1. **Project**

Team:
Si Shen
Yang Jiang
Albert Dayn

Title:
CampusRecycle

Vision:

A platform for CU students to post products they want to sell and organize meetups with potential buyers

Description

The main goal for this project is to make the useless resources (like old bikes or other products) recyclable and available among students and staff. This will be a platform for CU students to post products they want to sell and organize meetups with potential buyers.

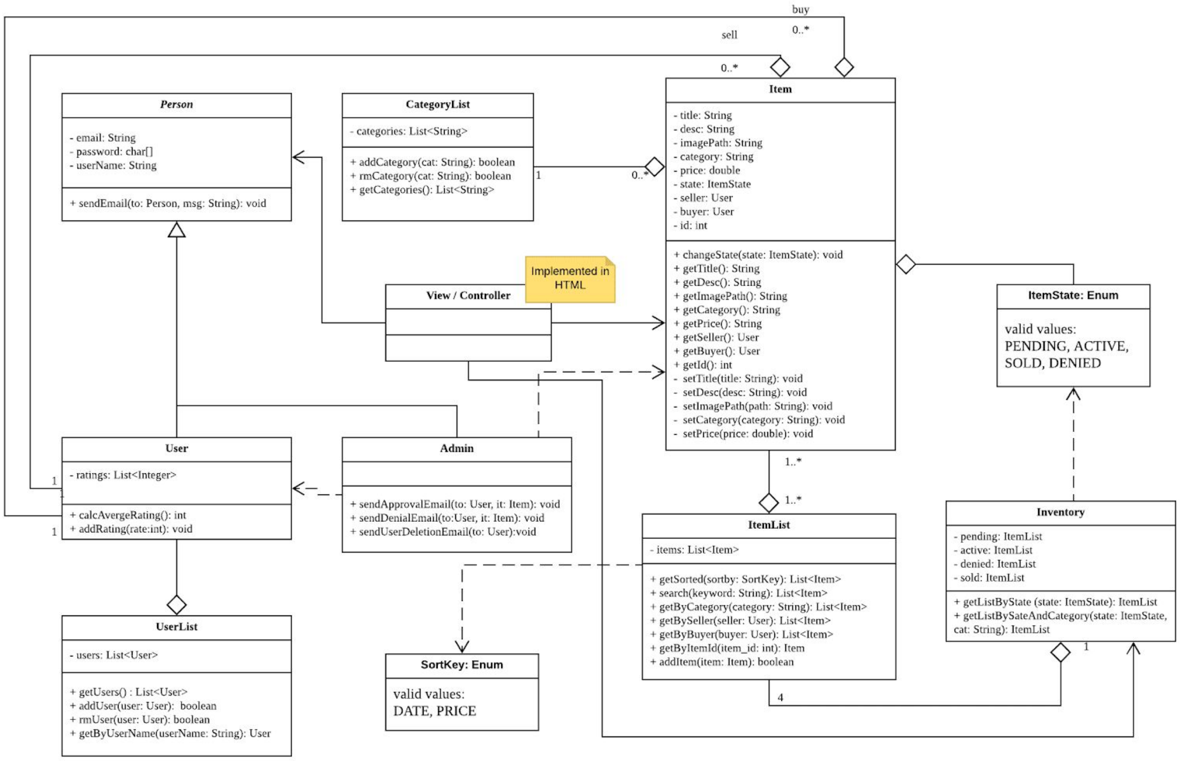
Functionality:

- User must register with @colorado.edu email
- User can view products that are published
- Users can sort products by price / date
- Users can search for products by keyword
- Users can place their items in categories
- User can publish his/her own product
- User can delete his/her published product
- User can update his/her published product (e.g. update price)
- User can mark his/her product as sold
- User can send message to each other
- Users can rate seller and buyer
- Admin can delete some product (e.g. illegal)
- Admin can add / delete categories

Stretch Goals (Optional)

