**RentAll**

**Team:** *Romaneasca*

**Members:**

*Branea Alexandru*

*Candet Andrei-Gabriel*

*Ceaca-Mocan Mircea-Radu (Team Leader)*

*Ceausoglu George-Eduard*

*Corlaci Andreea-Maria*

*Pop Daniel-Avram*

*Pop Vlad*

*Popa Alex-Ovidiu*

*Radu Ana-Maria*

**Coordinator:** *Borodi Flavia - Emiliana*

*Contents*

[*Presentation of the project*  3](#_Toc516435761)

[*Stages of implementing the application* 4](#_Toc516435762)

[*Task management* 5](#_Toc516435762)

[*Diagrams* 6](#_Toc516435762)

[*Class Diagram* 7](#_Toc516435762)

[*Use case diagram* 8](#_Toc516435762)

[*Database diagram* 9](#_Toc516435762)

[*Used technologies* 10](#_Toc516435763)

[*Application Testing* 12](#_Toc516435772)

[*Help manual*](#_Toc516435773) 13

**Presentation of the project**:

Did you ever want to repair something around the house or built something, but you didn’t have the necessary tools for it? And you also don’t want to spend too much money to buy those tools? Well, we just solved your problem.

Our website aims at helping people share their goods not by selling them, but by making them available for a specific amount of time, for free or for a price. With this platform, we want to make sure that anyone in Cluj that needs a certain item just for a limited amount of time can obtain it, be it in agriculture, real estate or education.

Our application provides a user-friendly interface, not difficult to learn and very intuitive the first time you see it.

**Stages of implementing the application**

1. **Concept**

After thinking about the project idea, we started wondering what functionalities the application should have and how to implement them.

1. **Modeling**

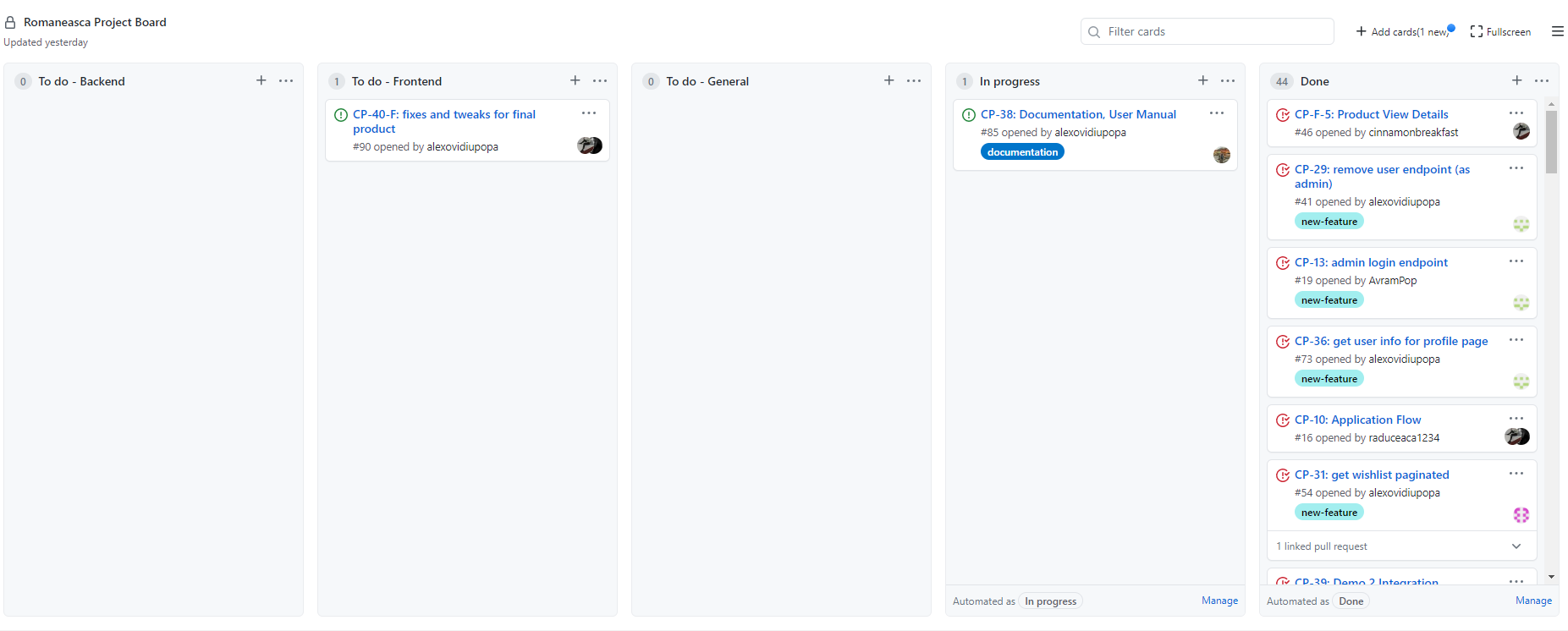
Then we extracted the concepts we will use for the app. The models for the app were: Admin, User, Announcement, Loan, ClosedLoan, Disscusion, Image and Wishlist. Then, we created the diagrams that should help us with the relations among concepts, to see the interaction user-app and to see how the app should work for different kinds of actors. The diagrams were Use-Case Diagram an Class-Diagram. Then we decided to split into two teams one team will work on the backend and the other on the frontend. The backend team created the database with all the tables holding these models as well as the relations among them, while the frontend worked on the ui-ux design. We chose the frameworks we will use for frontend and backend for our app,then created the repositories and controllers with CRUD operations.

