SW Engineering CSC648/848 Fall 2019

GatorTrader

Team 02
Sean Darryanto - Team Lead, Backend Lead | sdarryanto@mail.sfsu.edu
Aaditya Chokshi - Frontend Lead
Momo Sun - Frontend SWE
David Chung - Frontend SWE, Github Master
Christopher Smith - Backend SWE
Adam Bea - Backend SWE
Kam Khai - Fullstack SWE

Milestone 2
10/17/2019

Revisions

October 17, 2019	Initial Draft

Table of Contents

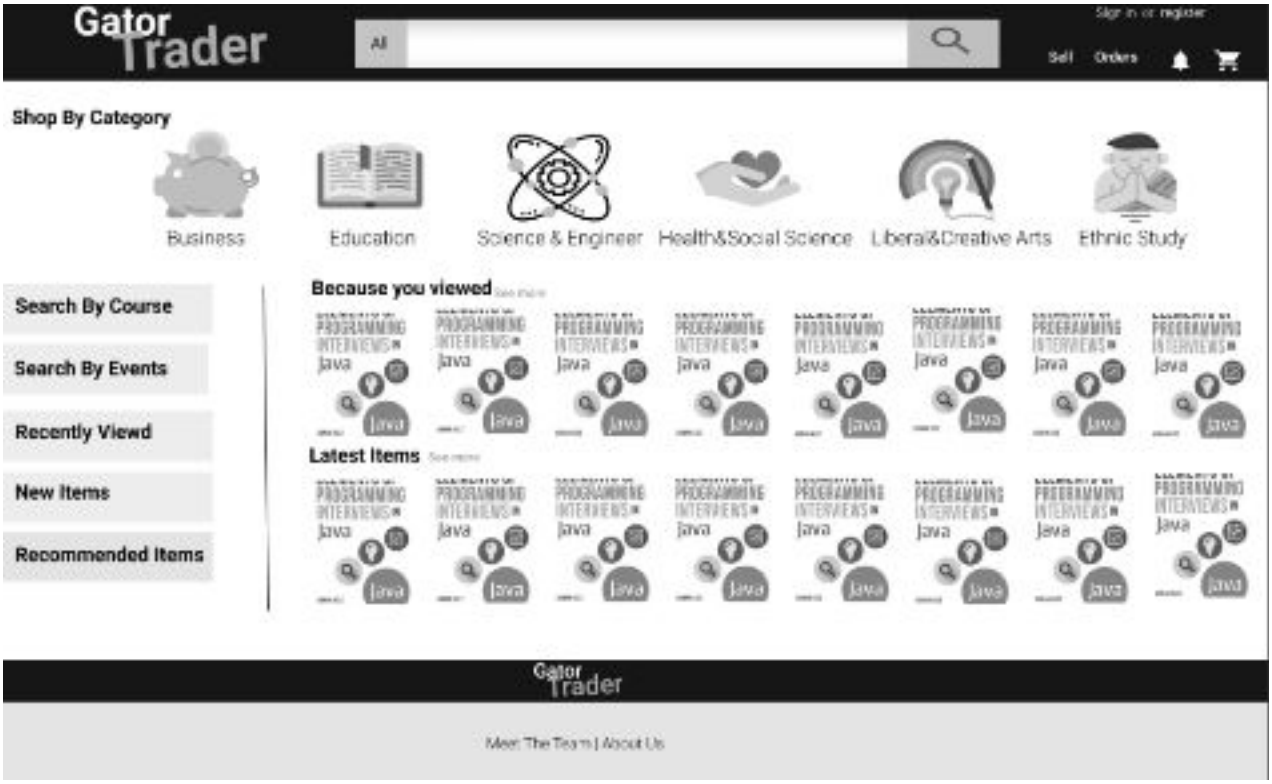
Functional Requirements

1. Guest users shall be able to browse through the website without logging in.
2. Guest users who want to register shall need to be verified by SFSU email.
3. Guest users shall not be able to contact the sellers.
4. Guest users shall be able to contact the administrator.
5. Guest users shall be able to create an account.
6. Registered users shall be able to contact both administrators and sellers.
7. Registered users shall be able to log in their account by username/ email and password.
8. Registered users shall be able to reset their password if they forget.
9. Registered users shall be able to create profiles from the user side.
10. Registered users shall be able to create listings and posts on the website.
11. Registered users shall be able to modify the post.
12. Registered users shall be able to upload pictures of the items they want to sell.
13. Registered users shall be able to download any items they purchased on the website.
14. Registered users shall be able to choose to pay either through the website or in person.
15. The administrator shall be able to postpone the posting if they violate the policy of the website.
16. The administrator shall be able to access the database.
17. The administrator shall be able to contact the sellers directly.
18. The administrator shall be able to reply to messages and emails from Guest users.
19. The site shall present the Terms, Conditions and Privacy Policy during the registration.
20. The site shall be searched by class' name, the professor's name, author, title, category, and keywords.

