Auction Marketplace

Back-End Project

Project description

Create REST server for auction marketplace where people (as a sellers) can present their products for sale at auctions' lots from one side, and from the other side - take part in the lots of other people (as a customers) to buy their products.

Project Details

Entities description

There are 4 entity types in the Auction Marketplace database:

Users. Users - are an entities that created when user registers in the system. Each user can be seller or/and customer of products. In general user's entities should have next attributes (all attributes are required):

- email;
- phone;
- first name;
- lastname;
- birth day.

You should add validation for the following attributes: email, phone, birth_day (age must be > 21).

Lots. If user want to sell any product he should create a *Lot* in the system. All users should have ability to see list of all lots that have *in Progress* status.

Lots should have the following attributes (all attributes are required):

- title (required);
- image;
- description;
- status (required);
- created at (required) - need to sort lot's list by creation time, for pagination;
- current price (required) - represent start price of the lot or max bids' proposed price, if they exist;
- estimated price (required) - price that is maximum for current lot - if user propose this price, than he can buy the lot immediately;
- lot start time (required) - time when the lot will be open;
- lot end time (required) - time when the lot will be closed.

Statuses:

- *pending* - by default;
- *inProcess* - changed, when *lot start time* become;
- *closed* - changed, when *lot end time* become or any customer propose *max estimated price*.

Validations: fields related to datetime, price (cannot be negative).
End time cannot be less than start time and start time cannot be less than current time

Bids. When lots status become *inProcess* other users (customers), that want to buy this lot can propose their price. Bid entities represent this propositions. User can create bids as many times as he wants while lot is in *inProcess* state.

Bid's attributes (all attributes are required):

- bid creation time;
- proposed price.

Validations: proposed price (cannot be negative).

Orders. When the status of the lot becomes *closed*, and there is one or more bids present for the target lot, system automatically sends notification to customer that won this lot.

If customer ready, he creates Order record for the corresponding bid, that represent information of the product delivery. Order fields (all fields are required):

- arrival location - text field where user should enter product destination address;
- arrival type (either pickup or one of the post delivery companies: Royal Mail, United States Postal Service, DHL Express);
- status

Statuses:

- *pending* - waiting while seller accept customer's order;
- *sent* - means that seller accept order and the process of product's delivering has been started;
- *delivered* - user setup this status after he got product.

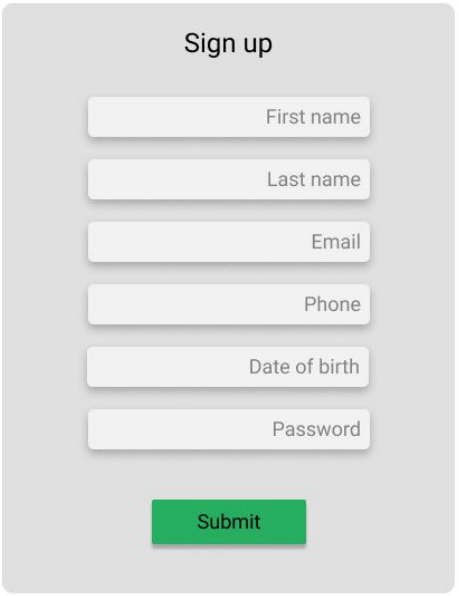
Application's Flows

Authentification flows

All the registration, auth, forgot password flow can be implemented using Devise gem.

Registration

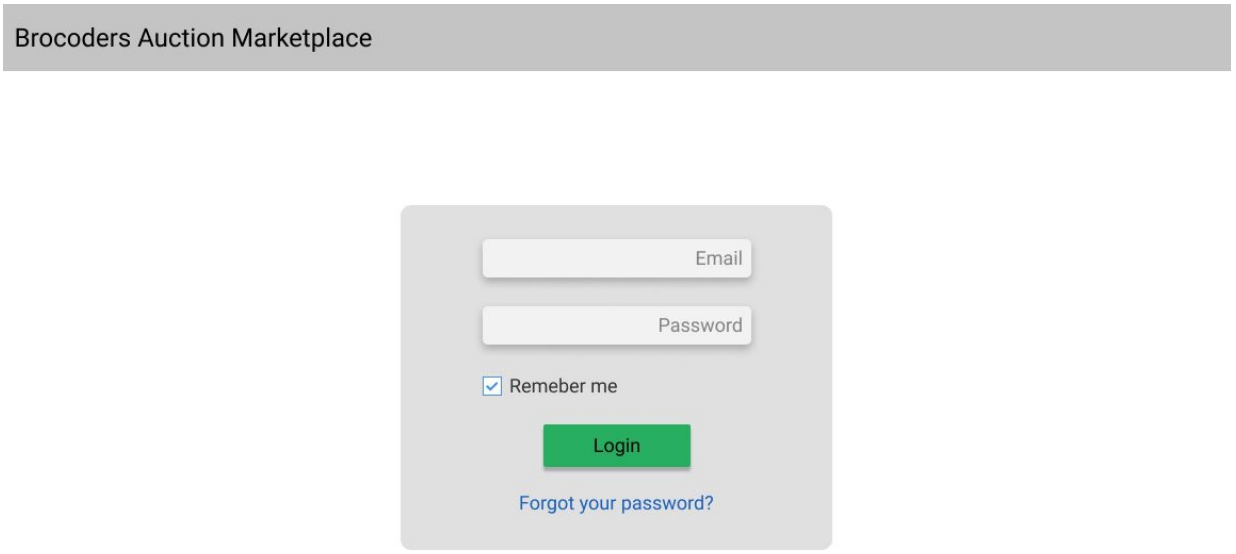
1. User goes to **Sign up** page. He should enter all fields - all of them are required. Email and phone is unique for all users.



2. When he enter all fields, he must click on *Submit* button. After that 1) system should send email with confirmation message and link; 2) He should go via the link address specified in the email. When user follows this link, system will finish the registration process.

