Grady’sBooch

Conference management for everyone

www.gradysbooch.com

By GradysBooch© software solutions

I. Short presentation of the brief

The Grady's Booch Conference Management System is an online tool for organizing and attending conferences. The project was developed over the course of 14 weeks, by a team of second-year Computer Science students at the Babes-Bolyai Faculty of Mathematics and Computer Science.

The Conference Management System allows users to create and attend conferences, as well as submit and review scientific papers related to a conference's subjects. Each process is simple and effortless, from creating an account, to hosting a conference or grading a paper.

Those who want to attend a conference only have to create an account, select one of the conferences displayed on the website, and buy a ticket. For those who want to submit a scientific paper to one of the upcoming conferences in order to try and get a speaker position, the process is almost identical, only they do not have to purchase a ticket.

Conference organizers should also have an easy time creating and managing conferences, as our Conference Management System gives them many powerful tools, such as splitting a conference into specialized sections, setting the number and price of tickets, and creating review committees for submitted papers.

In order to bring our vision of the Conference Management System to light, we had to split our team into smaller ones, each one learning new technologies and tackling different problems. For the planning part, we had to create multiple diagrams that helped us see eye to eye on the many functions of the project. Once everyone had a clear idea of the project, we got to work: the back-end team used Django and GraphQL to create the backbone of our application, while the front-end team used HTML, CSS, JavaScript, React and Apollo to make the website functional. All of this was done under the mentorship of our team leader, who also took care of implementing dev-ops technologies such as Nginx, Gunicorn and other AWS tools to get the website up and running on the Internet.

II. Stages of implementing the application

We took several steps before beginning to write any code. First, we analysed the given task and started by formulating software requirements, designing a use case diagram and drafting the first version of our UML class diagram. Then, we discussed on which technologies would help us reach our goal, in the fastest and safest way possible, by designing an architecture diagram, quickly followed by the UML Communication Diagram and basic sequence diagram. The last step of our designing journey was the database diagram, which we ensured was in 3NF.

After the planning phase was complete, everybody got to work. The technologies we used gave us the flexibility and ease of communication between the front-end and back-end teams, while our thorough planning and application layering ensured tasks were kept simple, modular and clear. Also, black-box testing of each component was paramount in saving us time from any future bugs that might have occurred.

The final step was a system-wide testing phase, which is described in the testing chapter.

We had a few big things we wanted to make sure we provided:

- **security**, so authentication is necessary and we generate a key for each user.
- **concurrency** is also not a problem as our app handles that easily, while keeping the experience smooth for multiple users
- **straightforward ui experience**, our app works great on various mobile and desktop browsers, regardless of window resolution.
- **modularity**, by respecting the layers we designed, which makes the codebase easy to maintain

We feel like all the points were achieved and furthermore, our app could be easily expanded and become a real product, which proves that every step we took on this journey was handled with proper care and attention. Our team gelled well together and we had no problem with sub-teams communication.

III. Technology description and UI prototyping

a) Technologies

Developed over the course of 14 weeks, Grady's Booch was a complex project that taught us the importance of design patterns, careful planning and team work. In order to bring the project to its current state, our team had to work with a plethora of front and back-end technologies:

- **Front-End Technologies:** HTML, CSS, JavaScript, React, Apollo
- **Back-End Technologies:** Django, GraphQL
- **Dev-Ops Technologies:** Nginx, Gunicorn and various other AWS tools
- **Version control:** Github

In the following section there will be short description of some of the more impactful technologies used in our project.

1. Github

GitHub, Inc. is a United States-based global company that provides hosting for software development version control using Git. It offers the distributed version control and source code management (SCM) functionality of Git, plus its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

2. React

React (also known as React.js or ReactJS) is an open-source JavaScript library for building user interfaces. It is maintained by Facebook and a community of individual developers and companies.

React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing. Redux and React Router are respective examples of such libraries.

Core attributes of react:

- **Declarative**

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

- **Component-Based**

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

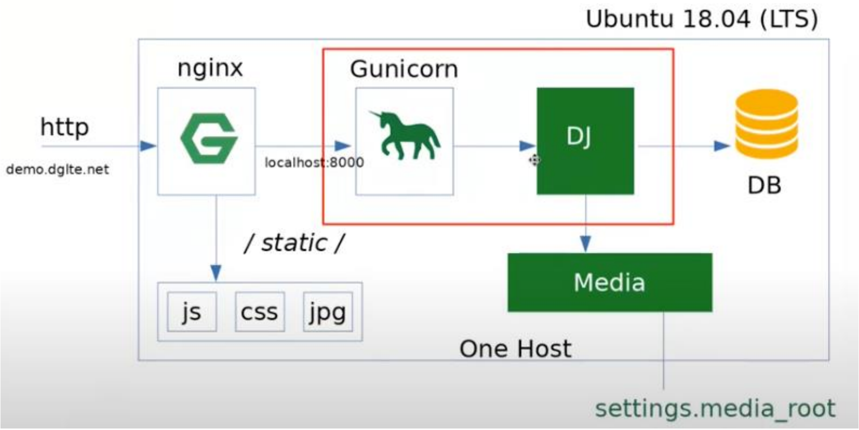
- **Learn Once, Write Anywhere**

We don’t make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.

3. Nginx & Gunicorn

When running a django application in production, runserver command is not used. Instead it is replaced by a combination of nginx and gunicorn. Niginx takes care of static and media files. if the request is not about static or media content, nginx forwards it to gunicorn via port 8000. Gunicorn takes care of running django application. All this process is happening on an EC2 instance from aws.



For ssl certification we used certbot and letsencrypt and nginx redirects the requests from http to https, more details can be discovered on their website (https://letsencrypt.org/).

For the DNS part we used Route53 from aws. This tool is used to route end users to Internet applications by translating names like gradysbooch.com into the numeric IP addresses, such as 192.0.2.1, that computers use to connect to each other.

4. Bootstrap

Bootstrap is a web framework that focuses on simplifying the development of web pages. The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents.

5. Python/Django

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Why python is used:

- **Readable and Maintainable Code**
- **Multiple Programming Paradigms**
- **Compatible with Major Platforms and Systems**
- **Robust Standard Library**
- **Many Open Source Frameworks and Tools**
- **Simplify Complex Software Development**
- **Adopt Test Driven Development**

Django is a Python-based free and open-source web framework that follows the model-template-view (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an American independent organization.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself.Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

Advantages of Django framework

- **Rich ecosystem** - Read Django like a system
- **Maturity** - Django has been around for 11 years and has gone through stages of significant improvement.
- **Admin panel by default** - Admin panels are designed to help you manage your application
- **Good for SEO** - With Django, you can generate readable website.
- **Pluggable** - Django is pluggable by nature and can be extended with plugins.
- **Libraries** - Some popular libraries include the Django REST framework, Django CMS and Django-allauth.
- **ORM** - Django is valued for its object-relational mapper that helps developers interact with databases

b) UI prototyping

There were some brainstorming sessions in which our zealous frontend team hand-crafted some simple sketches that soon were becoming the beautiful site that we have today.

The UI Prototyping was something we thought about a lot during the first stages of the project, although it was only a sketch back then. As is the case with any design work, this prototype we are talking about, which is the design of our application continued to evolve and change to accommodate the vision we ultimately had for this project.

We started with sketching the most basic features that an app should have, and we tried to imagine the needs of someone who would be using it. Of course, since we had no idea of who the users of our app are, and the requirements were vague regarding the design, this was a pretty hard task at hand. So, in order to deliver a quality product, we had to let our imagination and creativity kick in.