UI Mockups and Storyboards

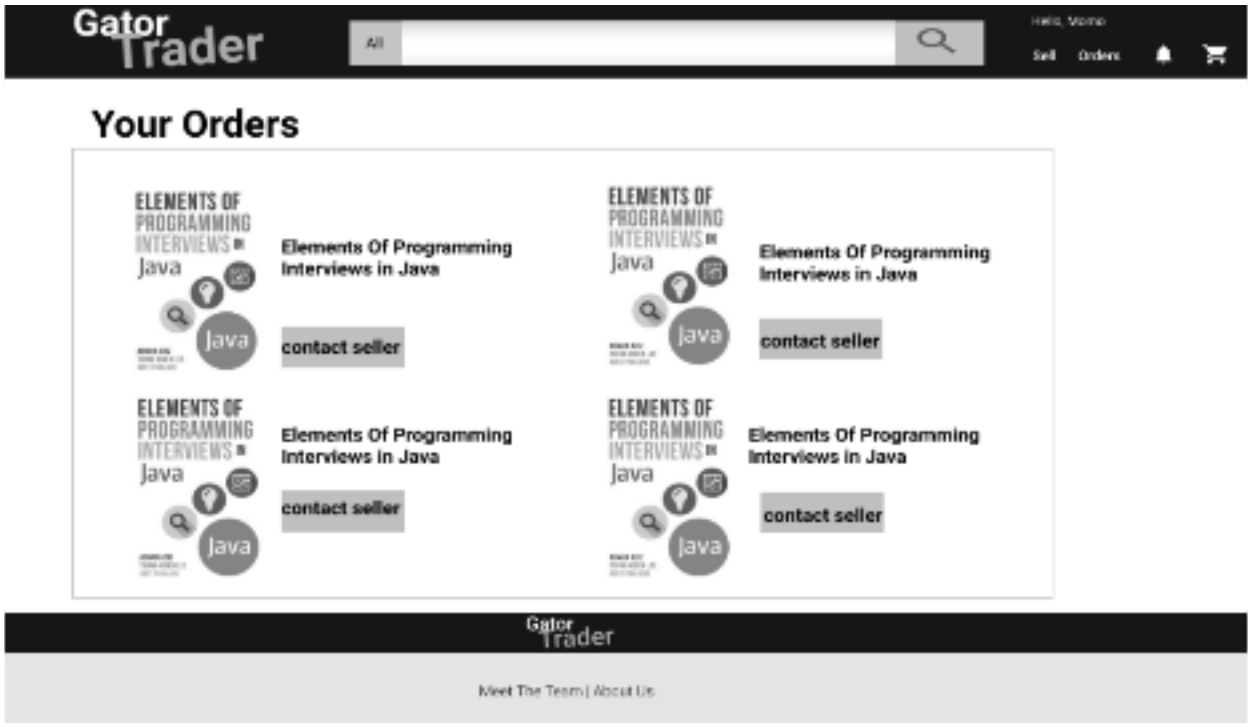
Sign in and register page which allows guest users be able to create an account with sfsu email and registered users be able to sign in with sfsu email.

Landing page allows guest users be able to browse through the website without logging in and search by category,course, events,recently viewed, new recommended items.

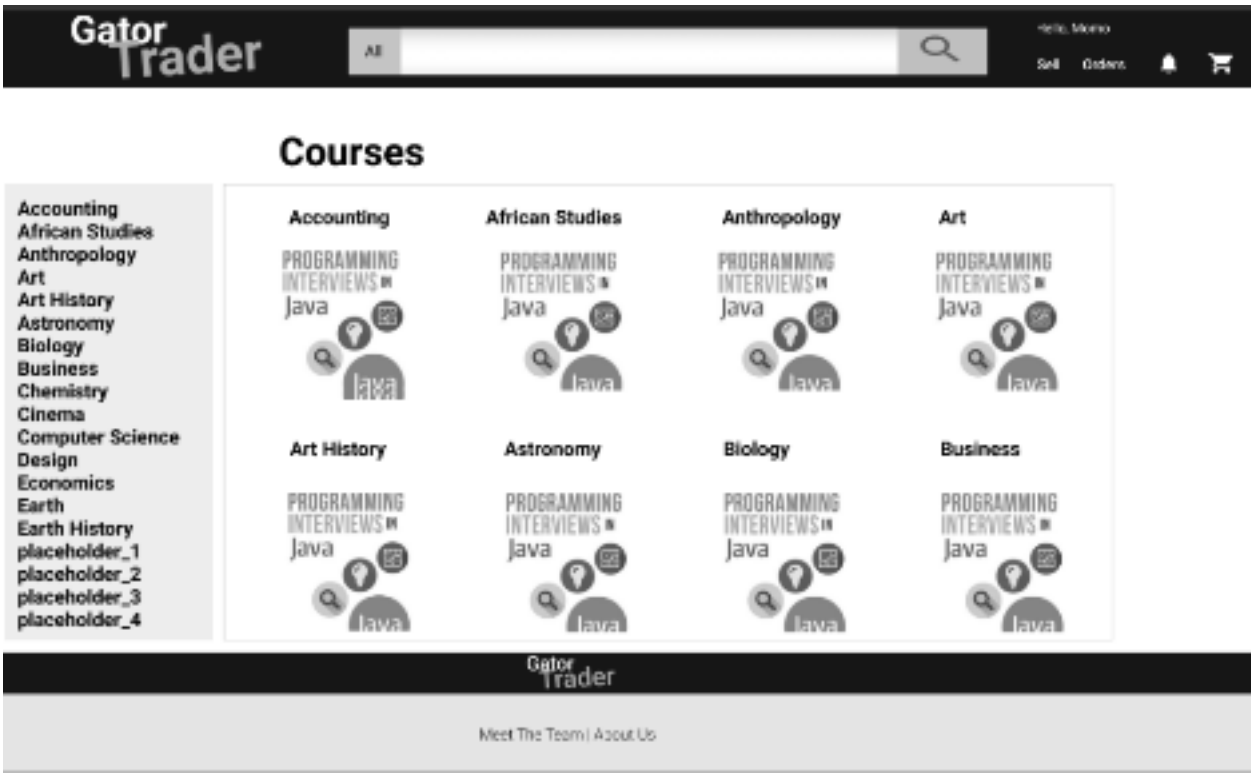


Searching results page will list the results after registered users do a search. This page also allows registered users go back to home page by clicking the gator trader and check their orders.

