Ikhsan Habibi
Julia Rakowiecka
Manh Cuong Tran

Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Lab 8: Hoagies on Rails

Alright, you now understand what customers will want to do when they purchase Hoagies at your store! Let's see how far you get implementing your shop using Ruby and Rails! You have studied the structure of your shop for many weeks and have produced many diagrams. They won't translate 1:1 to Ruby on Rails, but they will help you understand what your application needs to do.

Note that there are four weeks for this exercise, but only points for three weeks. The exercise should be done in three weeks, but since it is exam time, you get an extra week to finish up.
Here are some tutorials that might be useful to have read before you start:

- http://guides.rubyonrails.org/active_record_basics.html
- http://guides.rubyonrails.org/active_record_callbacks.html
- http://guides.rubyonrails.org/association_basics.html
- http://guides.rubyonrails.org/active_record_querying.html
- http://docs.seattlerb.org/rake/

1. Your first step is to get your database migration defined and populated with test data. Set up the scaffolding for all of your classes and generate those databases! Have a look around at all the cool stuff Rails created for you and document what you find in your report.

2. You should never be attached to manually created test data living in your development or test database. Instead, you should have scripts that generate and regenerate this test data on demand.
Prof. Kleinen has an example script he prepared for rake: FactoryGirl and the rake task.
You can use the task with “rake db:populate” which automatically creates all models defined in spec/factories to populate your development database. The database will be reset before the data is populated. Remember, you should not be attached to any data in your development database.

3. Now, can you get a web page set up that accesses this database? Think back to your installation party! Make sure you can enter and delete data before you go on. Can you use the generated code to upload a picture for your comments? You may have to install a gem for this if you didn't do it during the installation party.

4. Right, your interface could use a face-lift. Get down into the CSS and see how much more beautiful you can make it! Don't spend too much time on this, as there are other tasks.

5. Your first implementation was the admin side of your database, for entering in your data. Now make a customer application. It is able to configure a hoagie, and have it delivered. We don't need shopping carts and whatnot in this first iteration, they just purchase one if they click on a button "order".

6. Refactor your code as necessary, and see if you can manage to get the short reviews set up and working. Can you also add the selfies here? Wait a minute, if they can upload pictures, the users can upload any kind of photo. Better set up an application for the admin to review and delete fotos.

7. Let's see those statistics! You did remember to set up a table that is not linked to any screens yet that will persist your data? Make a new screen to show this data. Can you now get your configuration

page to update this? Remember, people can choose to put onions on and then take them off again before they order, so only store the data when the order is triggered.

8. Bored? Keep testing, refactoring and implementing! Did you remember to include an impressum? What is the maximum number of ingredients you can add to a hoagie? What happens if you want to order three hoagies, all the same? Do you have to re-implement everything? Can you save a configuration and give it a name? The sky's the limit! Just see how far you can go!
Your report, including all source code, is due at 8.00 on Feb. 4, 2019 and should include all materials (including copies of the scenarios used), properly marked with the authors of these scenarios. Don't forget to include your own names on your report, and post the materials in the Moodle area for each team member. Please include a reflection on how easy or how difficult this exercise was.

LAB PLAN

At the beginning, we read the previous exercises and we tried to think how we should have a good implementation based on all the feedbacks that we got. We analyzed our protocols and we decided which parts or features that we want to implement on Rails. We received a great list of tutorials, each of us chose which the good one for beginner to read is. We studied them a lot before we start with implementation. We want to make sure that we understand the idea of MVC (Model View Controller) pattern.

LAB PROTOCOL INDEX

LEARN FROM THE MISTAKES

Constructive comments from Professor Weber Wulff

- **Scenarios are very specific, people have names and they either sign in OR register. From this you derive the use cases, which are more abstract. You named all of your use cases scenarios, and have a mix of specific scenarios and abstract use cases.**
  We know that it is really complicated to generalize use cases. It is also quite different between what we planned and what we can implement. At least we looked up to our use cases and scenarios that we had at the beginning, so that we still have a similar concept to be executed.

- **"invalid email" -> how will you be determining this?**
  Email without @ character or contains only number. We can use the constraints by using the regex (Regular expression).

- **"illegal character" -> what is this? Pretty much any Unicode character should be acceptable.**
  For instance, customer input phone number with alphabet, it is illegal character. We should prevent this input.

- **"Delivery" -> don't you need to check first if you can produce and deliver by this time?**
  We ended up implementing delivery or picked up, but we do not show it when the customer order delivery. They can choose, but at the end they can see, if it is delivery or not.

- **Why is the administrator participating in "Register an account"?**
  We thought about this scenario, it is only the customer that register an account. The administrator are responsible only for the website and the functionalities of the website work.

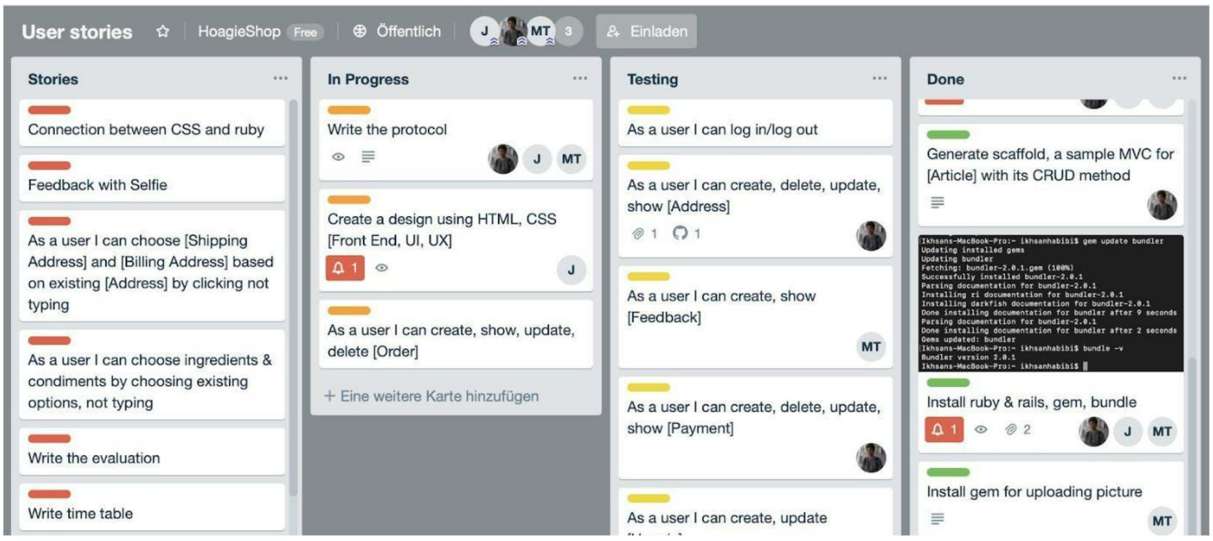
- **If you have the shipping and billing address in Order, the customer will have to type it in again every time they order....**
  We decided to put address only in the order. It means if customer choose delivery, then he/she should give the address where the order should be delivered.

- **Your ingredients are only strings? I would have thought that some sort of type would be helpful here.**
  We planned to create an enum for ingredients. Customer can choose only one of the ingredients. The ingredients that we have are 'None', 'Ham', 'Bacon', 'Chicken', 'Salami', 'Quinoa'.

- **You only have a URL to the photo, that is, you expect your users to host the images on some site under their control and then just post the URL? Saves you some work, but generally you need to store this on your servers.**
  We thought at the beginning, if we save only image URL, it would be faster to load. But then, we ended up to save the image, all of image format can be stored and uploaded.

SPRINT BOARD

The first step we did was creating a sprint board - for this, we used the Trello platform. It is a free platform to manage our progress, because if we just implement all the way without looking the organization of the hoagie production, it might be difficult and chaos in order to see, how far we have done the implementation. We do not create sprint board for every week, but we just created it for one big sprint board instead. We wrote all the user stories together, each member of our team has an access and each of us could add some cards and make some changes, so that we know who are working with which task. We give the color in order to prioritize our task. It is like mini simulation of web development.
Here is the link of our lovely trello board : https://trello.com/b/xcyqrRRS/user-stories



Stories
- Connection between CSS and ruby
- Feedback with Selfie
- As a user I can choose [Shipping Address] and [Billing Address] based on existing [Address] by clicking not typing
- As a user I can choose ingredients & condiments by choosing existing options, not typing
- Write the evaluation
- Write time table

In Progress
- Write the protocol
- Create a design using HTML, CSS [Front End, UI, UX]
- As a user I can create, show, update, delete [Order]
- + Eine weitere Karte hinzufügen

Testing
- As a user I can log in/log out
- As a user I can create, delete, update, show [Address]
- As a user I can create, show [Feedback]
- As a user I can create, delete, update, show [Payment]
- As a user I can create, update