1. **Development**

The next step was to implement the frontend of the app and to bind it with backend. Every member of the frontend team was responsible for some pages of the app and to make them work correctly and load correct data from backend, and to add functionalities in the backend, besides CRUD operations, that should help us, for example, to upload/download files, or to check if a loan is closed or not. Then we established the flow of the app, from signup to all the other operations, regarding the actor that uses it. Once all the pages in the frontend were implemented, the next step of implementation started.

1. **Testing**

The last part of implementation was testing. Firstly, we tested all the operations in all repositories and controllers, to make sure all the operations work correctly. When the tests were done, we started to check how the app is working, i.e. all the stages, from creating an user to loaning the objects, work correctly and the flow is right.

**Task management**

For managing the application development we used GitHub’s project board and separated each feature in different tasks following Agile development methodology.

We started by describing how the end product will be used and what problem it will solve. Once we clarified our expectations, the work began. We cycled through a process of planning, executing and evaluating – which changed the final product to fit our needs better. Continuous collaboration among our team members was the key to successfully accomplish what we set out to do.

**Diagrams**

* Class diagram
* Use case diagram
* Database diagram

**Class diagram**

**Text

Description automatically generated**

**Use case diagram**

**Diagram

Description automatically generated**

**Database diagram**

**A picture containing graphical user interface

Description automatically generated**

**Used technologies**

Here is a short presentation of the technologies we used. We have treated each of them separately considering where we used it and most importantly, why we chose to use that particular tool.

**BackEnd:**

1. *The Java Programming Language*:
   1. Where we used it – we have used java in our backend for many purposes using libraries in order to be able to perform all the required things starting from the REST APIs and ending in the ORM API required for the user.
   2. Motivation – Java has been chosen as the language that drive the backend for its object-oriented capabilities and its libraries, but especially because of its popularity (it is powering 3 billion devices!) which led to an extended amount of available resources and documentation.
2. *The Hibernate ORM*:
   1. Where we used it – being an ORM we used it to link out database to our main backend driven code. It had the capabilities of creating our database enabling us to automatize the processes of creating and handling the databases using only Java code
   2. Motivation – Hibernate has been chosen as the ORM for this application for its capabilities regarding automation being one of the most capable ORMs built on Java (also supported by official docs). It is also efficient enough for our purposes as it is built on the JPA (Java Persistence API), the “proper” way of handling persistent storage in Java
3. *The Spring framework*:
   1. Where we used it – this can be considered the “giant” of our backend as it ties together the architecture of the application. It allowed us to easily create an Model View Controller architecture for our application as it automatically injected our dependencies and tied together (For example there was no need for us to do any instantiation regarding the repositories, the entities that resided in them or the controller in charge of that specific repository. It was all done easily though the Spring framework)
   2. Motivation – Spring is a great framework that provides lots of functionalities. All the functionalities where used spring speak for themselves.
4. *Microsoft SQL Server dialect*
   1. Where we used it - Used for the database manipulated by the backend
   2. Motivation – The tools used for the sql server are very accessible and the dialect can’t be considered complicated.