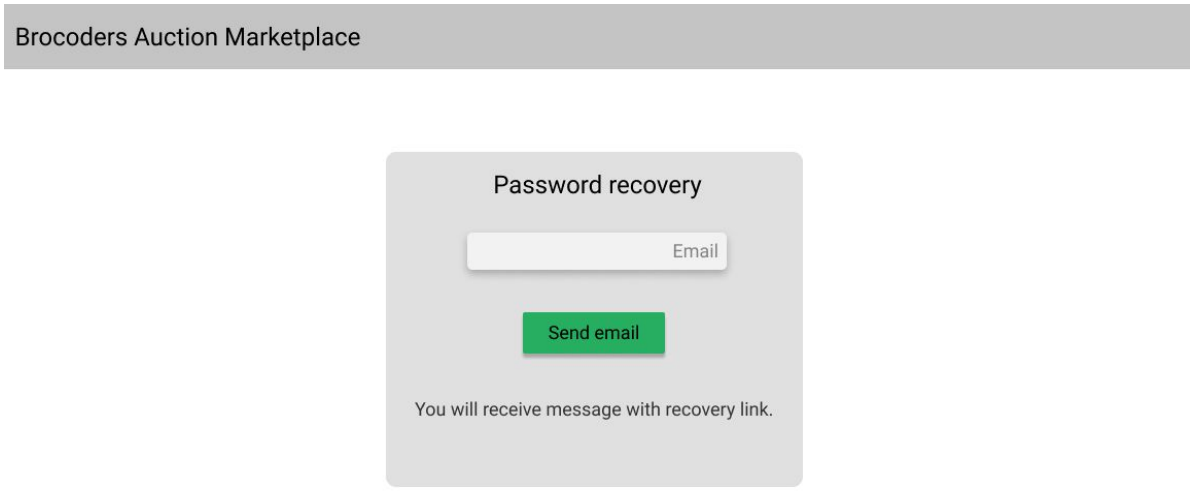
Login

1. User can login to the system using his email and password.
2. System must support ability to remember user in the system (by default).



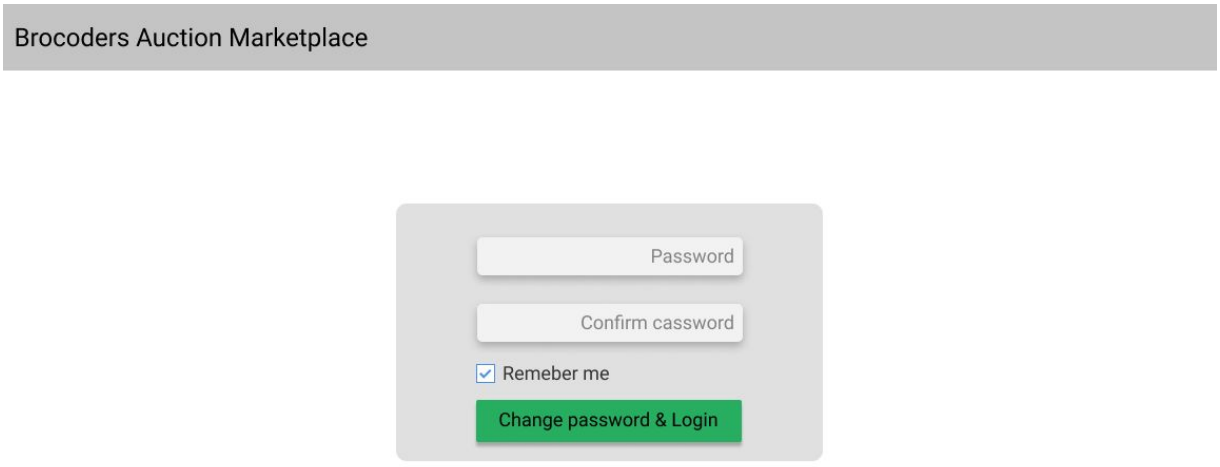
Password recovery

1. System must support password recovery. User goes to the corresponding page:



2. After that user should enter his email, click *Send email*, and than system sends recovery link to his email if it correct (means exist in the system);

3. Than user should check his email mailbox and find out letter with password recovery link. This link must redirect user to the next page:

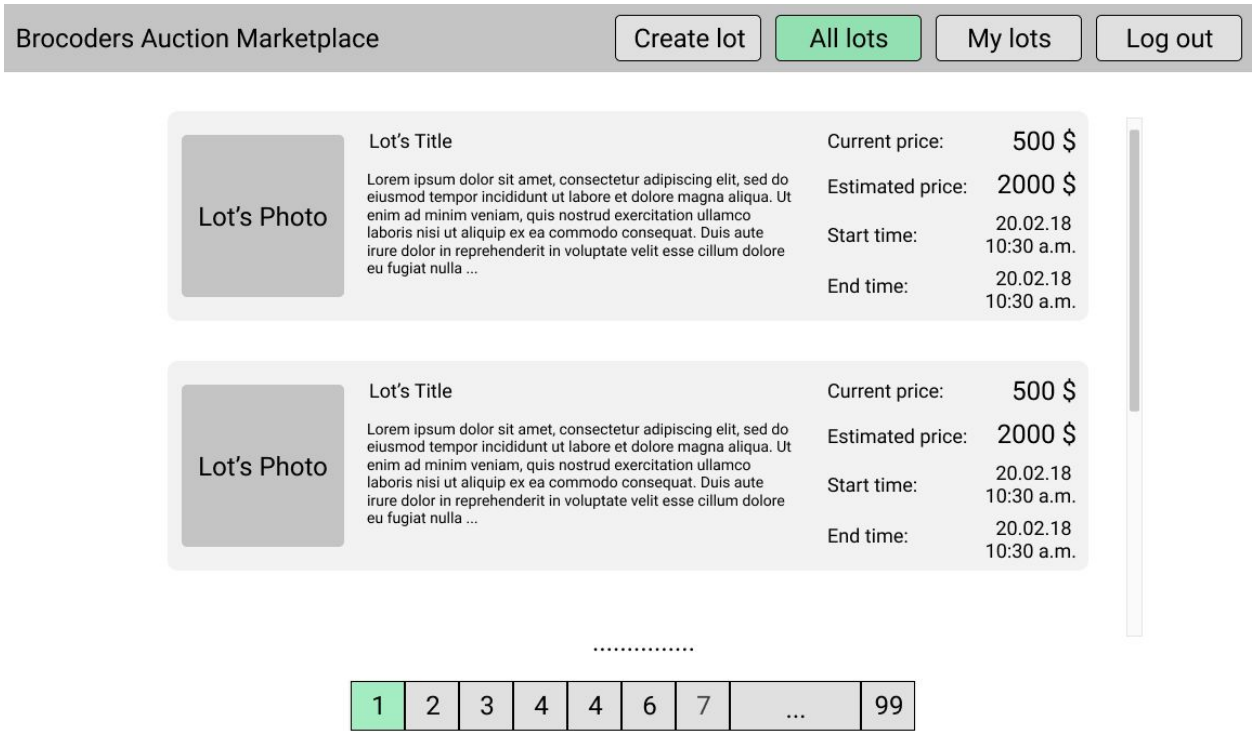


4. If user enter fields correctly and press *Change password and Login* system will update his password and login user to the system (return token).
   Also system allows to perform *Logout*.