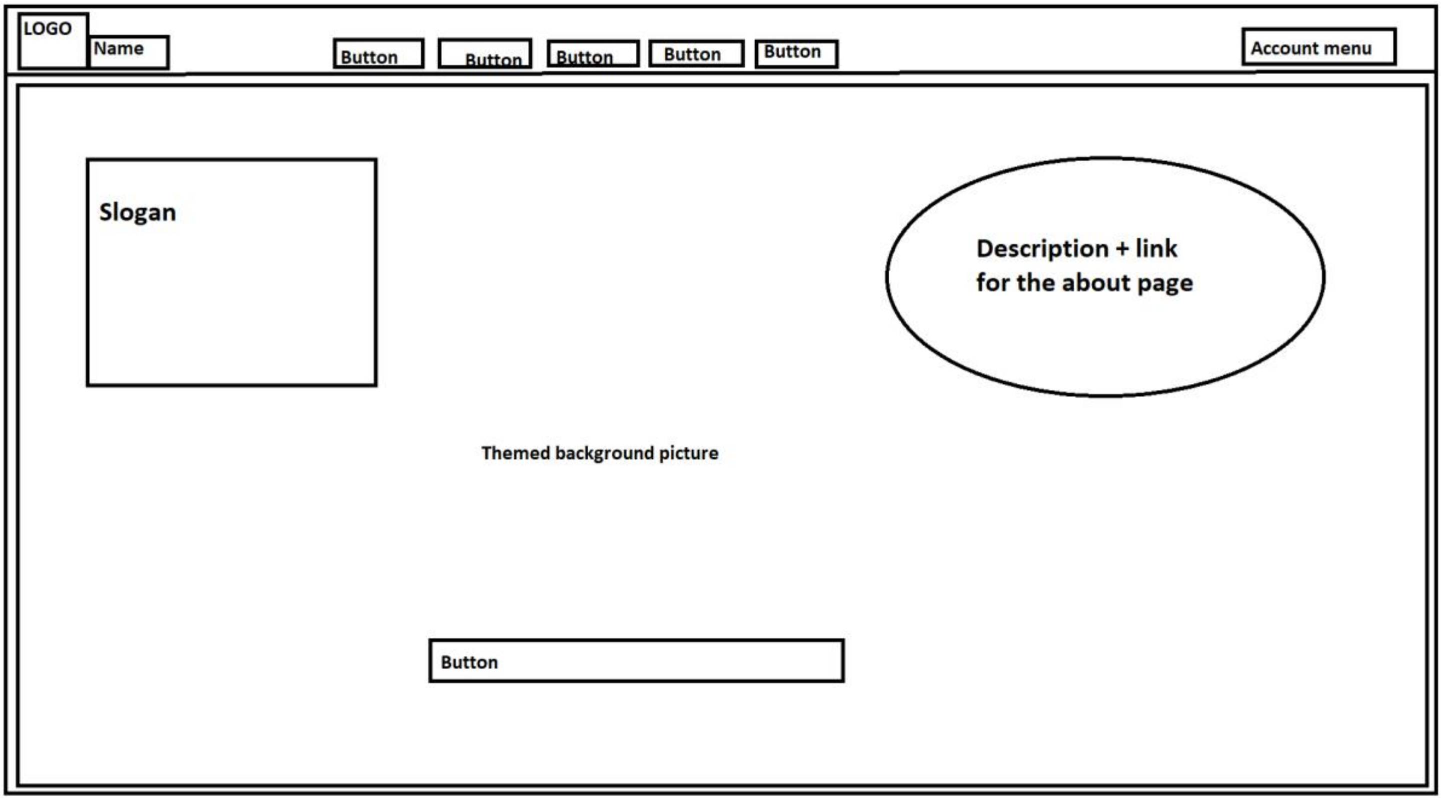
As the actual functionality of the app was beginning to grow, so did the number of pages we needed to design and think about. Of course, we tried to use the most well-known online tools for UI Prototyping, including but not limited to Figma, Moqups, Proto and others. Because the files we worked on were misplaced, we have nothing to show for this, but the final design of the app should be enough evidence supporting this.

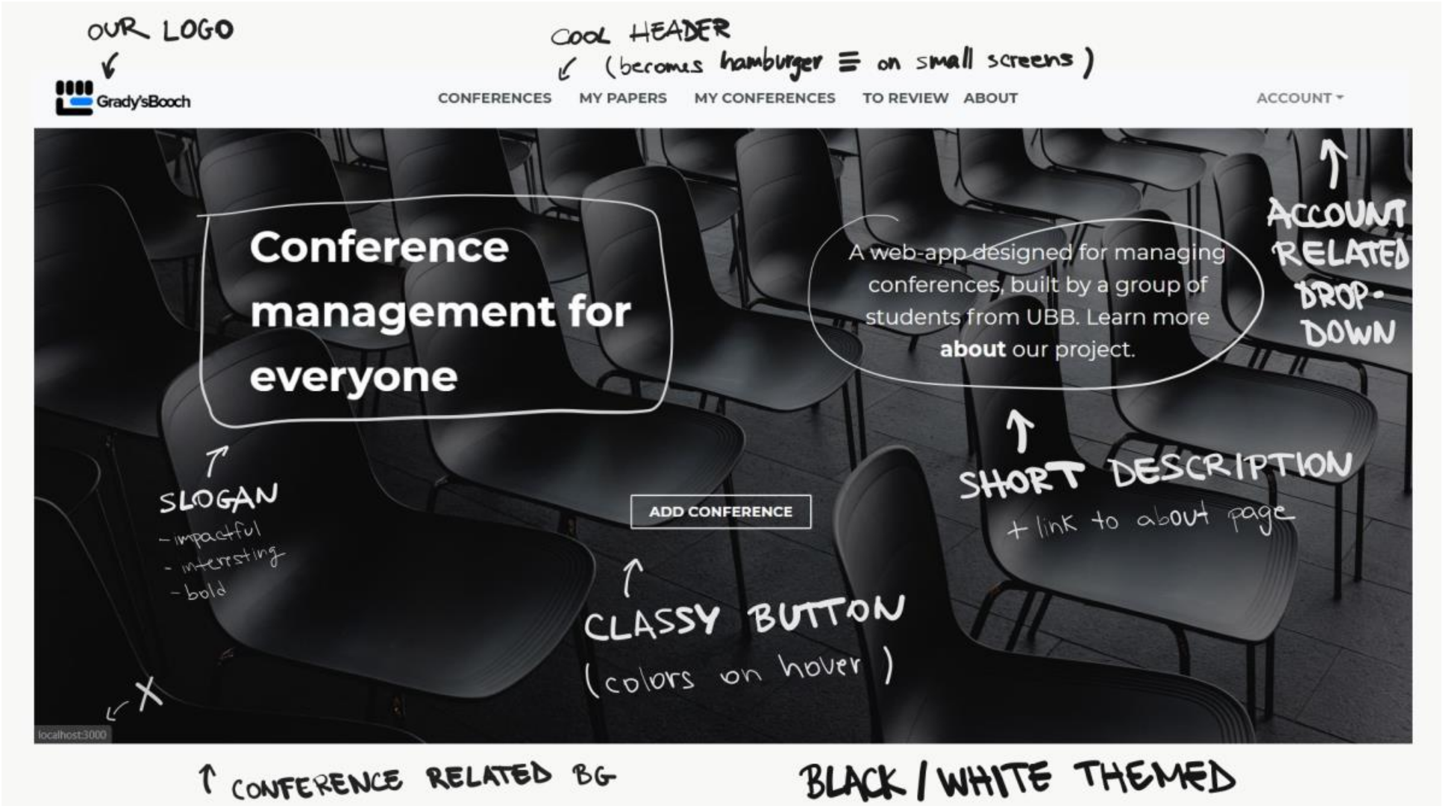
Since this whole process was and still is an iterative one, we take pride in what we have accomplished until now, considering the sketch we had at the beginning, and having in mind all the stages which the design went through to reach its final form (for now).

The work started changing a little bit when all of us decided that we want a responsive web-app that accommodates a whole range of devices, even mobile, regardless of the width of their display. This implied rethinking everything we have done until that point and cover almost all scenarios in which our app was going to be used.

Bearing all this in mind, and seeing how our app looks like today, we think that we have done a great job, even if we followed our own creative processes. And, of course, we also know that this design can always be improved as new ideas and needs appear.

