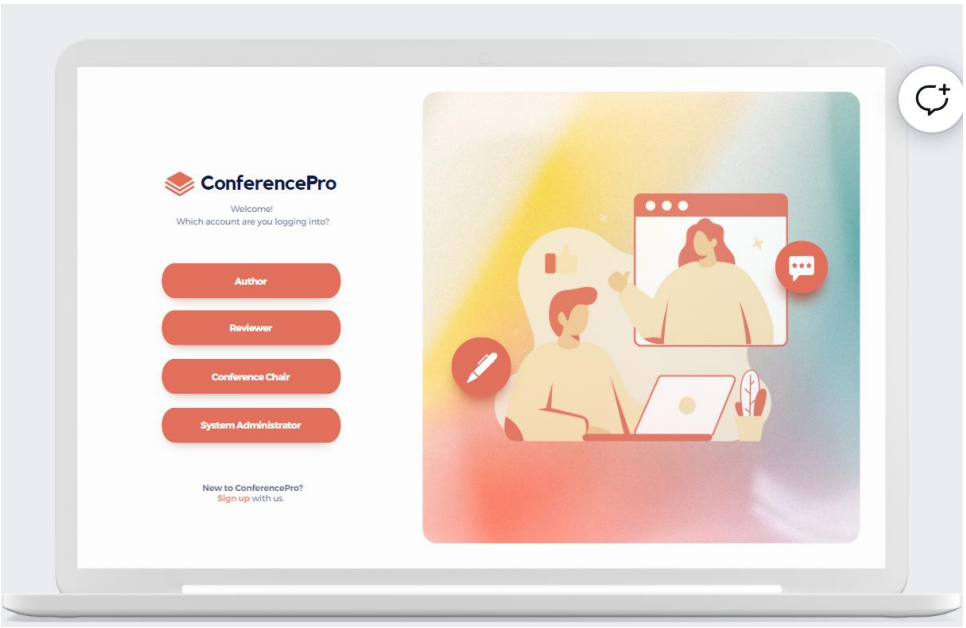


Figure 56: Landing Page for Sign In or Sign Up

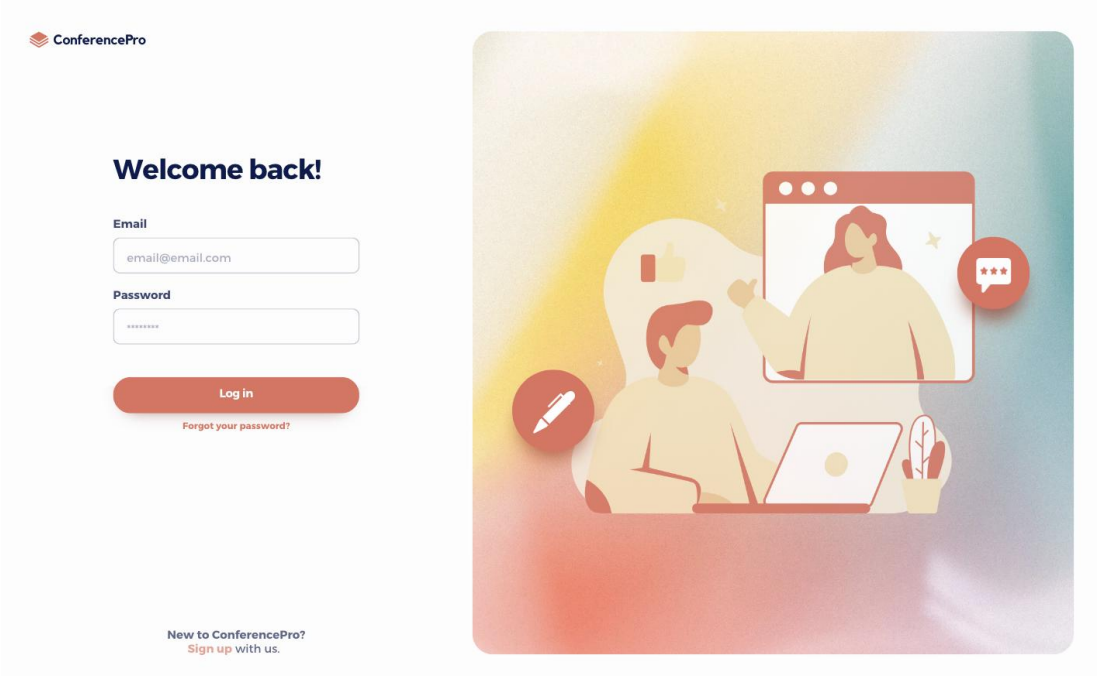


Figure 57: Sign In Option for those with Accounts

For authors and reviewers without an account, they are able to register for one by clicking the link at the bottom of the page. It is decided that Conference Chairs and System Administrator access are granted manually and cannot be registered for.

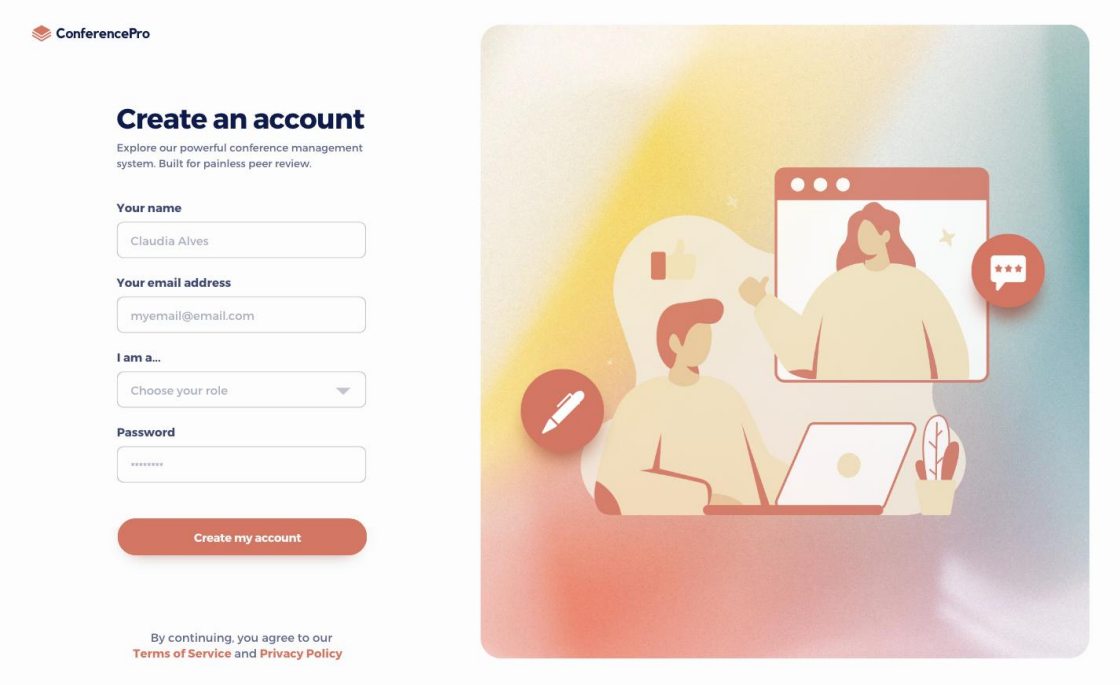


Figure 58: Creating New Accounts for Authors and Reviewers

1.8.1. Authors' UI

Flowing on from the dashboard, when an Author clicks 'New Submission', it brings them to some general information about the process of submitting their papers. Clicking 'Start Workflow' will guide the Author through the steps needed to successfully submit a paper. They are also able to have a quick glance of the steps included by the vertical navigation menu on the left – this is to help with information processing. It will also help the user feel less overwhelmed with the entire process.

If they were to click on a paper that has been marked 'Reviewed' from the "My Submissions" option in the menu, the user will be navigated to a page where they are able to rate the overall review they have received by the Reviewer(s) assigned to their submission.

1.8.2. Reviewers' UI

Flowing on from the dashboard, a Reviewer is able to click 'Edit Preferences'. This leads them to a page where they can edit their basic information and manage their preferences associated with the papers they will be assigned. This includes selecting topics that align with their specialty to assist with the automated assignment, and they can also set the maximum number of papers they can review.

The "Paper" menu option leads the users to the list of unassigned papers that have been submitted for review by the authors. The reviewers can then choose the papers they want to review, read their abstracts, and bid for them if they would like to.

Clicking 'View All Papers' brings them to a page listing out all of their assigned papers. Selecting a paper then brings a Reviewer to a page with all of the information associated with the assigned paper where they can confirm whether or not they would like to proceed with reviewing it.

1.8.3. Conference Chairs' UI

Flowing on from the dashboard, the user is able to click 'View Reviewers'. This leads them to a page which shows all the reviewers registered for the conference. The user can view each reviewer's specialty and their workload.

Clicking on the 'Papers' tab brings the user to a page showing all the papers in the system. The page contains two tabs: assigned and unassigned. Assigned papers consist of those assigned to

reviewers where they are either accepted, rejected or pending decision. Clicking on a paper that is pending a decision allows the user to view details about the paper including reviews and ratings given to it by reviewers. The user is able to use these details to then accept or reject the paper by using the buttons at the bottom of the page. Going back to the papers page, if the user clicks on an unassigned paper, they are taken to a page showing further details on the paper. If the user wishes to assign the paper to reviewers, they can click on the ‘View Reviewers’ button at the bottom of the page. The user is then taken to a page showing all users who are available to review the paper, ranking them by most suitable. The user can select desired reviewers and then confirm assignment by clicking on the ‘Confirm’ button at the bottom of the page.

1.9. Design Patterns

1.9.1. Creational Patterns

1.9.1.1. Factory Method

An interface for making objects in a superclass is provided by the factory method, a creational design pattern, although subclasses are free to change the type of objects that are created.

The factory switched from having different endpoints for each type of user to only having one endpoint. UserFactory.java is the class name. It simply takes an object that contains all user data and returns the specific user type to which the information applies.

1.9.2. Structural Patterns

1.9.2.1. Adapter

A structural design pattern called adapter enables items with disparate interfaces to work together. The steps it follow are:

An interface that works with one of the already-existing objects is provided to the adapter.

The existing object can call the adapter's methods securely by using this interface.

When an adapter receives a call, it transmits the request to the second object in the format and sequence that the second object anticipates.

In some cases, a two-way adapter that can convert calls both ways can even be built.

Any class from the repository serves as the adaptor. For instance ReviewRepository.java, ResearchPaperRepository.java and others, they extend the MongoDB class from a library that transforms messages from Spring Boot to the MongoDB Atlas instance.

1.9.2.2. Proxy

A structural design pattern called proxy enables you to create a stand-in or replacement for another object. Using a proxy, you may conduct an action either before or after the request reaches the original object, controlling access to it.

A proxy is crucial to the creation of a product in React. A website must be hosted on a server before release. However, a locally mimicked server called a proxy is needed since developers must continually be able to monitor, modify, and test changes during development. The design of the product makes advantage of React full stack. Using a proxy, testing was done on how the frontend and database communicate as well as how a prospective client could experience the design on their device.

1.9.3. Behavioral Patterns

1.9.3.1. Iterator

A collection's elements can be seen using an iterator, a behavioral design pattern, without having to reveal the collection's underlying representation (such as a list, stack, tree, etc.).

The program's different data processing techniques used the iterator design pattern to filter, search, and alter various types of data as well as looping over collections. Iterators were primarily utilised to generate the website's front-end components, which involved translating JSON from the back end.

1.9.3.2. Command

A request is transformed into a stand-alone object that contains all the information about the request using the behavioral design pattern known as command. With this transformation, you can allow undoable operations, delay or queue the request's execution, and pass requests as method arguments.

Any API endpoint in the system will convert JSON into objects and then send those objects to classes that can handle them. Any endpoint can be used to carry out any functionality using the associated object. For instance, all user functions are performed via the user endpoint, etc.

1.9.3.3. Decorator

With the use of a structural design pattern called decorator, you may give existing objects new behaviors by enclosing them in special wrapper objects that also contain the new behaviors. The alternate name for the Decorator pattern that succinctly conveys the pattern's fundamental principle is "Wrapper". The wrapped object's interface is implemented by the wrapper as well.

Because of this, these objects are the same in the client's perspective. Any object that adheres to that interface should be acceptable in the wrapper's reference field. This enables to add the combined behavior of all the wrappers to an object by wrapping it in more than one.

In the system, When the information is returned from an endpoint it's wrapped in a ResponseEntity class that enables extra HTTP functionality.

1.10. Refactoring Evidence

1.10.1 Factory creational design pattern

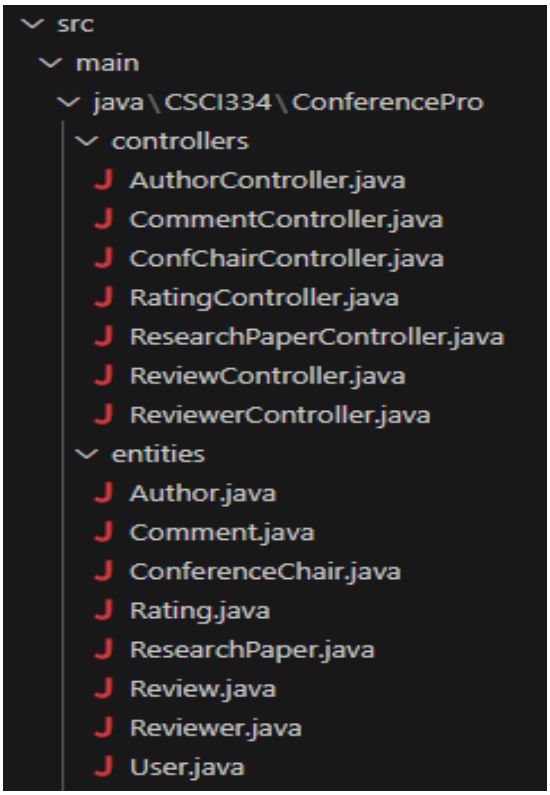


Figure 59: Entities and Controllers before Refactoring

The factory eliminated the need for distinct endpoints for each type of user in favor of a single endpoint. It simply takes an object that has all available user data and returns the user type to which the information applies.