Lots flows

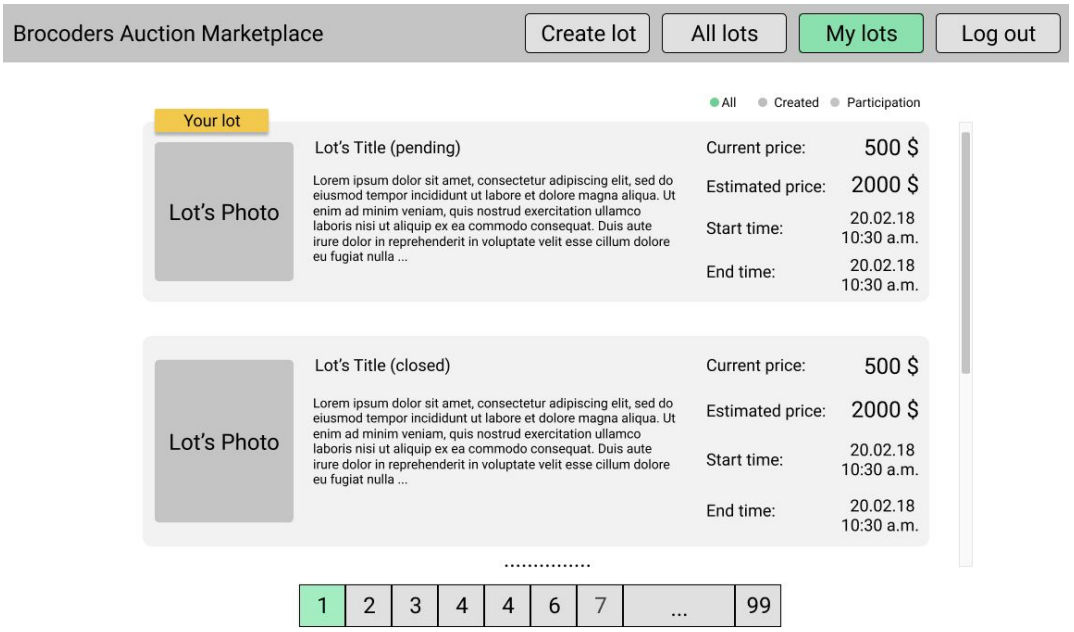
After login, system allows users to navigate to the next pages: *All lots, My lots, Create lot*.

All lots:



On this page user should see all lots with *inProcess* status. This pages should support pagination (10 pages per page) and provide information that represented on screenshot. If user want to see full information about lot, he should click on *Lot's Title*.

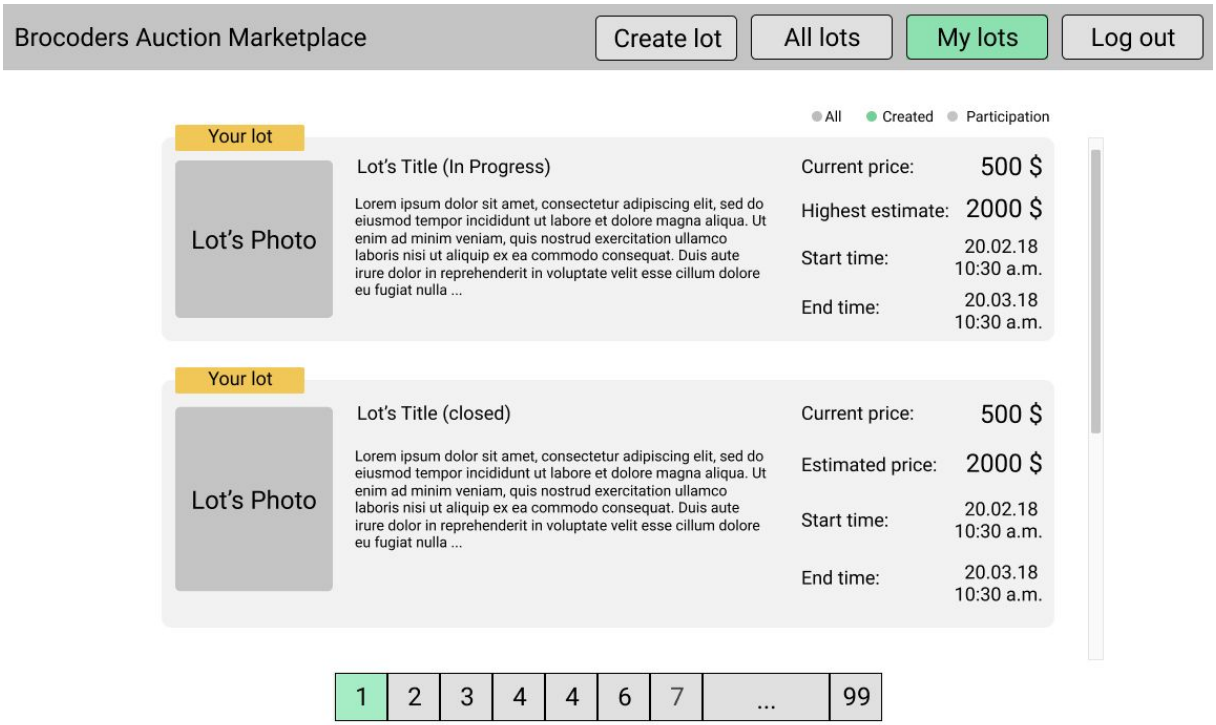
My lots:



My lots page contains all lots that user created as a seller or which user takes part in as a customer (means proposed any bid).

Main requirements to this page:

- pagination (10 lots per page);
- lot that user created as seller should be marked as *Your lot*;
- user must have ability to filter page by next criteria:
    - all
    - created (render only lots that user create for sale)
    - participation (render only lots that user won/try to win)



Lots creation:

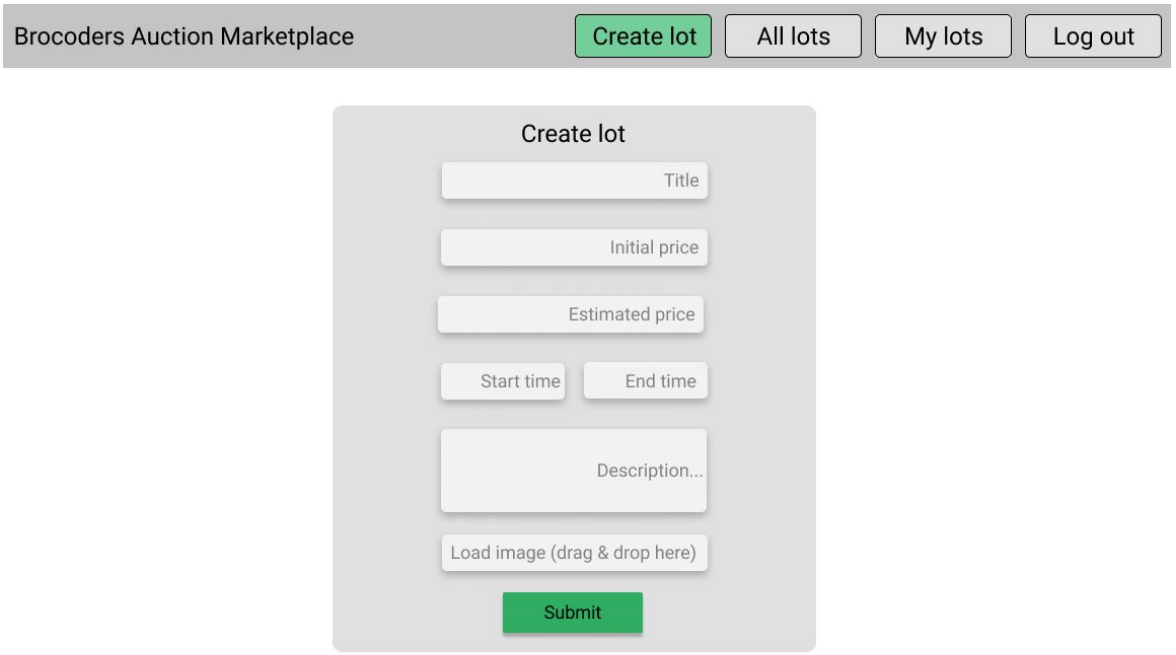
Any user can create a lot. He can specify all fields in lots creation form. Image and description fields can be empty but all other fields is mandatory.

Current price represent maximum proposed price by bids or initial price (if there is no bids present).

Estimated price - when customer proposes this price (greater than or equal to) as a bid lot will be closed, and this customer becomes a winner.

Initial and Estimated prices couldn't be negative.

End time cannot be less than start time and start time cannot be less than current time.

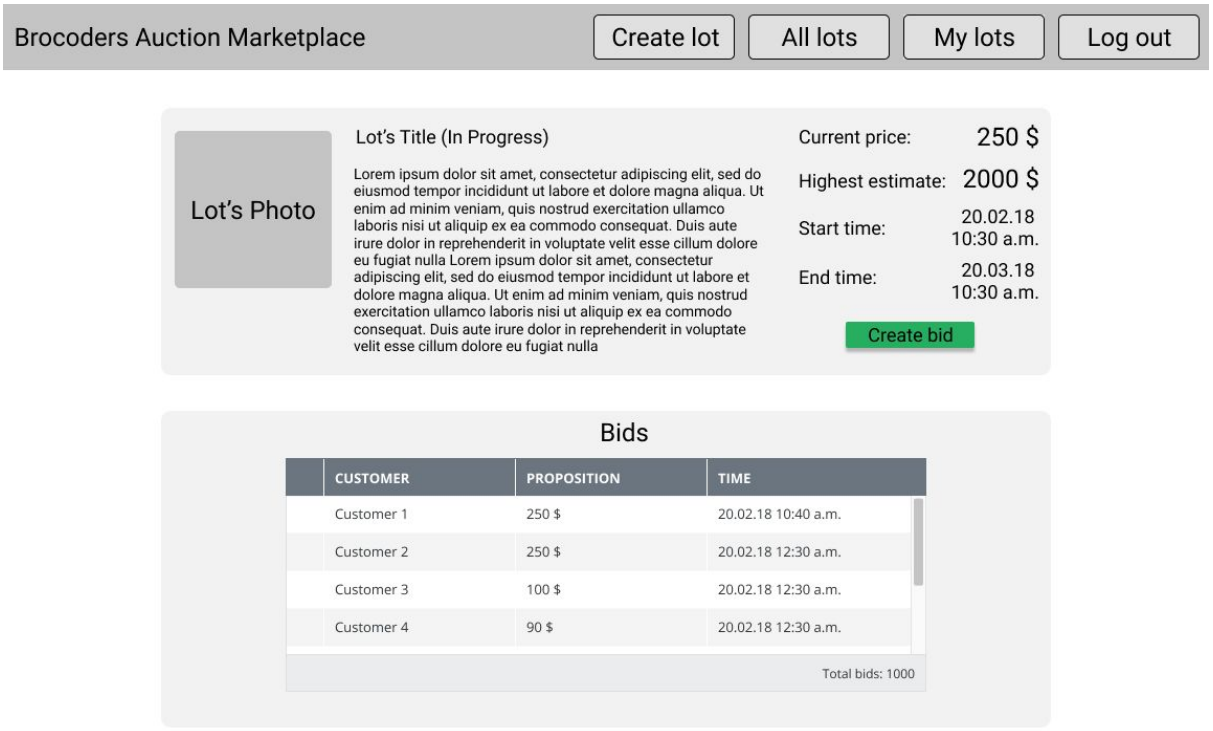


After submission lot will have *pending* status. That means that the *Start time* has not come yet. While lot has this status saler can delete or update lot. Update-page has the same UI as *Lot's creation page*. Lots with *pending* status should not be present on the All lots page.

When start time is coming, system should change lot's status from *pending* to *in Process*. That means that:

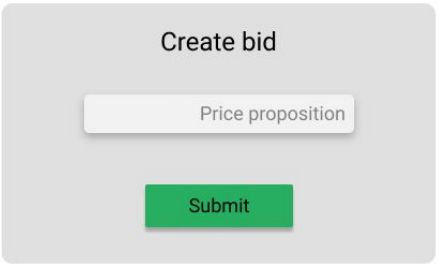
- all users can see this lot on the *All lots page* ;
- customers (any user other than the current seller) able to create bid under this lot;
- seller cannot update or delete lot;
- Bids rating table become visible on Lot's page.