Done
- Generate scaffold, a sample MVC for [Article] with its CRUD method

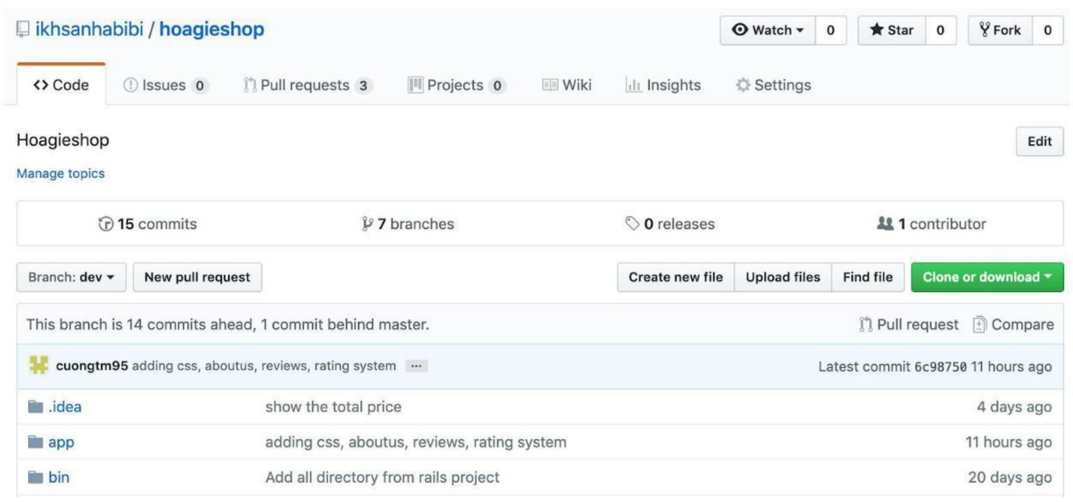
```
Ikhsans-MacBook-Pro:~ ikhsanhabibi$ gem update bundler
Updating installed gems
Updating bundler
Fetching: bundler-2.0.1.gem (100%)
Successfully installed bundler-2.0.1
Parsing documentation for bundler-2.0.1
Installing ri documentation for bundler-2.0.1
Installing darkfish documentation for bundler-2.0.1
Done installing documentation for bundler after 9 seconds
Parsing documentation for bundler-2.0.1
Done installing documentation for bundler after 2 seconds
Gems updated: bundler
Ikhsans-MacBook-Pro:~ ikhsanhabibi$ bundle -v
Bundler version 2.0.1
Ikhsans-MacBook-Pro:~ ikhsanhabibi$
```

- Install ruby & rails, gem, bundle
- Install gem for uploading picture

GITHUB PAGE

We always posted our work on GitHub, so that everyone can pull the current project and push the new part or feature. We don't want implement on local only. If something is wrong or the current project is accidentally deleted, we do not cry and hope that everything will come back like before. Just make sure everything is fine. In dev branch we have the final version of hoagie shop web. **ALL OF OUR CODES FOR THE EXERCISE CAN BE ACCESSED ON GITHUB:**

https://github.com/ikhsanhabibi/hoagieshop/tree/dev



Hoagieshop

15 commits · 7 branches · 0 releases · 1 contributor

This branch is 14 commits ahead, 1 commit behind master.

cuongtm95 adding css, aboutus, reviews, rating system … Latest commit 6c98750 11 hours ago

.idea	show the total price	4 days ago
app	adding css, aboutus, reviews, rating system	11 hours ago
bin	Add all directory from rails project	20 days ago

PROBLEM SOLUTION TABLE

We listed in the table all kinds of problems. Because we faced this problems every time, that is why we want to minimize the stress, we can just look at the table, what we actually need. The left side is the problems, on the right side we described the solutions. Zack zack, done.

Problem	Solution
“undefined method `title' for nil:NilClass” Rails Guides Tutorial Ask Question	Remember where you put private keyword. all the methods below that, will become private
NoMethodError in ArticlesController#edit	`def edit @article = Article.find params[:id] end`

how to remove MVC	`rails destroy controller (Name)`
	`rails destroy model (Name)`
	`rails destroy scaffold (Name)`

form syntax	`<%= form_for :person do |f| %>`
	`First name: <%= f.text field :first name %><br />`
	`  Last name : <%= f.text field :last name %><br />`
	`<% end %>`

add link in view	`<%= link to "Name for link", mission path %>`
how to make a drop down list / enum	`<%= form.select :desired_attribute, ['MoneyPal', 'Credit Card', 'Money Transfer']%>`
how to drop database	`rake db:drop`
email validation	`validates :email, format: { with: URI::MailTo::EMAIL REGEXP }`
'required' validation in Ruby on Rails forms	`<%=t.text area :content, :rows => 3, :required => true%>`
Validate only numbers but possible with [+-]	`validates :number, format: { with: /\A[+-]?\d+\z/, message: "Integer only. No sign allowed." }`
validating min and max	`validates :password, length: {minimum: 6}`
back to the root path	`<%= link to "Back", root path %>`

create checkbox list	`https://www.sitepoint.com/save-multiple-checkbox-values-database-rails/`
	
	`<%= label_tag 'tomato', 'Tomato' %>`
	`<%= check_box_tag 'order[vegetable][]', 'Tomato', id:'tomato' %>`

Ikhsan Habibi
Julia Rakowiecka
Manh Cuong Tran

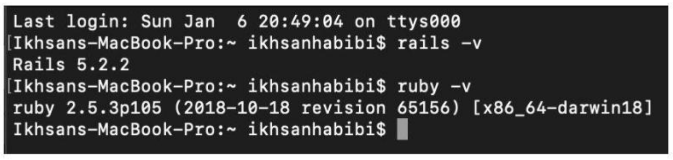
Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

DID WE ANSWER THE QUESTIONS? YES. YES. YES.

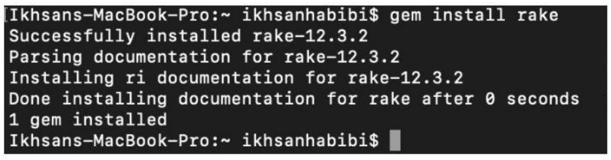
1. Your first step is to get your database migration defined and populated with test data. Set up the scaffolding for all of your classes and generate those databases! Have a look around at all the cool stuff Rails created for you and document what you find in your report.

Installing ruby, rails, sqlite3, bundler was done pretty confusing, especially for beginner like us. After installing the tools we need, we are ready to go to the next step. We spent really much time to understand, how the data migration works. But the first important thing to do is just generate scaffolding, then we can migrate our database. The good thing to know for beginner is that we don't have to create the controller and view manually, if we use generate scaffold. It is like magic and automatic. Here is an example of command, how we can generate scaffold and migrate db.

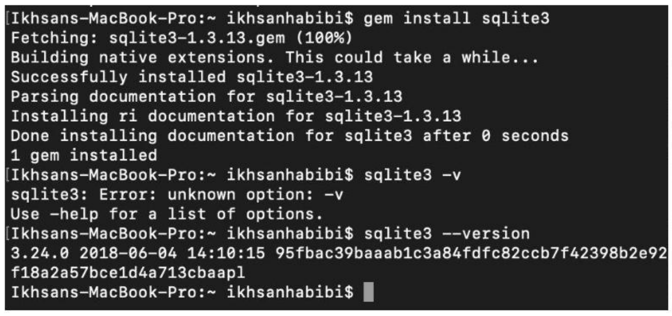
```
$ rails generate scaffold Customer email:string password string
$ rails db:migrate
```



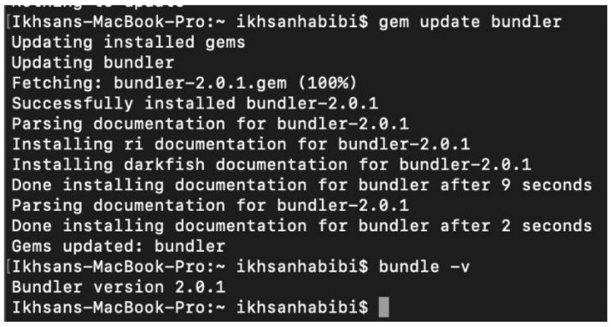
```
Last login: Sun Jan  6 20:49:04 on ttys000
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ rails -v
Rails 5.2.2
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ ruby -v
ruby 2.5.3p105 (2018-10-18 revision 65156) [x86_64-darwin18]
Ikhsans-MacBook-Pro:~ ikhsanhabibi$ 
```



```
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ gem install rake
Successfully installed rake-12.3.2
Parsing documentation for rake-12.3.2
Installing ri documentation for rake-12.3.2
Done installing documentation for rake after 0 seconds
1 gem installed
Ikhsans-MacBook-Pro:~ ikhsanhabibi$ 
```