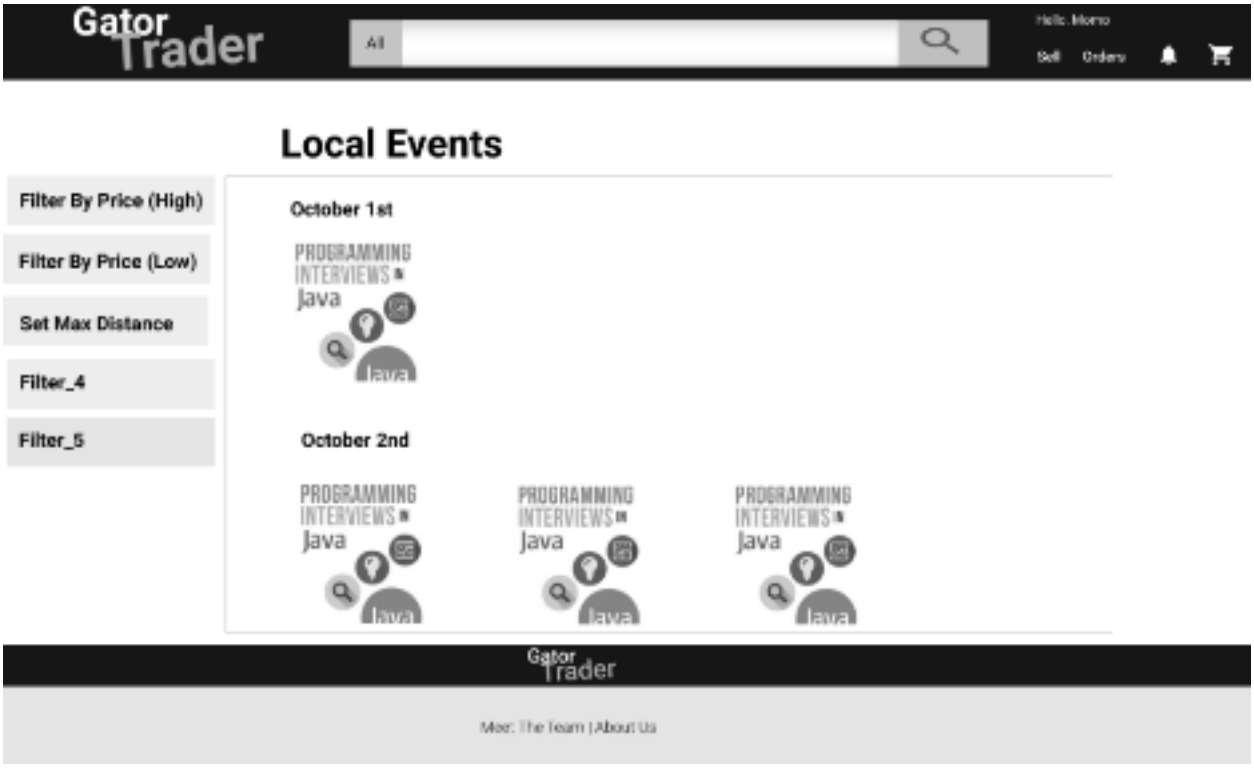
Your orders page will list the orders registered users purchased and allow them to contact with sellers.



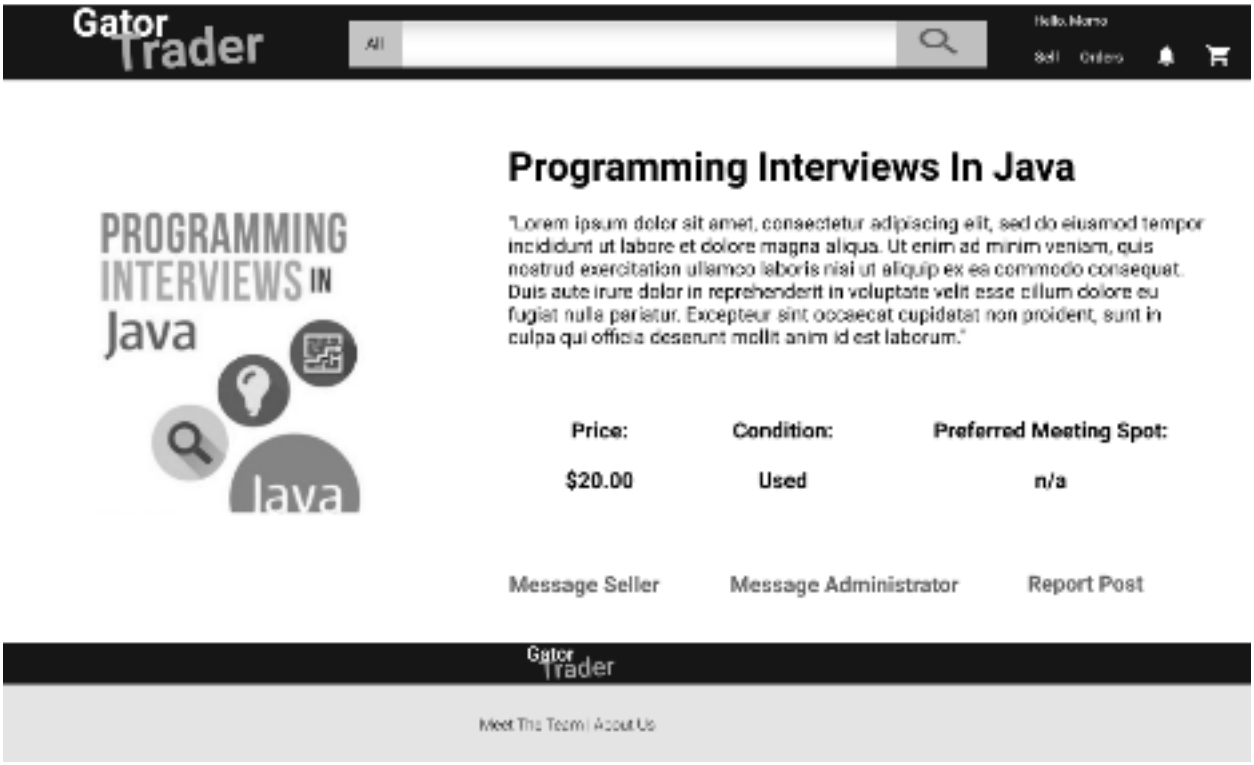
Courses page will list the courses after registered users do a search.