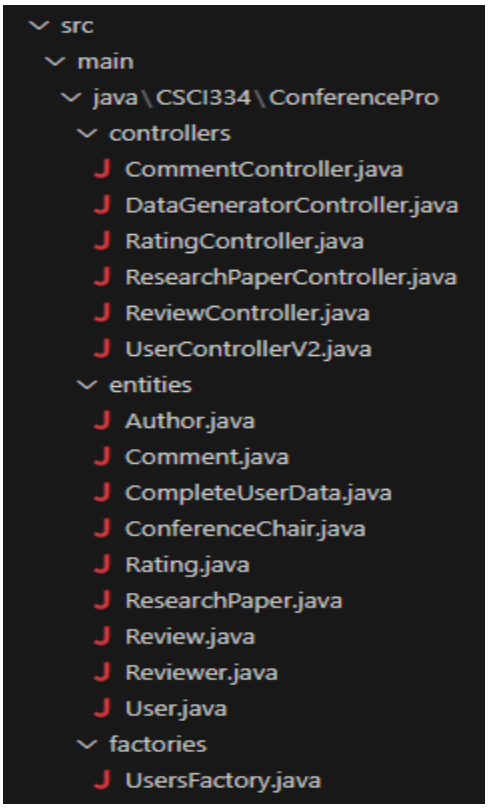


Figure 60: Entities and Controllers afterRefactoring

Due to the endpoint's ability to defer creation to the factory rather than requiring a user to declare the user type, refactoring enabled the design of a more user-friendly solution. By increasing the factory's capabilities, it makes future changes simple without requiring the endpoint to be modified.

1.11. Object Constraint Language (OCL)

1.11.1. Invariants and Functions

1. A paper needs to have 1 or more authors.

    Context ResearchPaper

        inv : self.authors.size() >= 1

2. The rating for a paper must be between -3 and 3.

    Context Review

        inv : self.score >= -3 and self.score <= 3

3. The workload preference of a reviewer must be between 1 and 10.

    Context Reviewer

        inv : self.paperLimit >= 1 and self.paperLimit <= 10

4. A maximum of 3 reviewers can be assigned to a paper.

    Context ResearchPaper

        inv : self.reviewers >= 0 and self.reviewers <= 3

5. A rating can only belong to one author.

    Context Rating

        inv : self.createdBy → isUnique(authorEmail)

6. An entity cannot be its own child: the intersection between the one element for the set that includes one entity _self and the set of the children of _self, has to be empty

    Set{self} → intersection (self.children} → isEmpty()

1.12. Software Engineering Principles

1.12.1 Basic Principles

Basic Software Engineering Principles	
Feasibility	The feasibility of the system's design has been demonstrated in its implementation as it has been developed to fully cover the critical functional and non-functional requirements that have been identified during the requirements analysis stage of the project. The use of React for the frontend has promoted the efficiency of the development as reusable components, such as the vertical navigation bar, dashboard for users, and row items, were created that helped in the effective development of pages that are similar in design and layout. The Backend development, on the other hand, has utilized Spring Boot which provides HTTP communication functionality, automatic instantiation and control of objects (there is no need to write a main() function telling the application what to do), and provides boiler plate functionality with MongoDB via another library. This further facilitates the development of the system and enhances its feasibility. MongoDB's Atlas is used as the database host. This service automates the creation and setup of databases, and because it's a NoSQL solution, there is no need to define strict schemas and relations before using the database which facilitates the development of the system. The system is also considered to be feasible in terms of its costs as there are direct, indirect, variable, nor fixed costs. This is because the system is an in-house software that did not incur setup costs nor labour costs since accessible tools were used to develop it with the help of an unpaid team. Being an in-house software also means that there are no external clients that could redirect the project's course. This saves time and allows the team members to efficiently use the limited amount of time (ten weeks) available to complete the project.
Adequacy	Several constraints, such as the commercial constraints [the limited time (ten weeks), budget ($zero), and manpower (seven members)], were used to develop the system that satisfies the stakeholders' needs by implementing the previously identified critical functional and non-functional requirements. Creating private attributes with public accessors and functions that are only made public when external classes need to call them has also helped in ensuring that the privacy

requirements in the compliance constraints are followed as only the required information is accessed by the relevant classes. This indicates that the data of the users is securely held within the system and is displayed only when necessary.

- Implemented functional requirements – *functional constraints*:

Functional Requirement	User
Choose a relevant role upon accessing the system	Authors, Reviewers, and Conference Chairs
Sign up	Authors and Reviewers
Login	Authors, Reviewers, and Conference Chairs
Set review and workload preferences	Reviewers
Submit papers for review	Authors
View papers that they have submitted	Authors
View all submitted papers	Conference Chairs
View all unassigned papers	Reviewers
View papers that have been assigned to them	Reviewers
Bid for papers	Reviewers
View a list of reviewers that bid for a paper	Conference Chairs
Manually allocate paper(s) to a bidder(s)	Conference Chairs
Automatically allocate paper(s) to a bidder(s)	Conference Chairs
Rate papers that have been assigned to them based on a given ratings system	Reviewers
Review papers that have been assigned to them	Reviewers
Edit a rating that they have given to a paper	Reviewers

Edit a review that they have given to a paper	Reviewers
View other reviewers' reviews	Reviewers
Discuss other reviewers' reviews with them	Reviewers
View papers' reviews and ratings	Conference Chairs
Accept/reject papers	Conference Chairs
View papers' ratings and reviews	Authors
View papers' acceptance/ rejection decisions	Authors
Rate papers' reviews	Authors

- Implemented non-functional requirements – *non-functional constraints*:

Non-functional Requirement	Constraint Type
System fully supports English	- Non-functional constraint - Usability constraint
System has good color combinations	- Non-functional constraint - Compliance constraint
System has visible buttons	- Non-functional constraint - Compliance constraint
System supports web browsers such as Chrome, Firefox, and Edge	- Non-functional constraint - Usability constraint
System is easy to use and navigate through	- Non-functional constraint - Usability constraint
System stores users' data securely	- Non-functional constraint - Compliance constraint
System is able to support many users simultaneously	- Non-functional constraint
System delivers the correct information when required	- Non-functional constraint
System is able to store many profiles, papers, and reviews with	- Non-functional constraint

	<table><tr><td>no issues arising</td><td></td></tr><tr><td>System is able to work 24/7 except during maintenance</td><td>- Non-functional constraint</td></tr></table>
Economy	As mentioned before, this project was carried out with no funding and just seven unpaid team members who used tools that were accessible. Therefore, there are no direct, indirect, fixed, or variable costs. Additionally, the project was completed in around ten weeks which is a reasonable amount of time for small projects and would not be expensive to carry out in the real world. Moreover, since the project was internal, no external clients could offer suggestions that might alter the project's course, eliminating the risks of the project taking longer than expected. Only the team members were involved in the project and were able to identify the requirements after conducting extensive research. As a result, the risk of scope creep was reduced as the identified requirements were labeled according to their importance. The fact that the project is internal also suggests that it would be less expensive in a real-world situation because no external organization would be employed to develop the system.
Changeability	Using React for the frontend development has helped in the creation of reusable components such as the vertical navigation bar, dashboard for users, and row items. These components support and facilitate the processes of updating the system and maintaining it. The backend of the system, on the other hand, uses components that are neat vertical slices (a controller, persistence class, and repository class). This allows the developers to easily add extra functionalities in the system, accommodating any changes.

1.12.2. Constructive Principles

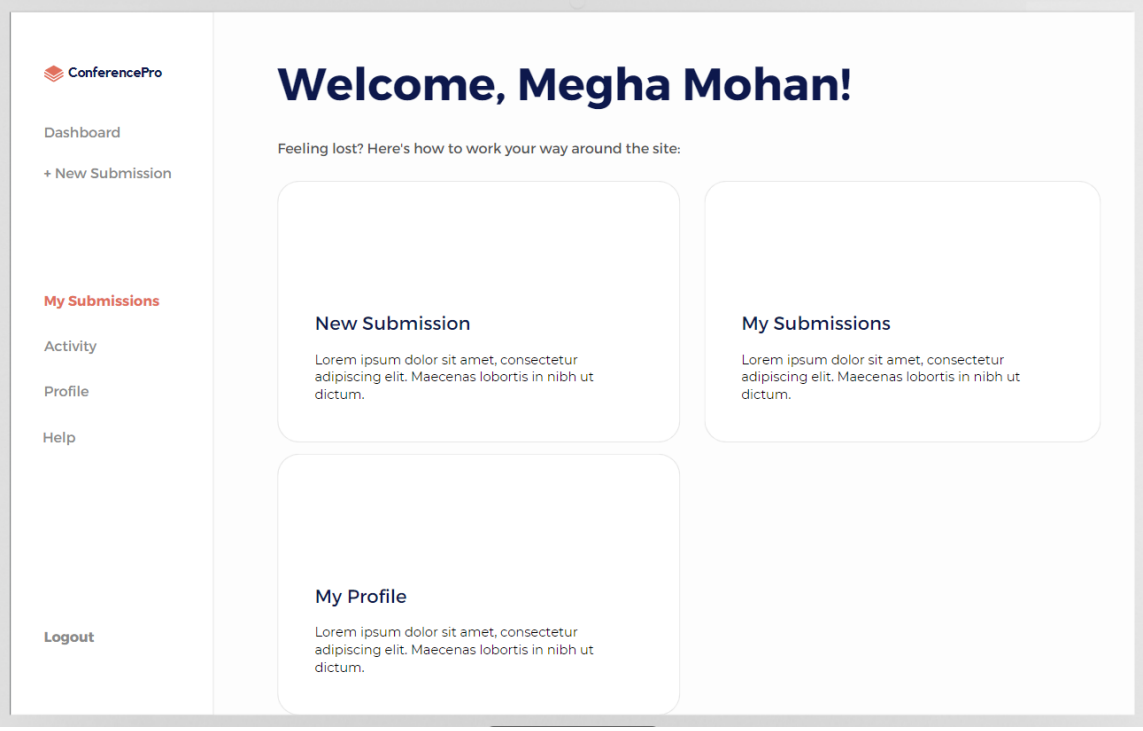
Constructive Software Principles		
Modularity	**Small Modules**	The system has been broken down into separate subsystems to keep classes small and focused. Using the Spring Boot framework allows dependencies to be added and custom build classes to inherit from existing superclasses that provide base functionality. More specific

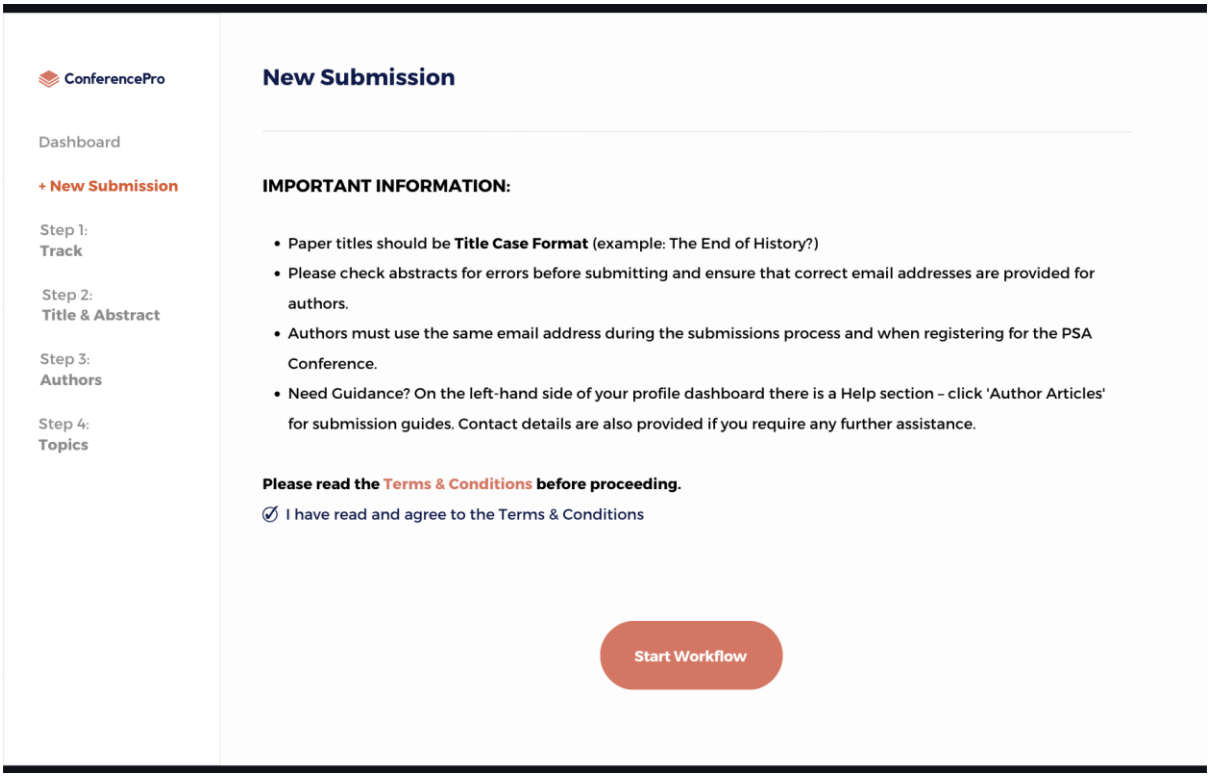
		functionality is added through developed classes that inherit from these superclasses.
	Information Hiding	All attributes within created classes are set to private with public accessors. Functions are only set to public if there is a need for external classes to call them. The three-tier architecture allows business logic to be hidden from clients and ensures that the system only provides information that is necessary for the operation of the browser client.
	Least Privilege	Classes are only made aware of the resources needed for their operation. This prevents unintended changes or functionality from unexpected parts of the system and increases security by limiting the attack surface of system components.
	Coupling	REST requests are used to connect the frontend to the backend, leaving them well decoupled. Any new program that needed to replace either the frontend or backend would only need to communicate using the appropriate entities and endpoints to effectively replace the old system, without the need to edit the existing system it would communicate with. Coupling between the backend and the database is very low. If the backend is pointed to another MongoDB Atlas instance it will read and write from the new instance without any additional configuration.
	Cohesion	The system is broken down into subsystems to ensure that each subsystem will be one cohesive unit. Restricting subsystems to only related use cases ensures that the subsystems are only accessing necessary common resources.
Implementability	**Simplicity**	As illustrated in the prototype, there are multiple design elements that can be reused. Such examples include the headings on each page and the vertical navigation tab. This is to ensure that it is simple to implement, while ensuring consistent branding throughout the website.
	Design with Reuse	As this website will be an entirely new product, there are no existing artifacts to build on. However, the production of these new assets will be reusable and allows for newer, additional features to easily reuse these existing ones effectively.