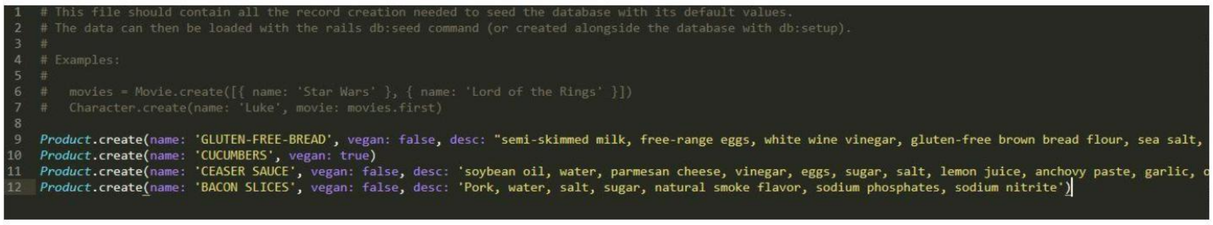
```
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ gem install sqlite3
Fetching: sqlite3-1.3.13.gem (100%)
Building native extensions. This could take a while...
Successfully installed sqlite3-1.3.13
Parsing documentation for sqlite3-1.3.13
Installing ri documentation for sqlite3-1.3.13
Done installing documentation for sqlite3 after 0 seconds
1 gem installed
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ sqlite3 -v
sqlite3: Error: unknown option: -v
Use -help for a list of options.
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ sqlite3 --version
3.24.0 2018-06-04 14:10:15 95fbac39baaab1c3a84fdfc82ccb7f42398b2e92
f18a2a57bce1d4a713cbaapl
Ikhsans-MacBook-Pro:~ ikhsanhabibi$ 
```



```
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ gem update bundler
Updating installed gems
Updating bundler
Fetching: bundler-2.0.1.gem (100%)
Successfully installed bundler-2.0.1
Parsing documentation for bundler-2.0.1
Installing ri documentation for bundler-2.0.1
Installing darkfish documentation for bundler-2.0.1
Done installing documentation for bundler after 9 seconds
Parsing documentation for bundler-2.0.1
Done installing documentation for bundler after 2 seconds
Gems updated: bundler
[Ikhsans-MacBook-Pro:~ ikhsanhabibi$ bundle -v
Bundler version 2.0.1
Ikhsans-MacBook-Pro:~ ikhsanhabibi$ 
```

2. You should never be attached to manually created test data living in your development or test database. Instead, you should have scripts that generate and regenerate this test data on demand.
Prof. Kleinen has an example script he prepared for rake: FactoryGirl and the rake task.
You can use the task with "rake db:populate" which automatically creates all models defined in spec/factories to populate your development database. The database will be reset before the data is populated. Remember, you should not be attached to any data in your development database.

We generated all models automatically, although we can create it manually. But the idea here is to make things done faster and easier. Instead of using FactoryGirl, we populate the database using seed.rb.



```ruby
# This file should contain all the record creation needed to seed the database with its default values.
# The data can then be loaded with the rails db:seed command (or created alongside the database with db:setup).
#
# Examples:
#
#   movies = Movie.create([{ name: 'Star Wars' }, { name: 'Lord of the Rings' }])
#   Character.create(name: 'Luke', movie: movies.first)

Product.create(name: 'GLUTEN-FREE-BREAD', vegan: false, desc: "semi-skimmed milk, free-range eggs, white wine vinegar, gluten-free brown bread flour, sea salt,
Product.create(name: 'CUCUMBERS', vegan: true)
Product.create(name: 'CEASER SAUCE', vegan: false, desc: 'soybean oil, water, parmesan cheese, vinegar, eggs, sugar, salt, lemon juice, anchovy paste, garlic,
Product.create(name: 'BACON SLICES', vegan: false, desc: 'Pork, water, salt, sugar, natural smoke flavor, sodium phosphates, sodium nitrite')
```

With the file db/seeds.rb, Rails have given us a way of feeding default values easily and quickly to a fresh installation. This is a normal Ruby program within the Rails environment. We then populated it with data via **rake db:seed.** We used the file db/seeds.rb at this point because it offers a simple mechanism for filling an empty database with default values.

For the sake of the exercise we decided to not create too many data, because it will take times. Instead we created just 4 to show it work.

3. Now, can you get a web page set up that accesses this database? Think back to your installation party! Make sure you can enter and delete data before you go on. Can you use the generated code to upload a picture for your comments? You may have to install a gem for this if you didn't do it during the installation party.

We created MVC (Model, View, controller) by generating scaffold and then we migrate the database. After doing this step, we tested whether the MVC works as it suppose to work. We tried basic CRUD operation to create, show/read, edit, delete/destroy. We installed a gem for feedback MVC. The gem that we used is carrierwave. Installing that gems is pretty clear, the instruction is also on the GitHub documentation.

4. Right, your interface could use a face-lift. Get down into the CSS and see how much more beautiful you can make it! Don't spend too much time on this, as there are other tasks.

This was really exhausting, seriously. The core of its system is more important to do, but if we don't have a beauty interface, who would have an interest to buy our hoagies? That is why, we did trial and error to implement CSS, make the hoagie shop look pretty. At least, we touched the CSS, because pretty is not just outside, but beautiful inside the system. There is no fancy CSS style that we implemented here, we did only the background color, some layouts and position. If we have more time, we could have done better designing our web and it will be mesmerizing.

5. Your first implementation was the admin side of your database, for entering in your data. Now make a customer application. It is able to configure a hoagie, and have it delivered. We don't need shopping carts and whatnot in this first iteration, they just purchase one if they click on a button "order".

On the admin side, we can remove the order, but on the customer side, they just can modify the order, not destroy the order. Thank God, we do not have to implement shopping cart, it makes life easier. But we implemented a mini multiplication, we just sell the hoagie for 3 Euro, means we multiply the amount of hoagies that customer ordered. If customer see the list of orders, the total price will be shown, how much the customer should pay for the orders.

6. Refactor your code as necessary, and see if you can manage to get the short reviews set up and working. Can you also add the selfies here? Wait a minute, if they can upload pictures, the users can upload any kind of photo. Better set up an application for the admin to review and delete fotos.

Refactoring code is difficult, because as an owner of the project, we always think that our code is fine, we do not see the problems with the code. Maybe we can make our code clean and prettier. We can minimize the amount of comments or using the same indentations. Yes, we implemented also the selfies. If we think logically, customer can only post a selfie after they bought the hoagies. No one could easily upload the selfies without buying the hoagie first. If the picture is inappropriate, we should control and delete that photo. This is only administrator rights.

7. Let's see those statistics! You did remember to set up a table that is not linked to any screens yet that will persist your data? Make a new screen to show this data. Can you now get your configuration page to update this? Remember, people can choose to put onions on and then take them off again before they order, so only store the data when the order is triggered.

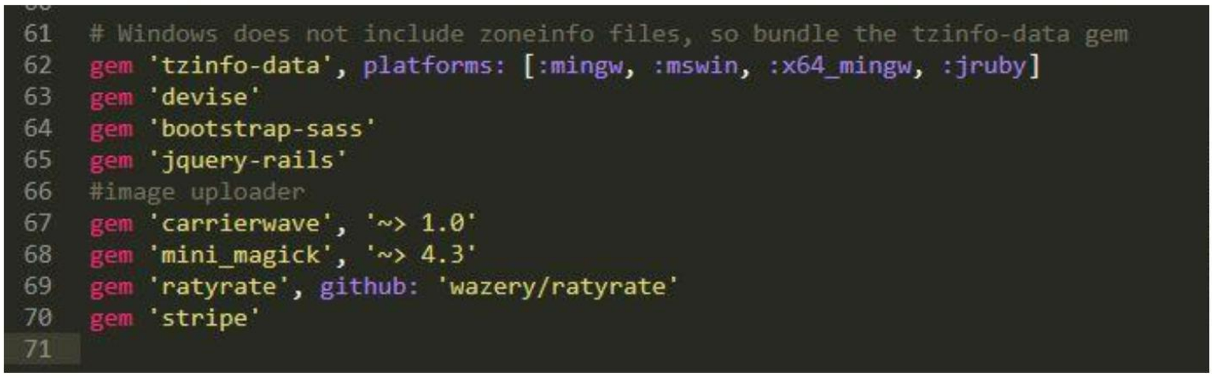
We did not have enough time to implement this statistic data. We just implemented that the customer can see their order and the details. Although it is nice to have, because management team can create a marketing strategy. We just imagined as we created a start up, it takes time to be on that step.

8. Bored? Keep testing, refactoring and implementing! Did you remember to include an impressum? What is the maximum number of ingredients you can add to a hoagie? What happens if you want to order three hoagies, all the same? Do you have to re-implement everything? Can you save a configuration and give it a name? The sky's the limit! Just see how far you can go!

We implemented an impressum, because it is easy to write about the owner, address of hoagie shop. The maximum amount of ingredients is one. Customer can choose one and only ingredient or without it. Customer can order the ame hoagie more than one. But there is no limit amount to order hoagies. We did not implement a configuration in order to save the custom hoagie that customer wants. We just created a general hoagie with different ingredients. You can see our ingredients in products.

