PJ04 - Handout

Read This First

This is a team project. You must contact your team members on your chosen communication platform to verify your participation status by Friday, March 31st. Failure to do so will result in removal from the team and a 0 on both Project 4 and Project 5. There is no option to complete the project individually. We highly recommend that you start early and communicate with your team members far before the deadline. Peer reviews will have a significant impact on your score.

Additionally, you and your entire team should be attending lab for the rest of the semester. Your lab GTA will help you as a Project Manager, checking in and making sure things stay on track.

Start early!

Contact your lab GTA and CC the head TA if you have any concerns. For team-related issues, reach out **as soon as possible**. There might not be much that can be done if you reach out at the last moment.

Description

Project 4 is the first part of a two part project. Both Project 4 and Project 5 are team projects focusing on the same topics. You'll be working on the initial solution with your team over the next two weeks. In Project 4, we'll establish the foundational implementation requirements and core functionality. Project 5 will expand upon that foundation with new features. Not only are these team projects, but they are also an opportunity to explore every aspect of software development independently. Think of it like the capstone for your CS 18000

experience.

This project is worth 11% of your final grade. We recommend that you take it, along with the other projects in the class, very seriously.

There are two choices. First, you'll need to select the target industry for your project (described in **The Sharing Economy** below). Next, your team will select one of the three implementation prompts to focus on. These prompts have implementation requirements, as well as optional features. You must implement the required features. Implementing optional features may earn you bonus points, depending on their complexity and polish. Bonus points are capped at the maximum possible earned points for the project (that is, your score will not exceed 100, regardless of bonus points earned). Note: The optional features we list are only suggestions. If you have ideas about other features to implement, feel free to do so.

The Sharing Economy

You will be developing a limited form of a sharing economy marketplace. Make sure to select an approach that allows you to specialize in the aspects of the course you find interesting. For Project 4, you'll be implementing an application that utilizes the keyboard (System.in) for input and the screen (System.out) for output.

The sharing economy connects customers with something they need. It might be a person with a specific skill (make an app! Fix my car!) or an object that they need to access (renting a scooter or apartment). It's common to find new apps that aim to service these markets frequently. Select one potential option for your project. It can be anything you like, provided it is appropriate for an education environment (you should be able to share the idea with the class, professors, potential employers, and family members).

You can choose your own target market and product. Want a tutoring match service? A scooter rental platform? Connecting volunteers with service work? The possibilities are endless, so pick one to differentiate your product. You could even develop a platform that connects users to other platforms, but the level of abstraction may be difficult to maintain. The implementation requirements for your platform are included in subsequent sections of the handout.

Note: Several features of your implementation involve users interacting with one another. For Project 4, you can assume that users are using the same machine, logging into and out of accounts sequentially. That is, only one user accesses the application at a time.

Lastly, this project will be manually graded after submission. We'll post scores as soon as possible to ensure you have adequate time to consider feedback while

soon as possible to ensure you have adequate time to consider feedback while working on Project 5.

The requirements for each option are grouped into two categories: Core and Selections. Each team must implement the core requirements, and select the appropriate number of selective requirements based on the number of students in their team. Teams with 5 members should select 3 selective requirements, 4 member teams should select 2, and 3 member teams should select 1. Leftover requirements will count towards the "optional" requirements for extra credit.

Note: 5 points of your grade is based on Coding Style. Use the "Run" button to check your Coding Style without using a submission.

Team Work Expectations

There will be 3, 4, or 5 team members on each team. We expect each member of every team to contribute to the project. You are permitted to divide the work in any way you see fit as long as responsibilities are evenly distributed and every team member contributes to the project source code. You will be required to document your contributions to the project in the planning document.

Each of you must send a message to your teammates by Friday, March 31st, 11:59 pm. If a team member misses this deadline and is reported for doing so, a team meeting will be scheduled with the project manager. The individual who was reported for missing the deadline will be required to provide evidence of sending a message before the deadline. If they are unable to do so, they will receive a 0 on both Project 4 and Project 5. There are no exceptions to this policy.