Lot page (status: *in Process*):



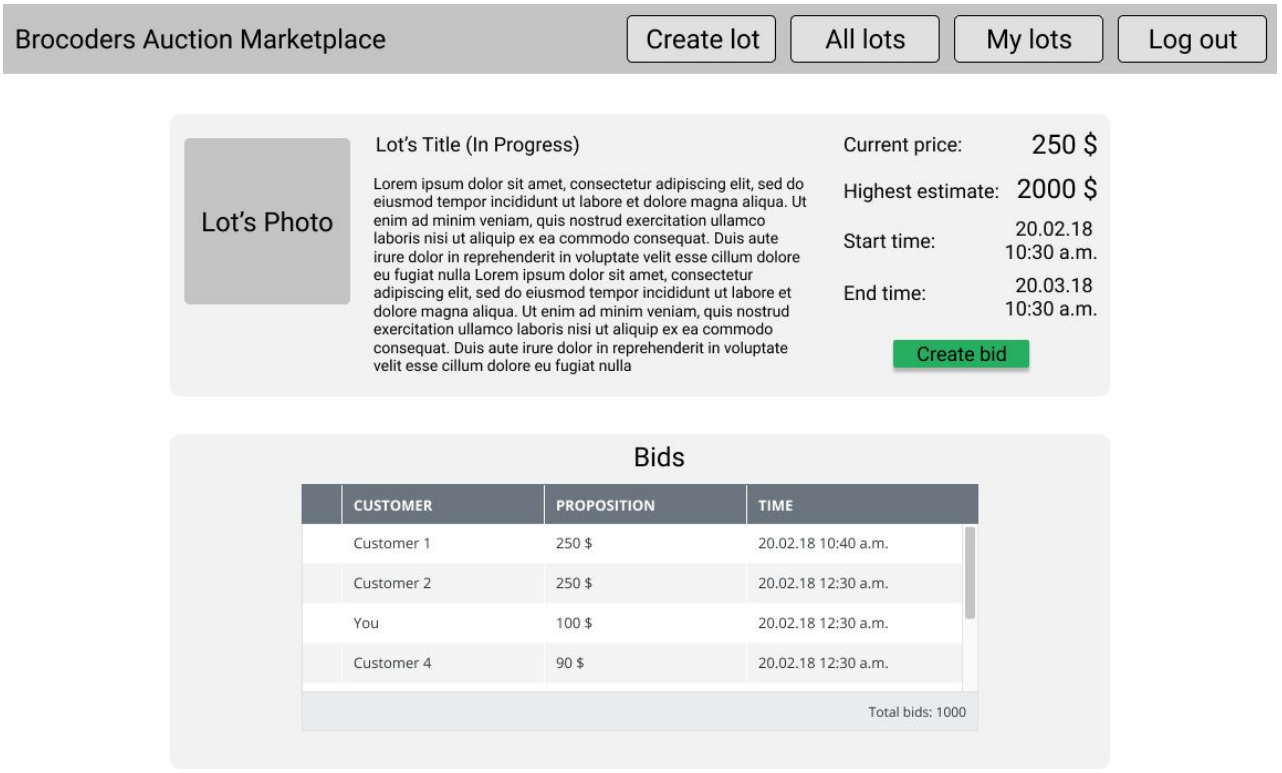
Bid creation:

If user wants to buy any lot that he sees on the All lots page, he should create bid - see *Create bid* button on the previous screenshot. This button redirects customer on the simple form:



When user create bid, it should become visible on Bids table, that on the Lot details page. **Important**:

1. Seller and customers must see new bids in the table of the lots detail page **in real time, without refreshing the page**;
2. Customers and seller **shouldn't see any personal data** of each others (for example name). Names in the Bids table should be replaced with "*Customer #"* or "*You*". See example below (please come up with defining numbers by your own, they are not necessarily sequence numbers):

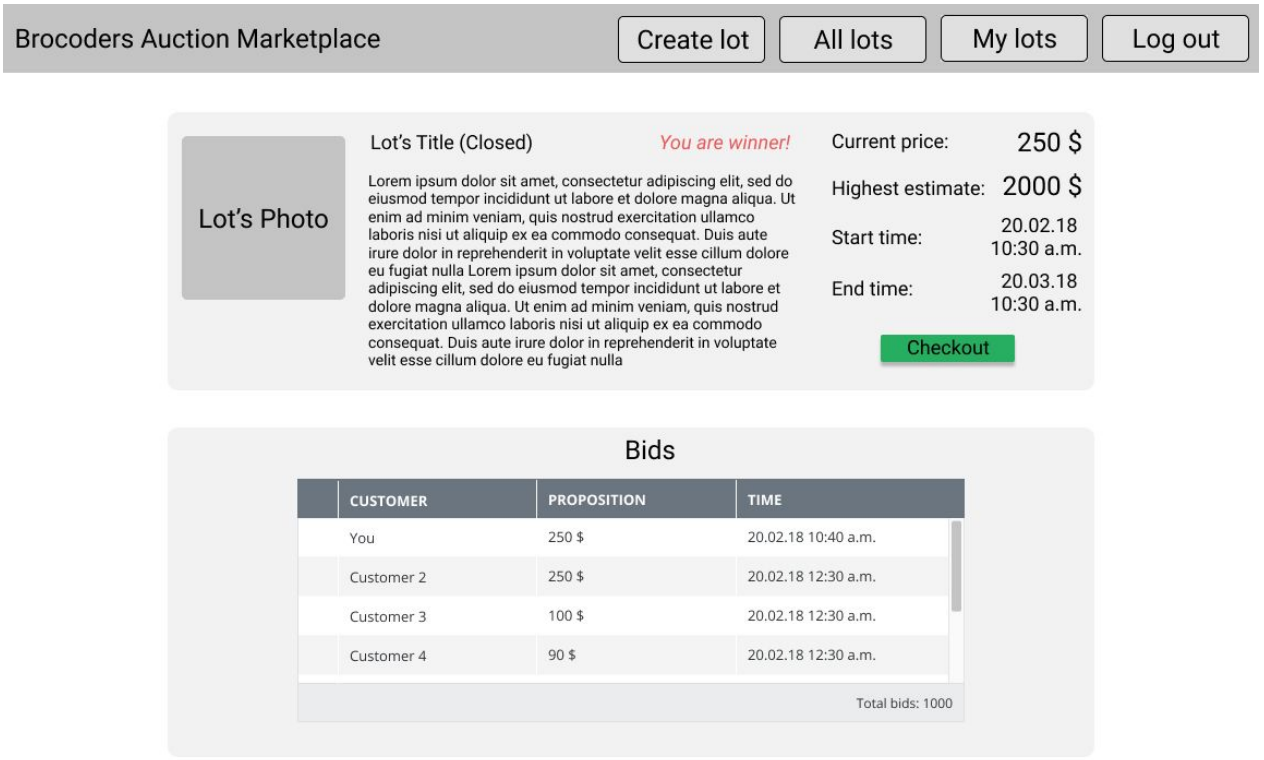


User wins if he proposed highest estimated price or *the End time* become and user proposed the max price on the Bids rating table. In case when multiple users propose the

