Find tutor platform

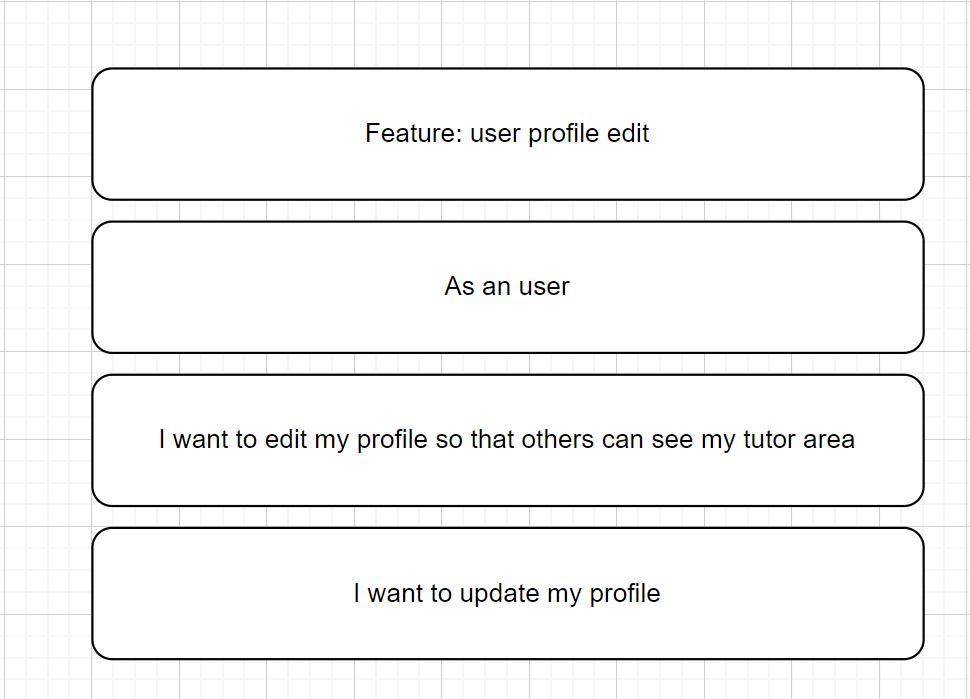
# Project background

The main customer need is: to build a web platform for students to find proper tutors. Based on a large demand of tutoring in this university of 55,000 undergraduate students, our customer want to build a web platform for senior student to post tutoring information and for junior students to seek proper teachers in the website. She plans to run this website as a non-profit information sharing platform.

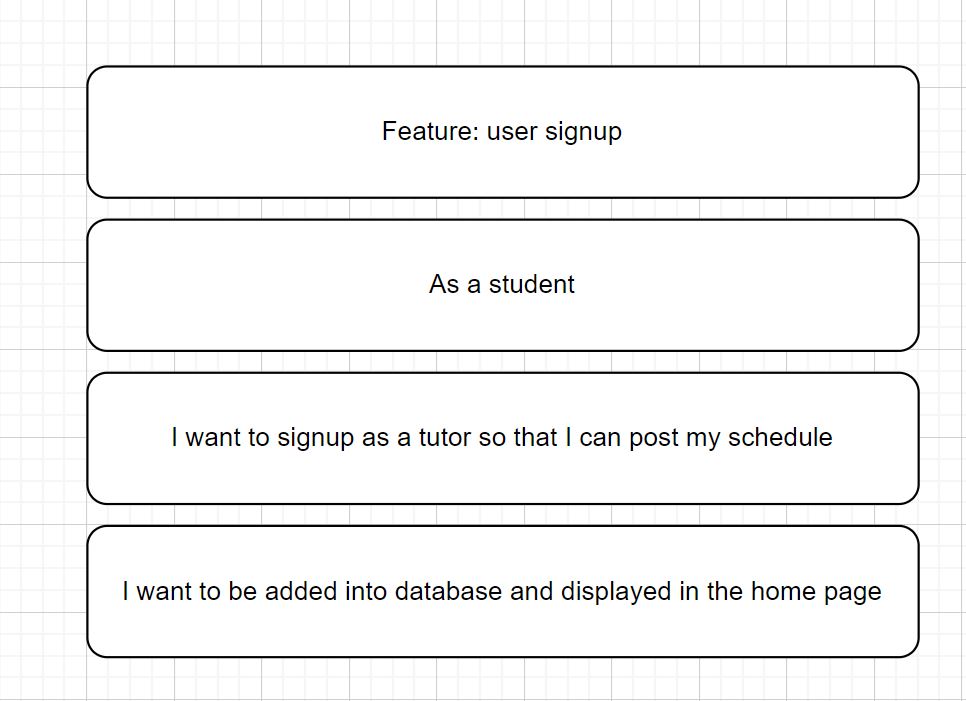
Based on these requirements, we implemented the project as a digital board for those tutors to post their information to match those potential students. Students could register on our website as tutor and post their available tutoring courses. The post includes title, tutor major, available date, self-description and price. Other students could search for tutors or posts that match their interests on our website. In the future, this application can be extended into broader fields like used things sales information, activities information center that anything happens on that physical board.