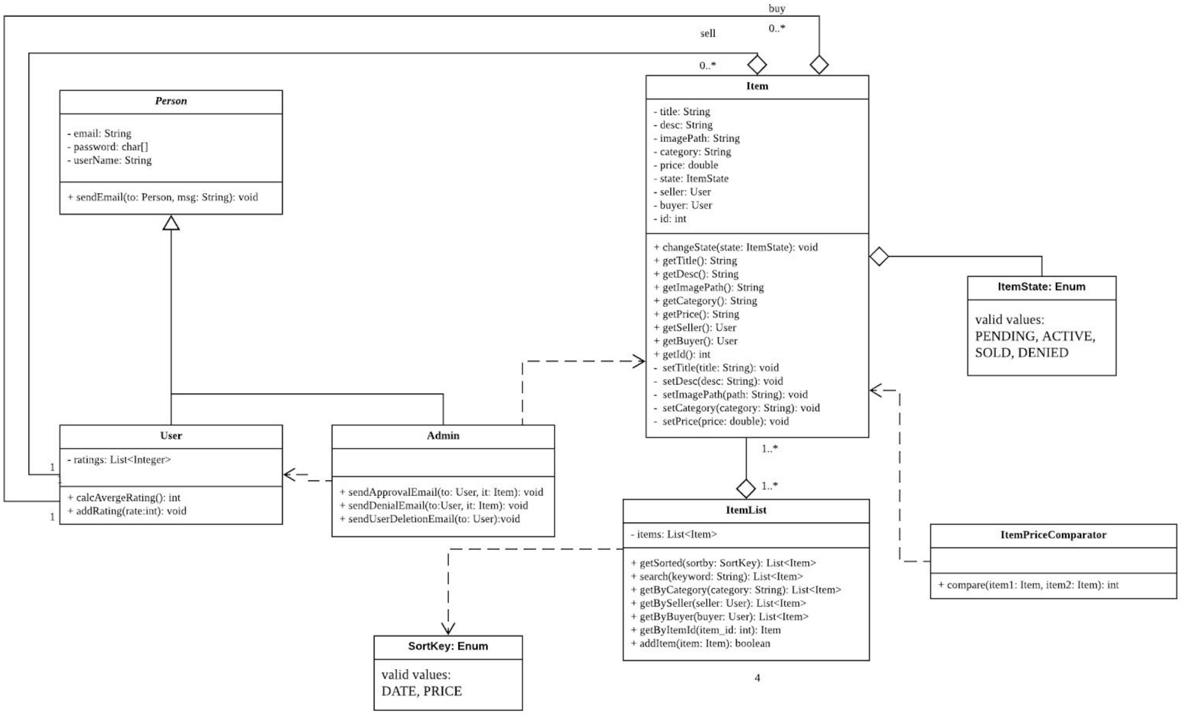
- Admin can delete a seller if they have bad reviews

2. Previous Class Diagram


buy
0..*
sell
0..*
Person
- email: String
- password: char[]
- userName: String
+ sendEmail(to: Person, msg: String): void
CategoryList
- categories: List<String>
+ addCategory(cat: String): boolean
+ rmCategory(cat: String): boolean
+ getCategories(): List<String>
1
0..*
Item
- title: String
- desc: String
- imagePath: String
- category: String
- price: double
- state: ItemState
- seller: User
- buyer: User
- id: int
+ changeState(state: ItemState): void
+ getTitle(): String
+ getDesc(): String
+ getImagePath(): String
+ getCategory(): String
+ getPrice(): String
+ getSeller(): User
+ getBuyer(): User
+ getId(): int
- setTitle(title: String): void
- setDesc(desc: String): void
- setImagePath(path: String): void
- setCategory(category: String): void
- setPrice(price: double): void
Implemented in HTML
View / Controller
ItemState: Enum
valid values:
PENDING, ACTIVE,
SOLD, DENIED
User
- ratings: List<Integer>
+ calcAvergeRating(): int
+ addRating(rate:int): void
1
1
Admin
+ sendApprovalEmail(to: User, it: Item): void
+ sendDenialEmail(to:User, it: Item): void
+ sendUserDeletionEmail(to: User):void
1..*
1..*
ItemList
- items: List<Item>
+ getSorted(sortby: SortKey): List<Item>
+ search(keyword: String): List<Item>
+ getByCategory(category: String): List<Item>
+ getBySeller(seller: User): List<Item>
+ getByBuyer(buyer: User): List<Item>
+ getByItemId(item_id: int): Item
+ addItem(item: Item): boolean
Inventory
- pending: ItemList
- active: ItemList
- denied: ItemList
- sold: ItemList
+ getListByState (state: ItemState): ItemList
+ getListBySateAndCategory(state: ItemState, cat: String): ItemList
1
4
UserList
- users: List<User>
+ getUsers() : List<User>
+ addUser(user: User): boolean
+ rmUser(user: User): boolean
+ getByUserName(userName: String): User
SortKey: Enum
valid values:
DATE, PRICE