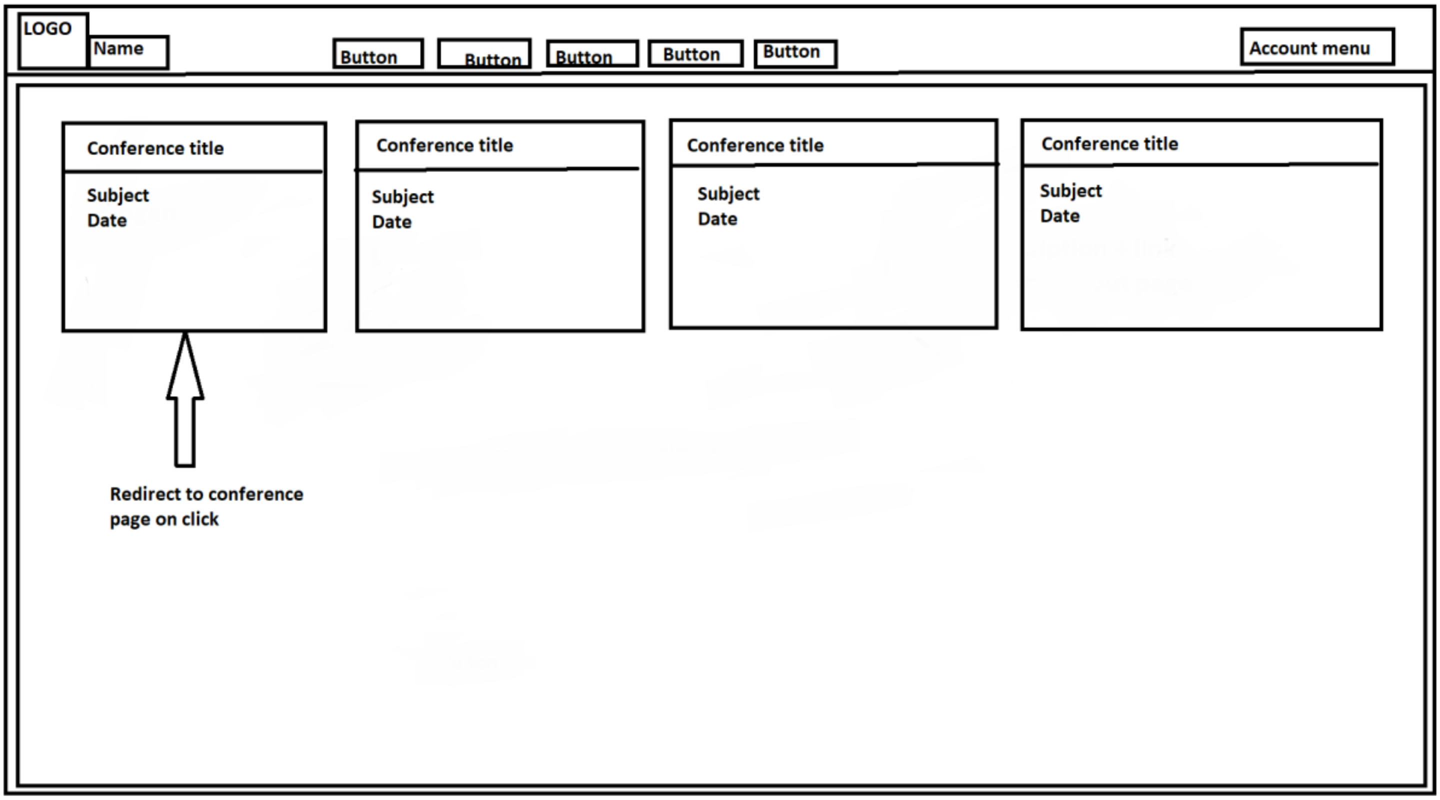
Some of the sketches and the end result are available below:

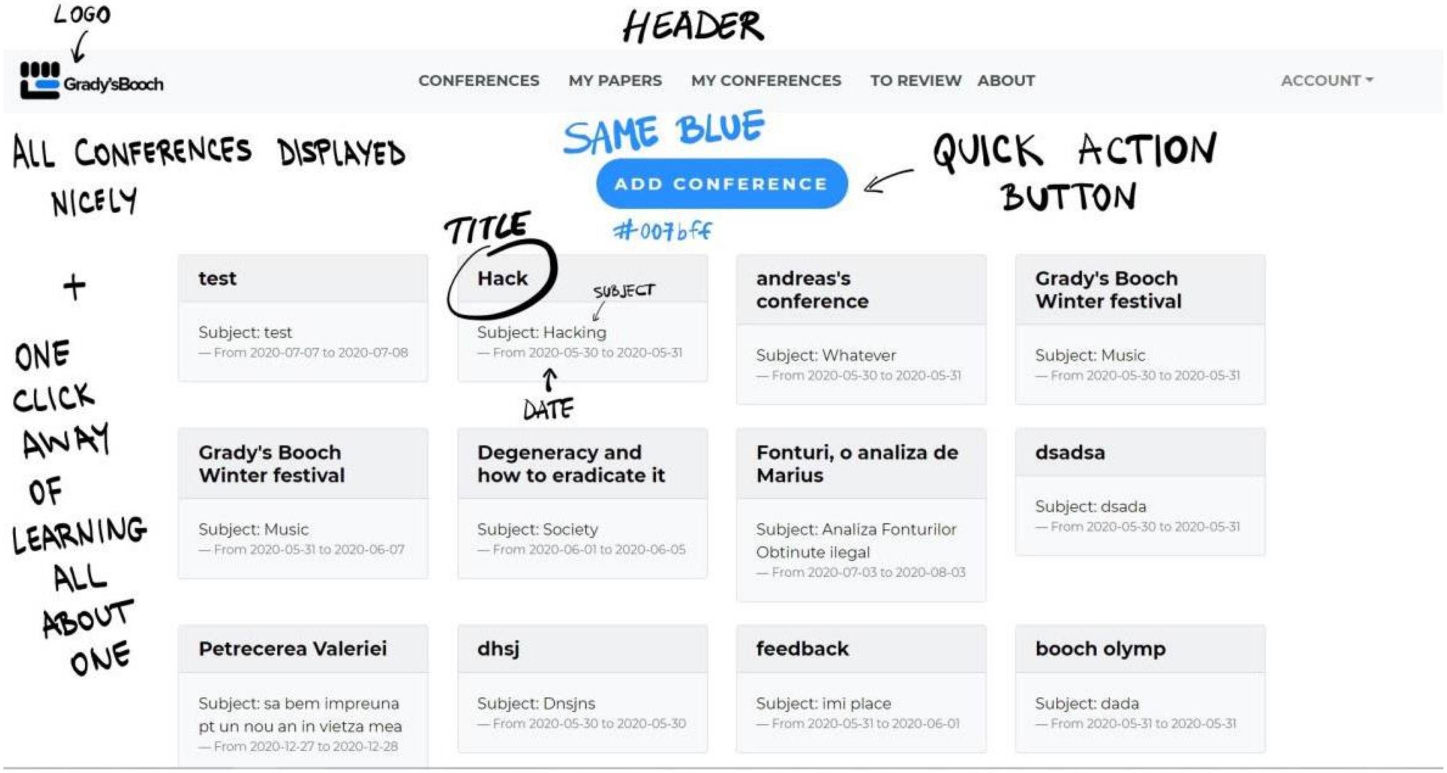

LOGO
Name
Button
Button
Button
Button
Button
Account menu
Slogan
Description + link for the about page
Themed background picture
Button



Conference management for everyone

A web-app designed for managing conferences, built by a group of students from UBB. Learn more **about** our project.





test
Subject: test
— From 2020-07-07 to 2020-07-08

Hack
Subject: Hacking
— From 2020-05-30 to 2020-05-31

andreas's conference
Subject: Whatever
— From 2020-05-30 to 2020-05-31

Grady's Booch Winter festival
Subject: Music
— From 2020-05-30 to 2020-05-31

Grady's Booch Winter festival
Subject: Music
— From 2020-05-31 to 2020-06-07

Degeneracy and how to eradicate it
Subject: Society
— From 2020-06-01 to 2020-06-05

Fonturi, o analiza de Marius
Subject: Analiza Fonturilor Obtinute ilegal
— From 2020-07-03 to 2020-08-03