PROJECT PROGRESS

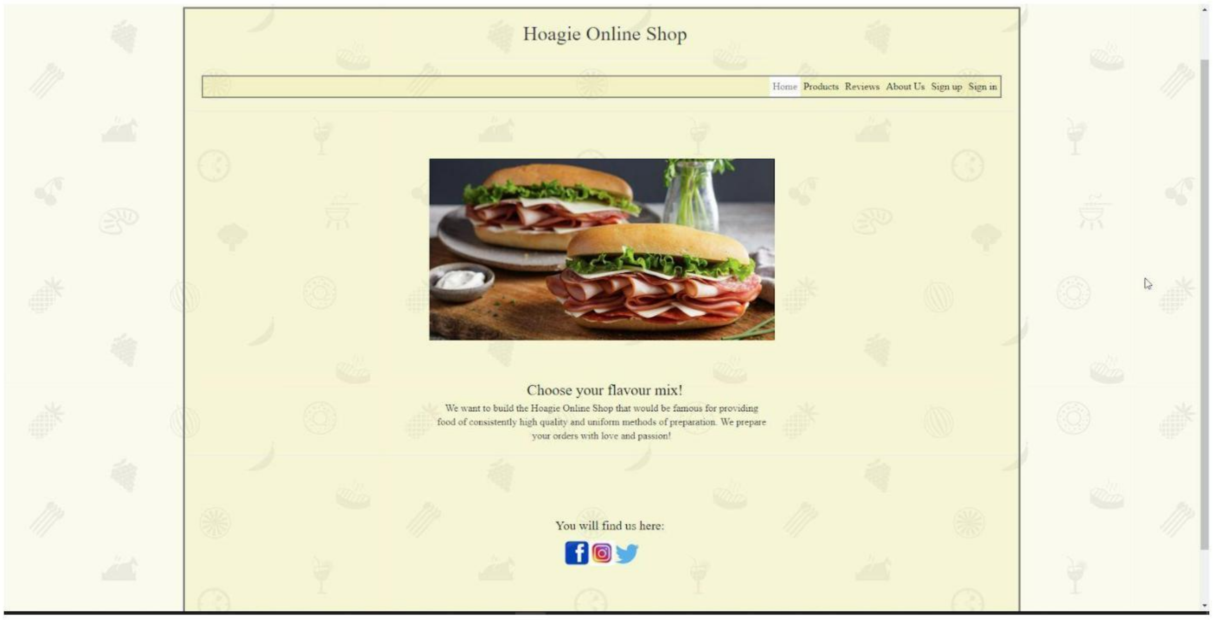
There was a lot to learn and do during this exercise. The back-end part was a mess but we managed to the most of it. We have to use google and youtube to figure out how to do certain things. But that was also a good way to learn new stuffs.



```ruby
# Windows does not include zoneinfo files, so bundle the tzinfo-data gem
gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]
gem 'devise'
gem 'bootstrap-sass'
gem 'jquery-rails'
#image uploader
gem 'carrierwave', '~> 1.0'
gem 'mini_magick', '~> 4.3'
gem 'ratyrate', github: 'wazery/ratyrate'
gem 'stripe'
```

Here is our gemfile that we used to create the site.

- devise: Devise gem is the solution for Authentication, for creating User and Admin role. We can control what the customer as User account can see and do.
  - As a guest (without signing in), one can only see and access a limited of sections such as: **Homepage** - Showing the home page

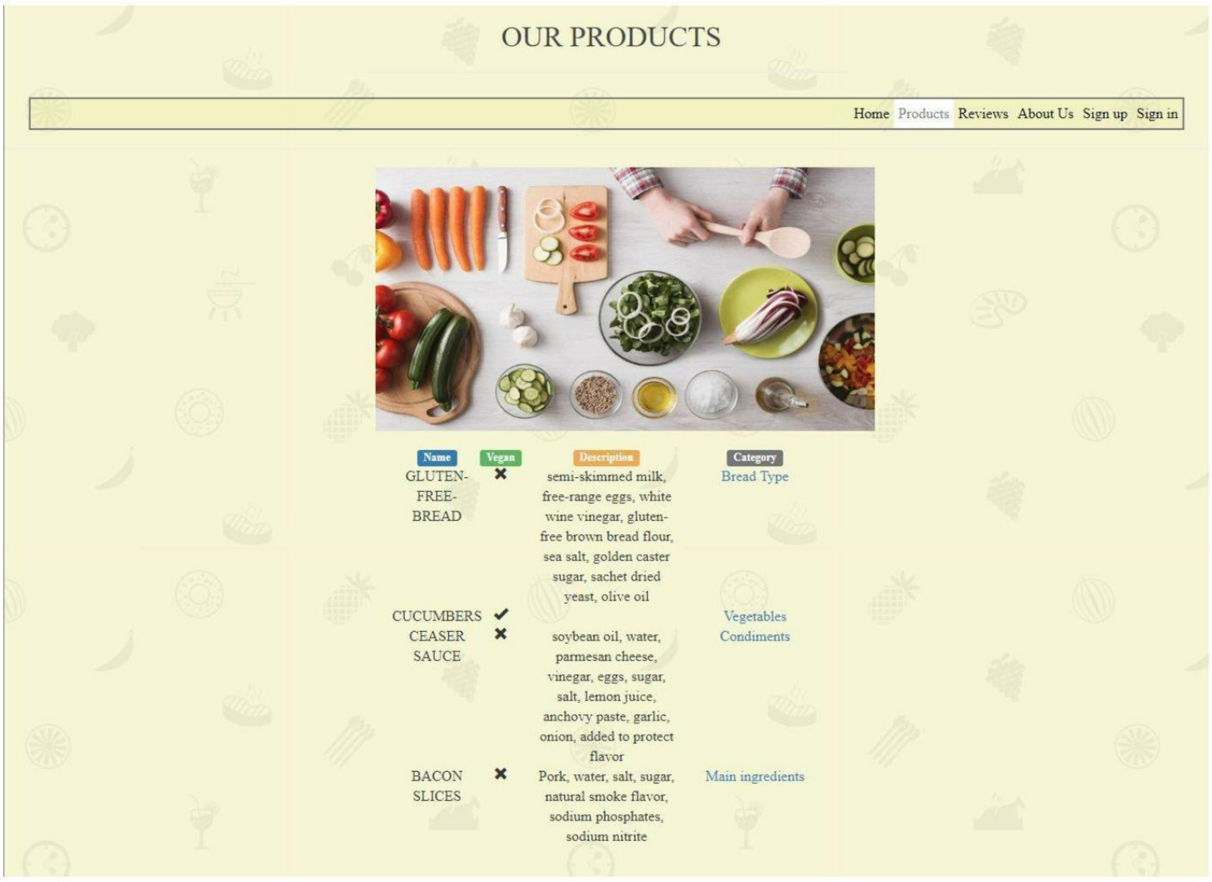


Choose your flavour mix!

We want to build the Hoagie Online Shop that would be famous for providing food of consistently high quality and uniform methods of preparation. We prepare your orders with love and passion!

You will find us here:

  - **Products** - Showing the ingredients menu, guest can only see, can't edit or add a new ingredients

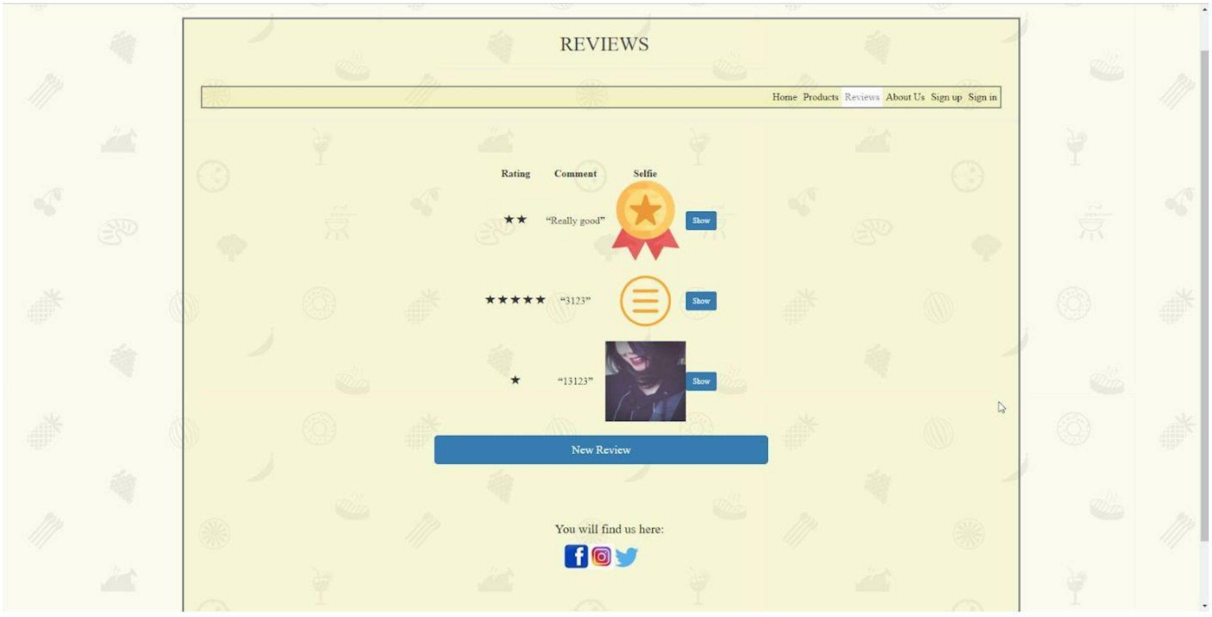


OUR PRODUCTS

Home Products Reviews About Us Sign up Sign in

Name	Vegan	Description	Category
GLUTEN-FREE-BREAD	✗	semi-skimmed milk, free-range eggs, white wine vinegar, gluten-free brown bread flour, sea salt, golden caster sugar, sachet dried yeast, olive oil	Bread Type
CUCUMBERS	✓		Vegetables
CEASER SAUCE	✗	soybean oil, water, parmesan cheese, vinegar, eggs, sugar, salt, lemon juice, anchovy paste, garlic, onion, added to protect flavor	Condiments
BACON SLICES	✗	Pork, water, salt, sugar, natural smoke flavor, sodium phosphates, sodium nitrite	Main ingredients