# User stories

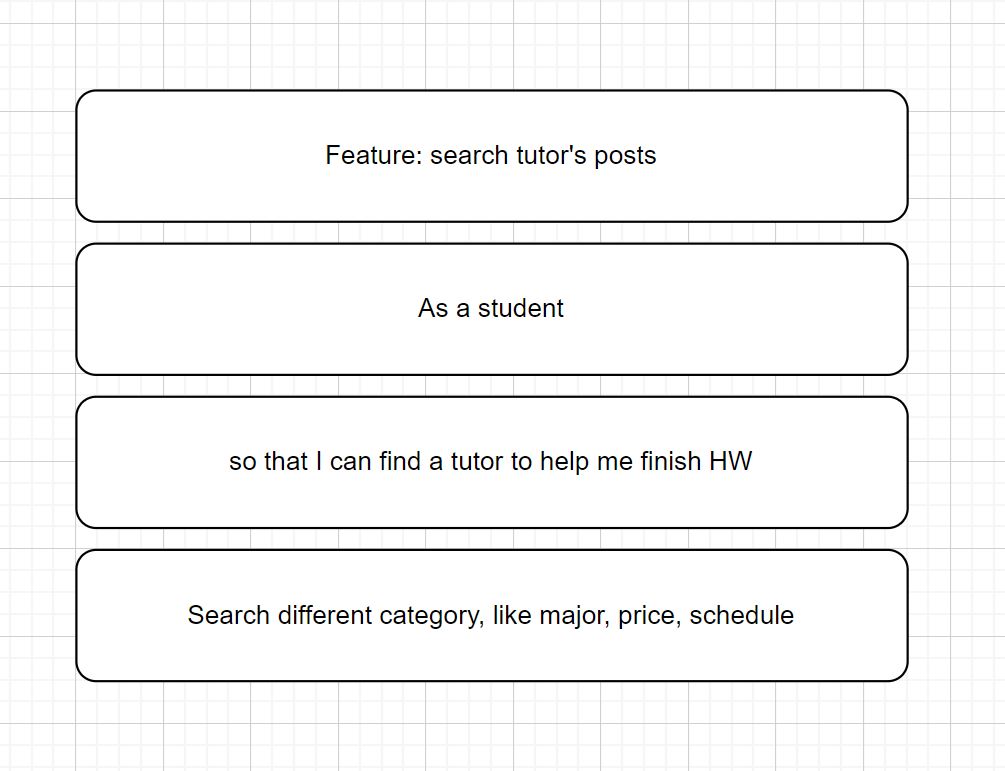
Story 1: user profile edit (3 points, finished)