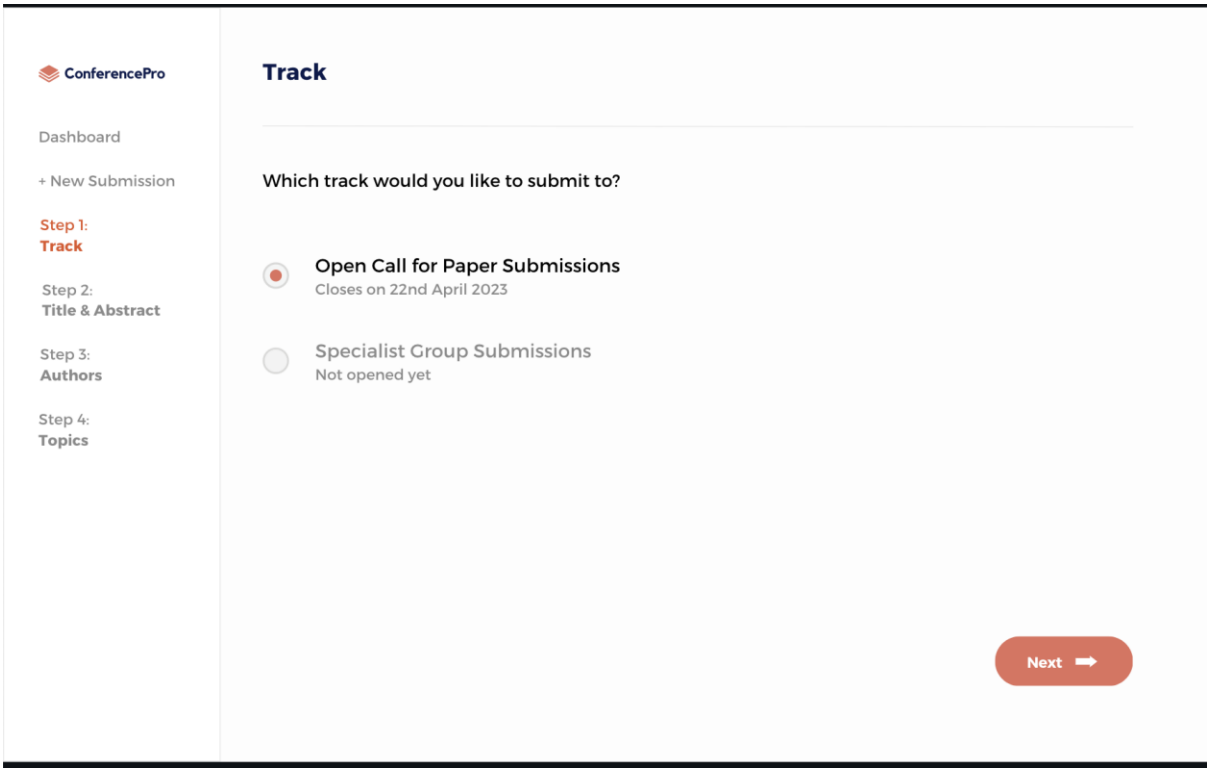
	Design for Reuse	As mentioned above, the design produces reusable assets, which contributes to a better design and implementation overall. This includes the use of the reusable vertical navigation bar, dashboard for users, and row items that were developed using React.
Aesthetic	**Beauty**	The design elements incorporated in the chosen prototype is meant to be simple, concise and is designed to support user learnability. The team designed this while keeping in mind to ensure that it is straightforward, easy to use and implement, and supports the stakeholders' requirements.