- **Reviews -** Showing the reviews of other User and also their Images, guest can only see this



REVIEWS

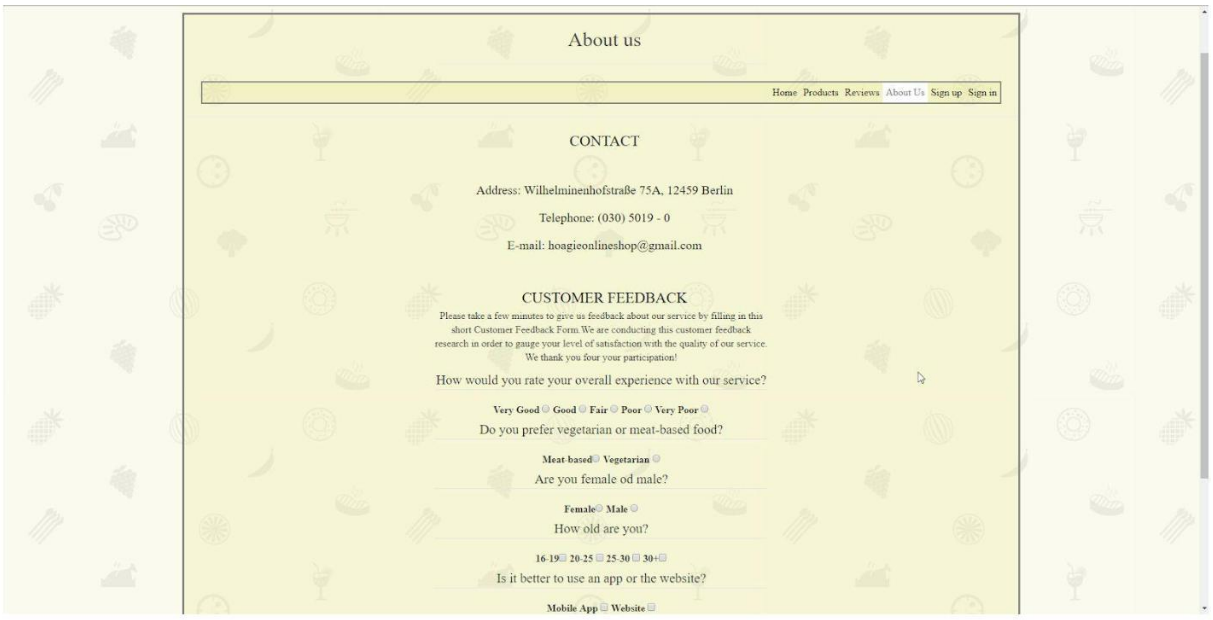
Home Products Reviews About Us Sign up Sign in

Rating	Comment	Selfie	
★★	"Really good"		Show
★★★★★	"3123"		Show
★	"13123"		Show

New Review

You will find us here:

- **About us -** Showing details about the shop.



CONTACT

Address: Wilhelminenhofstraße 75A, 12459 Berlin

Telephone: (030) 5019 - 0

E-mail: hoagieonlineshop@gmail.com

CUSTOMER FEEDBACK

Please take a few minutes to give us feedback about our service by filling in this short Customer Feedback Form.We are conducting this customer feedback research in order to gauge your level of satisfaction with the quality of our service. We thank you four your participation!

How would you rate your overall experience with our service?

Very Good ○ Good ○ Fair ○ Poor ○ Very Poor ○

Do you prefer vegetarian or meat-based food?

Meat-based ○ Vegetarian ○

Are you female od male?

Female ○ Male ○

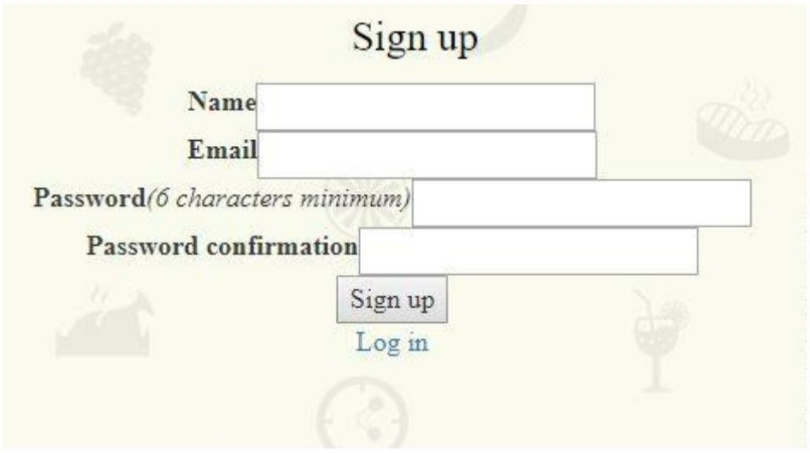
How old are you?

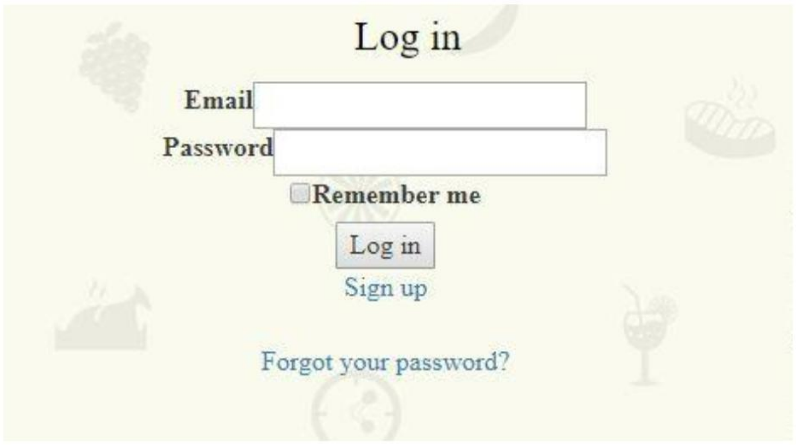
16-19 ☐ 20-25 ☐ 25-30 ☐ 30+ ☐

Is it better to use an app or the website?

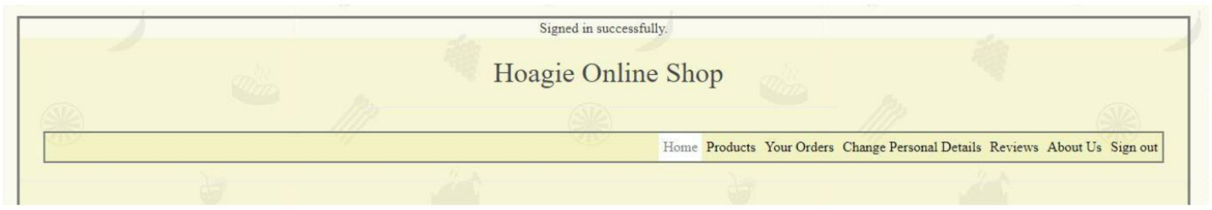
Mobile App ☐ Website ☐

- Also guest will have options to Sign in (if he/she already have an account) and Sign up (if not). The devise gem allows us to easily create this with a simple registering form:

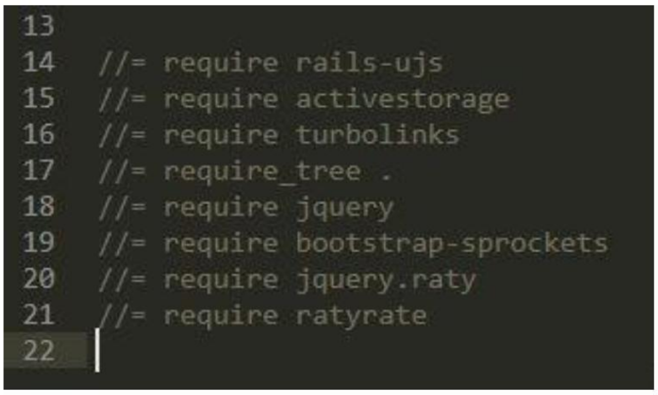




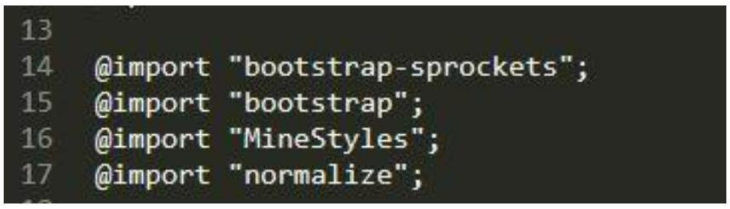
- ○ After logging in, user will be allowed to access more sections.