Local events page will allow registered users to check out the days and local events, and they sort the events by price, distance and etc.

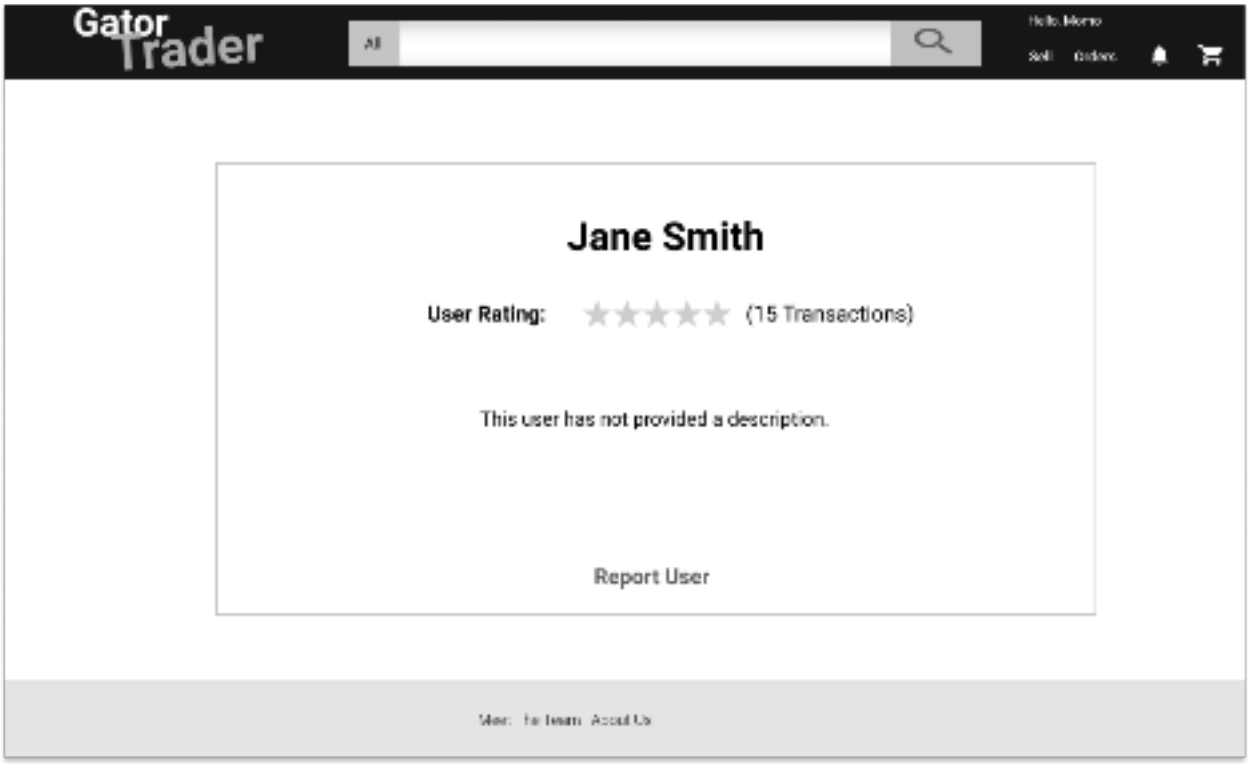


Product description page will show the brief introduction, price, condition of the product. Registered users can message the seller, administrator and be able to report the post.