2. Appendix

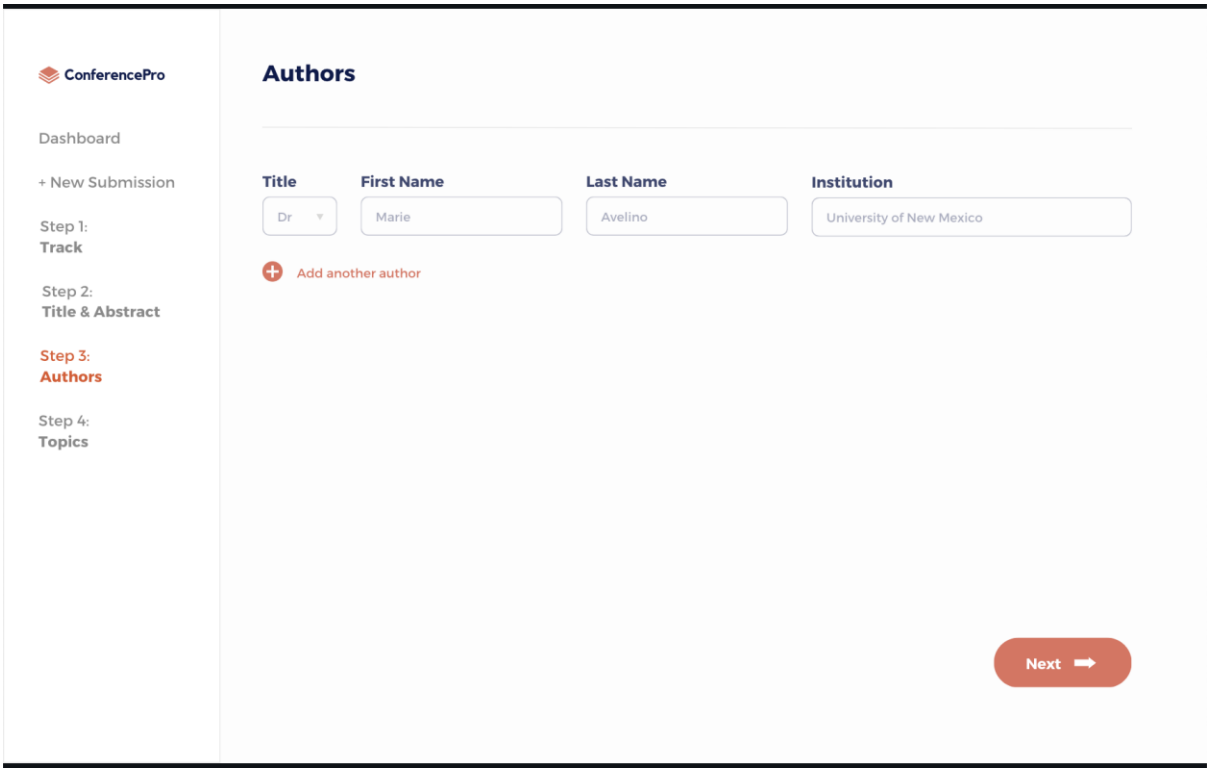
Authors' UI

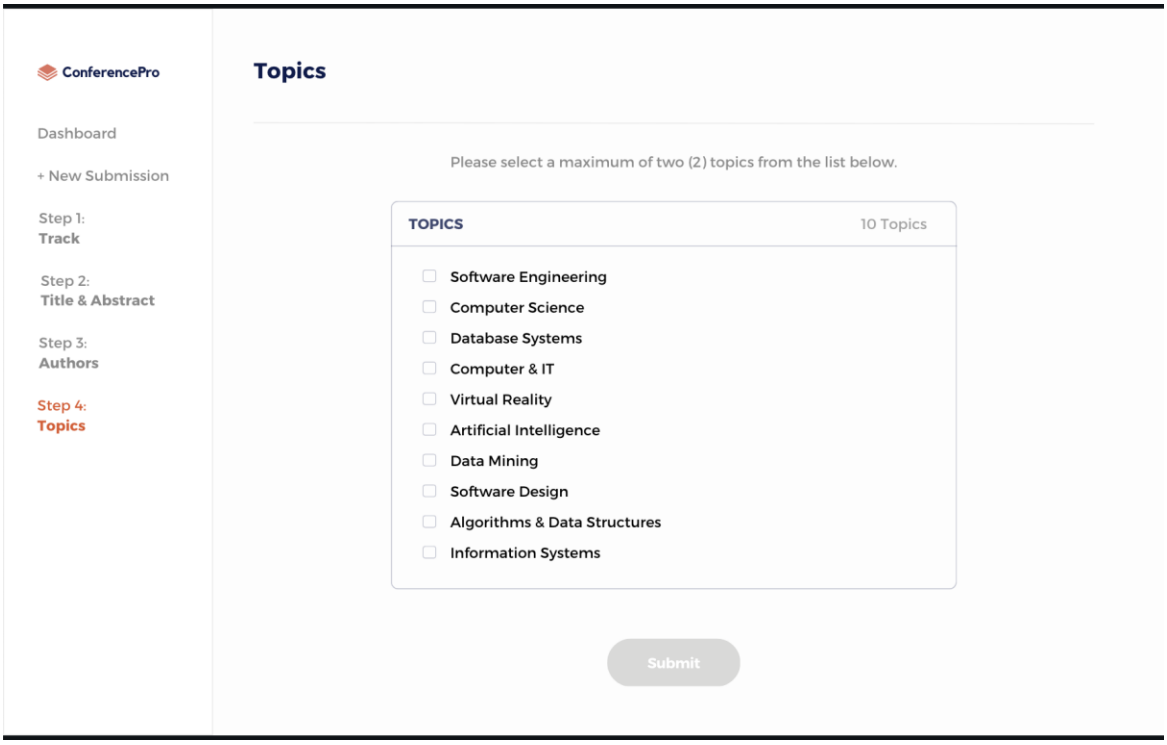


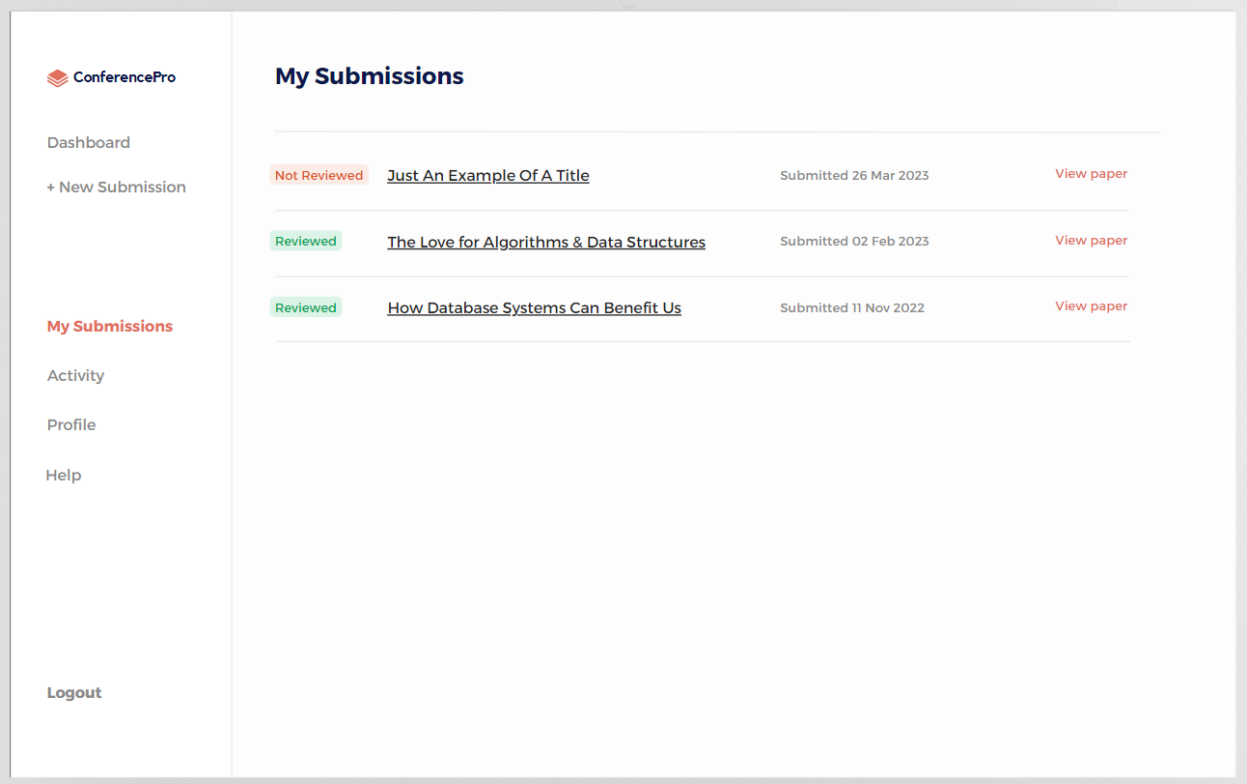


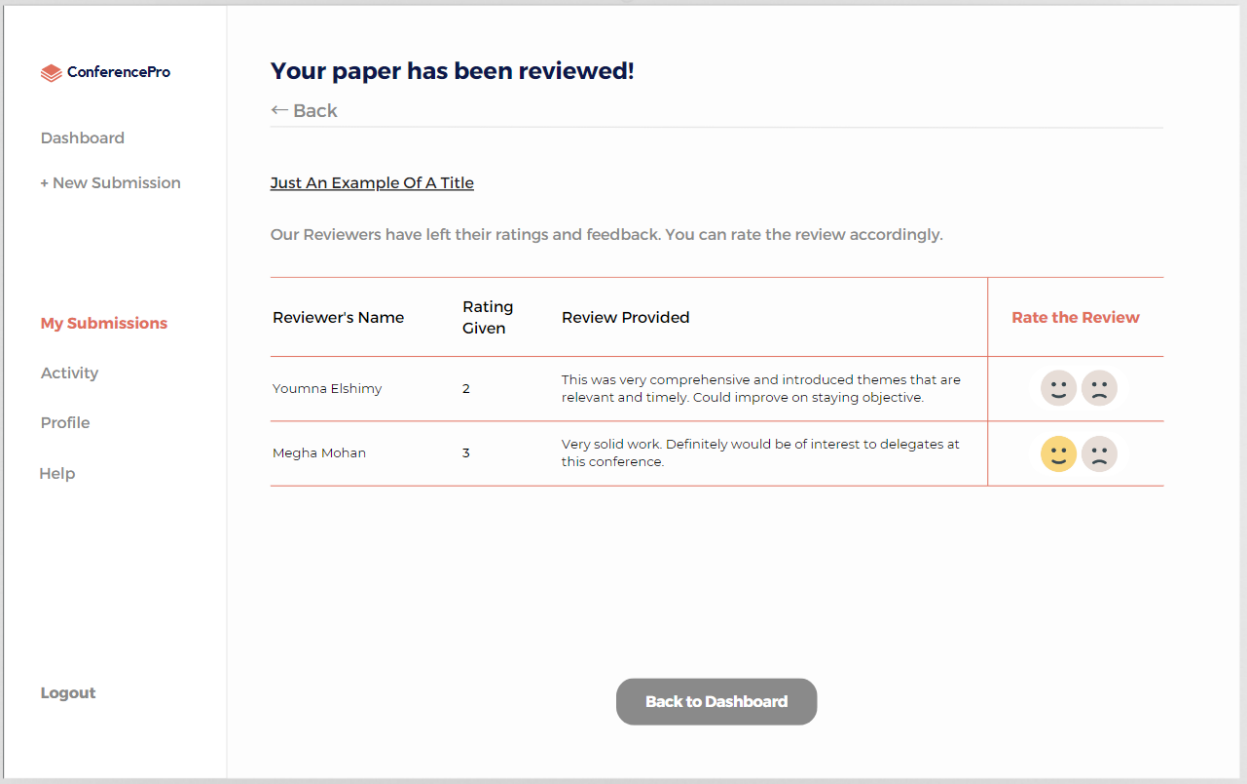

ConferencePro
Dashboard
+ New Submission
Step 1:
Track
Step 2:
Title & Abstract
Step 3:
Authors
Step 4:
Topics
Track
Which track would you like to submit to?
Open Call for Paper Submissions
Closes on 22nd April 2023
Specialist Group Submissions
Not opened yet
Next

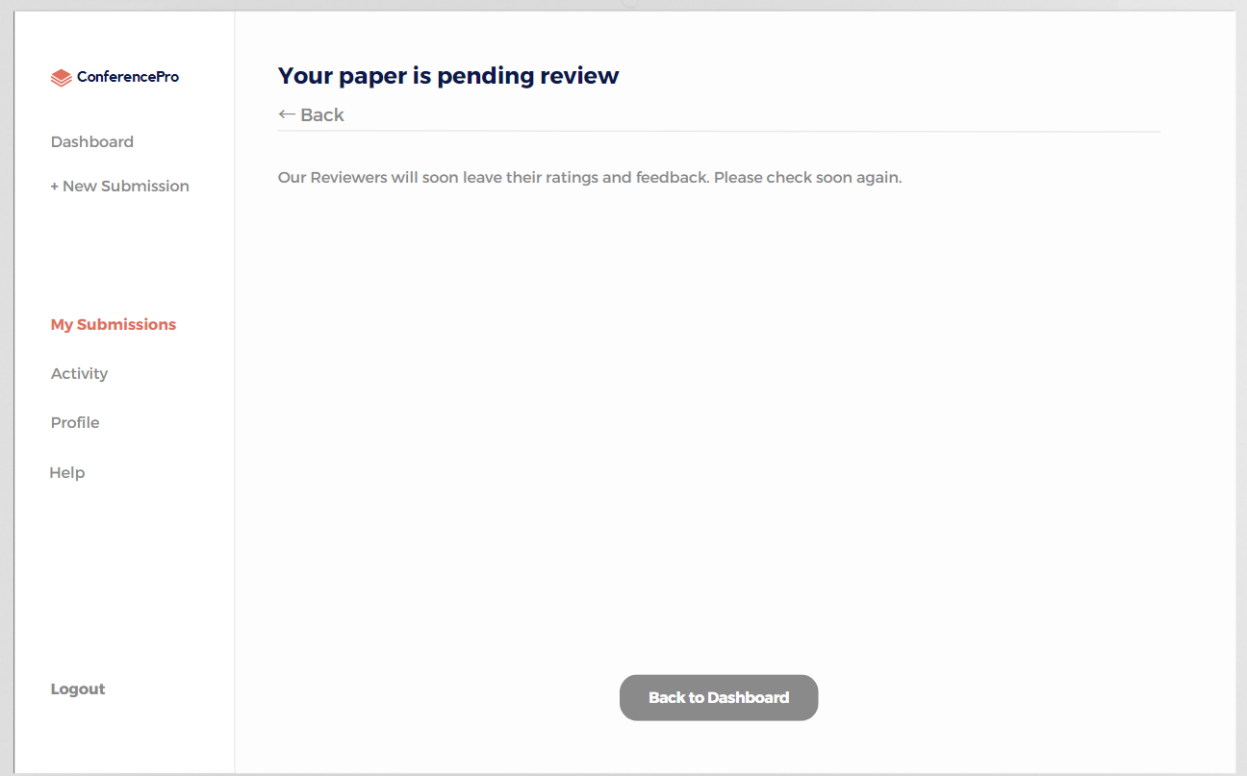

ConferencePro
Dashboard
+ New Submission
Step 1:
Track
Step 2:
Title & Abstract
Step 3:
Authors
Step 4:
Topics
Title & Abstract
Title *
Abstract *
Next


ConferencePro
Dashboard
+ New Submission
Step 1:
Track
Step 2:
Title & Abstract
Step 3:
Authors
Step 4:
Topics
Authors
Title
First Name
Last Name
Institution
Dr
Marie
Avelino
University of New Mexico
Add another author
Next

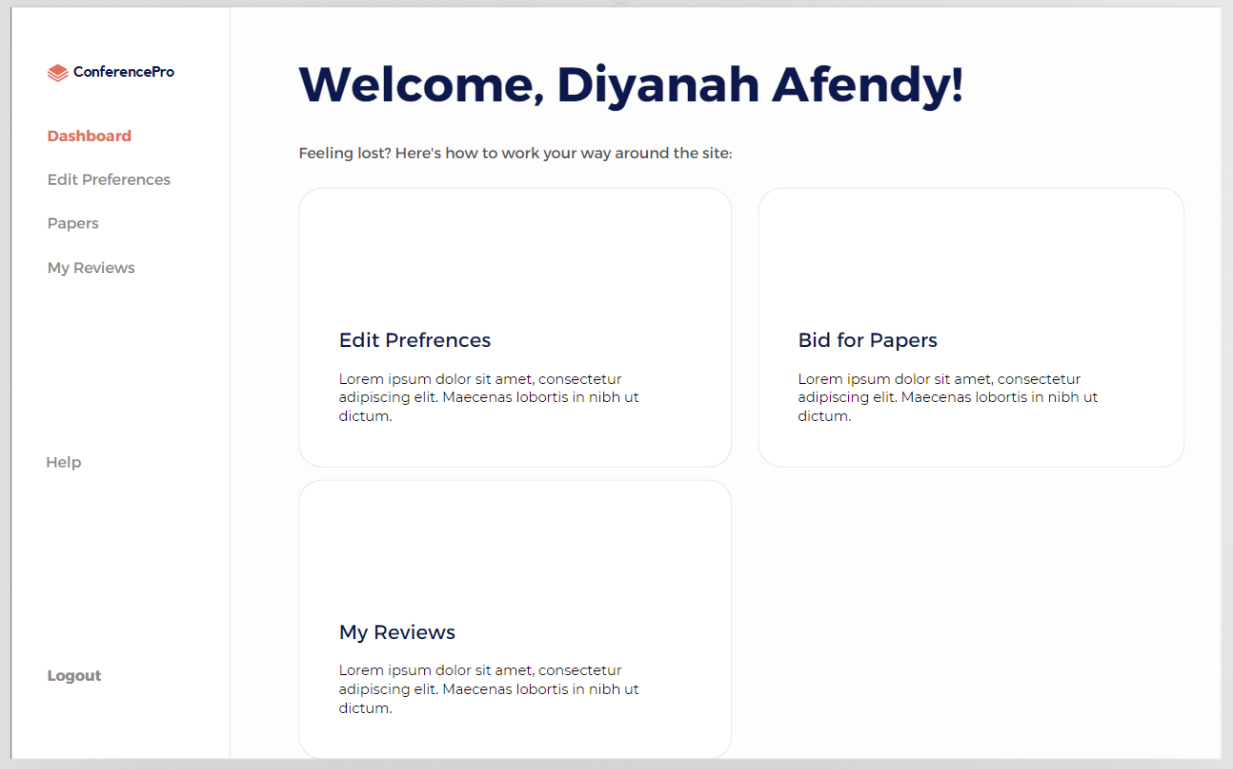

ConferencePro
Dashboard
+ New Submission
Step 1:
Track
Step 2:
Title & Abstract
Step 3:
Authors
Step 4:
Topics
Topics
Please select a maximum of two (2) topics from the list below.
TOPICS
10 Topics
Software Engineering
Computer Science
Database Systems
Computer & IT
Virtual Reality
Artificial Intelligence
Data Mining
Software Design
Algorithms & Data Structures
Information Systems
Submit

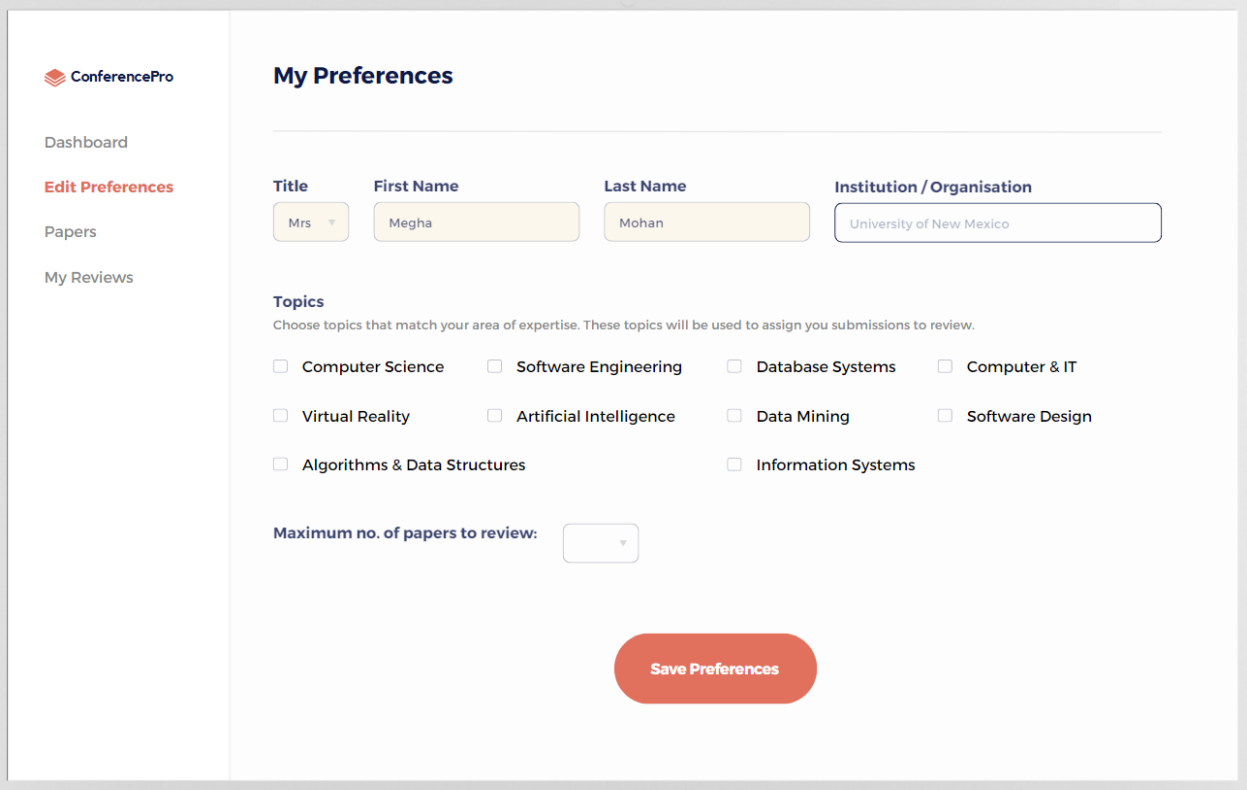

ConferencePro
Dashboard
+ New Submission
My Submissions
Activity
Profile
Help
Logout
My Submissions
Not Reviewed
Just An Example Of A Title
Submitted 26 Mar 2023
View paper
Reviewed
The Love for Algorithms & Data Structures
Submitted 02 Feb 2023
View paper
Reviewed
How Database Systems Can Benefit Us
Submitted 11 Nov 2022
View paper

