

SW Engineering

CSC648-848

Spring 2025

RedJr: SFSU Marketplace

Ria Thakker (Team Lead, rthakker@sfsu.edu), Rohith Gannaju (Backend Lead), Danny Duenas (Frontend Lead), James Richards (GitHub Master), Eric/Po-Han Chen(Frontend Dev)

Milestone 2 Part I, 03/25/2025

History

Date Submitted	03/25/2025
Date Revised	04/01/2025

01. Executive Summary

In today's fast-paced university environment, students and faculty often struggle to find affordable and convenient ways to buy and sell essential items like textbooks, electronics, furniture, and learning materials. While general online marketplaces exist, they are not tailored to students, and can have unrelated listings, scams, and unreliable sellers. RedJr is a platform designed specifically for the San Francisco State University (SFSU) community, offering an easy-to-use and trusted marketplace where students and faculty can connect to buy and sell items.

RedJr provides essential e-commerce functionalities, including item browsing, keyword-based searching, and direct communication between buyers and sellers. Sellers can effortlessly upload product listings and manage them through a personalized dashboard, while buyers can explore verified items with confidence. A key differentiator of our platform is its high degree of safety due to SFSU-only verification and administrator monitoring of all content. To further distinguish itself from existing platforms, RedJr features student-specific tools such as the ability to rent items from other members of the SFSU community, the ability to set up sales on campus at specific locations, and a rating system for both buyers and sellers to provide feedback on their experiences.

RedJr is a project built by SFSU students for the SFSU community. It's developed by a passionate team: Ria Thakker (Team Lead), Rohith Gannoju (Backend Lead), Danny Duenas (Frontend Lead), James Richards (GitHub Master), and Eric Chen (Frontend Contributor). As students ourselves, we are familiar with the challenges faced by college students. Our mission is to reduce waste by promoting selling and giving away old items and to provide an affordable, reliable solution for buying and selling needs to the campus community.

02. Data Items & Entities

User	Description
Administrator	User responsible for system maintenance, views a different UI compared to other users
User	<p>General term for any non-administrator user who is registered and can buy or sell items</p> <ul style="list-style-type: none">• Buyer: A user that has made an offer on an item and can rent listings<ul style="list-style-type: none">○ Boolean indicating if the user is able to buy/rent items• Seller: A user that has items up for sale or has put a listing up for rent<ul style="list-style-type: none">○ Boolean indicating if the user is able to sell/rent out items <p>A user is able to be both buyer and seller or can be one or the other.</p>

Entities	Description
item	<p>Can be listed and bought from users and includes:</p> <ul style="list-style-type: none">• Title: Name of the item - mandatory• Category: Furniture, Tech, Clothing, etc. - mandatory• Description: Brief details about the item - mandatory<ul style="list-style-type: none">○ Minimum 50 characters○ Maximum 1500 characters• Condition: Brand New, Used but Like New, Some Wearing, etc. - mandatory• Photos: Images of the item - mandatory<ul style="list-style-type: none">○ Must have at least one photo or video (limit 10MB)

	<ul style="list-style-type: none"> ● Price: The amount the seller is asking for - mandatory
Offer	<p>Bid on an item that's up to be sold - mandatory to have a price</p>
Rental item	<p>An item that is only being offered for a temporary time then returned to seller</p> <p>Includes</p> <ul style="list-style-type: none"> ● Title: Name of the item - mandatory ● Category: Furniture, Tech, Clothing, etc. - mandatory ● Description: Brief details about the item - mandatory <ul style="list-style-type: none"> ○ Minimum 50 characters ○ Maximum 1500 characters ● Condition: Brand New, Used but Like New, Some Wearing, etc. - mandatory ● Photos: Images of the item - mandatory <ul style="list-style-type: none"> ○ Must have at least one photo or video (limit 10MB) ● Rental period: How long the item will be with the buyer - mandatory <ul style="list-style-type: none"> ○ Minimum 1 hour ○ Maximum 1 week ● Rental rate: Price of renting the item through one rental period - mandatory
Sale	<p>A pre-arranged meetup for buyers and sellers to exchange items and payment. Based on the outcome/fulfillment of both parties, the users' ratings are impacted.</p> <p>Includes</p> <ul style="list-style-type: none"> ● Meeting location - mandatory ● Meeting time - mandatory ● Post-sale, buyer rating - optional ● Post-sale, seller rating - optional

Seller Rating	Score out of 5 buyers give based on quality of transaction process
Buyer Rating	Score out of 5 sellers give based on quality of transaction process

Data Structure	Description
Messaging	Log of chats sent between buyer and seller users
Locations	List of on-campus areas that a sale/pickup is able to be fulfilled
Items bought/sold	Structure contains list of items a user has purchased and sold on the site

03. Functional Requirements, Prioritized

Priority 1

- Unreg user
 - 1. Unregistered users shall only be able to access the register/login/home page/listing page/about pages.
 - 2. Unregistered users can create an account using a verified SFSU email address.
 - 3. A registered/unregistered user shall be able to browse and search for items listed on the website and descriptions of the items.
- Reg user
 - 3. A registered/unregistered user shall be able to browse and search for items listed on the website and descriptions of the items.
 - 4. A registered user shall be able to upload item listings and descriptions for sale.
 - 5. A registered user shall be able to make an offer to purchase an item of their choice.
 - 10. A registered user shall be able to message another registered user.
 - 11. A registered user shall have a dashboard for their posted items and view messages from prospective buyers.
 - 12. Registered users who have agreed to a purchase shall be able to create a sale that contains the meeting location on campus and time.
- Admin
 - 17. A site administrator shall be able to approve all items uploaded to the site before they are available for public view.
 - 19. A site administrator shall be able to delete/ban users that have been reported, violating platform policies.