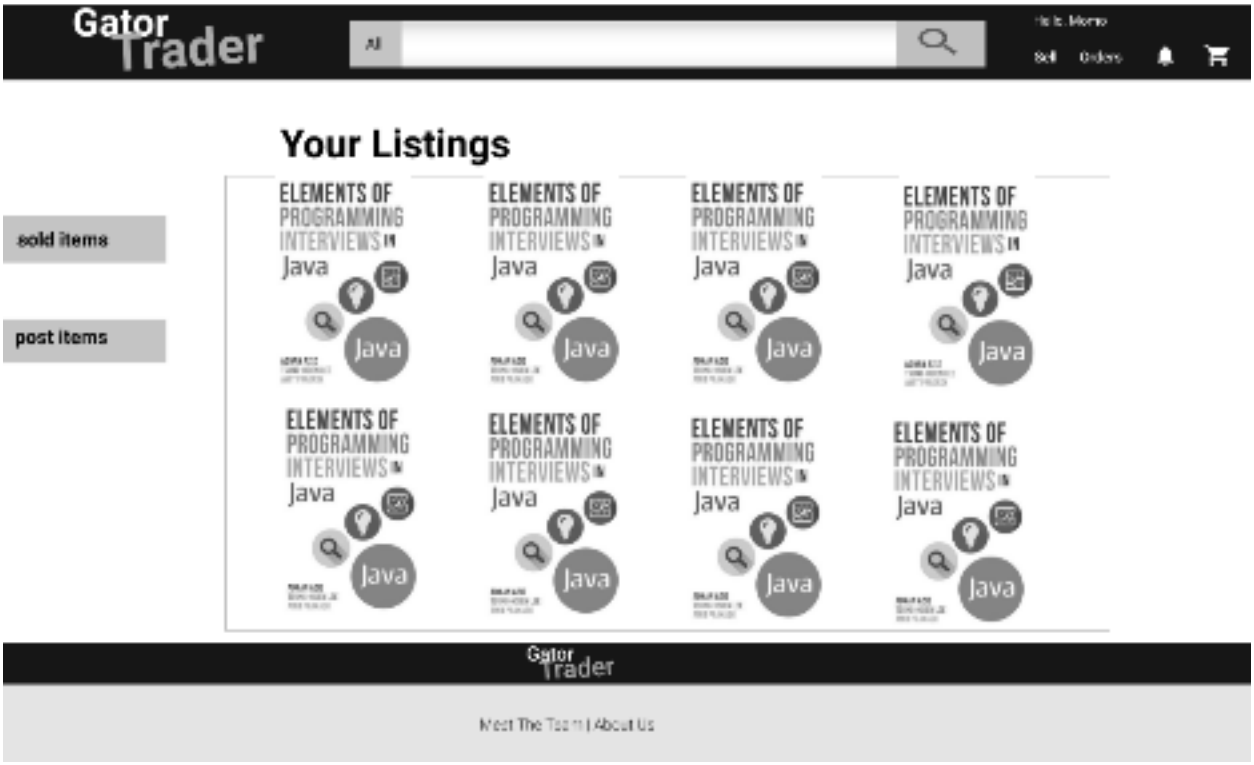


Chat page will show the messages exchanged between registered users and the sellers.

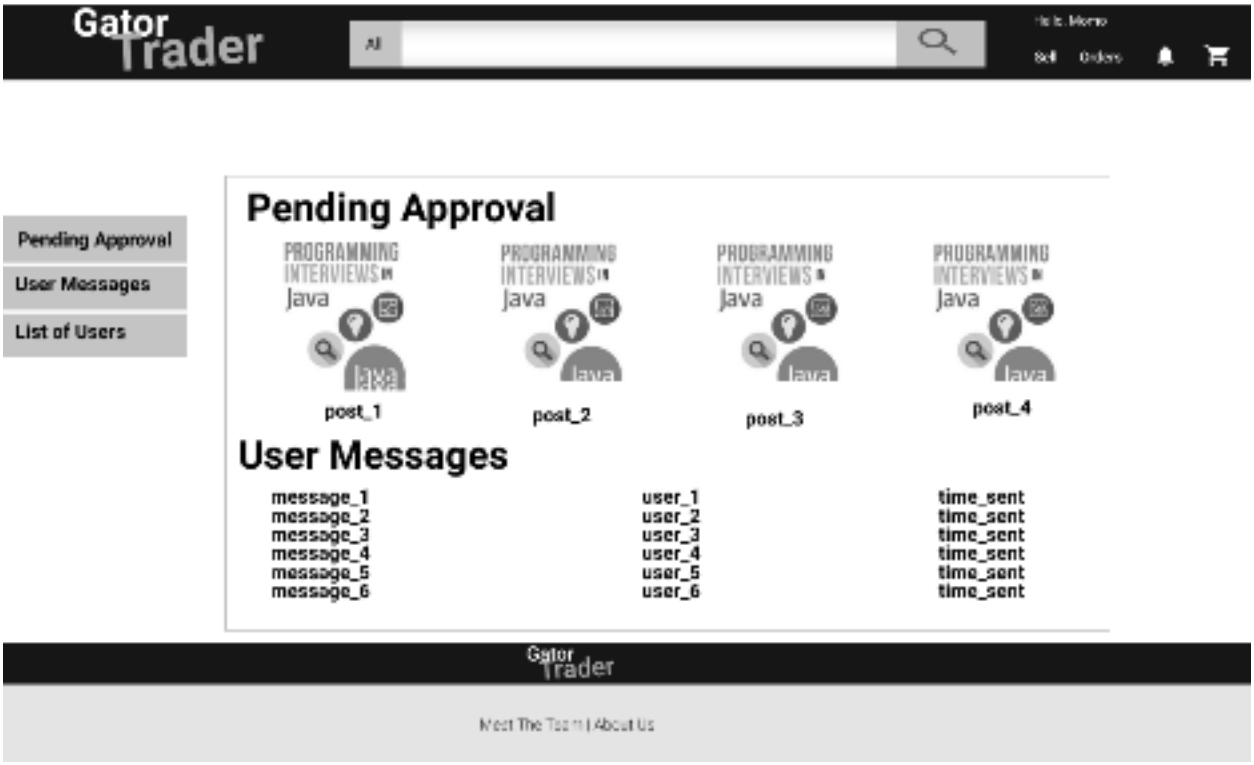
Information page of a seller will display seller's name, user rating and allow registered users to report.