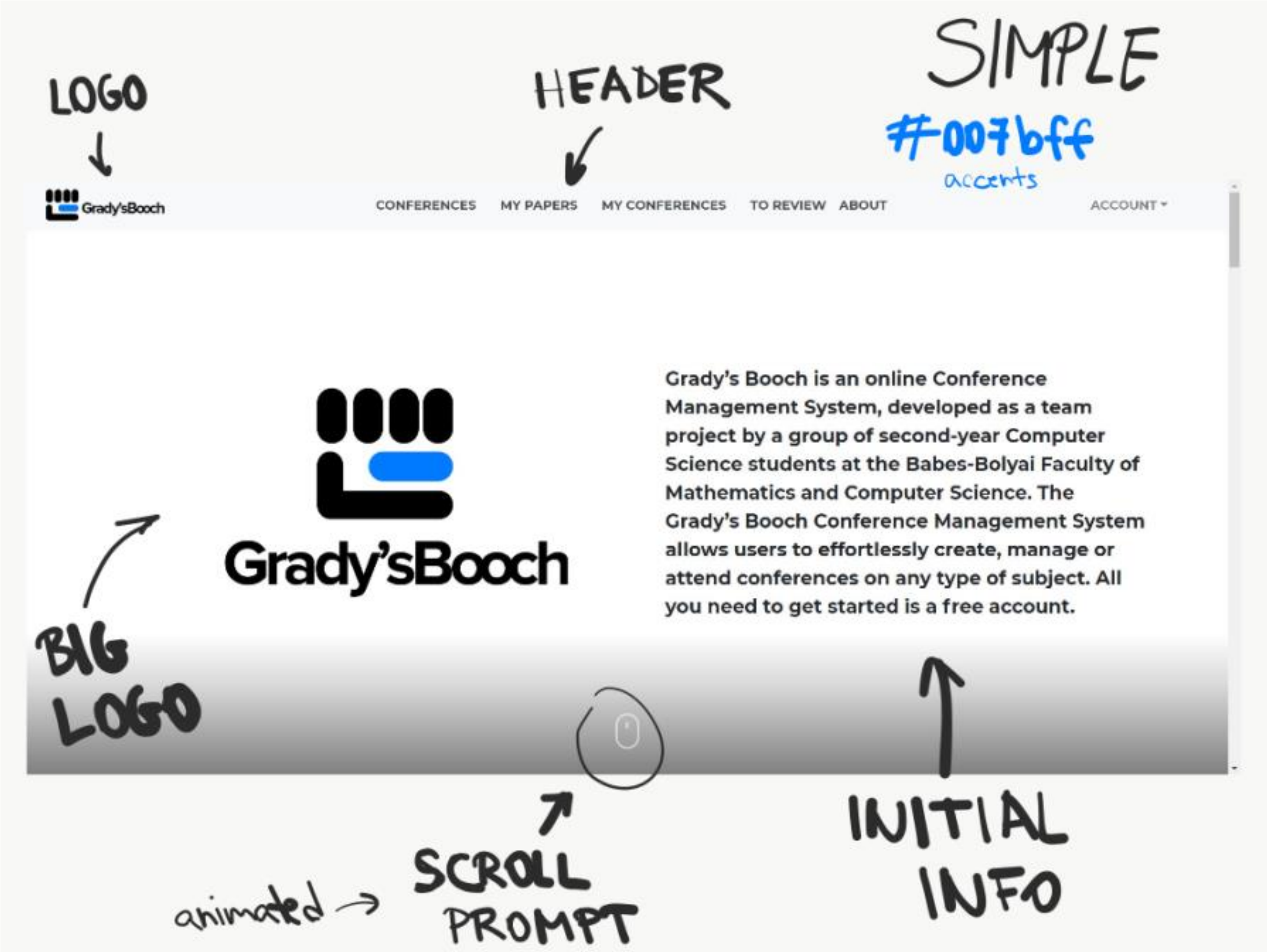
dsadsa
Subject: dsada
— From 2020-05-30 to 2020-05-31

Petrecerea Valeriei
Subject: sa bem impreuna pt un nou an in vietza mea
— From 2020-12-27 to 2020-12-28

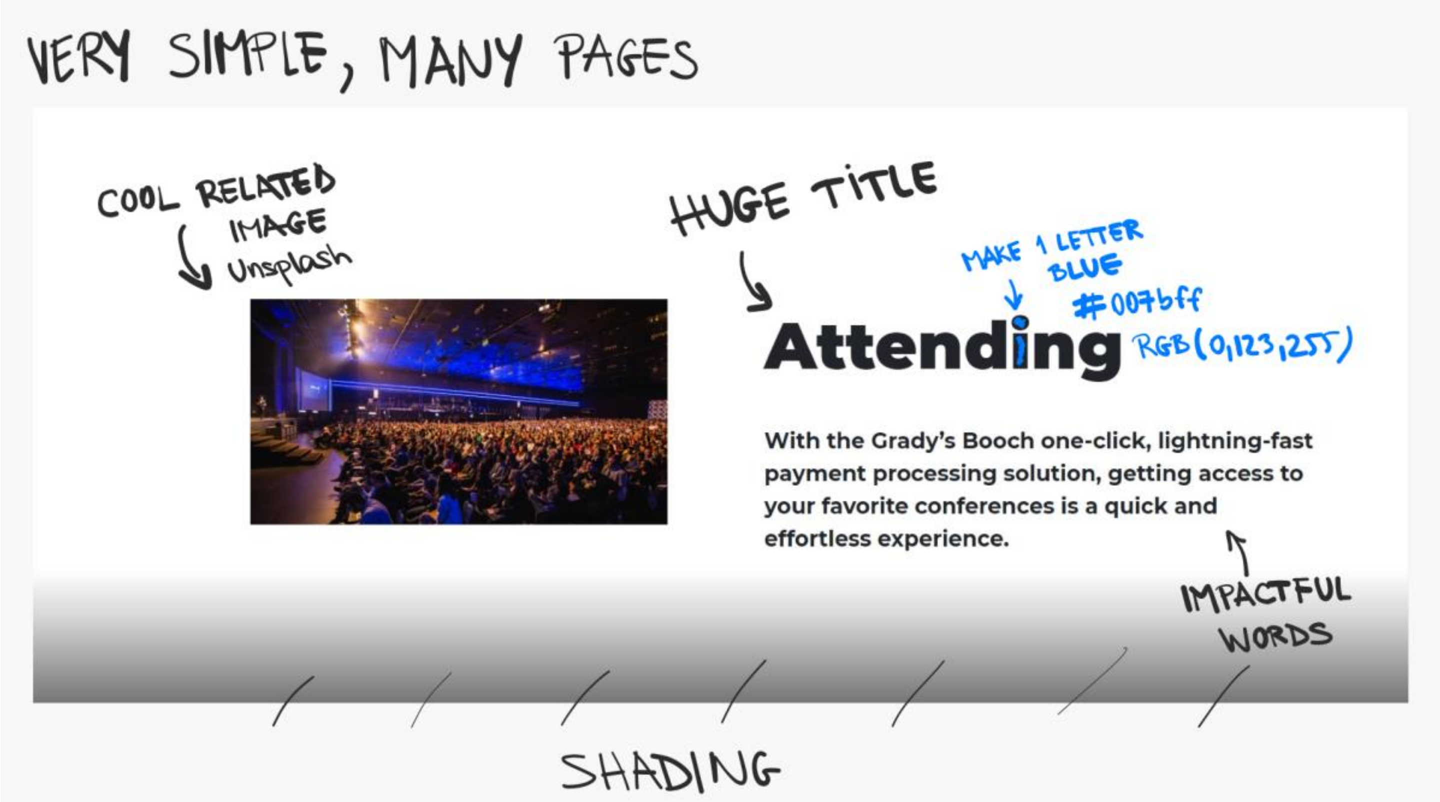
dhsj
Subject: Dnsjns
— From 2020-05-30 to 2020-05-30

feedback
Subject: imi place
— From 2020-05-31 to 2020-06-01

booch olymp
Subject: dada
— From 2020-05-31 to 2020-05-31



Grady's Booch is an online Conference Management System, developed as a team project by a group of second-year Computer Science students at the Babes-Bolyai Faculty of Mathematics and Computer Science. The Grady's Booch Conference Management System allows users to effortlessly create, manage or attend conferences on any type of subject. All you need to get started is a free account.