You may use any chat or communication platform to coordinate your development efforts. Regardless of your choice of communication tool, you must contact your teammates by Friday, March 31st, 11:59 pm.

Any team member who fails to contribute will receive a 0 on the project.

Be aware that team collaboration is limited to the members of your team. You should not be sharing code with individuals outside of your team. All work must be your own. Copying code from the internet is academic dishonesty, as is requesting help from someone who is not on your team. Any team member who is caught copying code will receive a 0 on the project and the other team members will be given an opportunity to continue without them. Remember to follow the course academic honesty policy. If you have concerns about whether or not something is okay, just ask us.

To simplify collaboration, you must make use of a Code Repository on Gitlab or Github. It will make sharing code, tracking changes, and debugging significantly

easier. However, keep in mind that any repository you use must be private, with access limited only to members of your team. Code made available publicly is academic dishonesty. You will be required to submit a copy of your repository on Vocareum by cloning it into your work folder.

- Cloning the repository on Vocareum follows the same process as cloning it locally. Use the Linux terminal in your workspace and enter the appropriate commands!
- Alternatively, you may upload a zip folder with the repository.

Note: Every team member must commit to the repository. A lack of commits may be used as evidence that a team member did not participate.

We reserve the right to modify your grade based on participation. You may receive a 0 for not contributing at all, or you may receive a 75% deduction on your team score for only contributing superficial content. If you wait until the last minute to work on the assignment, you will receive a 0 or a significant deduction as well. These cases will be judged on a case-by-case basis using evidence provided by teams.

Your team will share a workspace on Vocareum. Only one team member needs to submit.

In the event of a team disagreement, dispute, or lack of participation from any individual, you should contact your project manager as soon as possible. We can only help if we are aware of the situation. Please reach out, even if it is only to notify us that you are worried that an issue may develop.

Details about Peer Evaluations will be announced after you begin working.

Role Distribution

Every team member needs to work on this project. Before you begin coding, we highly recommend that you meet and identify roles for everyone. For example:

- Role 1: Develop menus and application control flow.
- Role 2: Develop log in details and user permissions.
- Role 3: Develop option specific requirements.
- Role 4: Develop option specific requirements.
- Role 5: Develop data persistence solutions and test cases.

You can define the roles in any way that your team prefers. Two team members may work on the same topic, as we note above. Alternatively, you may find it simpler to have everyone complete a specific task separately. Communication will be key!

Remember, this is a team project. Your project score will reflect your contribution to the team.

Common Features

While the specifics of each implementation will be related to the option you select, there are several common features for every project.

User Interactions

- All Project 4 input and output should be handled via the keyboard (System.in) and screen (System.out), just as with all of your previous projects. There is no GUI for this project.

Data

- Data must persist regardless of whether or not a user is connected. If a user disconnects and reconnects, their data should still be present.
    - If a user creates an account and then closes the application (it is no longer running), they should still be able to log in when running it again.
- Descriptive errors should appear as appropriate. For example, if someone tries to log in with an invalid account. The application should not crash under any circumstances.

Roles

- There will be two defined roles in the application: Sellers and Customers.
    - Sellers will be able to create stores to sell their products and maintain relationships with customers.
    - Customers will be able to purchase products and contact sellers.
- Users will select their role while creating an account, with permissions associated with their actions accordingly.
- Permissions details will be noted in the option requirements.

Accounts

- Users can create, edit, and delete accounts for themselves.
    - The attributes you collect as part of account creation are up to you. At a minimum, there should be an email and password associated with each account.
    - Users should be required to either create an account or sign in before gaining access to the application.
    - Whichever identifier you maintain for the user must be unique.
    - During account creation, a user will specify their role.

Stores

- Each application must support multiple stores.

- Sellers will be able to create as many stores as they like.
- Customers can access any store that has been created.

Note: The term "user" is used to refer to any user of the application, including both customers and sellers.

Option 1

The first option is to implement a marketplace calendar. The calendar will allow for booking and managing appointments between sellers and customers.