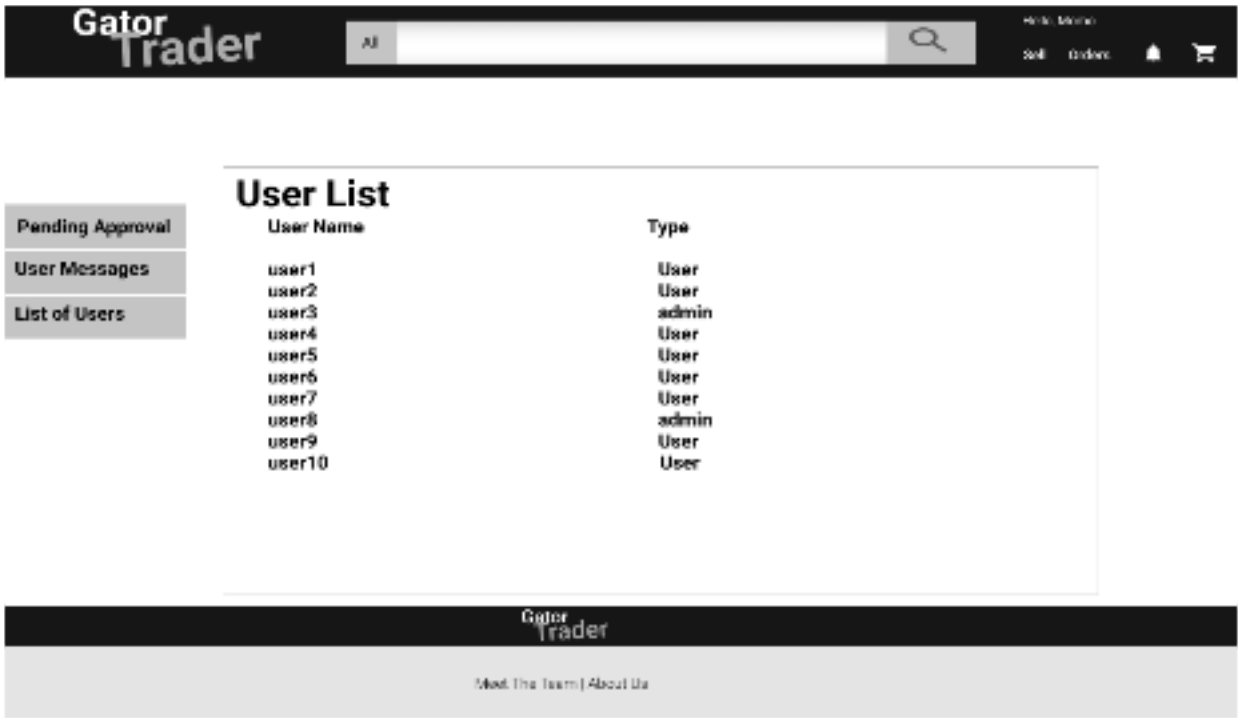
Your listings page allows sells to check their own listings, track sold items and be able to post new items.



Admin page will show the information of pending approval and users' messages .



Admin page will show the list of users.



Search Results

Your Orders

Sign In

Landing Page (Start)

Your Listings

Search By Course Results

Product Page

Search By Local Events Results

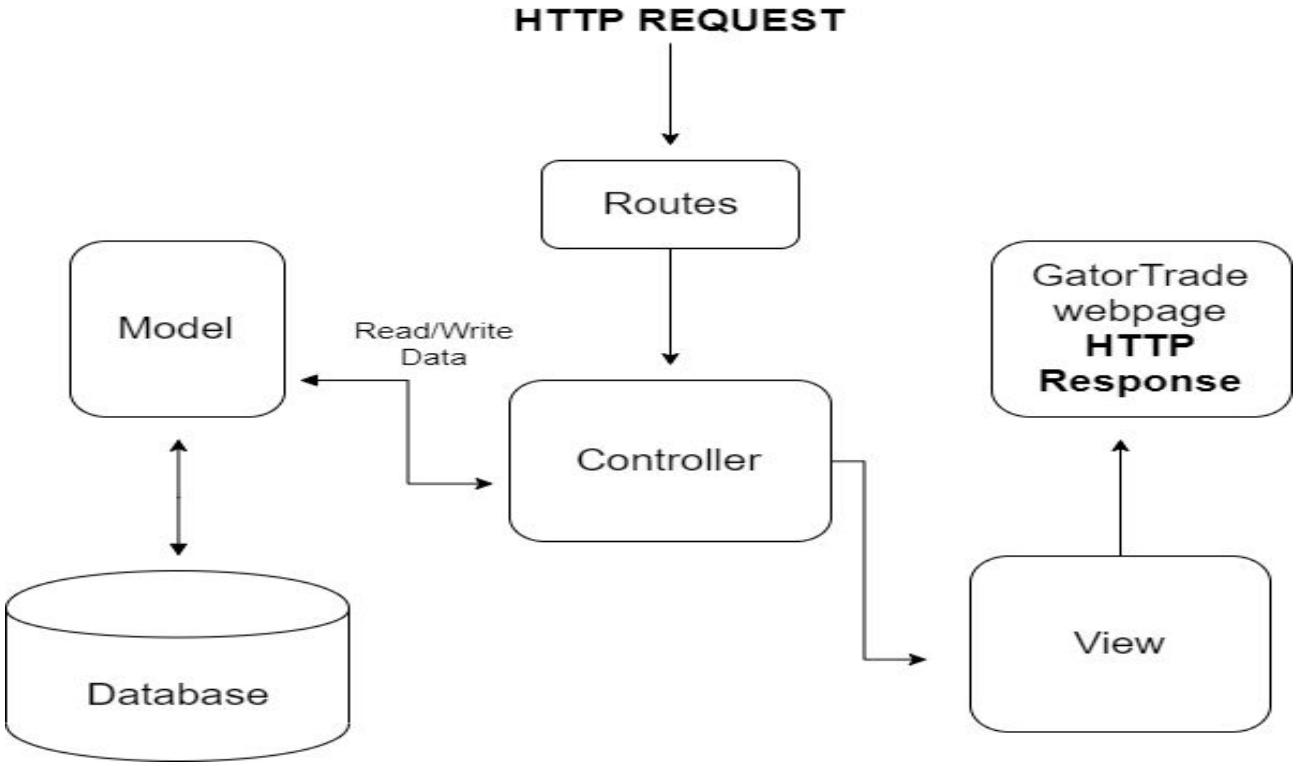
Message Seller

Seller Page

Recommended Items

Admin Pages

High-Level Architecture, Database Organization



The diagram above illustrates the MVC pattern that we will be using Flask/Python when developing the GatorTrader web app. In this diagram, when user requests to view a page by entering a URL, the application tried to match that URL to the routes which are associated with the Controller. Within the Controller, the Models are used to retrieve all the requested data from a MySQL Database, place the data in an array, and loads the View, passing along the data structure. The View will then access the data and uses it to form the HTTP response that the controller will send to the client display screen.