- bootstrap-sass: this gem allows us to use bootstrap css and its component for the site. To use bootstrap we need to include some "require" code in application file, both in css and js file.



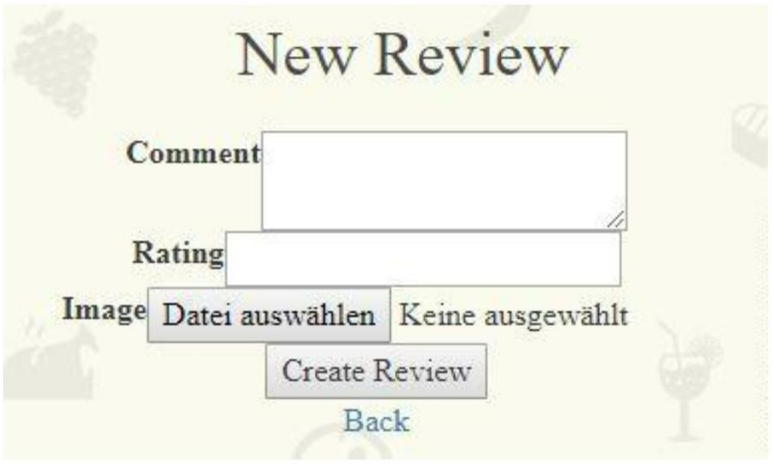
```
//= require rails-ujs
//= require activestorage
//= require turbolinks
//= require_tree .
//= require jquery
//= require bootstrap-sprockets
//= require jquery.raty
//= require ratyrate
```



```
@import "bootstrap-sprockets";
@import "bootstrap";
@import "MineStyles";
@import "normalize";
```

  - ○ Because we also have our css files, to use them we need to import them into application scss file
  - ○ Thanks to bootstrap we can created a simple 5 star rating (by replacing the number with the star icons) and also a lot of component that make the site look better.
- jquery-rails: this allows us to use jquery on rails

- carrierwave: This gem provides a simple and extremely flexible way to upload files from Ruby applications. It works well with Rack based web applications, such as Ruby on Rails. We used this to make an Upload function for our Review page.



New Review

Comment

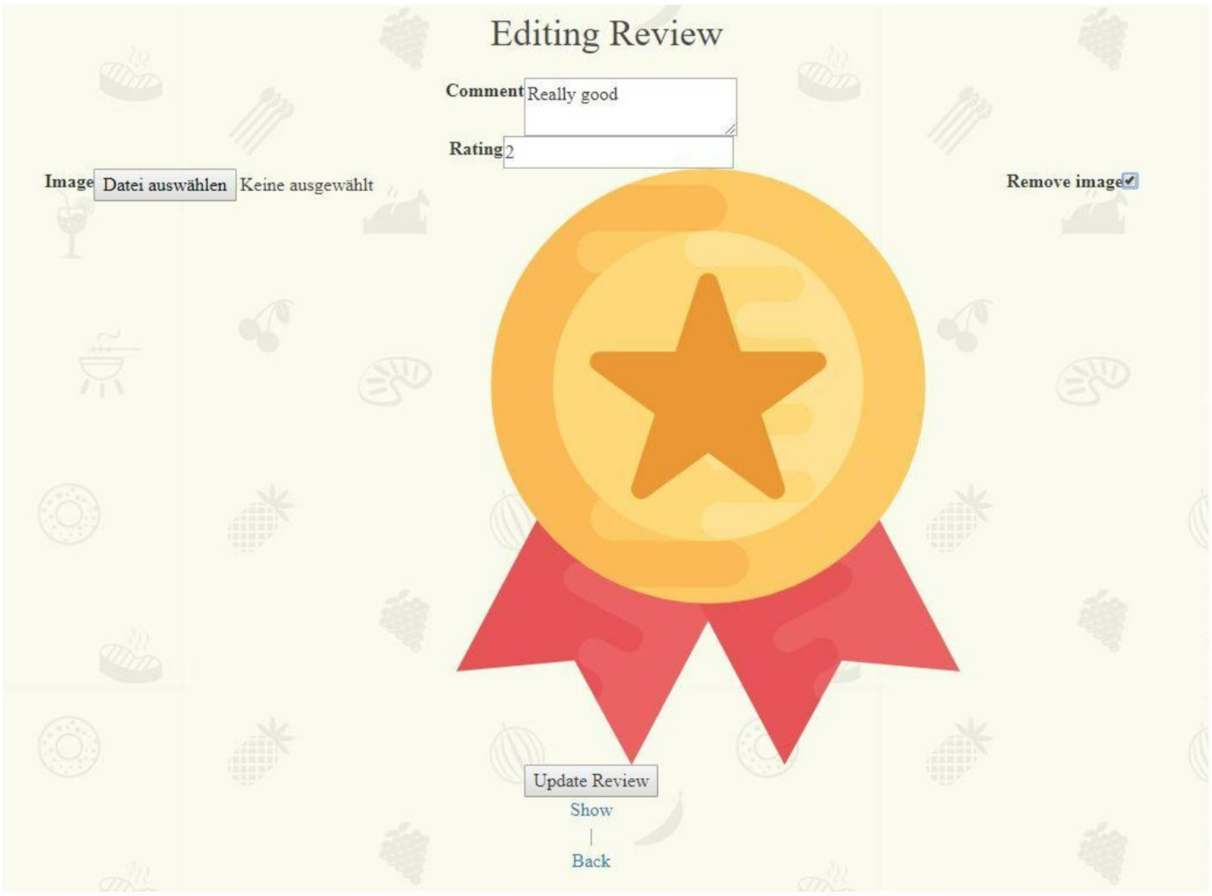
Rating

Image Datei auswählen Keine ausgewählt

Create Review

Back

  - The Image uploaded will be store in our database and can access anytime by admin
  - Admin can also edit and remove what he/she want, even just only the picture by checking the Remove image box then Update.



Editing Review

Comment Really good

Rating 2

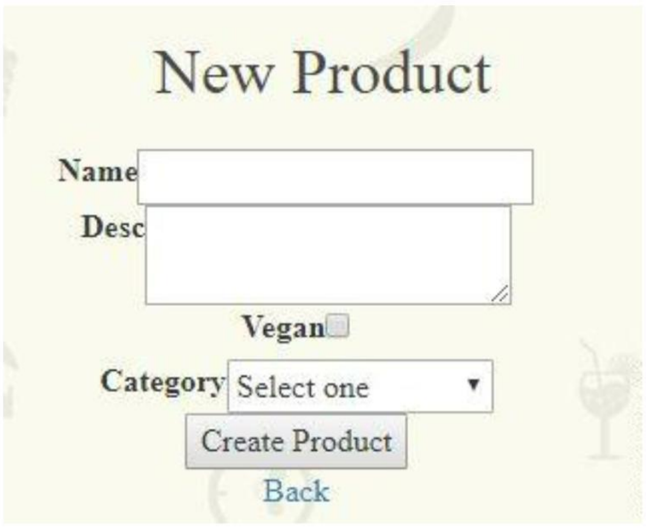
Image Datei auswählen Keine ausgewählt

Remove image ✓

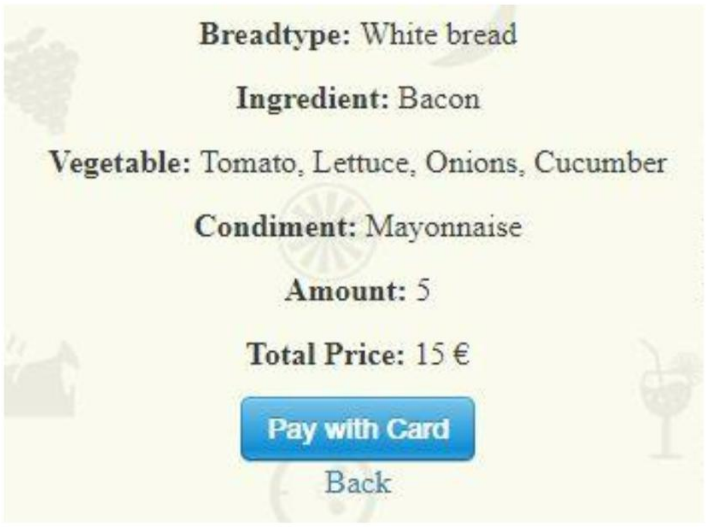
Update Review

Show | Back

  - Admin will also allowed to add new product, also edit or delete everything.