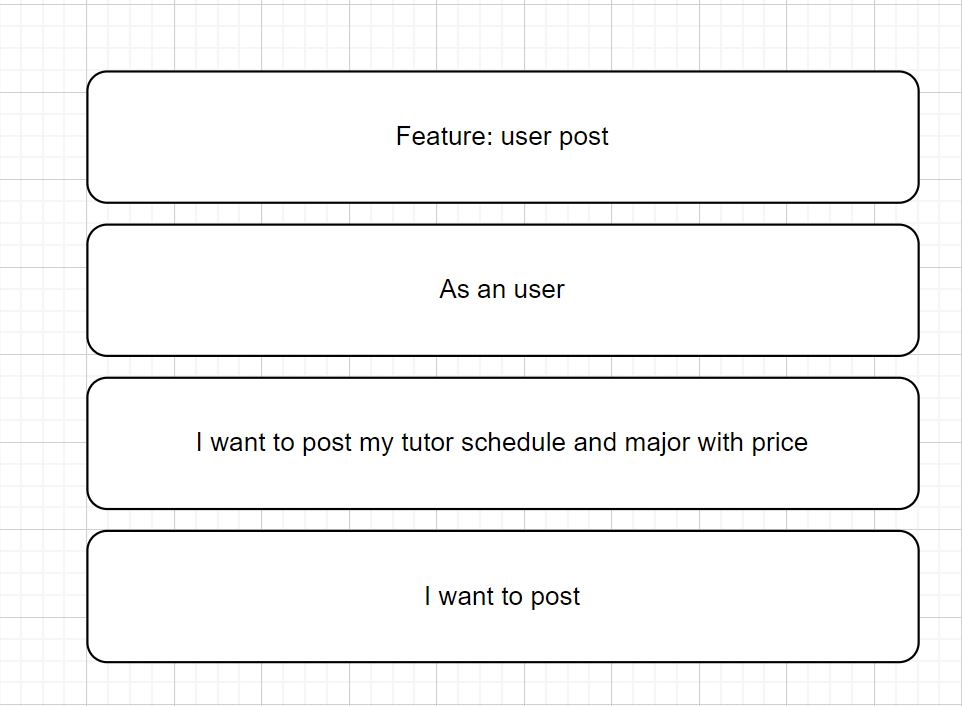
Story 2: user signup (3 points, finished)