Database Organization

User: The user data model for all users in the system.

- User_id (Primary Key)
- Username
- Name
- Email
- Password (encrypted)
- Is_admin
- Major (nullable)

Listing: A post about an object/service a seller wants to offer. Listings shall be viewed by users, and must be approved by an administrator. Each listing shall have preferred locations to meet at, designated by the creator. They can be deleted by either the creator or an administrator, but can only be edited by the creator.

- Listing_id (Primary Key)
- Title
- Description
- Category
- Type (object/service)
- Price
- Thumbnail
- Created_on
- Last_edit
- Created_by (Foreign key)
- Approved_on
- Approved_by (Foreign key)

Location: A place of safe locations created by the Buyers and the Sellers can choose to meet at to conduct transactions. Listings shall have preferred locations designated by the creator.

- Location_id (Primary key)
- Description
- Thumbnail

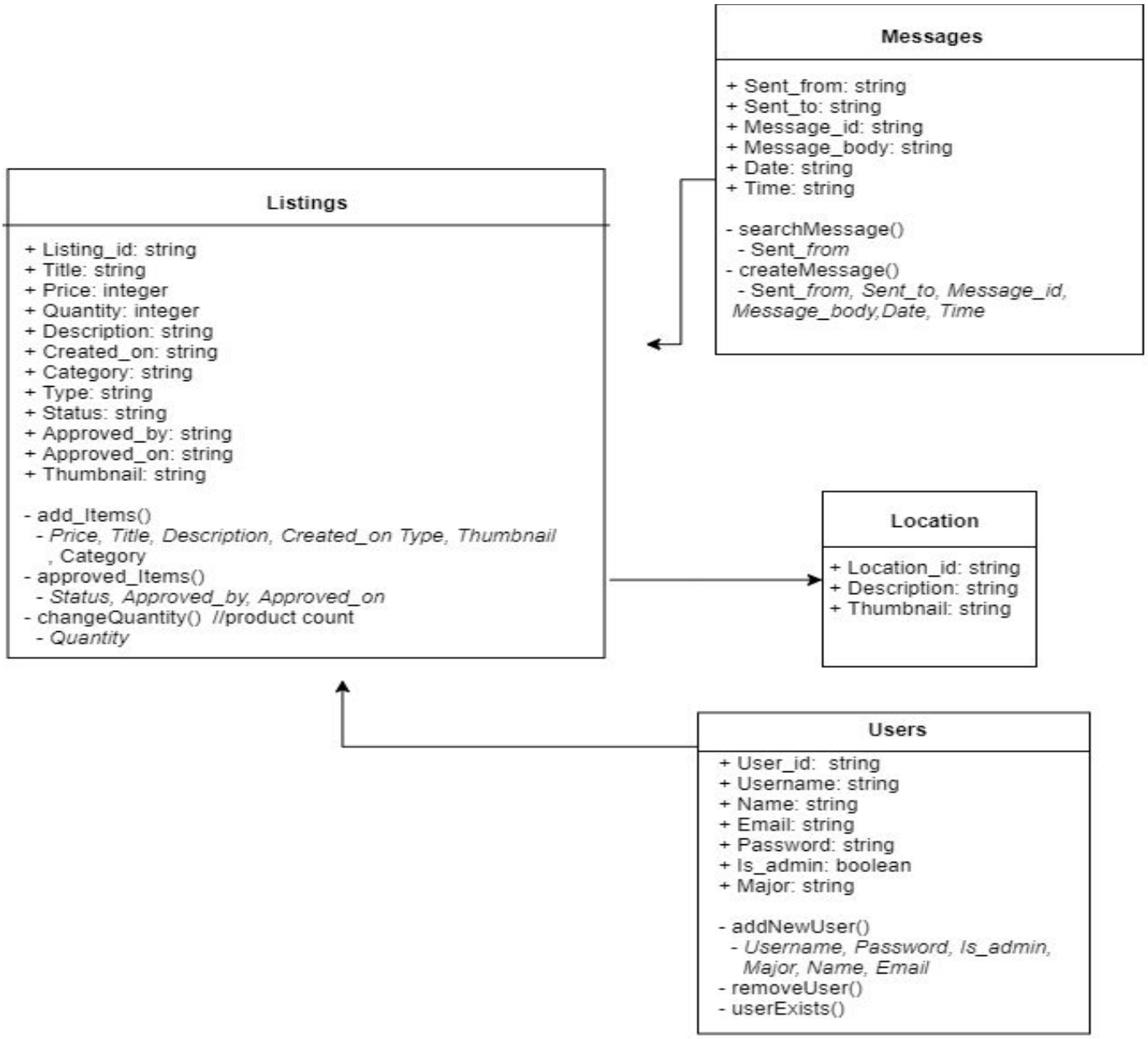
Message: A message between two users, timestamped.