Looking for an example? Try commonly scheduled services, such as salons or car maintenance. Moreover, remember that booking a window of time is applicable to all industries (renting a scooter for a number of hours, meeting with a tutor, consulting a lawyer, etc.).

Reminder: You can assume that only one user is accessing the application at a time. A customer might log in, make an appointment, then log out. Another customer on the same machine can then log in, view the new availability schedule, and make an appointment of their own.

Your implementation must have the following:

Core

- Calendar
  - Any number of appointment calendars may be added to a store.
  - A calendar title and description must be listed at the top of the page.
  - Appointment windows will be listed below the title ordered by time.
    - Details on the appointments will appear beneath each entry.
      - Title of the appointment
      - Maximum number of attendees.
      - Current number of approved bookings.
      - Start and end time
  - All created text content must display a timestamp tracking the most recent modification.
- Sellers
  - Sellers can create, edit, and delete calendars with individual appointment windows.
  - Sellers can approve or decline customer appointment requests.
  - Sellers can view a list of currently approved appointments by store.
- Customers
  - Customers can view any of the created calendars for a store.
  - Customers can make or cancel appointment requests.
  - Customers can view a list of their currently approved appointments.

Selections

- Files
  - All file imports must occur as a prompt to enter the file path.
  - Sellers can import a csv file with a calendar to automatically create a calendar and appointment window list.
  - Customers can export a file with all of their approved appointments.
- Statistics
  - Sellers can view a dashboard that lists statistics for each of their stores.
    - Data will include a list of customers with the number of approved appointments they made and the most popular appointment windows by store.
    - Sellers can choose to sort the dashboard.
  - Customers can view a dashboard with store and seller information.
    - Data will include a list of stores by number of customers and the most popular appointment windows by store.
    - Customers can choose to sort the dashboard.
- Rescheduling
  - Sellers can search for a specific customer and move their appointment to a different time at the same store.
  - Customers can search for a specific appointment and move it to a different time at the same store.

Optional Features:

- Add rosters by appointment window, listing approved and requested bookings for sellers. Customers can view a roster with all approved bookings for that appointment. Both customers and sellers can export rosters.
- Prevent customers from booking two appointments at the same time. Warn customers when they attempt to do so and offer a choice for which to select.
- Waitlists that allow customers to add themselves to a list on full appointment windows, with sellers being able to override the enrollment limit and allow specific users to book the session. Customers can view their current position on the waitlist.

Option 2

The second option is to implement a marketplace messaging system. The messenger will allow for communication between sellers and customers.

Looking for an example? Try checking out the messaging features in large online marketplaces with third party sellers, such as eBay or Etsy.

Reminder: You can assume that only one user is accessing the application at a

time. A customer might log in, send a message then log out. A seller could then log in and read the message.

Your implementation must have the following:

Core

- Messages
  - Any customer can message any Seller, and any Seller may message any customer.
    - Customers cannot message another customer. Sellers cannot message another seller.
  - All users must have accounts to participate in a message.
    - No user may start a message to a user that does not exist.
  - Access to a specific message must be limited to the participants in the conversation.
  - Users will have create, edit, and delete access to their personal conversation history.
    - Creating a message will create it for both the sender and the recipient.
    - Editing a message will update the contents for both the sender and the recipient.
    - Deleting a message will only delete it for the user who initiated the delete operation.
  - Individual messages should be labeled with the senders name in the conversation.
- Sellers
  - Sellers should be able to view a list of customers to select an individual to message.
  - Sellers should be able to search for a specific customer to message.
- Customers
  - Customers should be able to view a list of stores to select a message recipient. Selecting a store will send a message to the seller associated with that store.
  - Customers should be able to search for a specific seller to message.

Selections

- Files
  - All file imports must occur as a prompt to enter the file path.
  - Users can export details for one or more of their conversations using a csv file.
    - All message details should be preserved: Participants, Message sender, timestamp, and contents.
  - Users can import a text file (.txt) to an existing conversation.
    - The file text will be sent as a message to the recipient.