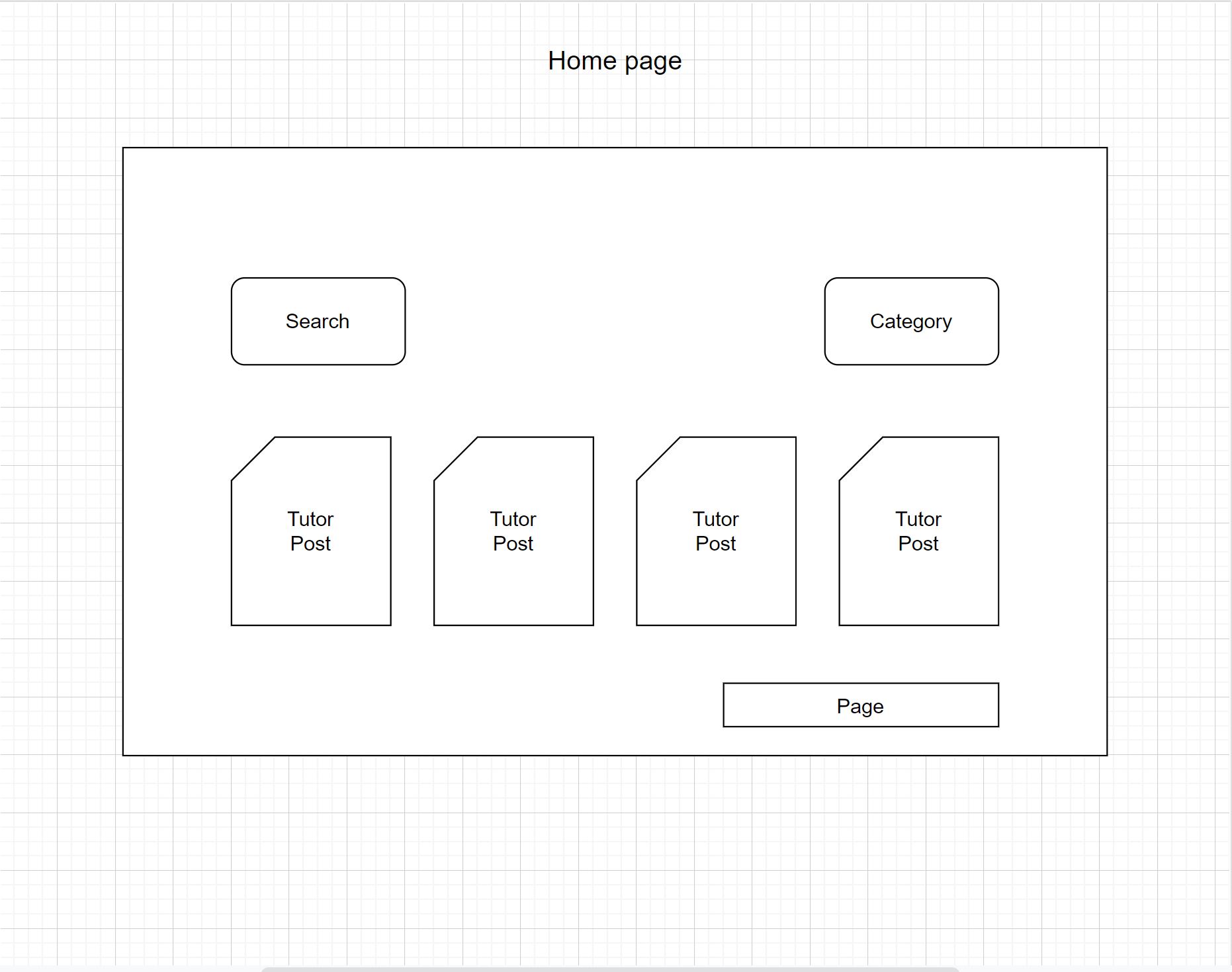
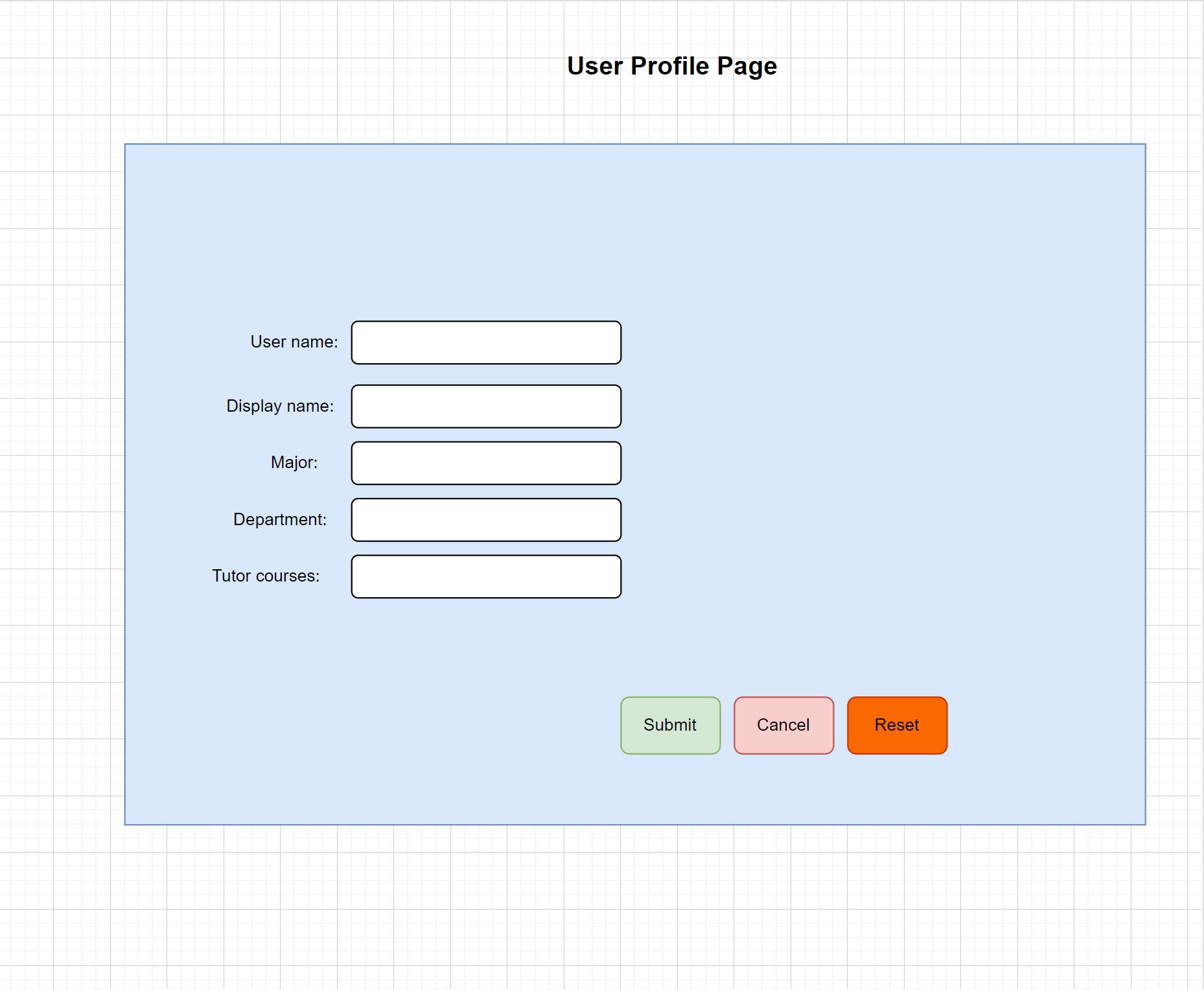