Attending

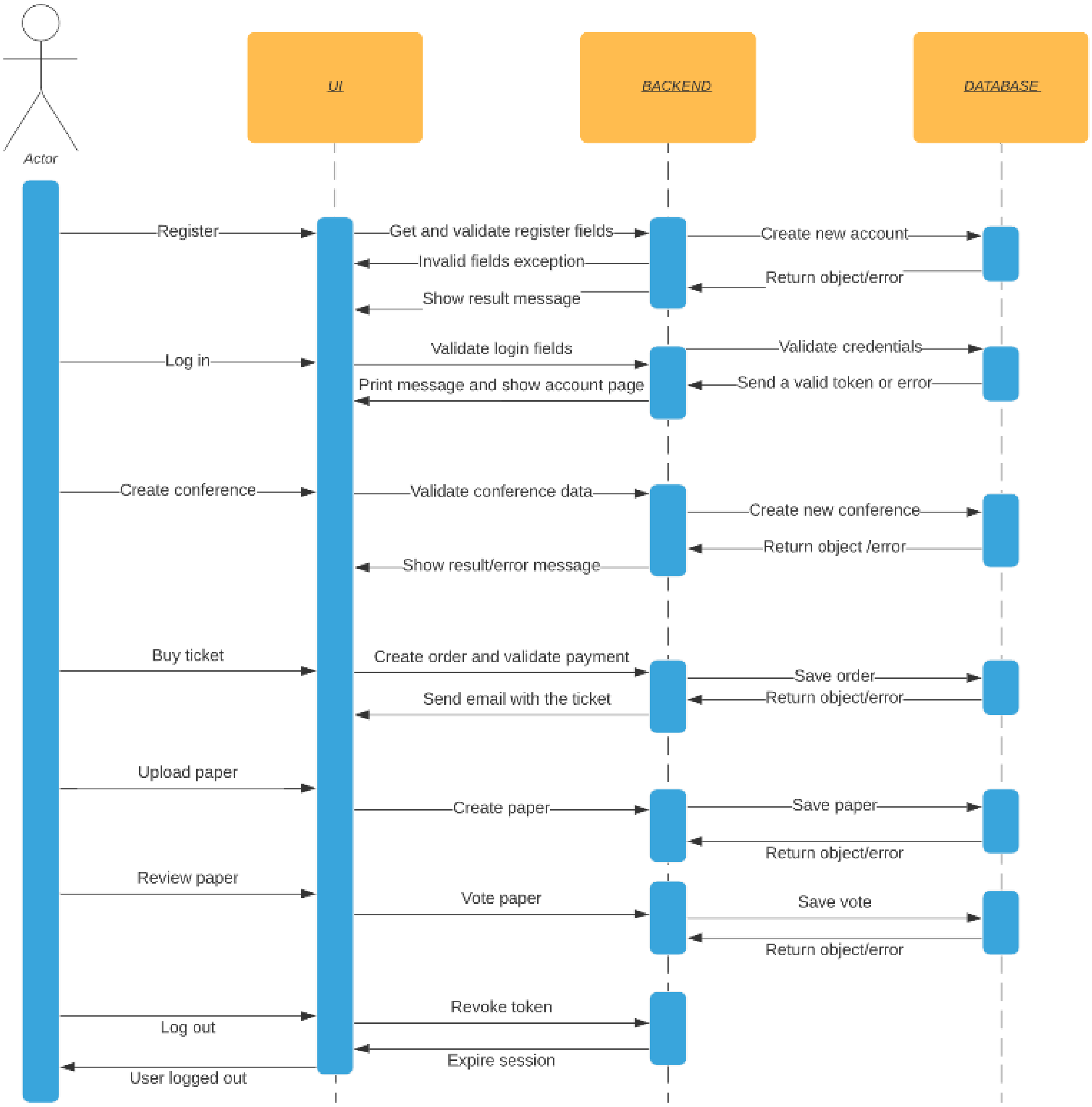
With the Grady's Booch one-click, lightning-fast payment processing solution, getting access to your favorite conferences is a quick and effortless experience.

IV. Diagrams

For our project planning several diagrams were designed, as follows:

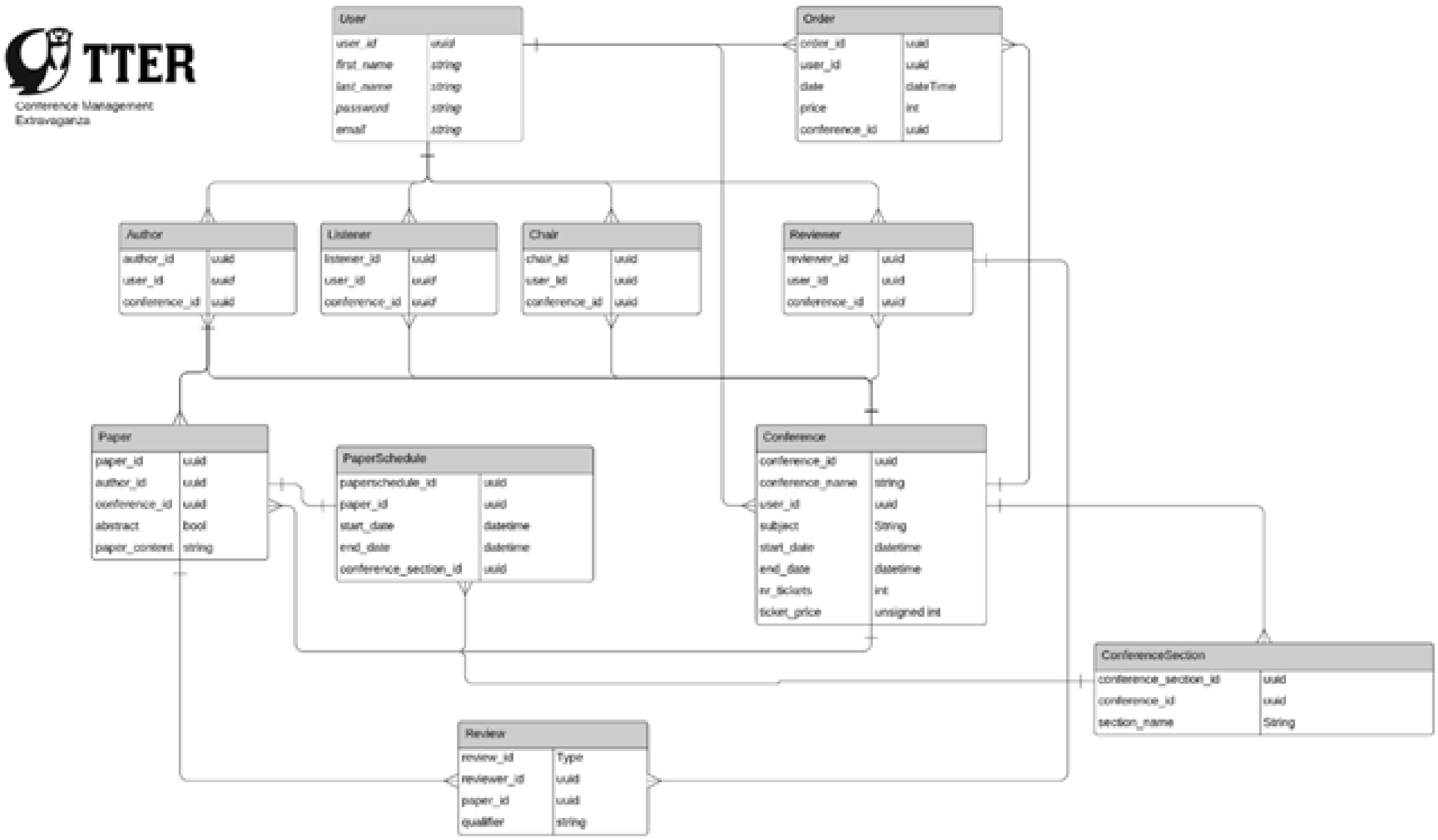
1. Basic sequence diagram

Sequence diagram is a special form of interaction diagram. Interaction diagrams are used to formalize the dynamic behavior of the system and to visualize the communication among objects. They are useful for identifying additional objects that participate in the use cases. The left-most column represents the timeline of the actor who initiates the use case. The other columns represent the timeline of the objects that participate in this use case. Labeled arrows are stimuli that an actor or an object sends to other objects.



2. Database diagram

An entity relationship diagram shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. By defining the entities, their attributes, and showing the relationships between them, an ER diagram illustrates the logical structure of databases. This type of diagrams are used to sketch out the design of a database.