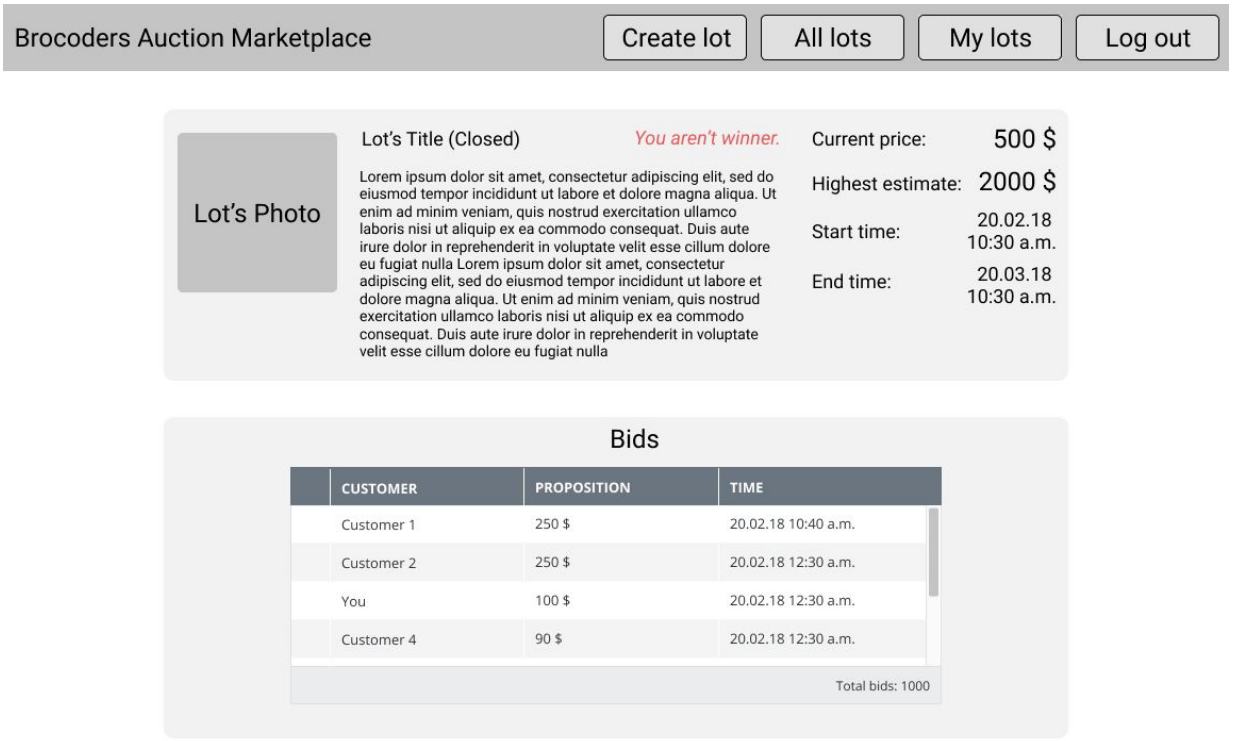
same price, compare the time when they create their bids and choose one that was created first. After that system should change Lot status from *in Progress* to *closed*. After that:

- customers cannot send bids for this lot;
- system sends email to the winner that he won lot and he can create an Order;
- system should allow winner to *Checkout* lot (show *Checkout* button on the *Lot's details* page, details in *Order flow* section);
- system sends email to the seller and notify him that the lot is closed, and about current price of lot (that become final);
- for lot participants (frontend logic):
    - if user won lot he must see "You are winner!" text near the lot title on the *Lot's details* page;
    - if user didn't win lot he must see "You aren't winner!" text near the lot title on the *Lot's details* page;
- system should hide this lot from *All lots* page.

If user is winner:

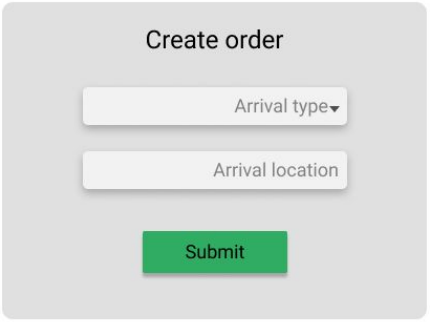


If user is not winner:



Orders flow

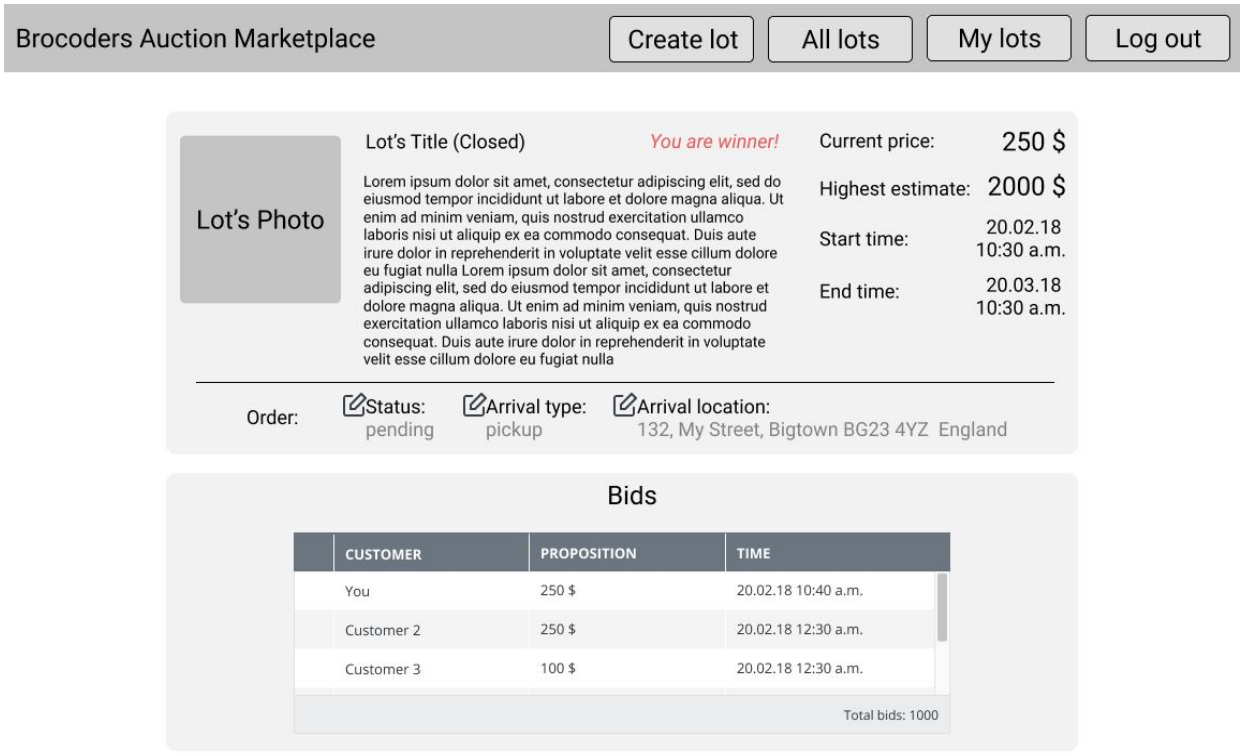
To get lot's product winner of lot should checkout product. It means that he should create Order - it's entity that represent in what way product should be delivered from the seller to the customer that won lot. After winner click *Checkout* button, he will be redirected to the *Create order* form:



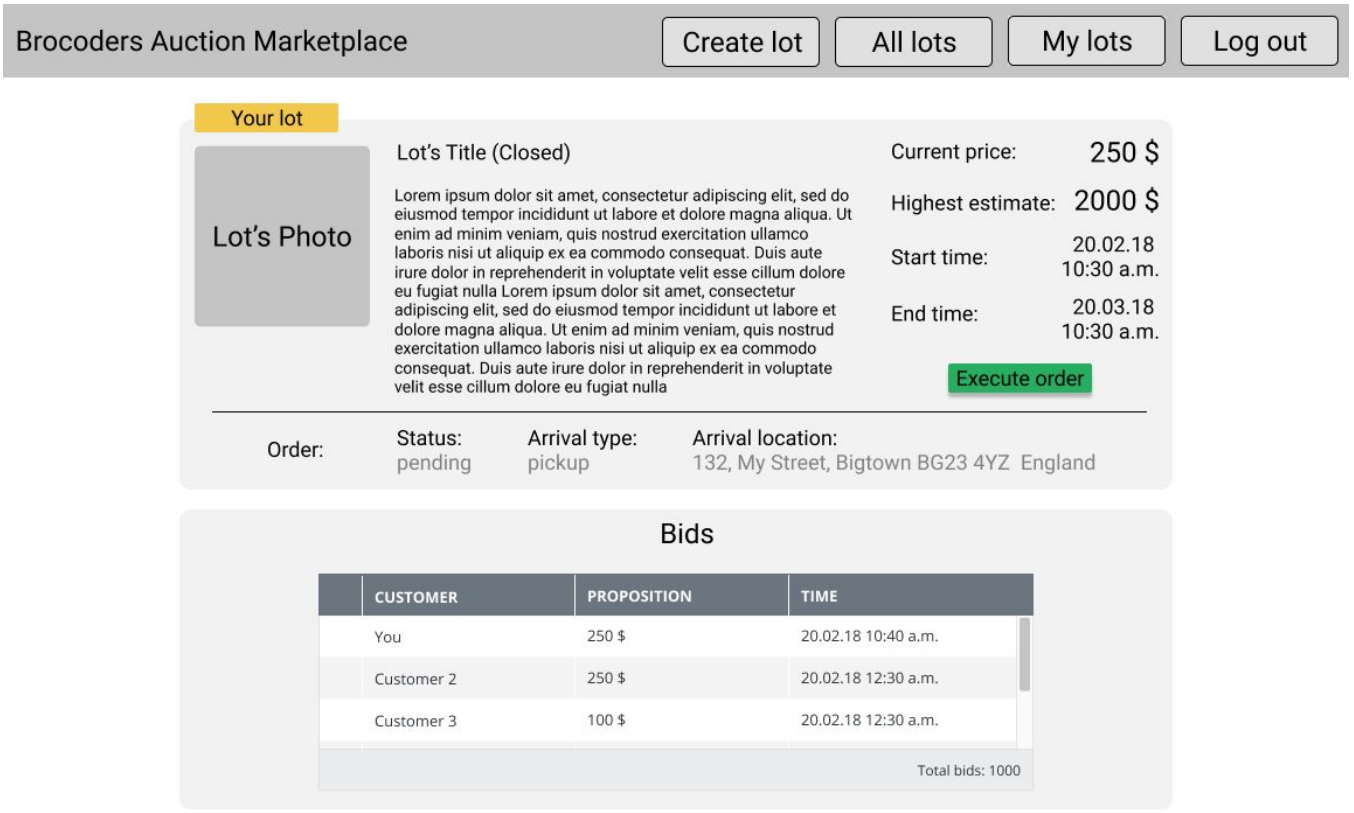
After customer press submit:

1. Order will be created and linked to the bid and lot;
2. System sends notification to the seller with the requirement to execute order (send it to the customer).

After creation order has pending status. When this status is active customer has ability to change order:

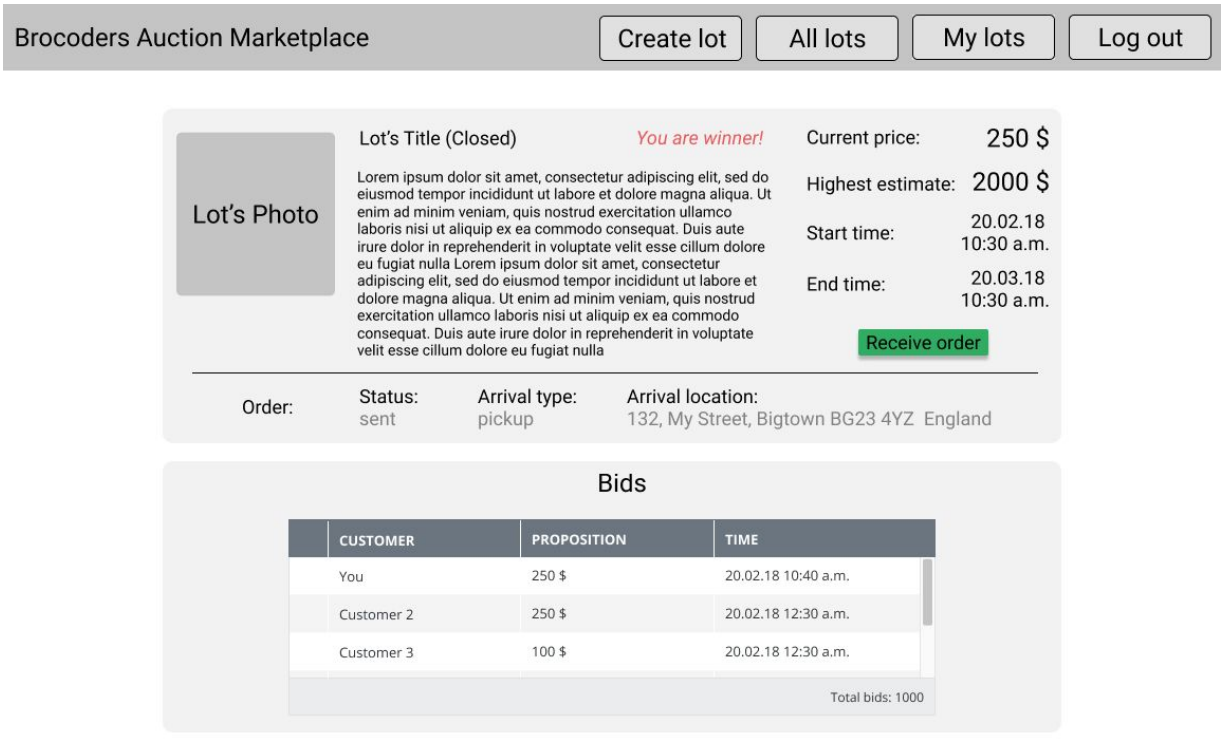


After seller send product to the customer he should press *Execute order*:

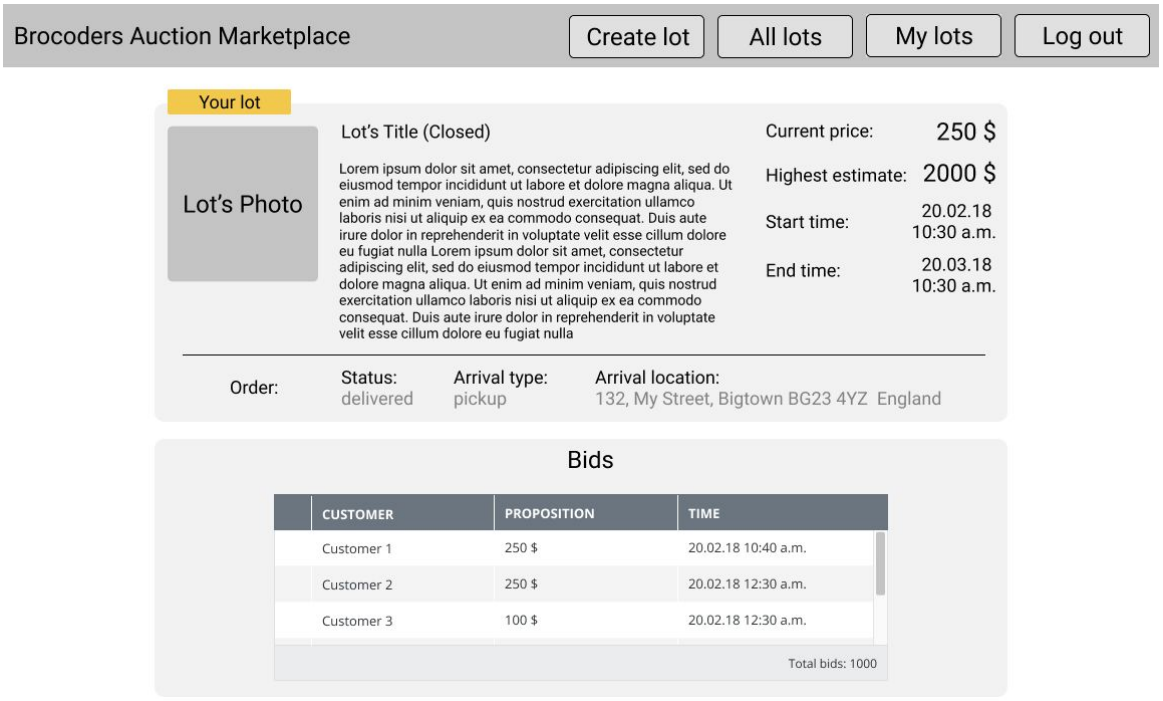


After that:

1. System changes status from pending to sent - that means that anybody can change order status;
2. System sends email to customer that seller change order status to 'sent';
3. System show *Receive order* button for customer. He should press it when he'll receive the order:



When user accept that he got the order system should change Order status to *delivered*. This is final step and nobody cannot do any changes with this lot, bids and order. Email notification should be send to customer and seller:



Mockups

If you want to see all prototyped site's pages, check out it via link:
https://www.figma.com/file/bFG0APn7Cfp782Ah9GuC9kSM/Brocoders-Auction-Marketplace

Technical requirements

Development process: your task implementation workflow must be based on **TDD methodology**, when you create tests first (red phase), then implement code (green phase) and after that - refactor existing code (refactoring phase)**.**

For tracking changes in your codebase, **use Git -** distributed VCS (version control system). **GitHub** repository is prefered, but you may use any other**.**

Technological stack:

- As main data storage you should choose one of next relational database:
  - PostgreSQL;
  - MySQL.
- Keep in mind that for some task you might want to use sidekiq (delayed job processing) and so - Redis, the NoSQL key-value database.
- For seeds creation you may use factory_bot gem, if you want. At least 5-10 seed should be created for each table.
- For image processing use carrierwave gem.
- Use device gem for handling authentication/authorization processes.
- For the real-time flows in application use WebSockets in Rails 5 with Action Cable.
- Your propositions, try to use something that become your development process easier and faster..

Testing details:

- **Required**: test should be implemented via *Rspec* + factory_bot gem + smth else (your choice).
- Check tests code coverage via SimpleCov gem. Discuss this metrics with your mentor.

Code quality:

- To check project code quality use Rubocop - it may be IDE plugin or any other service, also you may check it manually, via *rubocop* terminal command.

Database*:

- Feel free to change any attributes types to database specific. (for example PostgreSQL has built-in *email*, *json* types). Also you can change another attributes types - provide your propositions.
- Decide what default values you should add to specific attributes.

Development Process Iterations

Iteration #1 - Entity Relation Diagram creation

Create ERD that should describe:

- all entities, that were mentioned in Project description;
- relations between this entities;
- primary and foreign keys;
- entities attributes types;
- unique/nullable/indexed attributes.

Iteration #2 - User entities creation and Authentication flow

Create User entity. Implement all authentication flows **:**

- Sign up
- Login
- Password recovery process
- Logout

Iteration #3 Lots initial flow

Create Lot entity, implement all CRUD (create/read/update/delete) operation on it, described in lots flow description:

- list of lots for All lots and My lots (don't matter about lots where you take part as a customer on this iteration) pages;
- lot creation;
- lot update and deletion;
- don't forget about validations;
- other...

Iteration #4 Bids initial flow

Create Bid entity, implement all CRUD (create/read/update/delete) operation on it, described in bids flow description:

- bids creation;
- add lots where you takes/taken part as customer to My lots list;
- mastering bids table.

*don't matter about bids table realtime update during this phase.

Iteration #5 Handling of Lot statuses and Choosing Lot's winner

- Create job(s) that will handle lots statuses.
- Add validation related to changing lots status (means lot update, delete; creating new bids etc.)
- Implement logic of choosing lot's winner, don't forget about emails notifications.

Iteration #6 Realtime bid's table update

Implement real-time update of bid's table. Don't forget about bids sorting by price and date and about hiding customers names.

Iteration #7 Orders flow

Create Order entity, implement all CRUD (create/read/update/delete) operation on it, described in lots flow description. Don't forget about email notifications.