- Message_id (Primary key)
- Sent_by (Foreign Key)
- Sent_to (Foreign Key)
- Message_body
- Timestamp

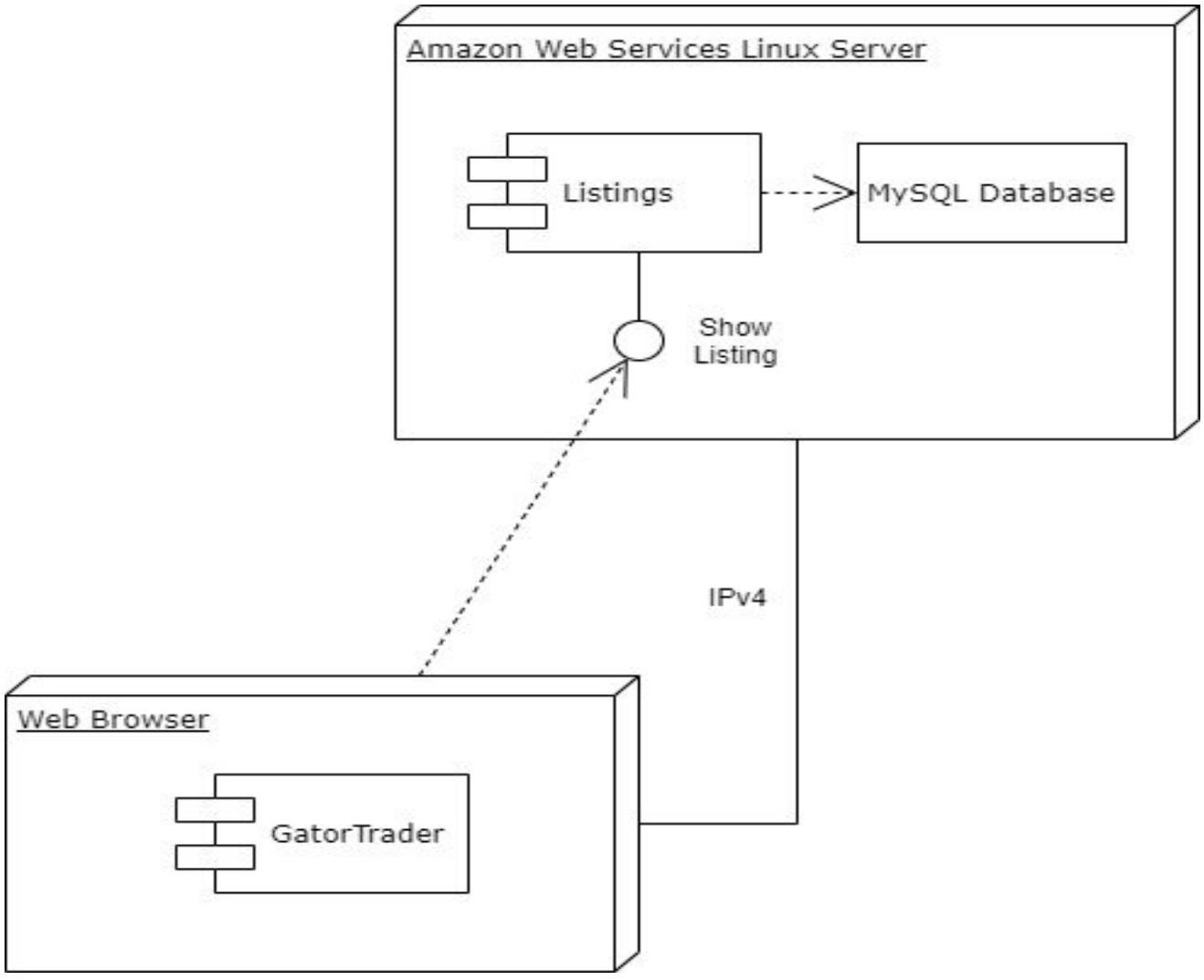
Media Storage:
We've decided to store media outside of the database and in our filesystem as it's a much cleaner and faster system for querying and overall management. The only form of media that we see ourselves collecting is photos, and we'll ensure to use proper relative paths.

Search and Querying:
Searching through a database is very simple. We have a few options to move forward, whether it be using the standard SQL library with stringified SQL queries or to use something like SQLAlchemy with Flask, in which we'll create models and use those POJO models to interact with tables in the DB. Both are fine, but typically we'll be using queries like:
" select * from listings where name LIKE %listing_name% and category=category_type "
" select * from listings inner join users on listings.Created_by=users.user_id where user.user_id=user_id_provided "

High Level UML Diagrams



UML Component and deployment diagrams



Identify key risks

Skills: While the team is passionate about the project, there is a level of overhead that's required to set the overall project up appropriately. While this isn't necessarily a huge concern, this will need to be taken into account when milestoning and scoping timelines. In order to mitigate the impact of this risk, we'll make sure to use our weekly meetings effectively for brown bags and workshops to ramp up newer engineers.

Bandwidth: The whole team has definitely felt the toll of midterm season being in full effect and as such, the bandwidth of each student has minimized drastically. This can be classified as a timing issue as well, but I think it goes past lack of time. This is a tougher issue to solve as the issues comes and goes depending on the time of the semester. We'll mitigate the impact of this issue by frontloading our efforts as much as we can to ensure that slowed down progress during busy times won't affect overall progress.

Project Management

We currently utilize Trello for all of our taskifying needs. While large tasks can be created and assigned by the team lead, each engineer has the option to make subtasks based off their assigned task (think epic, and story). The team meets every Wednesday at 6pm for an hour to discuss upcoming tasks, hold workshops to discuss architecture, brainstorm design and more. We also utilize Slack for all communication needs, the frontend team has their own pod channel, backend has theirs and we have a unified team channel as well. This is great because it allows both teams to operate independently while providing a means to collaborate all together.