- Statistics
  - Sellers can view a dashboard that lists statistics for each of their stores.
    - Data will include a list of customers with the number of messages they have sent and the most common words in overall messages.
    - Sellers can choose to sort the dashboard.
  - Customers can view a dashboard with store and seller information.
    - Data will include a list of stores by number of messages received and a list of stores by the number of messages that particular customer has sent.
    - Customers can choose to sort the dashboard.
- Blocking
  - Users may choose to block another user or become invisible to them.
  - An individual who has been blocked by another user may not send messages to the one who blocked them.
  - An individual who has become invisible to another user will not appear on the application when the other user searches for them.

Optional Features:

- Notify users of new messages when they log in. Move conversations with new messages to the top of the conversation history.
- Add an elective filtering mode where users can specify words or phrases they wish to be censored in their personal view. Users may enter replacement phrases or use a default **** to replace the text they wish to avoid. Filters are user-specific, one user's filter will not affect any other user.
- Users may create a special form of message as a disappearing conversation. Conversations in this mode will have no history, a message will disappear immediately after the recipient reads it.

Option 3

The third option is to implement the official marketplace of the application. The marketplace will allow sellers to list their products and customers to purchase them.

Looking for an example? Review the listing pages for any online store.

Reminder: You can assume that only one user is accessing the application at a time. A seller might log in, list a product, then log out. A customer could then log in and purchase the item.

Your implementation must have the following:

Core

- Market
  - The marketplace will be a centralized page listing available products for purchase.
    - Products will include the following information:
      - Name of the product
      - The store selling the product.
      - A description of the product
      - The quantity available for purchase
      - The price
  - The marketplace listing page will show the store, product name, and price of the available goods. Customers can select a specific product to be taken to that product's page, which will include a description and the quantity available.
  - When items are purchased, the quantity available for all users decreases by the amount being purchased.
- Sellers
  - Sellers can create, edit, or delete products associated with their stores.
  - Sellers can view a list of their sales by store, including customer information and revenues from the sale.
- Customers
  - Customers can view the overall marketplace listing products for sale, search for specific products using terms that match the name, store, or description, and sort the marketplace on price or quantity available.
  - Customers can purchase items from the product page and review a history of their previously purchased items.

Selections

- Files
  - All file imports must occur as a prompt to enter the file path.
  - Sellers can import or export products for their stores using a csv file.
    - All product details should be included, with one row per product.
  - Customers can export a file with their purchase history.
- Statistics
  - Sellers can view a dashboard that lists statistics for each of their stores.
    - Data will include a list of customers with the number of items that they have purchased and a list of products with the number of sales.
    - Sellers can choose to sort the dashboard.
  - Customers can view a dashboard with store and seller information.
    - Data will include a list of stores by number of products sold and a list of stores by the products purchased by that particular customer.
    - Customers can choose to sort the dashboard.
- Shopping cart
  - Customers can add products from different stores to a shopping cart to

purchase all at once, and can remove any product if they choose to do so. The shopping cart is preserved between sessions, so a customer may choose to sign out and return to make the purchase later.
    - Sellers can view the number of products currently in customer shopping carts, along with the store and details associated with the products.

Optional Features:

- Sellers can elect to hold sales that reduce the price of a product until a specified number of units are sold. Customers will be informed of the original and sale price when browsing the marketplace.
- Customers can leave reviews associated with specific products from sellers. Other customers can view the reviews after they post. Sellers may view reviews on their products.
- Sellers may set a per product order quantity limit that prohibits customers from ordering more units than the limit. Customers will not be able to place any additional orders for a product after they reach the limit, unless the seller increases or removes it.

Testing

There are no public test cases for this project. Each implementation will look different, and we do not want to restrict your creativity in any way.

However, you are expected to write your own custom test cases, specific to your team's implementation. We've given you examples of what this looks like in all your previous assignments. Here's an example from one of the Homework assignments:

```
@Test(timeout = 1000)
public void testExpectedOne() {
    // Set the input
    // Separate each input with a newline (\n).
    String input = "Line One\nLine Two\n";

    // Pair the input with the expected result
    String expected = "Insert the expected output here"

    // Runs the program with the input values
```