- mini_magick: This game we install along side of carrierwave, what this do is allow us to resize the image user upload to fit what we want to show on the page.
- ratyrate: this is a gem that allow us to create a 5 star rating system using jquery, but we decided not to use this anymore because it somehow not working, and we need a lot of time to figure out why.
- stripe: Stripe is a gem for Checkout/Pay function. Really easy to set up and create.
    - Stripe is our solution for Payment function.



Breadtype: White bread

Ingredient: Bacon

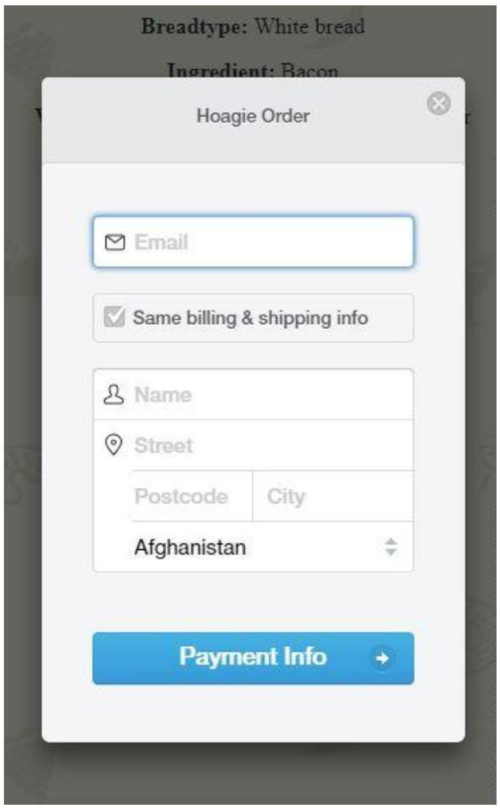
Vegetable: Tomato, Lettuce, Onions, Cucumber

Condiment: Mayonnaise

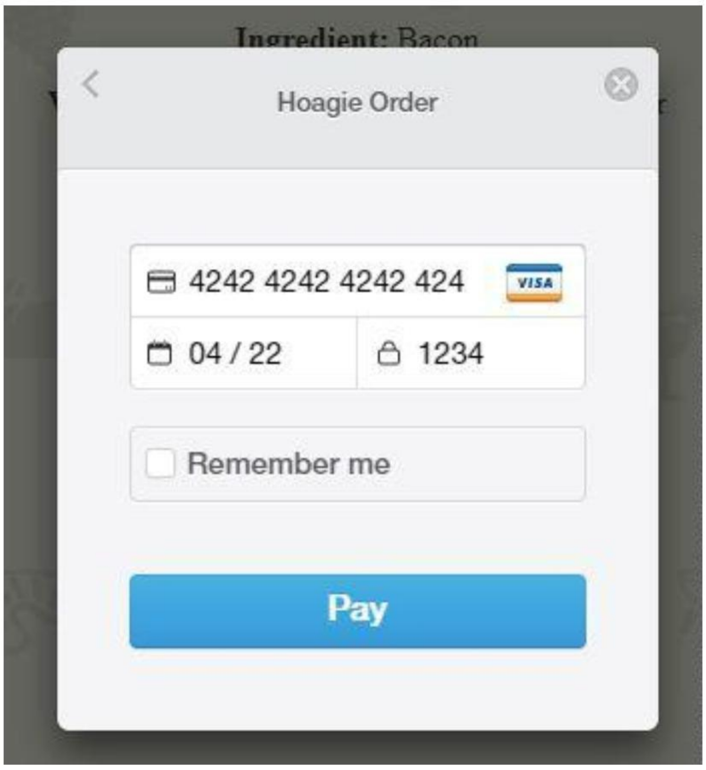
Amount: 5

Total Price: 15 €

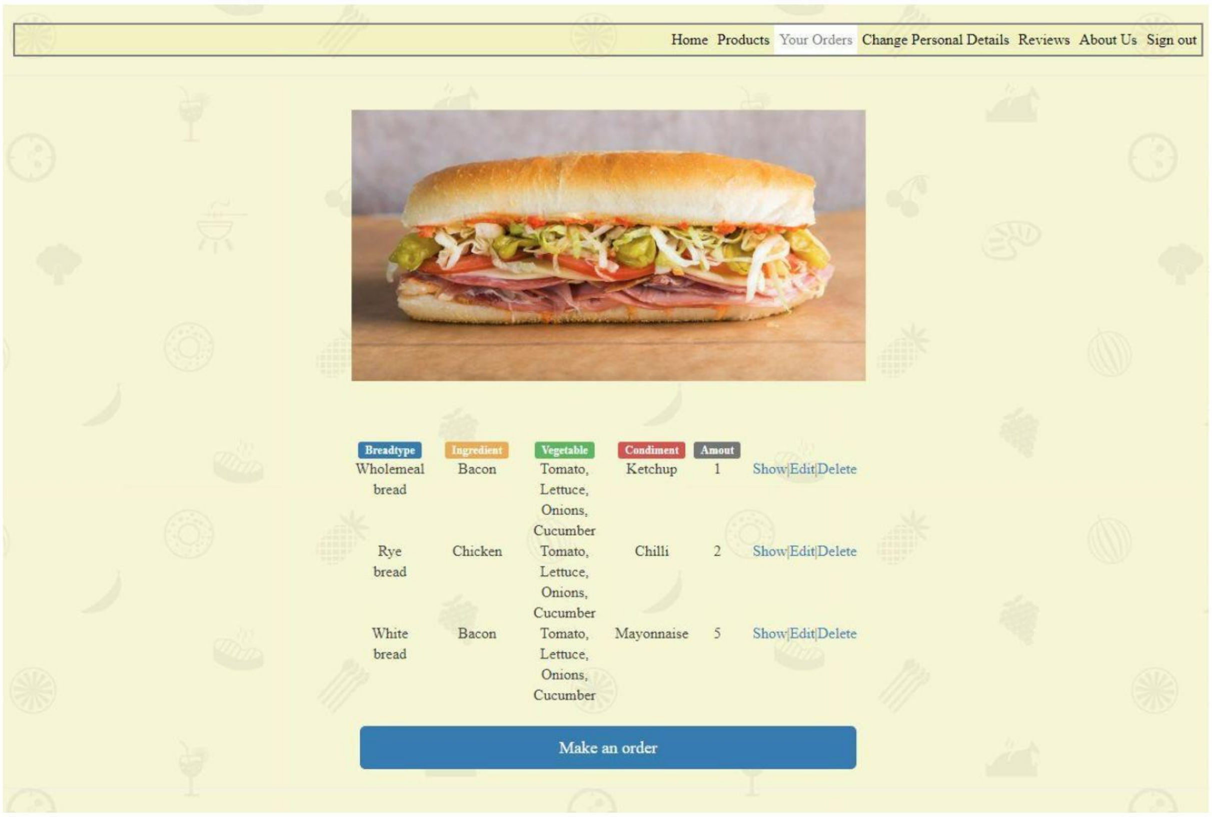
    - Basically when customer go into their order, they will see a Button that allow them to pay the order. Our idea for ordering was the customer can make as many order as he can but only when he pay, the order will send to the shop to process.



- The customer will be asked to give their information like email, addresses (for billing and shipping, there's also a check if 2 addresses are the same)..after that customer have to insert their payment info



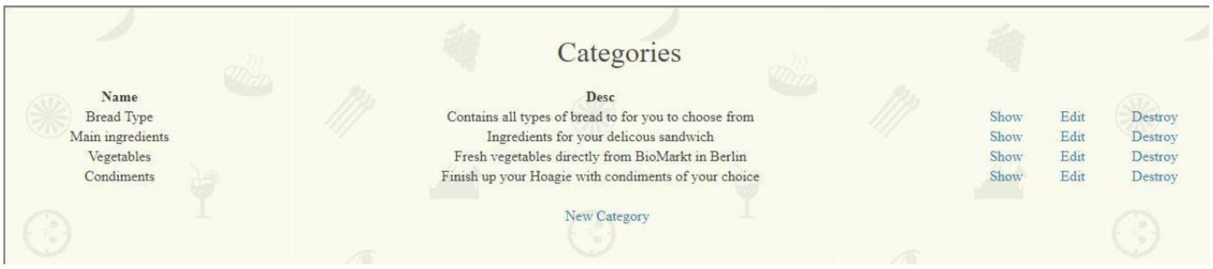
- The customer can check their order anytime by going to Your Order section. Our idea is to just showing the orders that are made by this User only. Our solution at first is using has_many and belongs_to and model file but not working. We think we are in the right way but we don't have much time since we have to prepare for the exams. We skipped this by only showing the order are made by everyone. This definitely not a good way to show the order information but we will try to finish it when we have time.



Breadtype	Ingredient	Vegetable	Condiment	Amout	
Wholemeal bread	Bacon	Tomato, Lettuce, Onions, Cucumber	Ketchup	1	Show|Edit|Delete
Rye bread	Chicken	Tomato, Lettuce, Onions, Cucumber	Chilli	2	Show|Edit|Delete
White bread	Bacon	Tomato, Lettuce, Onions, Cucumber	Mayonnaise	5	Show|Edit|Delete

- The customer can see each order and also edit and delete as they want.