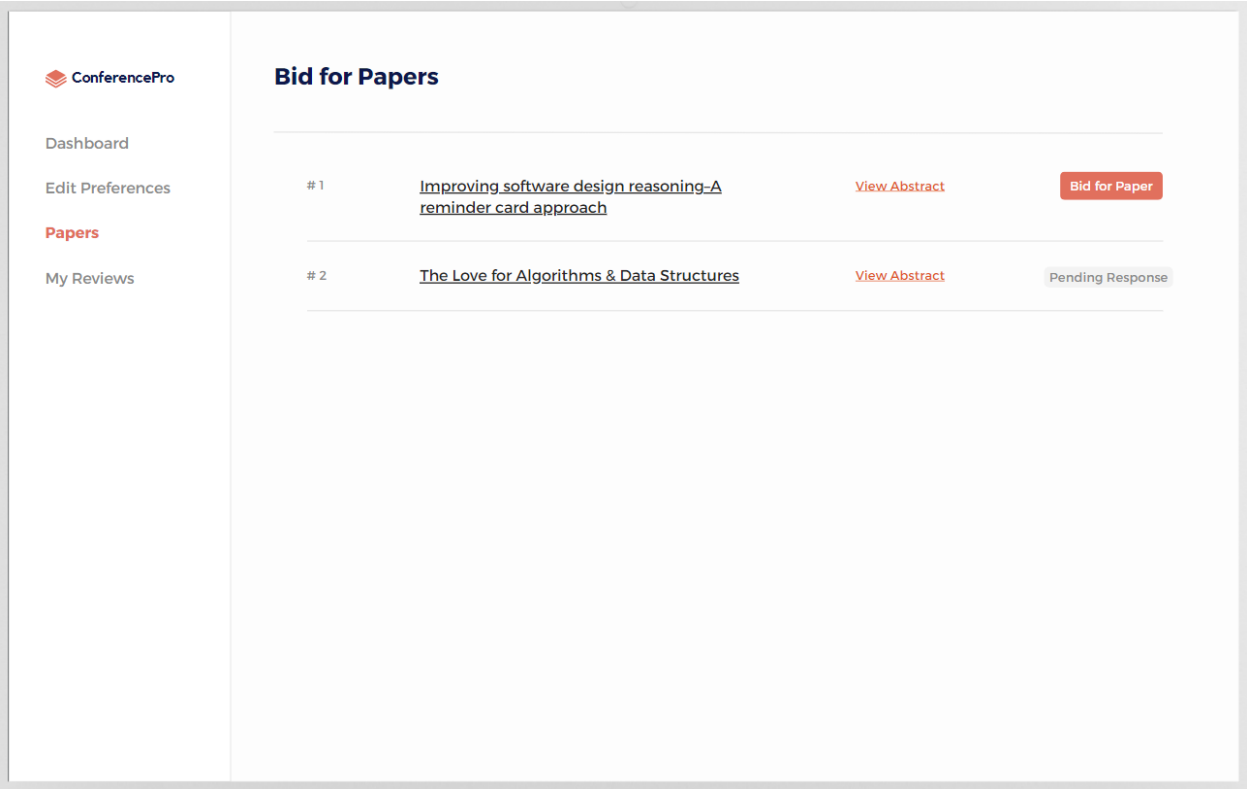

ConferencePro
Dashboard
+ New Submission
My Submissions
Activity
Profile
Help
Logout
Your paper has been reviewed!
← Back
Just An Example Of A Title
Our Reviewers have left their ratings and feedback. You can rate the review accordingly.
Reviewer's Name
Rating Given
Review Provided
Rate the Review
Youmna Elshimy
2
This was very comprehensive and introduced themes that are relevant and timely. Could improve on staying objective.
Megha Mohan
3
Very solid work. Definitely would be of interest to delegates at this conference.
Back to Dashboard