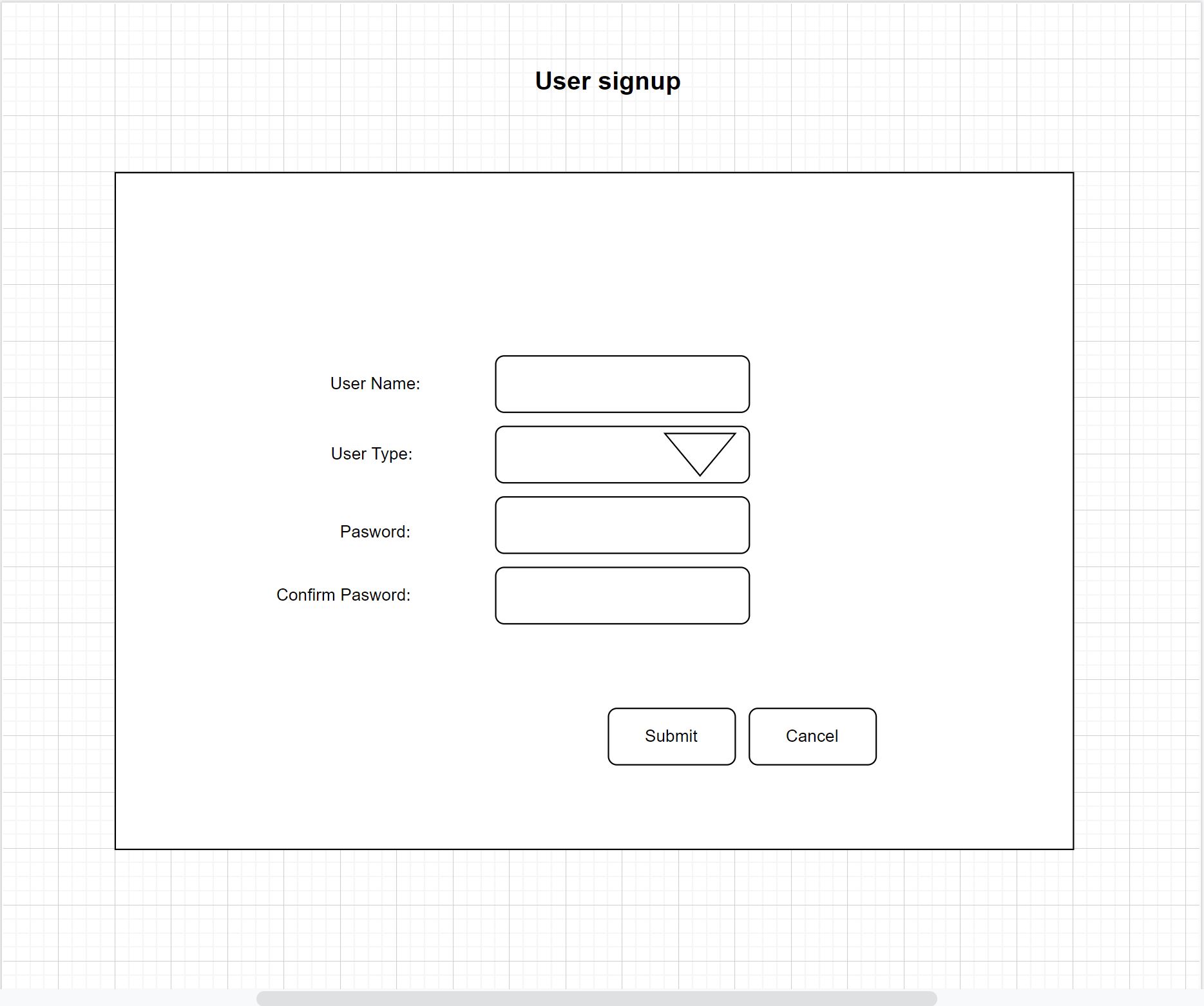
Story 3: user posts searching (3 points, finished)