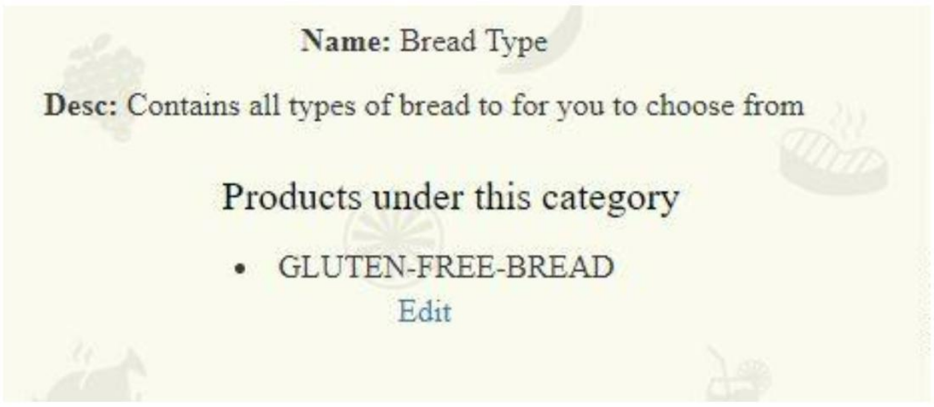
- There is also one part we're not include because we don't have time to finish, is the "Category" section.



Categories

Name	Desc			
Bread Type	Contains all types of bread to for you to choose from	Show	Edit	Destroy
Main ingredients	Ingredients for your delicous sandwich	Show	Edit	Destroy
Vegetables	Fresh vegetables directly from BioMarkt in Berlin	Show	Edit	Destroy
Condiments	Finish up your Hoagie with condiments of your choice	Show	Edit	Destroy

As the name say, this section will show every categories that the products belong to, like "bread type", "vegetable" etc..When we select "show" at any categories, it will show the details and every products inside under categories, like this:


Name: Bread Type

Desc: Contains all types of bread to for you to choose from

Products under this category

- GLUTEN-FREE-BREAD

Edit

We think this is a really smart thing to do because it will optimize our database and make it user-friendly. And of course only admin can add and edit this, other user can only see what inside.

END OF THE EXERCISE

There are definitely some bugs and function that we have not fixed or created yet but we have tried our best to do what are the most important. We are pleased with what we have done and what we have learned during this exercise. This was a really great opportunity to step outside of the comfort zone and do what you have to do.

EVALUATION

Ikhsan Habibi

During this exercise I learned how important is to plan everything. We could send this protocol one week after the final exam, but it was still really stressful for me. I'm sure that without a good planning practice I'll be totally lost. Together with the team I prepared the sprint board with help of the Trello platform and step by step I realized all of the tasks.

During last few lab sessions I learned a lot of about Ruby. Before this semester I had no idea about Ruby and now? I have already build my very first website with Ruby. It's really cool! I want to start in summer with Ruby, but I didn't find enough motivation, the teamwork and good structure during this semester was really helpful in this case. I found also many great tutorials for beginners and intermediate. If I have problems - I searched a lot on stackoverflow or I asked my teams.

I see also, that during this semester I improved my English skills what helps me a lot at my work. I'm working right now as a junior developer and I'm a part of the international team and we communicate in english. I have to write also the documentation in English and at the beginning it costs me a lot of time and stress and now I realised, that I made great progress and I feel much more comfortable. It's also really important for the future to be a developer.

I learned different project management methodologies and I can list without any problem the differences between modern and traditional management. As a team we worked a lot with agile method - it was a great experience and lesson for me, because right now I can use all of this knowledge in the real life. I think it's really important, that we learn at university things, which we can use in our future, not only the theory. but the real work. Thank you for this opportunity! :)

Julia Rakowiecka

The last lab was the most demanding and time-consuming. Before we start the whole implementation, each of us has to read all of the previous protocols and what's more important - all of the feedback. I made notes for each lab protocol - it was very useful during next steps and it saved a lot of time. I could use this time for more important tasks.

During this lab I learned more about Ruby. I saw some tutorials and use the recommended materials. I connect the acquired knowledge from Database and Info3 and I learned how I can implement the database in "the real world". It was pretty nice and stressful at the same time, because at the beginning I have a lot of problems with installation, but at least I used the tipps from my colleagues - they shared many useful information in the moodle platform. Sometimes is just good to know, that I'm not the only one who is totally lost! :)

During the first meeting Ikhsan offered us to use the Trello as our communication platform. I heard before about Trello, I think we spoke about this platform in first semester at Computer Systems, but I've never use it. At the beginning it was just a little bit complicated, but after first week I recognised the advantages of this platforms. It helps me a lot - each of us wrote there a short note after each task and thanks this opportunity each of us could see in which moment are we now and what we should make as next. This platform has also a great layout and in my opinion is very useful for different kind of documentation.

During this exercise I learned also, that the sprint board is always a good idea. Of course, we made a lot of changes in our plans and it happened, that we were not ready with all of the tasks that we planned, but at least we stayed focus on the main problem and we build kind of structure in our teamwork.

It was also a good refresh-experience for me, because I forgot a lot of from CSS rules. I have to check many times the w3school.com, but at least I found all of the answers for my questions.

I would like to conclude by saying, that I learned a lot during this semester - not only about modern software engineering, but also about myself. I spent many times with my colleagues, we discussed a lot. I gave and received feedback - thanks this opportunity I improved my soft skills and self-confidence. At the beginning was hard to get a criticism, but in my opinion it was really important. This constructive criticism shows me, that one perspective for more complex exercise/project is not enough. It is really important to communicate with other and shared our ideas - it could reduce the number of mistakes and saves our time. I improved also my knowledge about Agile methods.

Manh Cuong Tran

This semester I learned a lot important and cool stuff, not only technical knowledge, but also how to work and communicate in team. I don't want to say that it was an easy semester, but at least I'm done with all of the projects and I learned something new.

During this adventure I learned what are scenarios and use cases. I know also the difference between both of them. I know how to draw sequence, class and state diagrams. I know that I have different possibilities to test my code and I learned also how I can deal with legacy code - nice to know! I refreshed and improved also knowledge from the previous semester e.g.: building the database, css.

This exercise improved my Ruby skills, but also shows me that I have to learn pretty much more. It is clear that there is still a lack of certain things. I was really confused because of the whole process with installation. At the beginning I spent a lot of time looking for a solution, but I least I fixed all of my problems. I made great researched, that's why I found many interesting information about

Ruby. I'm satisfied that together we build the Hoagie Shop.

From the organisational perspective I learned how to write a good documentation and how important is communication with my colleagues. I still need to understand that notes on paper are good and sometimes(not always) is useful to create them. What was most important for me I finally learned what Agile exactly is and how I can handle with it. Why it was so important for me? I have already applied for working student position in IT company and during our first meeting they asked me about Agile method and if I have any experience with this methodology - I could answer yes, because I practised this method during the teamwork. I was really surprised, that thanks my knowledge from university(not programming skills) I increased my chance to get this position.

TIME TABLE

Name	Task	Time	Comment
Ikhsan Habibi	Exercises, Notes, Report	34 hours	During this exercise we worked together. We found the new communication platform - Trello. Each member of our team has an access - that's why each of us could add some comments and create check-list. It improved the tempo of our work and helps us to prepare documentation. We made also notes on paper.
Julia Rakowiecka	Exercises, Notes, Report	34 hours	It was a teamwork. We used Agile methods. We shared our roles and we prepared sprint boars. We shared our ideas, we created the docs file together. We used different platforms for communication e.g.: Whatsapp, Google Docs, Trello, Team Viewer. We built also the to-do list. It costs a lot of time to check previous protocols and follow all of the details and comments - that's why we used different tools for communication and notes.
Manh Cuong Tran	Exercises, Notes, Report	34 hours	We worked together as a team. We used the Google Docs as a platform for our protocol, we used also another tools to communicate. We prepared sprint for each week, which was very useful. After each exercise/week we made a short summary - it gives us a great view - what we should make as a next step and because of it we worked more structural.

WE SAY THANK YOU FOR TEACHING US!

It was a great pleasure to learn Software Engineering taught by you Professor Weber Wulff. The nice thing is that we learn how to organize to produce/develop a good software. The structure of the lectures materials were just insanely good structured. We couldn't ask for a structure that is better than the structure you made. Thank you for being a nice and friendly professor to all of students especially us Ikhsan, Julia and Tran. You know that every student has different way to solve problem and thoughts. Imagine if the students hate the professor, just because the students don't like the professor behavior or teaching style, there is no good future life ahead of them. The university should produce good alumnus to create a better future life. We as students really hope, there is more Professors like you. Clearly details, helpful and motivate us to learn something new every week. This kind of motivation is just seldom we found in other courses. Some professors just taught us how can we pass the exam with 4.0 without having deep understanding, not only theory but also in practice, what we have done so far and what actually we learned. I hope, your passion to teach student does not fade away. It is really great value or gift that you have in this world.