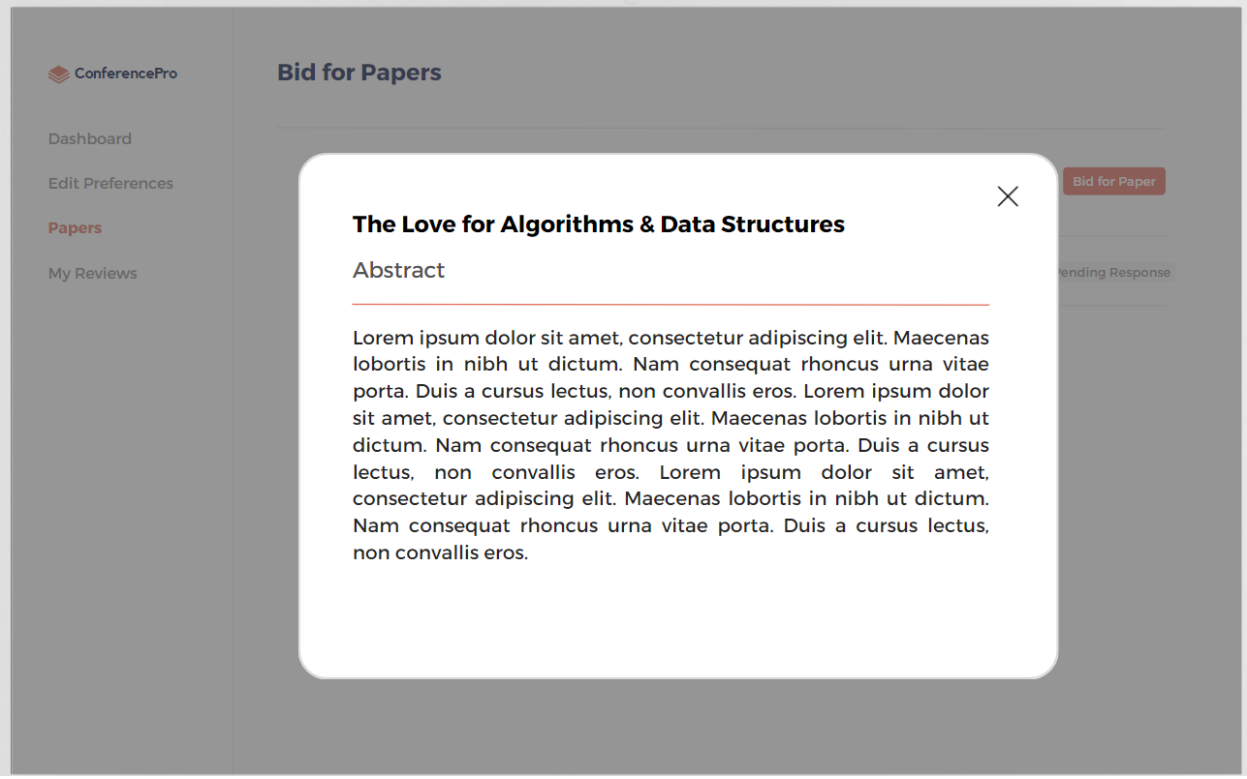


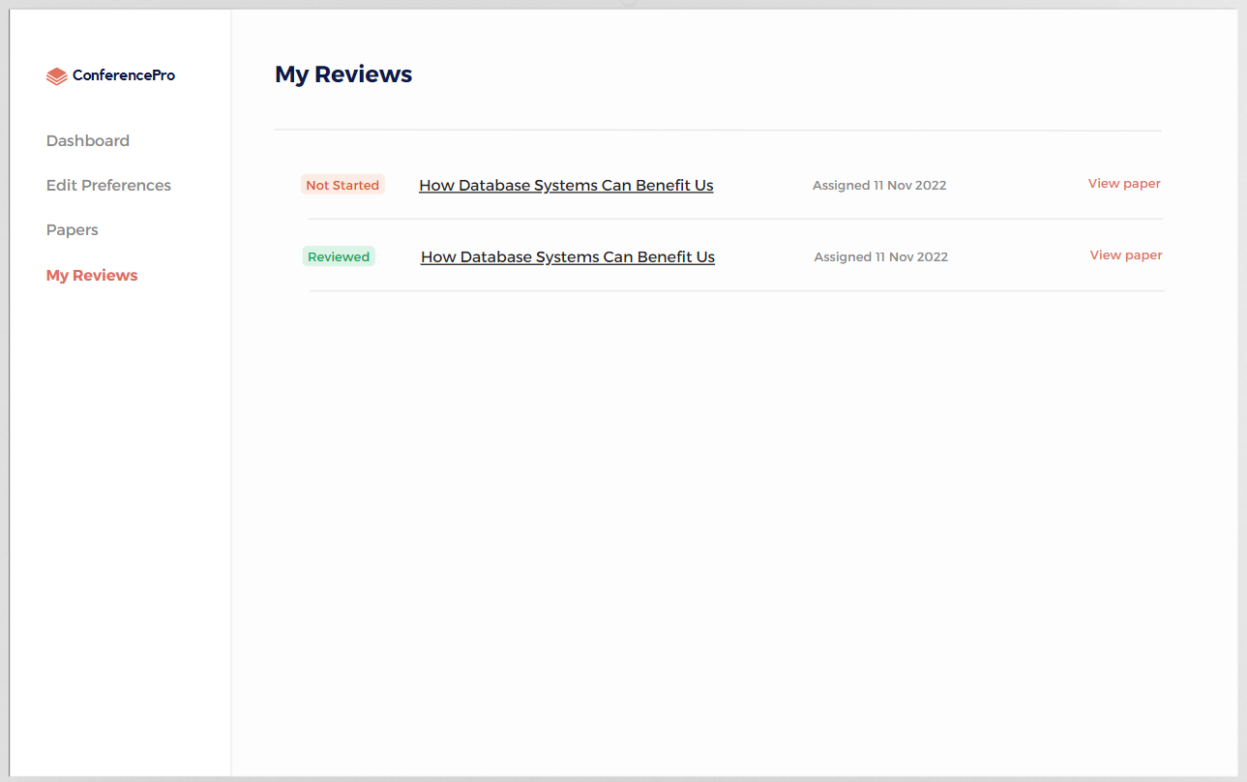
Reviewers' UI

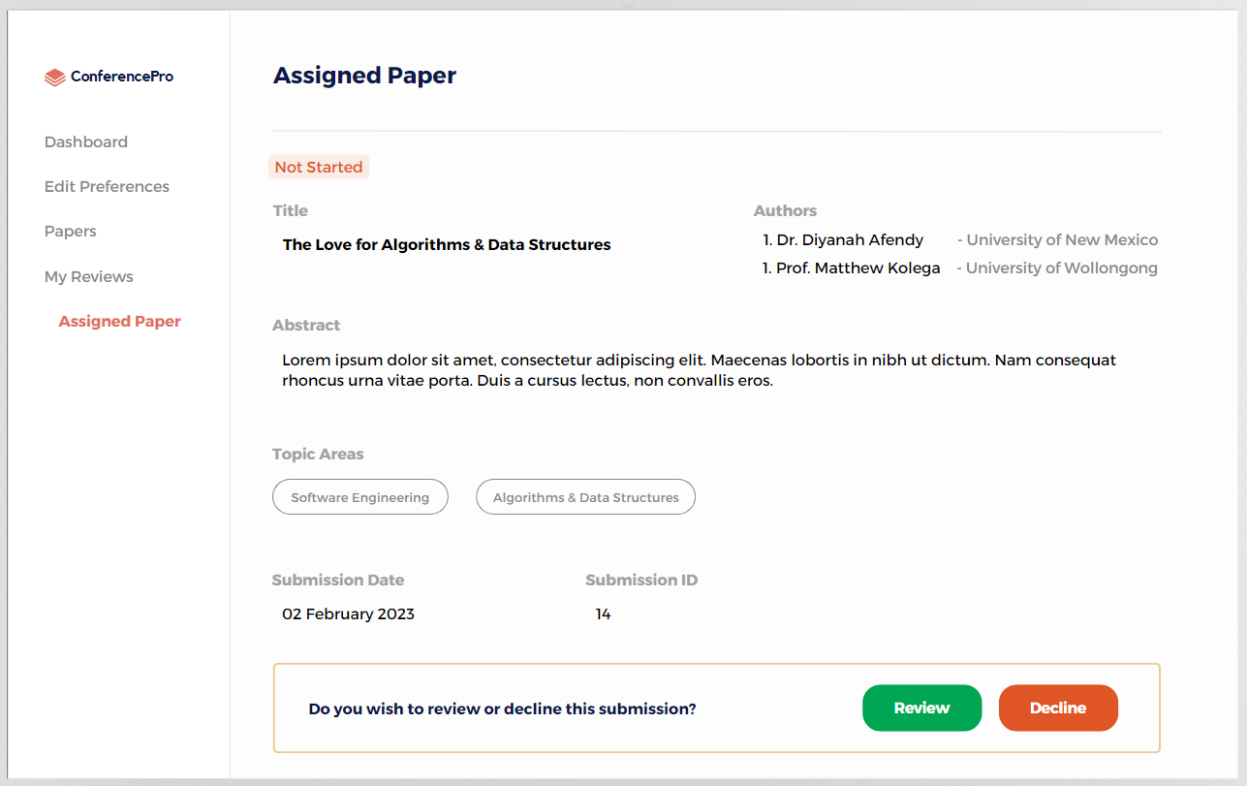


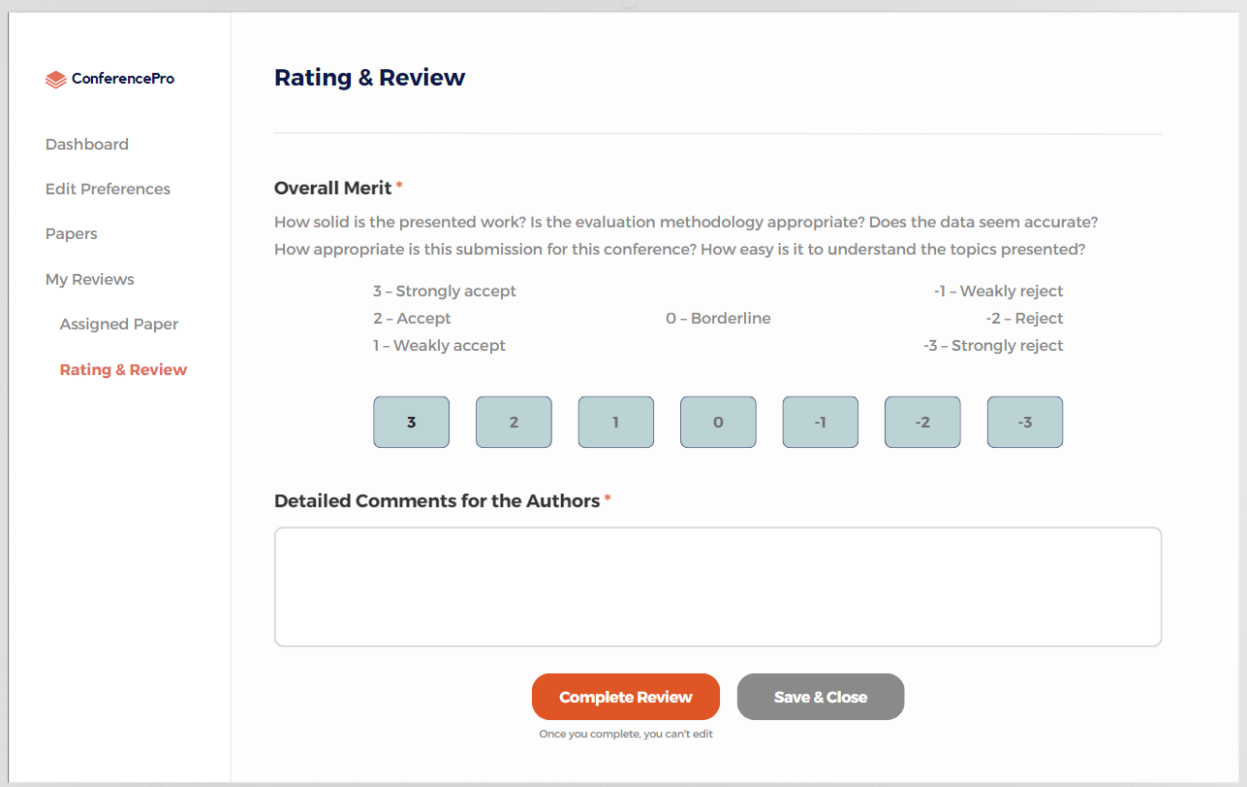

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
My Preferences
Title
Mrs
First Name
Megha
Last Name
Mohan
Institution / Organisation
University of New Mexico
Topics
Choose topics that match your area of expertise. These topics will be used to assign you submissions to review.
Computer Science
Software Engineering
Database Systems
Computer & IT
Virtual Reality
Artificial Intelligence
Data Mining
Software Design
Algorithms & Data Structures
Information Systems
Maximum no. of papers to review:
Save Preferences

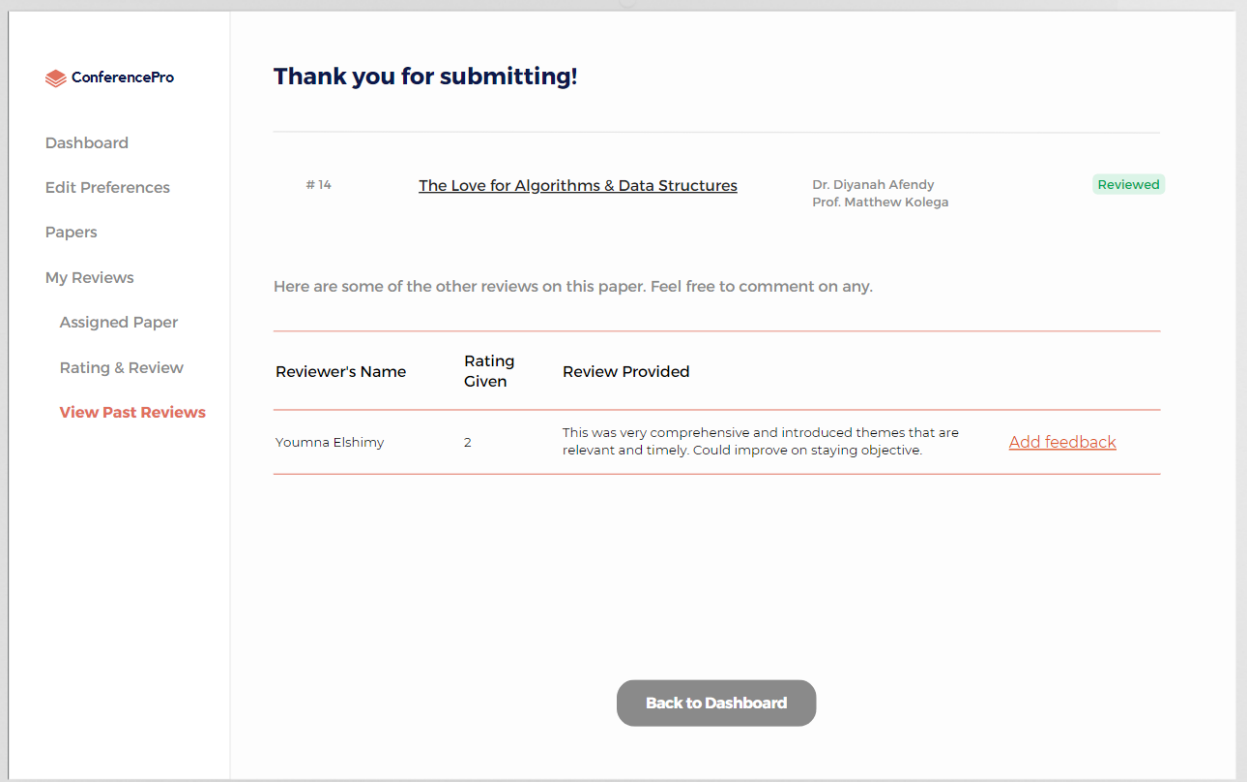

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
Bid for Papers
1
Improving software design reasoning–A reminder card approach
View Abstract
Bid for Paper
2
The Love for Algorithms & Data Structures
View Abstract
Pending Response

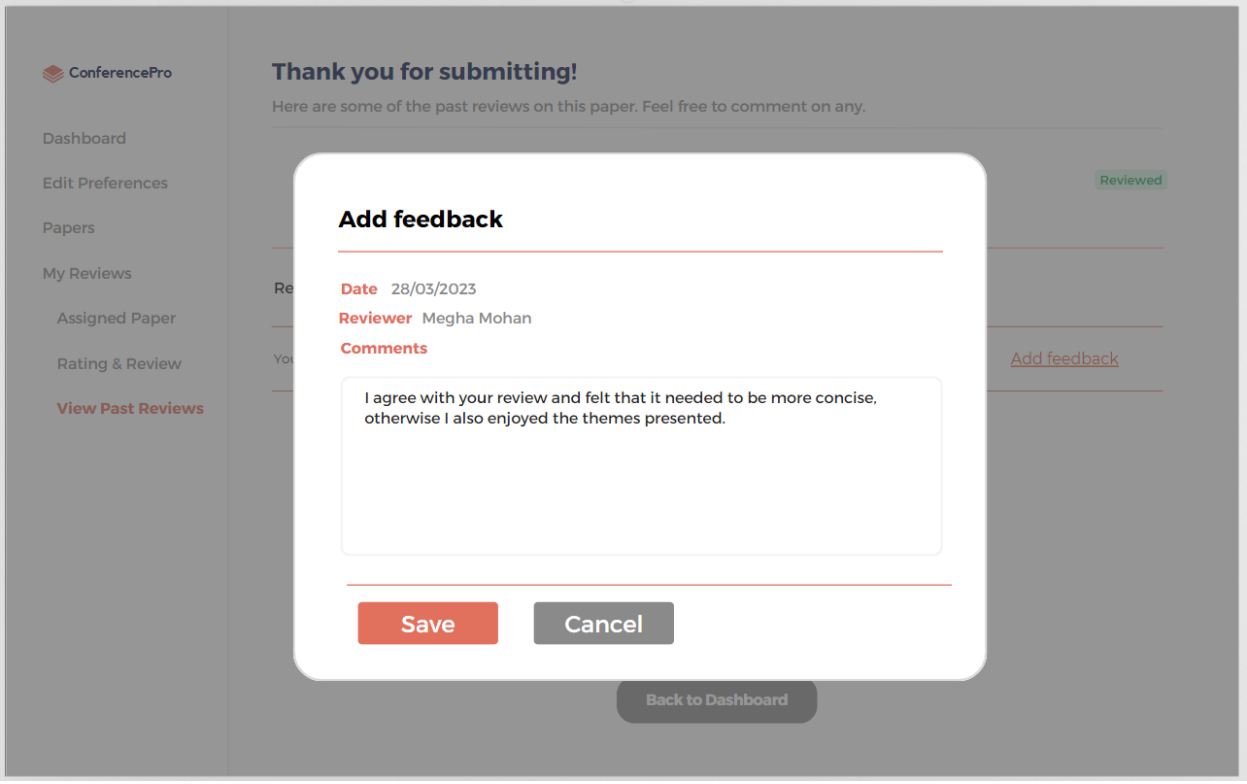

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
Bid for Papers
Bid for Paper
The Love for Algorithms & Data Structures
Abstract
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis in nibh ut dictum. Nam consequat rhoncus urna vitae porta. Duis a cursus lectus, non convallis eros. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis in nibh ut dictum. Nam consequat rhoncus urna vitae porta. Duis a cursus lectus, non convallis eros. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis in nibh ut dictum. Nam consequat rhoncus urna vitae porta. Duis a cursus lectus, non convallis eros.


ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
My Reviews
Not Started
How Database Systems Can Benefit Us
Assigned 11 Nov 2022
View paper
Reviewed
How Database Systems Can Benefit Us
Assigned 11 Nov 2022
View paper

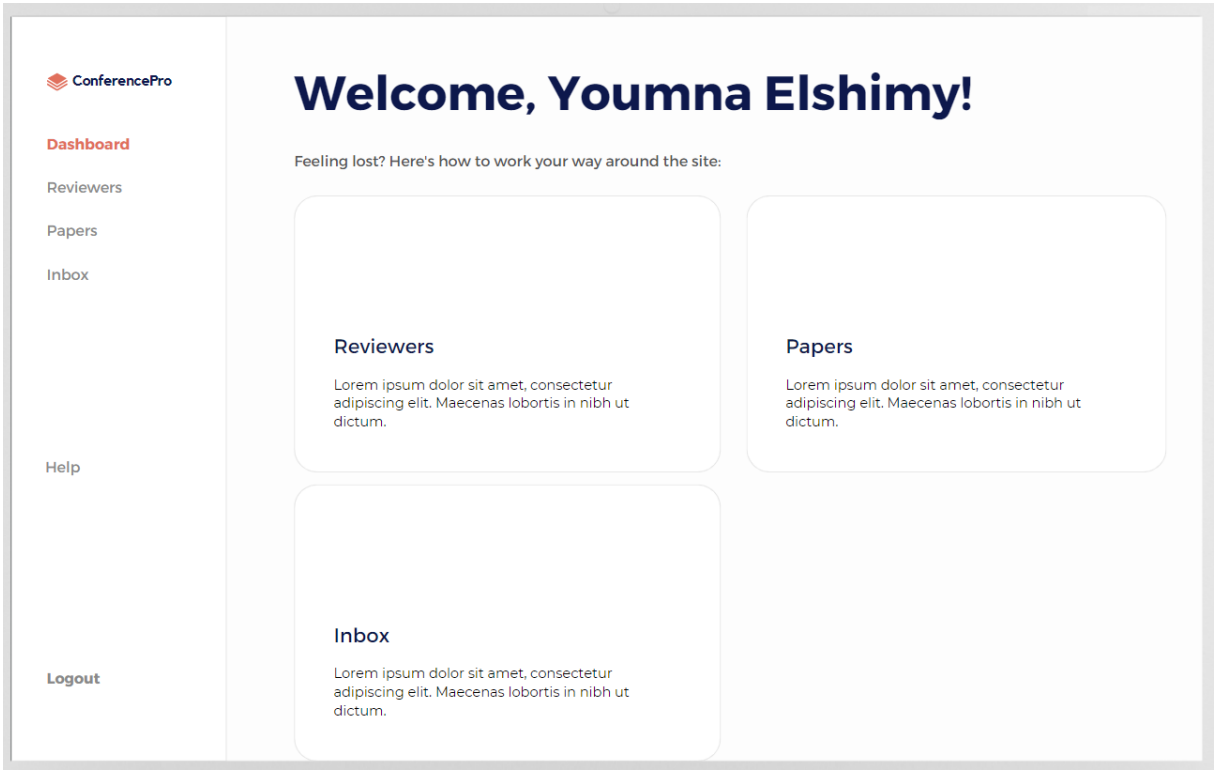

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
Assigned Paper
Assigned Paper
Not Started
Title
The Love for Algorithms & Data Structures
Authors
1. Dr. Diyanah Afendy - University of New Mexico
1. Prof. Matthew Kolega - University of Wollongong
Abstract
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis in nibh ut dictum. Nam consequat rhoncus urna vitae porta. Duis a cursus lectus, non convallis eros.
Topic Areas
Software Engineering
Algorithms & Data Structures
Submission Date
02 February 2023
Submission ID
14
Do you wish to review or decline this submission?
Review
Decline