Story 4: user post(3 points, finished)



The UI design



# 3. Team role:

* 1. Anmin Fang: Product Owner & full stack developer
  2. Dawei Xiang: Scrum Master &Back-end developer
  3. Yiming Li: front-end developer

Lei Zheng left our team halfway through.

# Iteratino summary

Iteration 0:

We met the client on 02/27/2022, and discussed the main function and requirement for the platform.

Iteration 1 (**finished 6 points**):

We met the client on 3/10/2022 and set up a few user stories to implement first. We showed our design for user model and post model. The client only required signup with username and password, and other detailed information would be filled when signing in. The client also said that post table should have columns as title, major, price, schedule and self-description.

user signup (4): to allow users to signup and create their profiles. We accomplished several tasks in this iteration, as database table design, backend model design and map to database and frontend implementation.

user profile edit (4): to allow users edit their profiles. We created a profile model, controller and view in rails.

We used rspec to develope test cases for user model and controller.

Iteration 2 (**finished 3 points**):

We met the client on 3/17/2022 and updated our progress. We showed the client the user signup and profile edit function. We discussed the next function of editing posts.

public post: to allow users to public their post. we added post’s model, view and controller.

Iteration 3 (**finished 3 points**):

We met the client on 4/8 2022 and updated our progress. We showed the client the post edit function. Client wanted us to add a search function at homepage and we discussed some detail about the search function.

add login in function: to allow users login in with their passcode.

Display posts: to show all the posts that tutor created.

We used rspec to develop test cases for post model and controller.

Iteration 4 (**finished 3 points**):

We met the client on 4/21/2022 and updated our progress. We showed the client the post searching function, and client wanted us to show the recent posts on the homepage.

search post: to allow users to search some topics in post. Thus, they could find the tutor they are interested in. We designed search controller, search function in model.

# Our BDD&TDD process

We developed our platform following BDD combined with TDD. At each iteration, we developed a few user process and user needs based on the user stories we finished at the beginning. Then we wrote test cases for these user stories. We first implemented our code to pass the test. Later we refactored our code to make it cleaner and more efficient.

BDD helped us capture what we should do in each iteration. Following the general target, we wrote our test cases to guide the detailed code implementation in TDD. Developing in TDD alone forces us to change the code every time we add new test implementation, while if in BDD alone, it’s hard to check special test case and we don’t have specific target. So we adopt BDD + TDD.

1. Discuss your configuration management approach. Did you need to do any spikes? How many branches and releases did you have?

# Configuration management

We didn’t use some specific configuration tools. We kept our configuration files in the config folder. We had two branches in github, each for one developer.

# Issues we met

We had several issues

Deployment:

1. The development database is different from the production database. So we configure database again in Heroku.

AWS cloud9:

1. We have already used cloud9 before, so we didn’t meet problems this time.

Github:

1. We have already used github before, so we didn’t meet problems this time.

# Other tools

We used SimpleCov to analyze our code. SimpleCov automatically test our project and generate test results reports, thus we could easily know the coverage across our test suites.

# Deployment scipts

1. Git clone our repo <https://github.com/anminfang-tamu/tamu-find-tutor>
2. Install rails
3. Install postgresql: sudo apt install postgresql postgresql-contrib
4. Run database migration: rails db:migrat
5. Add heroku remote: heroku git:remote -a <https://github.com/anminfang-tamu/tamu-find-tutor>
6. Deploy to heroku: git push heroku main
7. Or deploy locally: rails server

# Other links:

github: https://github.com/anminfang-tamu/tamu-find-tutor

heroku: https://tamu-find-tutor.herokuapp.com/

pivotal tracker: https://www.pivotaltracker.com/n/projects/2556191