3. Completed Class Diagram

Completed Class Diagram


buy
0..*
sell
0..*
Person
- email: String
- password: char[]
- userName: String
+ sendEmail(to: Person, msg: String): void
Item
- title: String
- desc: String
- imagePath: String
- category: String
- price: double
- state: ItemState
- seller: User
- buyer: User
- id: int
+ changeState(state: ItemState): void
+ getTitle(): String
+ getDesc(): String
+ getImagePath(): String
+ getCategory(): String
+ getPrice(): String
+ getSeller(): User
+ getBuyer(): User
+ getId(): int
- setTitle(title: String): void
- setDesc(desc: String): void
- setImagePath(path: String): void
- setCategory(category: String): void
- setPrice(price: double): void
ItemState: Enum
valid values:
PENDING, ACTIVE,
SOLD, DENIED
User
- ratings: List<Integer>
+ calcAvergeRating(): int
+ addRating(rate:int): void
1
1
Admin
+ sendApprovalEmail(to: User, it: Item): void
+ sendDenialEmail(to:User, it: Item): void
+ sendUserDeletionEmail(to: User):void
1..*
1..*
ItemList
- items: List<Item>
+ getSorted(sortby: SortKey): List<Item>
+ search(keyword: String): List<Item>
+ getByCategory(category: String): List<Item>
+ getBySeller(seller: User): List<Item>
+ getByBuyer(buyer: User): List<Item>
+ getByItemId(item_id: int): Item
+ addItem(item: Item): boolean
ItemPriceComparator
+ compare(item1: Item, item2: Item): int
SortKey: Enum
valid values:
DATE, PRICE
4

*Note, Class ItemPriceComparator implements Java Comparator<Item> interface. This class is added to simplify the ItemList.getSorted() method. There is no need to add variables in this class.