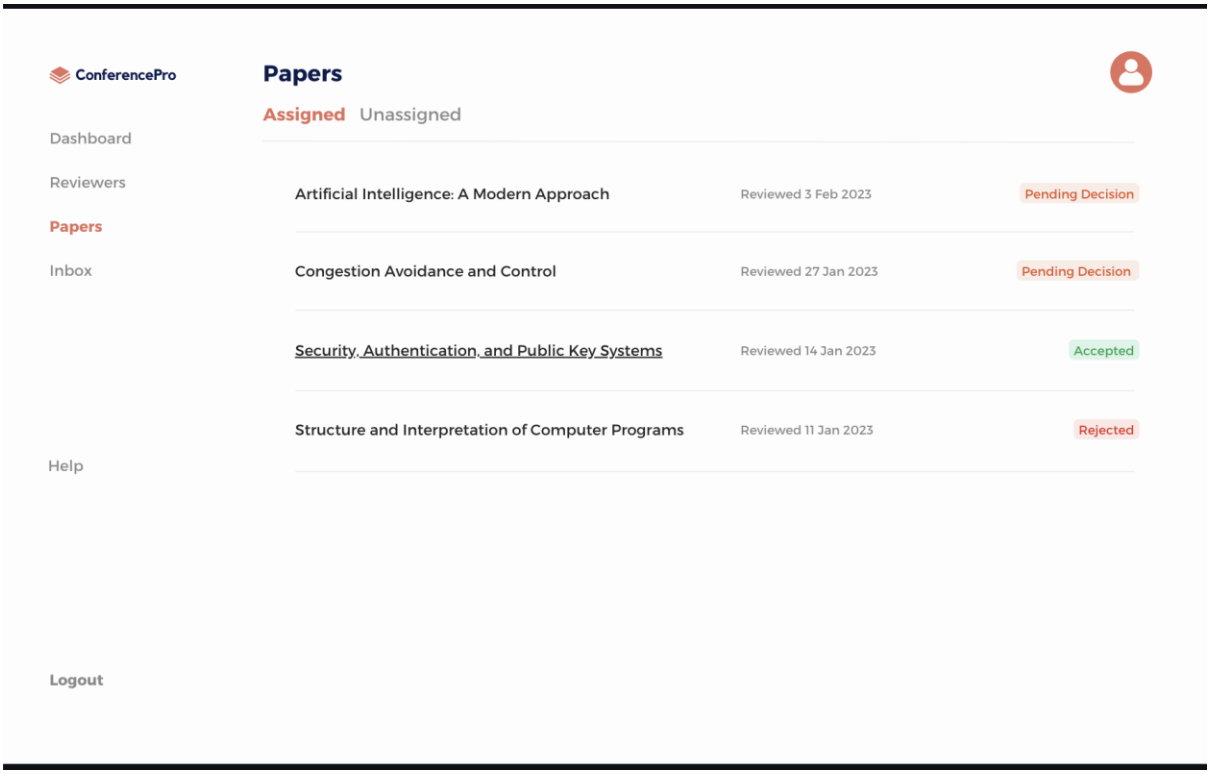

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
Assigned Paper
Rating & Review
Rating & Review
Overall Merit *
How solid is the presented work? Is the evaluation methodology appropriate? Does the data seem accurate?
How appropriate is this submission for this conference? How easy is it to understand the topics presented?
3 – Strongly accept
2 – Accept
1 – Weakly accept
0 – Borderline
-1 – Weakly reject
-2 – Reject
-3 – Strongly reject
3
2
1
0
-1
-2
-3
Detailed Comments for the Authors *
Complete Review
Save & Close
Once you complete, you can't edit

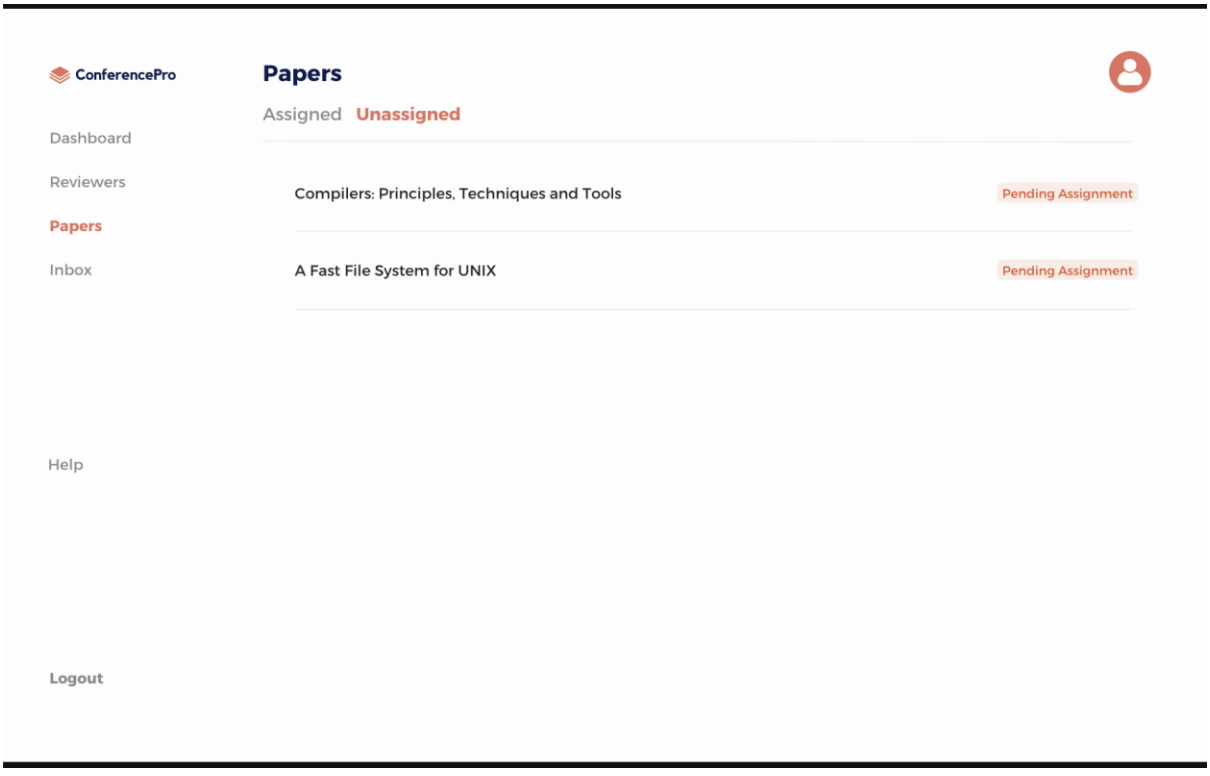

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
Assigned Paper
Rating & Review
View Past Reviews
Thank you for submitting!
14
The Love for Algorithms & Data Structures
Dr. Diyanah Afendy
Prof. Matthew Kolega
Reviewed
Here are some of the other reviews on this paper. Feel free to comment on any.
Reviewer's Name
Rating Given
Review Provided
Youmna Elshimy
2
This was very comprehensive and introduced themes that are relevant and timely. Could improve on staying objective.
Add feedback
Back to Dashboard

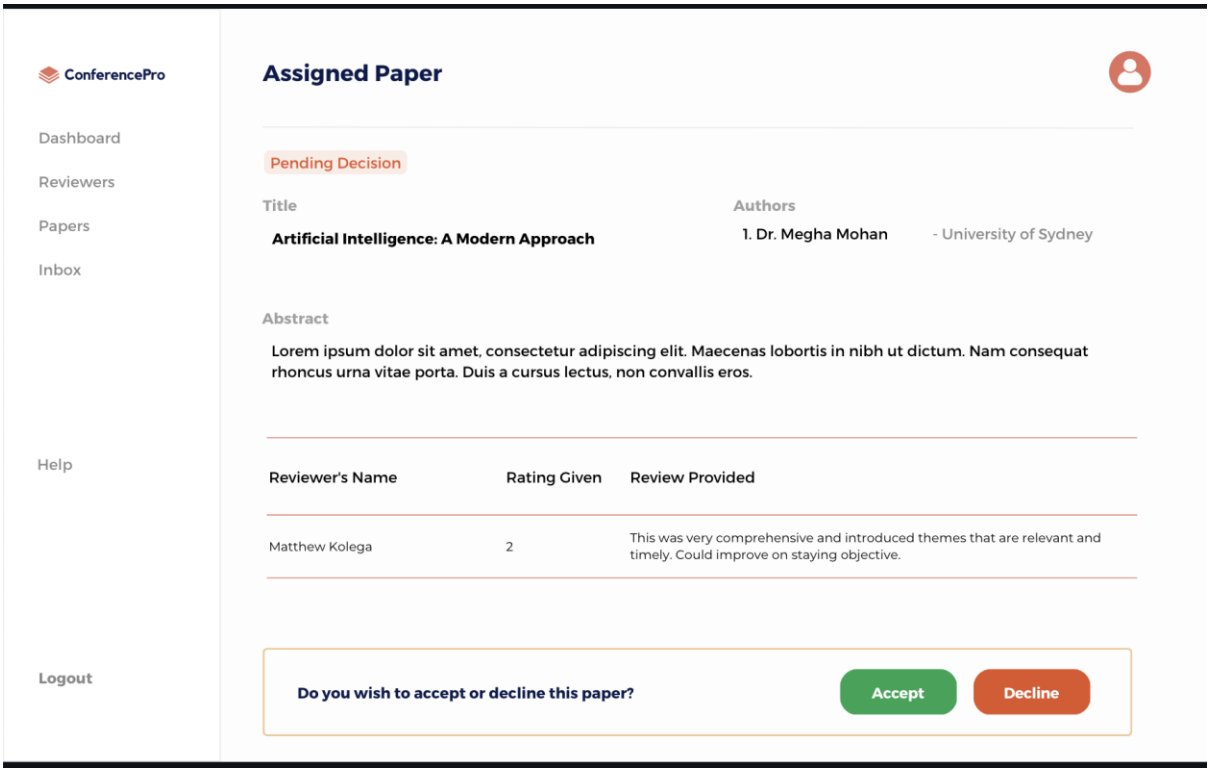

ConferencePro
Dashboard
Edit Preferences
Papers
My Reviews
Assigned Paper
Rating & Review
View Past Reviews
Thank you for submitting!
Here are some of the past reviews on this paper. Feel free to comment on any.
Reviewed
Add feedback
Add feedback
Date 28/03/2023
Reviewer Megha Mohan
Comments
I agree with your review and felt that it needed to be more concise, otherwise I also enjoyed the themes presented.
Save
Cancel
Back to Dashboard