**FrontEnd:**

1. *JavaScript, SCSS*:
   1. Where we used it – since this is a web application, these have been used everywhere under the hood of the front-end framework.
   2. Motivation –you can’t create web applications without these.
2. *ReactJS:*
   1. Where we used it – just as the Spring framework is the main drive of the backend, ReactJS is the main drive of the frontend. It has facilitated the creation of the user interface part of the application using its JSX dialect and specific libraries.
   2. Motivation – ReactJS is one of the most popular JS libraries with a wide array of uses, and we required an environment that would provide uniformity for our application user interface. It has been chosen because compared to JS frameworks, it is much more lightweight and faster to implement and use.
3. *Redux*
4. Where we used it – mostly for the application’s state management
5. Motivation - Redux is a predictable state container, designed to help you write JavaScript apps that behave consistently across client, server, and native environments and are easy to test
6. *NPM:*
   1. Where we used it – this was our package manager for creating the project to use react, adding and adding libraries without having conflicts between versions.
   2. Motivation – Not having your dependencies managed by a package manager is a recipe for disaster when multiple persons work on the same project and especially if they use different IDEs, Java versions or operating systems.

**Application testing**

In the testing phase of the project, for every API endpoint we implemented unit tests for the controllers using JUnit and MockMvc.

JUnit is a unit testing framework for the Java programming language that plays a big role in regression testing. Complementing unit-testing, regression testing makes certain that the latest fix, improvement or patch did not break existing functionality, by testing the changes we have made to our code.

MockMvc provides a powerful way to mock Spring MVC for testing MVC web applications. Through MockMvc, we send mock HTTP requests to the controller and test how the controller behaves without running the controller within a server.

As for the CI/CD part, we created a pipeline with the only job of building the application and implicitly running all the tests every time we push a new feature. For the CD, we have a Docker file that helps us save the latest version of the API on docker hub.

**Help manual**

**Graphical user interface

Description automatically generatedFirst look at our website**

**How to make an account?**

**Graphical user interface, application, Teams

Description automatically generated**To sign up for our website, you will need to press the Sign up button from the right corner of the page and fill in the required information.

Then you can sign in to your account, by pressing the Sign in button

Graphical user interface, application

Description automatically generated

**How to search for objects?**

Graphical user interface, application

Description automatically generatedYou can use the filtering and ordering fields on the left side of the page to search for anything you want.

**How to view more details about a certain product?**

You just have to click on the product’s image and the details page will open.

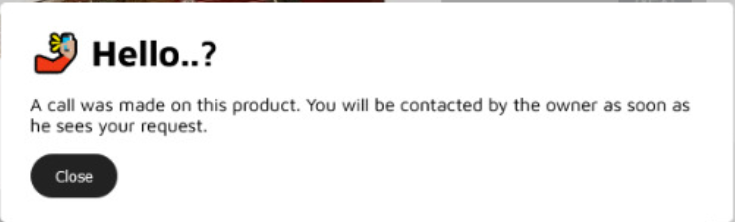
**Graphical user interface, application, Teams

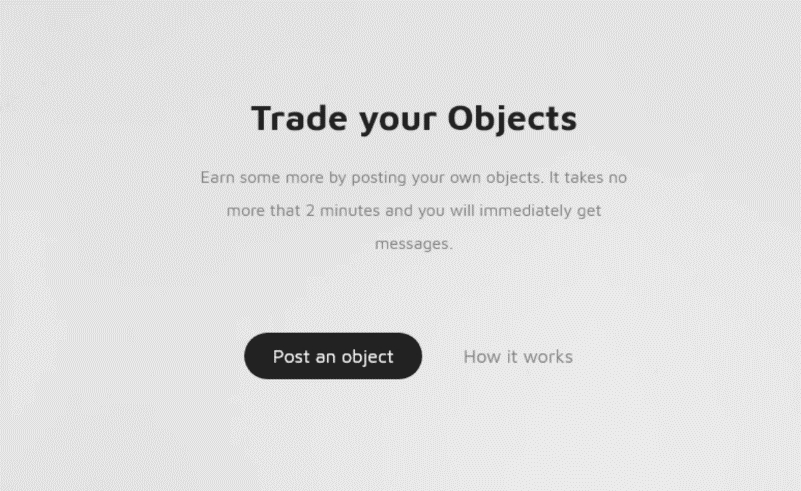
Description automatically generated**

**How to get a product?**

**Graphical user interface, application

Description automatically generated**On the detail page you will find the contact information for the owner of the product. You fill in your information and a request will be sent to the owner.

****

**How to post an announcement?**

On the main page you click on the Post an object button that will redirect you to the add page, where you cand post the object you want to loan.

Graphical user interface, application, Teams

Description automatically generated

**How do you know someone is interested in your product?**

Graphical user interface, application

Description automatically generatedOnce a request was made for your product, you will be able to see who did it and contact the person back if you want to accept.