User	
user_id	uuid
first_name	string
last_name	string
password	string
email	string

Order	
order_id	uuid
user_id	uuid
date	dateTime
price	int
conference_id	uuid

Author	
author_id	uuid
user_id	uuid
conference_id	uuid

Listener	
listener_id	uuid
user_id	uuid
conference_id	uuid

Chair	
chair_id	uuid
user_id	uuid
conference_id	uuid

Reviewer	
reviewer_id	uuid
user_id	uuid
conference_id	uuid

Paper	
paper_id	uuid
author_id	uuid
conference_id	uuid
abstract	bool
paper_content	string

PaperSchedule	
paperschedule_id	uuid
paper_id	uuid
start_date	datetime
end_date	datetime
conference_section_id	uuid

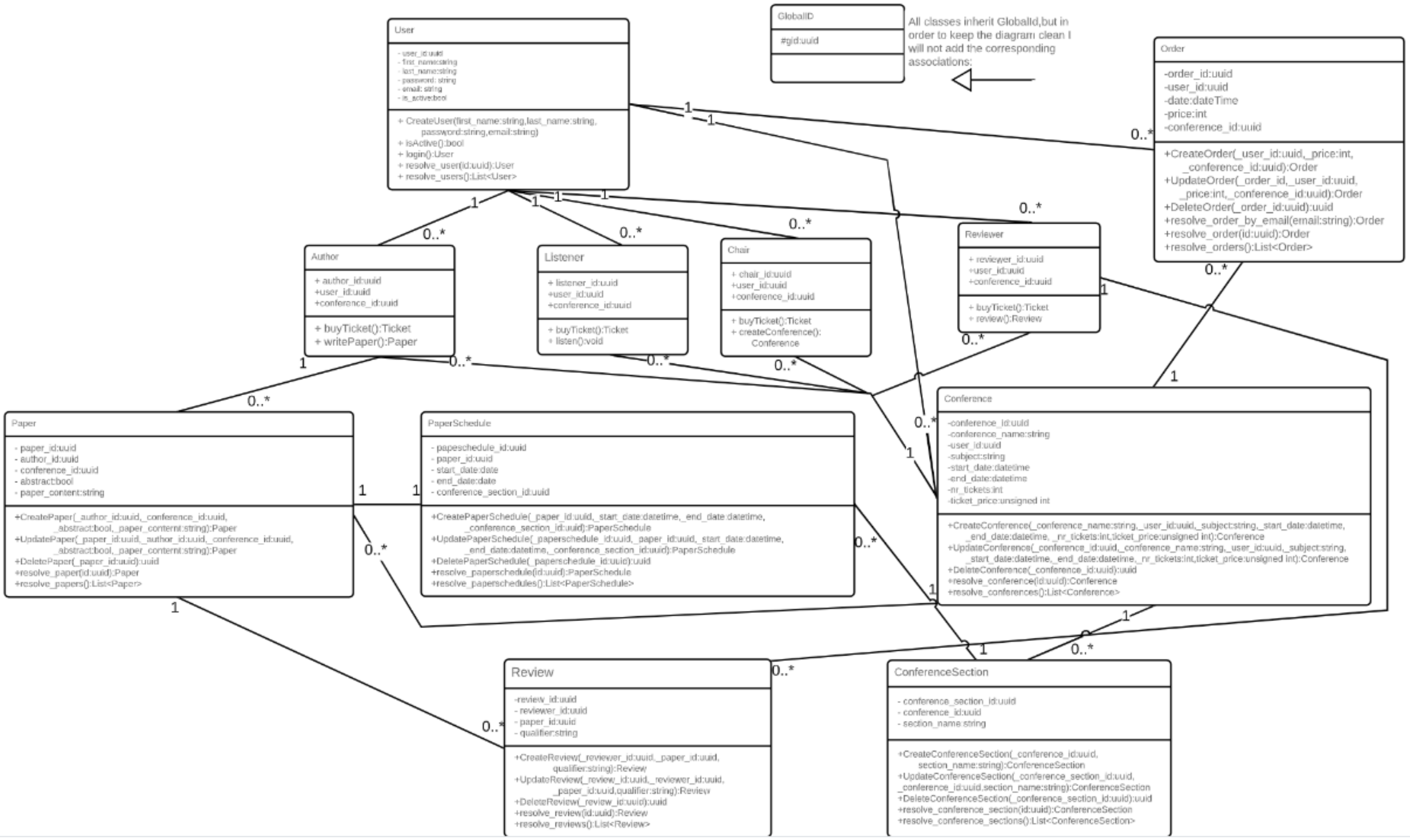
Conference	
conference_id	uuid
conference_name	string
user_id	uuid
subject	String
start_date	datetime
end_date	datetime
nr_tickets	int
ticket_price	unsigned int

ConferenceSection	
conference_section_id	uuid
conference_id	uuid
section_name	String

Review	
review_id	Type
reviewer_id	uuid
paper_id	uuid
qualifier	string

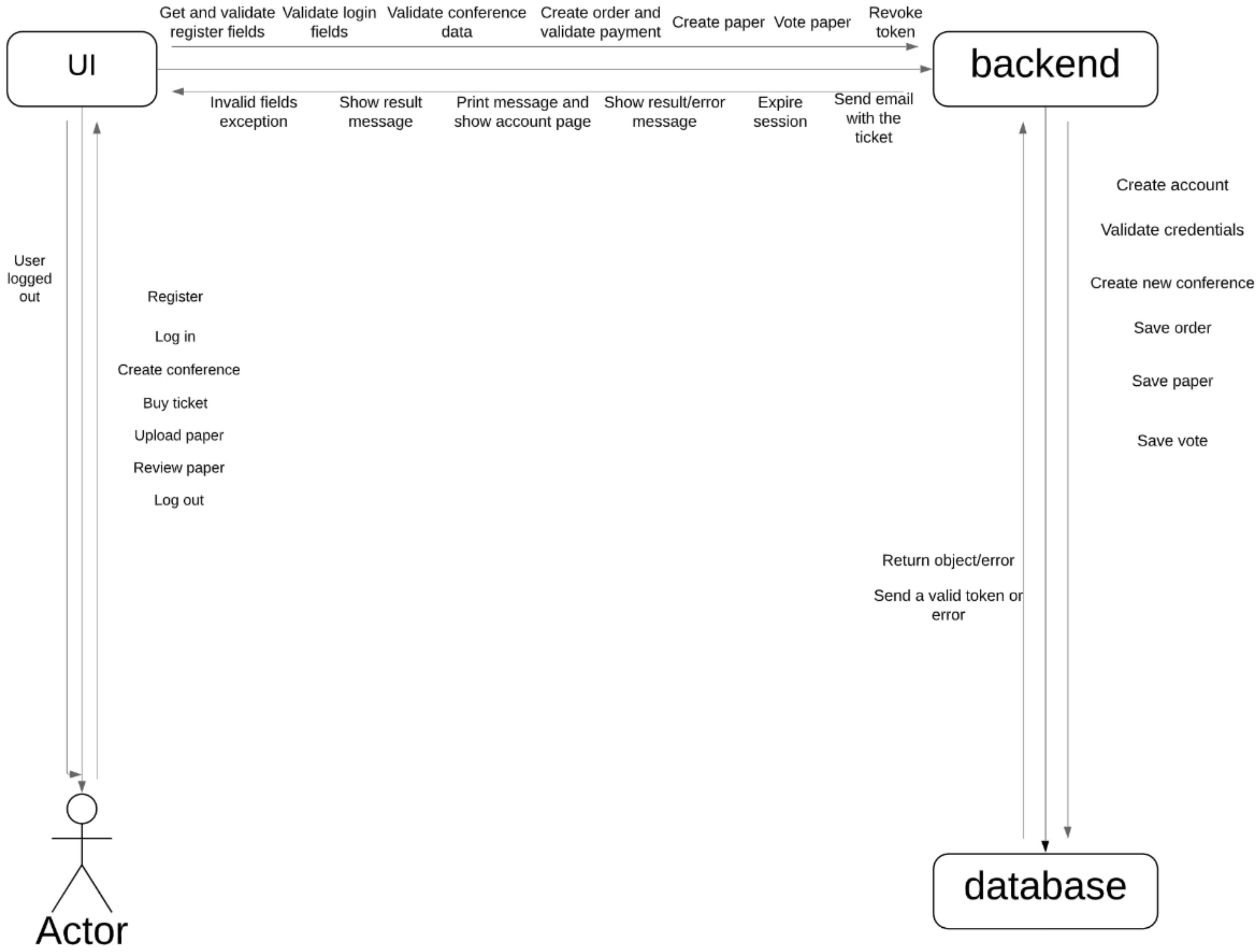
3. UML Class diagram

Class diagrams are used to describe the structure of the system. Classes are abstractions that specify the common structure and behavior of a set of objects. Objects are instances of classes that are created, modified, and destroyed during the execution of the system. An object has state that includes the values of its attributes and its links with other objects. Class diagrams describe the system in terms of objects, classes, attributes, operations, and their associations.