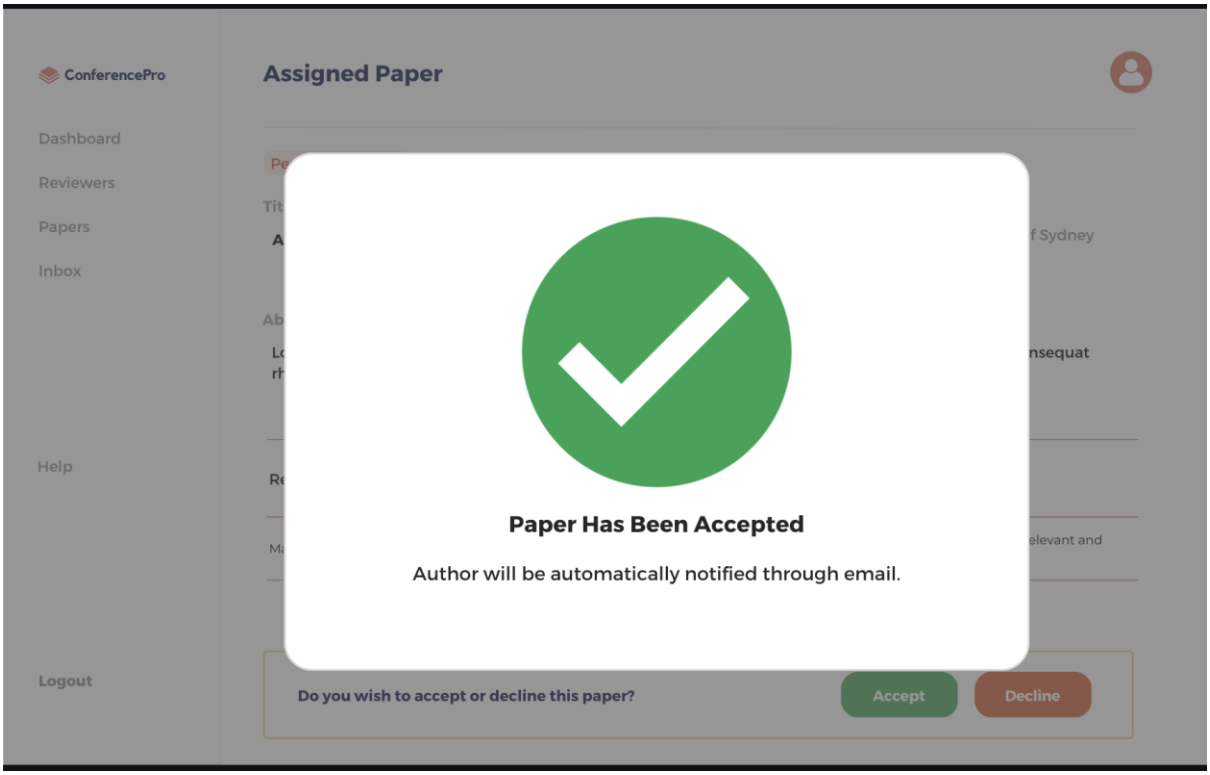
Conference Chairs' UI

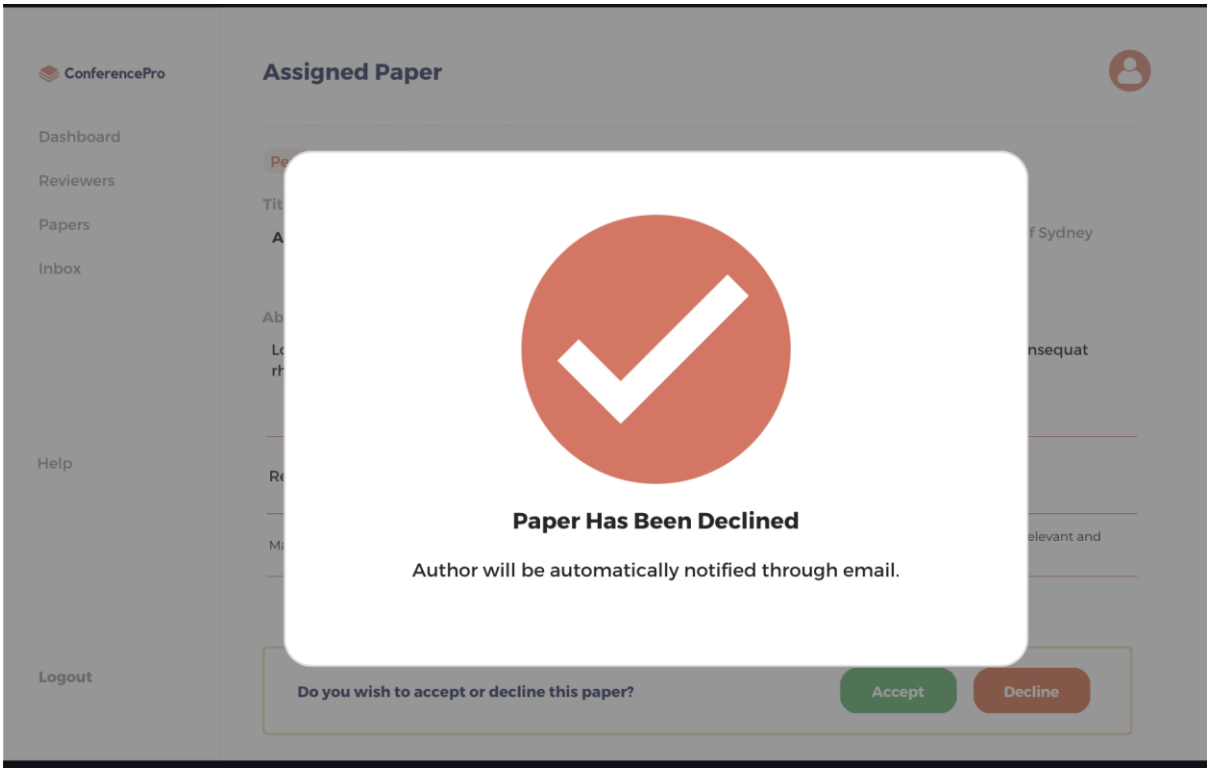


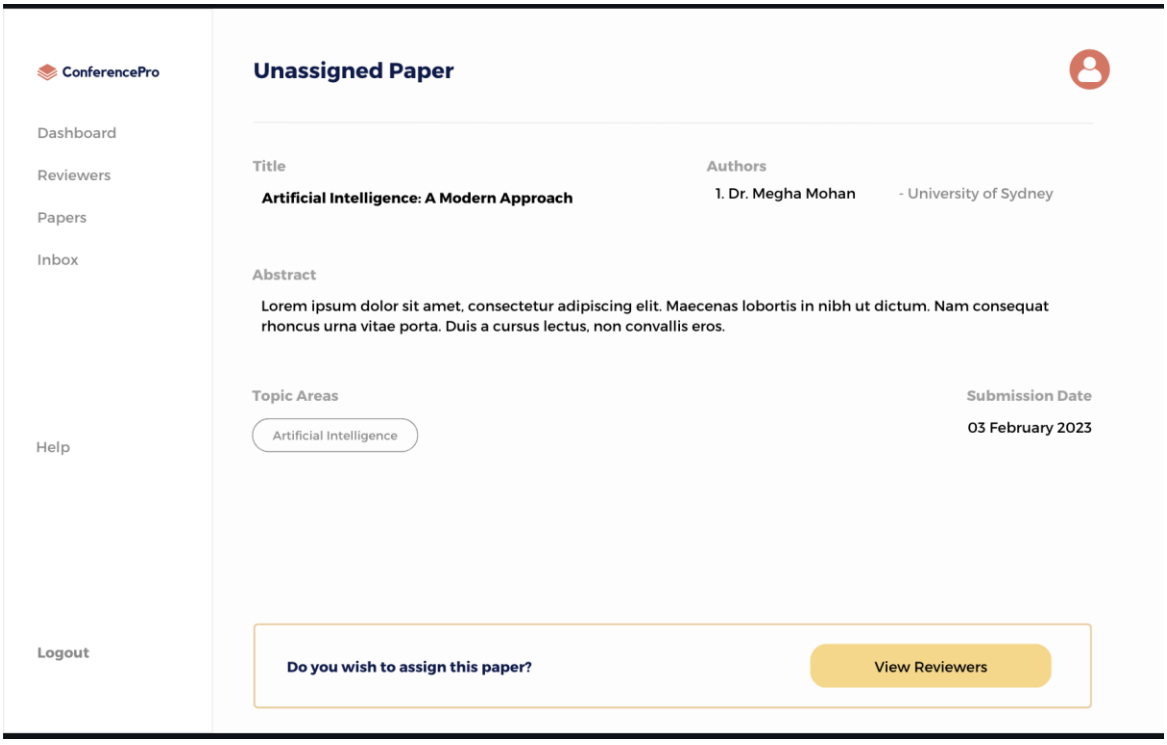

ConferencePro
Papers
Assigned Unassigned
Dashboard
Reviewers
Papers
Inbox
Artificial Intelligence: A Modern Approach
Reviewed 3 Feb 2023
Pending Decision
Congestion Avoidance and Control
Reviewed 27 Jan 2023
Pending Decision
Security, Authentication, and Public Key Systems
Reviewed 14 Jan 2023
Accepted
Structure and Interpretation of Computer Programs
Reviewed 11 Jan 2023
Rejected
Help
Logout


ConferencePro
Papers
Assigned Unassigned
Dashboard
Reviewers
Papers
Inbox
Compilers: Principles, Techniques and Tools
Pending Assignment
A Fast File System for UNIX
Pending Assignment
Help
Logout

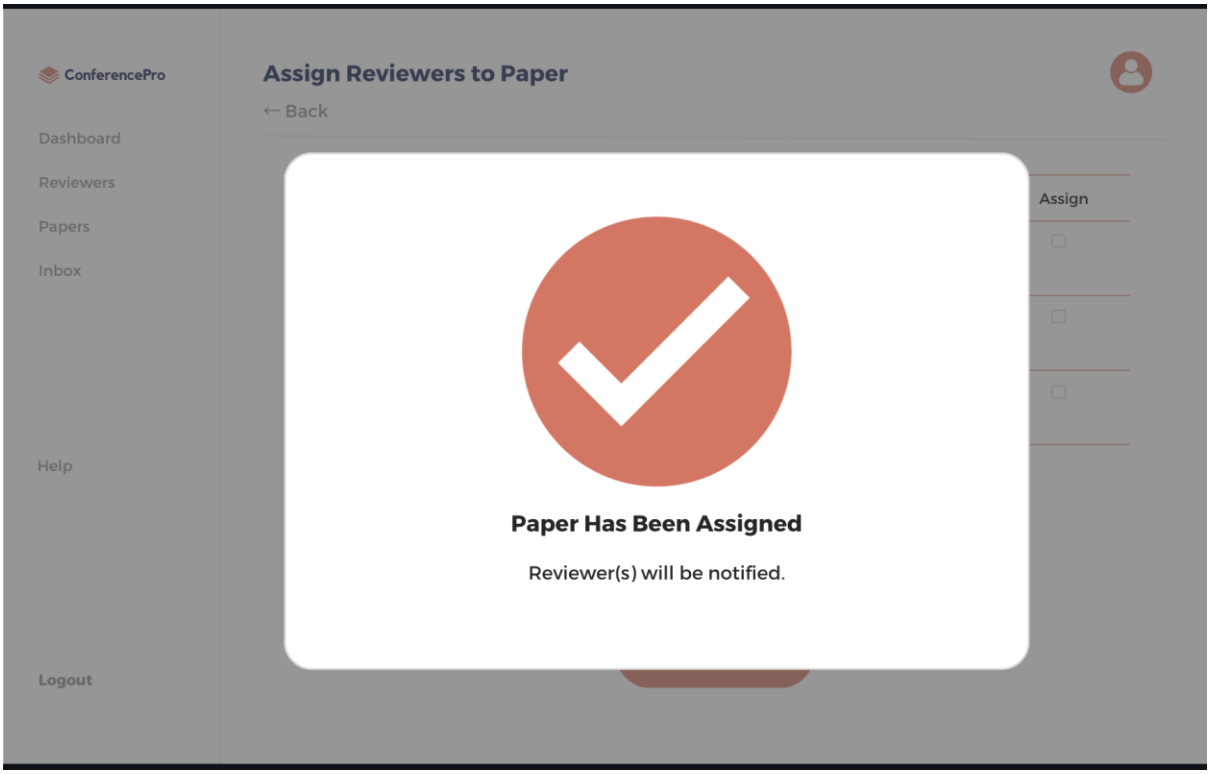

ConferencePro
Dashboard
Reviewers
Papers
Inbox
Help
Logout
Assigned Paper
Pending Decision
Title
Artificial Intelligence: A Modern Approach
Authors
1. Dr. Megha Mohan
- University of Sydney
Abstract
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis in nibh ut dictum. Nam consequat rhoncus urna vitae porta. Duis a cursus lectus, non convallis eros.
Reviewer's Name
Rating Given
Review Provided
Matthew Kolega
2
This was very comprehensive and introduced themes that are relevant and timely. Could improve on staying objective.
Do you wish to accept or decline this paper?
Accept
Decline


ConferencePro
Dashboard
Reviewers
Papers
Inbox
Help
Logout
Assigned Paper
Paper Has Been Accepted
Author will be automatically notified through email.
Do you wish to accept or decline this paper?
Accept
Decline


ConferencePro
Assigned Paper
Dashboard
Reviewers
Papers
Inbox
Help
Logout
Paper Has Been Declined
Author will be automatically notified through email.
Do you wish to accept or decline this paper?
Accept
Decline


ConferencePro
Unassigned Paper
Dashboard
Reviewers
Papers
Inbox
Help
Logout
Title
Artificial Intelligence: A Modern Approach
Authors
1. Dr. Megha Mohan
- University of Sydney
Abstract
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas lobortis in nibh ut dictum. Nam consequat rhoncus urna vitae porta. Duis a cursus lectus, non convallis eros.
Topic Areas
Artificial Intelligence
Submission Date
03 February 2023
Do you wish to assign this paper?
View Reviewers


ConferencePro
Dashboard
Reviewers
Papers
Inbox
Help
Logout
Assign Reviewers to Paper
← Back
Name
Specialties
Workload
Status
Assign
Youmna Elshimy
• Artificial Intelligence
• Computer & IT
7/9
Available for Assignment
Diyanah Afendy
• Data Mining
6/8
Available for Assignment
Hibah Jasim
• Software Design
7/10
Available for Assignment
Confirm


ConferencePro
Dashboard
Reviewers
Papers
Inbox
Help
Logout
Assign Reviewers to Paper
← Back
Assign
Paper Has Been Assigned
Reviewer(s) will be notified.