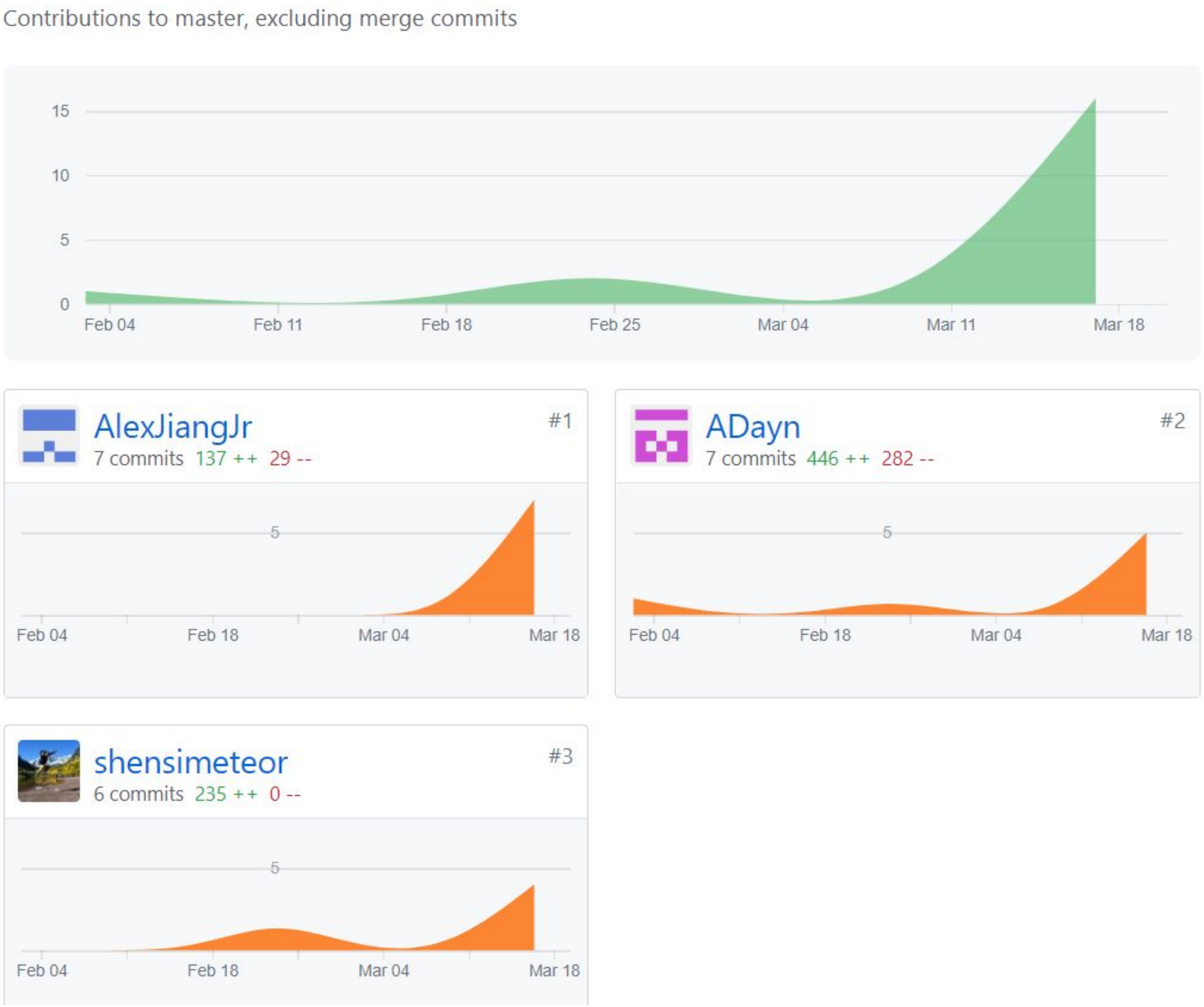
4. Summary of progress

During the past two weeks, we learned, installed and configured Spring MVC framework, Spring Boot, MySQL, database connection (e.g. Spring JPA, Hibernate). Now the basic web application and database connection works well. We designed the database schema. We implemented several basic classes (as shown in the last section) as well as some test Controller classes and View HTMLs.

5. Breakdowns

Si Shen: learned and set up Spring MVC framework, implement SortKey enum class, ItemList class, ItemPriceComparator class.
Yang Jiang: made use of MySql and created tables in it, coded for Person, User and Admin classed.
Albert Dayn: Learned and set up Spring Boot and Hibernate. Implemented “Item” class and “ItemState” enum.

6. Github graph



Contributions to master, excluding merge commits

AlexJiangJr #1
7 commits 137 ++ 29 --

ADayn #2
7 commits 446 ++ 282 --

shensimeteor #3
6 commits 235 ++ 0 --

7. **Estimate Remaining Effort**

We still have some classes to implement (as seen from the difference of section 2 diagram and section 3 diagram). We also need to write the View HTMLs and Controllers to handle URL requests. This will include designing and realizing the UI in CSS and HTML templates for Spring MVC. We have not yet developed the ability to upload files or send emails, so these will be the largest “devops” challenges. The server is running, accepting connections, and sending out HTML and JSON. We also have not completely integrated the Hibernate plugin, which means not all classes are persisted in the database.

To make sure our code compiles and functionally works well, we will communicate each other and come up with better concrete implementations. Our design choices still seem to be the best approach to the problem, but this may change as we get closer to realizing the product.

8. **Next Iteration:**

As we still do not have a functioning UI, our next iteration will involve adding a front page and ensuring some basic functions work (e.g. user registration, item adding/removing). To do this, we will need to add controllers and HTML views for each of the pages associated with these actions (as described in our UI mockups).

We will also need to Implement the rest of the classes in our design. Meanwhile, we will use our knowledge of refactoring to improve our code. As design decisions need to be changed, the knowledge that refactoring needs to be done in small steps will be invaluable in ensuring we don’t introduce new bugs as we change our system to better model the OO concepts we are learning.