User
- user_id:uuid
- first_name:string
- last_name:string
- password: string
- email: string
- is_active:bool
+ CreateUser(first_name:string,last_name:string, password:string,email:string)
+ isActive():bool
+ login():User
+ resolve_user(id:uuid):User
+ resolve_users():List<User>
GlobalID
#gid:uuid
All classes inherit GlobalId,but in order to keep the diagram clean I will not add the corresponding associations:
Order
-order_id:uuid
-user_id:uuid
-date:dateTime
-price:int
-conference_id:uuid
+CreateOrder(_user_id:uuid,_price:int, _conference_id:uuid):Order
+UpdateOrder(_order_id,_user_id:uuid, _price:int,_conference_id:uuid):Order
+DeleteOrder(_order_id:uuid):uuid
+resolve_order_by_email(email:string):Order
+resolve_order(id:uuid):Order
+resolve_orders():List<Order>
1
1
0..*
1
1
1
1
0..*
0..*
0..*
0..*
Author
+ author_id:uuid
+user_id:uuid
+conference_id:uuid
+ buyTicket():Ticket
+ writePaper():Paper
Listener
+ listener_id:uuid
+user_id:uuid
+conference_id:uuid
+ buyTicket():Ticket
+ listen():void
Chair
+ chair_id:uuid
+user_id:uuid
+conference_id:uuid
+ buyTicket():Ticket
+ createConference(): Conference
Reviewer
+ reviewer_id:uuid
+user_id:uuid
+conference_id:uuid
+ buyTicket():Ticket
+ review():Review
0..*
1
1
0..*
0..*
0..*
0..*
0..*
1
1
Paper
- paper_id:uuid
- author_id:uuid
- conference_id:uuid
- abstract:bool
- paper_content:string
+CreatePaper(_author_id:uuid,_conference_id:uuid, _abstract:bool,_paper_content:string):Paper
+UpdatePaper(_paper_id:uuid,_author_id:uuid,_conference_id:uuid, _abstract:bool,_paper_content:string):Paper
+DeletePaper(_paper_id:uuid):uuid
+resolve_paper(id:uuid):Paper
+resolve_papers():List<Paper>
1
1
PaperSchedule
- papeschedule_id:uuid
- paper_id:uuid
- start_date:date
- end_date:date
- conference_section_id:uuid
+CreatePaperSchedule(_paper_id:uuid,_start_date:datetime,_end_date:datetime, _conference_section_id:uuid):PaperSchedule
+UpdatePaperSchedule(_paperschedule_id:uuid,_paper_id:uuid,_start_date:datetime, _end_date:datetime,_conference_section_id:uuid):PaperSchedule
+DeletePaperSchedule(_paperschedule_id:uuid):uuid
+resolve_paperschedule(id:uuid):PaperSchedule
+resolve_paperschedules():List<PaperSchedule>
0..*
0..*
1
Conference
-conference_id:uuid
-conference_name:string
-user_id:uuid
-subject:string
-start_date:datetime
-end_date:datetime
-nr_tickets:int
-ticket_price:unsigned int
+CreateConference(_conference_name:string,_user_id:uuid,_subject:string,_start_date:datetime, _end_date:datetime, _nr_tickets:int,ticket_price:unsigned int):Conference
+UpdateConference(_conference_id:uuid,_conference_name:string,_user_id:uuid,_subject:string, _start_date:datetime,_end_date:datetime,_nr_tickets:int,ticket_price:unsigned int):Conference
+DeleteConference(_conference_id:uuid):uuid
+resolve_conference(id:uuid):Conference
+resolve_conferences():List<Conference>
1
1
1
0..*
0..*
1
0..*
Review
-review_id:uuid
- reviewer_id:uuid
- paper_id:uuid
- qualifier:string
+CreateReview(_reviewer_id:uuid,_paper_id:uuid, qualifier:string):Review
+UpdateReview(_review_id:uuid,_reviewer_id:uuid, _paper_id:uuid,qualifier:string):Review
+DeleteReview(_review_id:uuid):uuid
+resolve_review(id:uuid):Review
+resolve_reviews():List<Review>
ConferenceSection
- conference_section_id:uuid
- conference_id:uuid
- section_name:string
+CreateConferenceSection(_conference_id:uuid, section_name:string):ConferenceSection
+UpdateConferenceSection(_conference_section_id:uuid, _conference_id:uuid,section_name:string):ConferenceSection
+DeleteConferenceSection(_conference_section_id:uuid):uuid
+resolve_conference_section(id:uuid):ConferenceSection
+resolve_conference_sections():List<ConferenceSection>

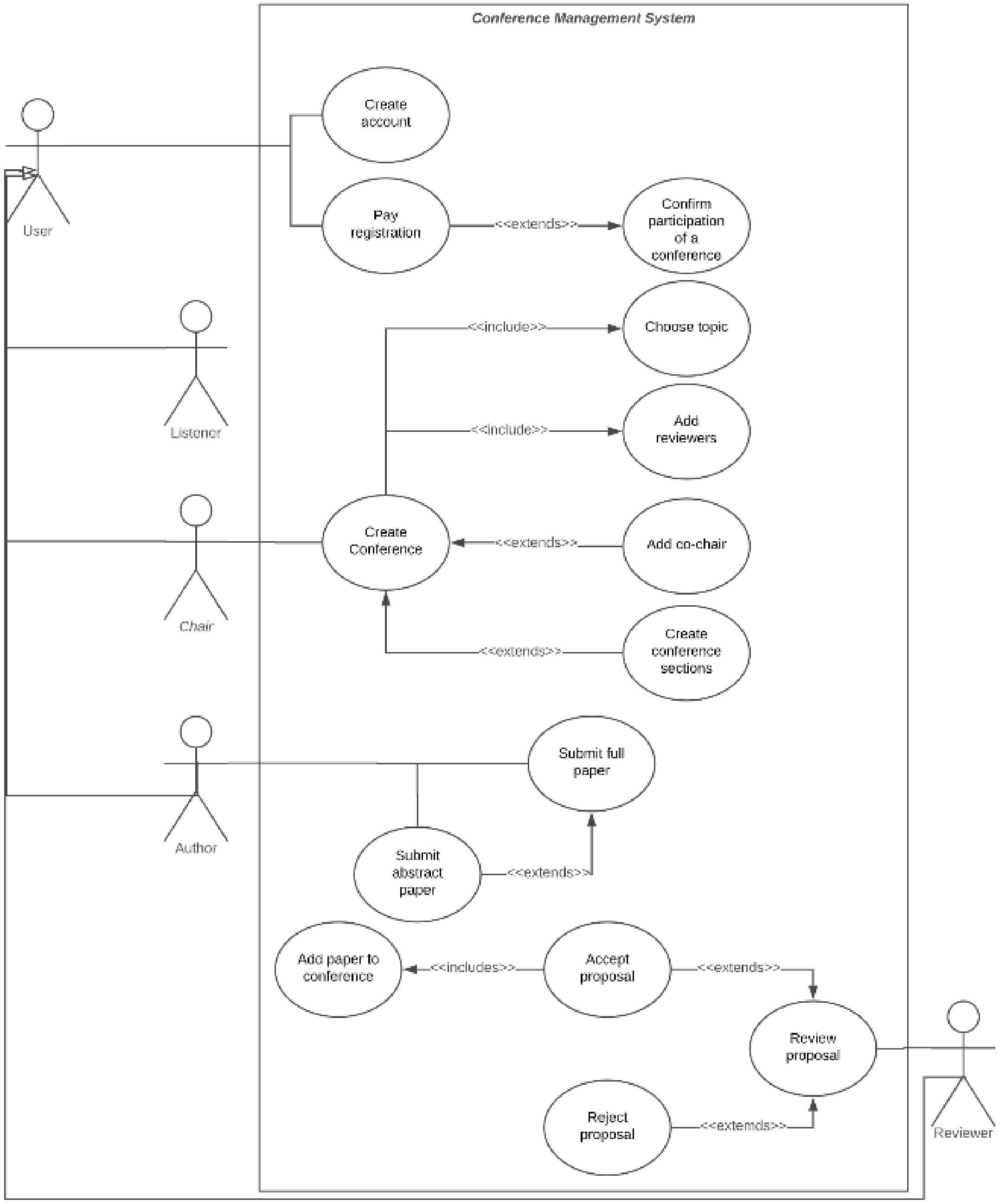
4. UML Communication diagram

Communication diagrams depict the same information as sequence diagrams, and they are a particular form of interaction diagram, too. Communication diagrams represent the sequence of messages by numbering the interactions. On one hand, this removes the need for geometrical constraints on the objects and results in a more compact diagram. On the other hand, the sequence of messages becomes more difficult to follow.



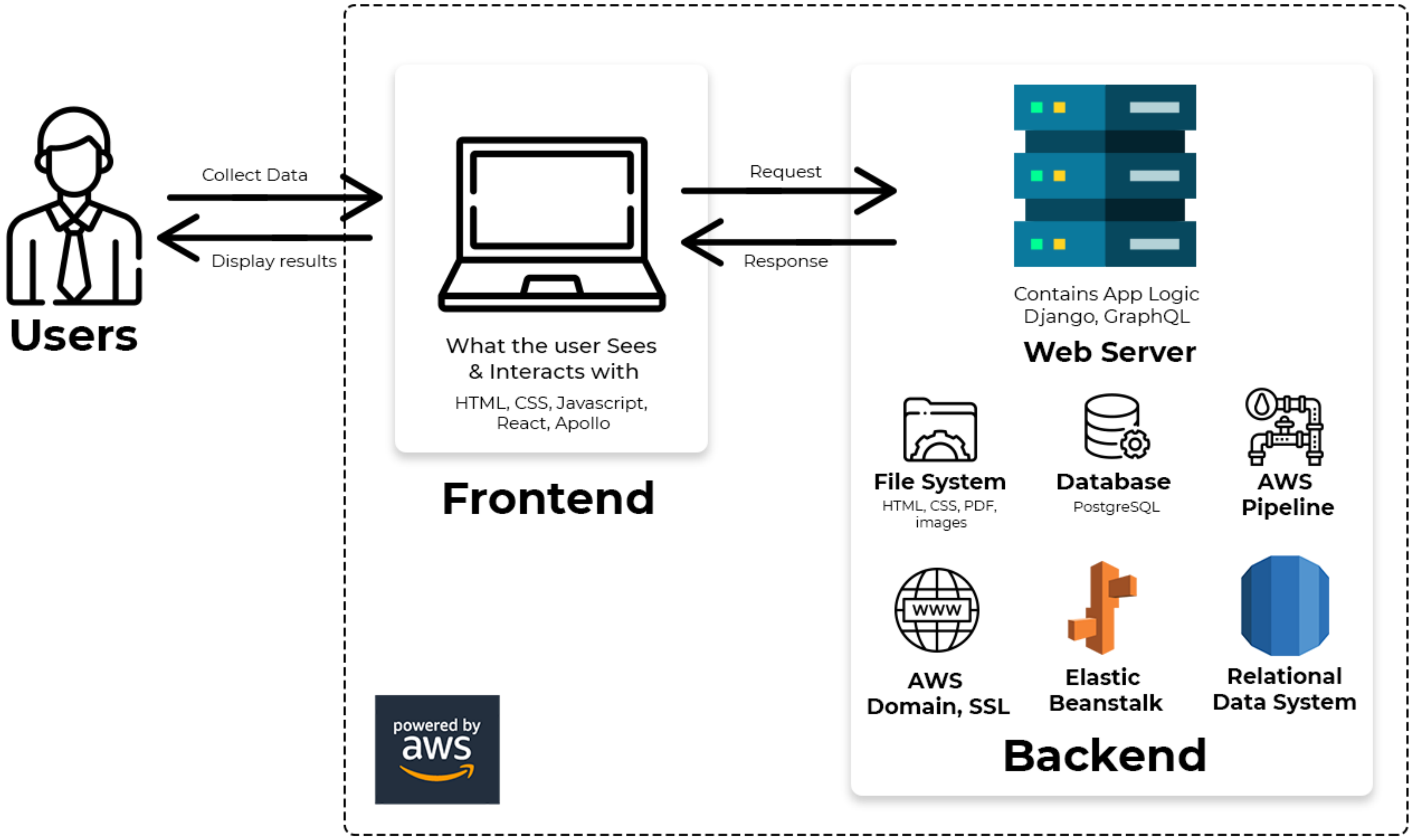
5. Use case diagram

Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from an external point of view. A use case describes a function provided by the system that yields a visible result for an actor. An actor describes any entity that interacts with the system. The identification of actors and use cases results in the definition of the boundary of the system, that is, in differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system.



6. Arhitecture diagram

A software architecture includes system decomposition, global control flow, handling of boundary conditions, and interconnected communication protocols.



V. Testing

Our team thoroughly tested every single aspect of the application whilst it was being developed. Every new function of our application was black-box tested for edge cases to ensure no bugs could get through and every error is caught before any changes could compromise our database.

Some basic data validation (e.g. Conference start date should be in the future) is done on the front-end level, then our data goes to the back-end, which communicates with our database and makes sure the data it received is valid. We believe we have achieved a strong and robust application in this way.

Furthermore, once all functionalities were implemented, we began the final testing phase in which we invited our fellow colleagues to join us in trying to break our app, both on desktop and mobile versions of the website. Fortunately, all our attempts were in vain and every error was handled, as expected, by our system.

In conclusion, we took testing seriously and it can evidently be seen when using our application, be it on the phone or on a PC.

VI. Help

Getting started

- **Register:**
  - Input a first name, last name, email and password on the register page
  - You will receive an email confirming you were registered in the system
- **Login (once you are registered)**
  - On the login page, input the email and password you used for registering
  - Choose a permission level from the list
  - Click the LOGIN button

Manage conferences

- **Add a conference – method 1**
  - On the main page, click the ADD CONFERENCE BUTTON
  - This will take you to the page where you can input all the information about the conference you want to add
- **Add a conference – method 2**
  - From the Conferences (or My Conferences) page, click the blue ADD CONFERENCE button at the top
  - Then you can add the information the same as in method 1
- **Submit a paper**
  - From the Conferences page, click on a conference to be taken to its page
  - Click the SUBMIT PAPER button
  - From here, input a title for the paper, write it down and check the checkbox if it is abstract
- **Buy a ticket for a conference**
  - From the Conferences page, click on a conference to be taken to its page
  - Click the BUY TICKET button, located near the price of tickets for the conference
  - Confirm or cancel the transaction
- **Manage sections for your conferences – add section**
  - To add a new section, click one of your conferences from the My Conferences page
  - Click the blue NEW SECTION button located under the conferences name and subject
  - Input a name for the section and press the CREATE SECTION button
- **Manage sections for your conferences – accept papers**
  - To accept a paper, scroll down on the conference view you clicked from the My Conferences page

- Click the green ACCEPT PAPER button corresponding to the paper you want
- Select the interval for the paper and assign it to a section
- Click SUBMIT

- **Manage sections for your conferences – add reviewer to a paper**
    - To assign a reviewer to a page, scroll down on the conference view you clicked from the My Conferences page
    - Click the blue ADD REVIEW button corresponding to the paper you want to assign a reviewer to
    - Choose an email from the list to assign the paper to that reviewer
    - Click SUBMIT

- **Review papers you have been assigned**
    - Go to the TO REVIEW page
    - Click the blue SUBMIT GRADE button located right-most on the row corresponding to the paper you want to review
    - Choose a qualifier from the list
    - Click UPGRADE

- **Other functionalities**
    - To see all your submitted papers, click on MY PAPERS right at the top of the page (on the header)
    - From the ACCOUNT dropdown in the right corner of the page you can:
        - View your profile
        - See all your orders (tickets bought)
        - See your conferences, papers and papers to review
    - Find more about the app and project on the ABOUT page, accessible from the header at the top of the page