Priority 2

- Unreg user
- Reg user
 - 6. A registered user shall be able to edit their item listings after uploading them.
 - 7. A registered user shall be able to edit an offer they have made on an item.
 - 8. A registered user shall be able to cancel an item listing and remove it from public view even if it is not sold.
 - 9. A registered user shall be able to cancel an offer on an item at any time before purchase is finalized.
 - 15. A registered user shall be able to mark items as sold and remove them from public view after a sale is finalized.

- 16. A registered user shall be able to report an item to an administrator if they feel the item violates a term or condition of the website.
- Admin
 - 18. A site administrator shall be able to delete items deemed inappropriate from public view.

Priority 3

- Unreg user
- Reg user
 - 13. A registered user who has sold an item shall be able to provide feedback and contribute to the rating of the user who has bought their item after a sale.
 - 14. A registered user who has bought an item shall be able to provide feedback and contribute to the rating of the user who has sold the item after a sale.
- Admin
 - 20. A site administrator shall be able to view the chat logs of registered users.

04. Storyboards

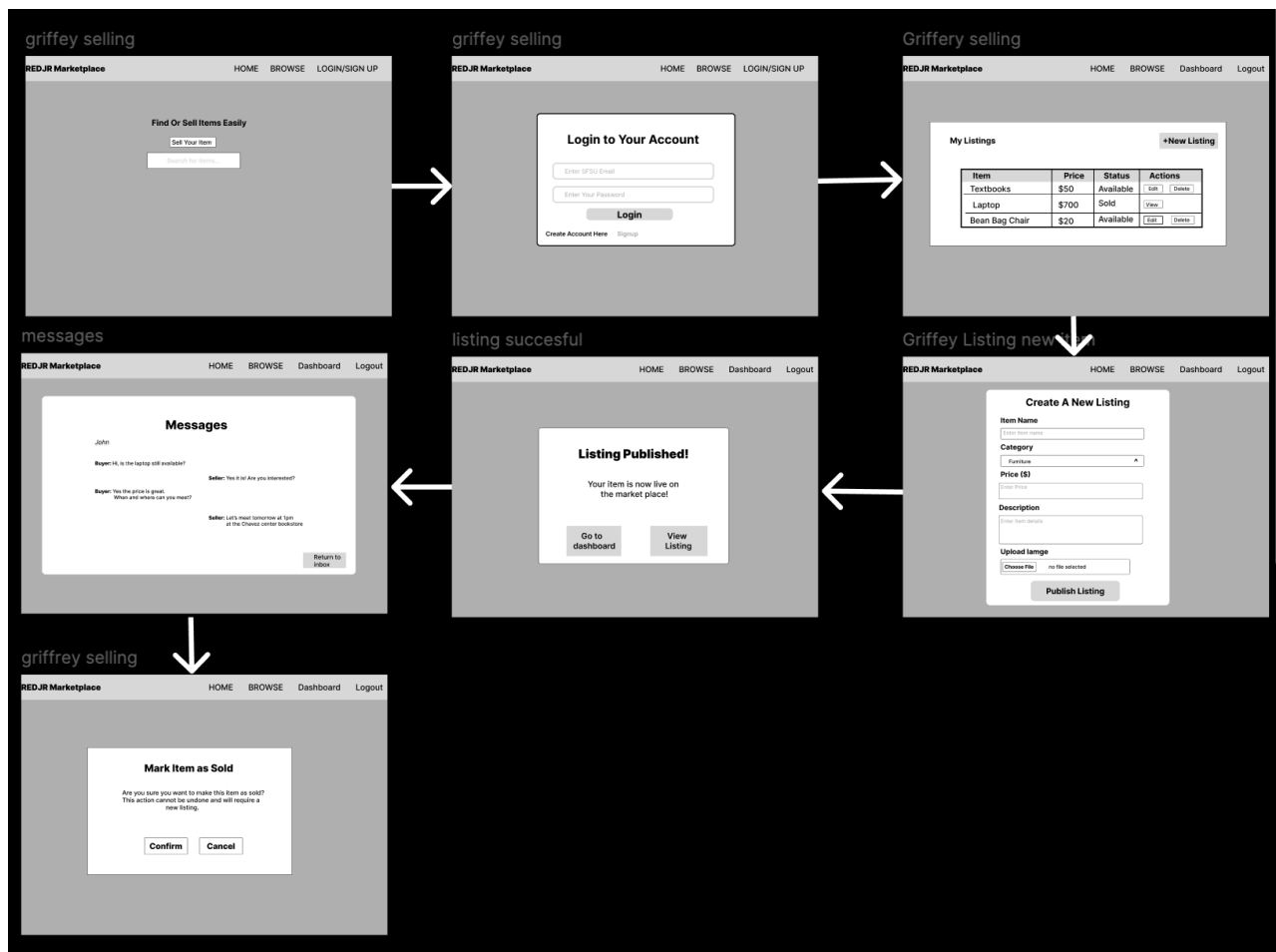
This presents the low-fidelity wireframes and UI storyboards for our REDJR Marketplace project. These designs focus on desktop browser interaction only and represent the key user flows identified in Milestone 1. The purpose of these mockups is to visualize the basic layout, navigation, and interaction flow without distractions like color or styling.

We created wireframes for the three major use cases:

- Griffey's selling flow
- Olivia's sign-up and buying flow
- Emily's buying flow

Griffey's Selling Flow:

Use Case 1: Seller logs in, creates listings, and manages them.

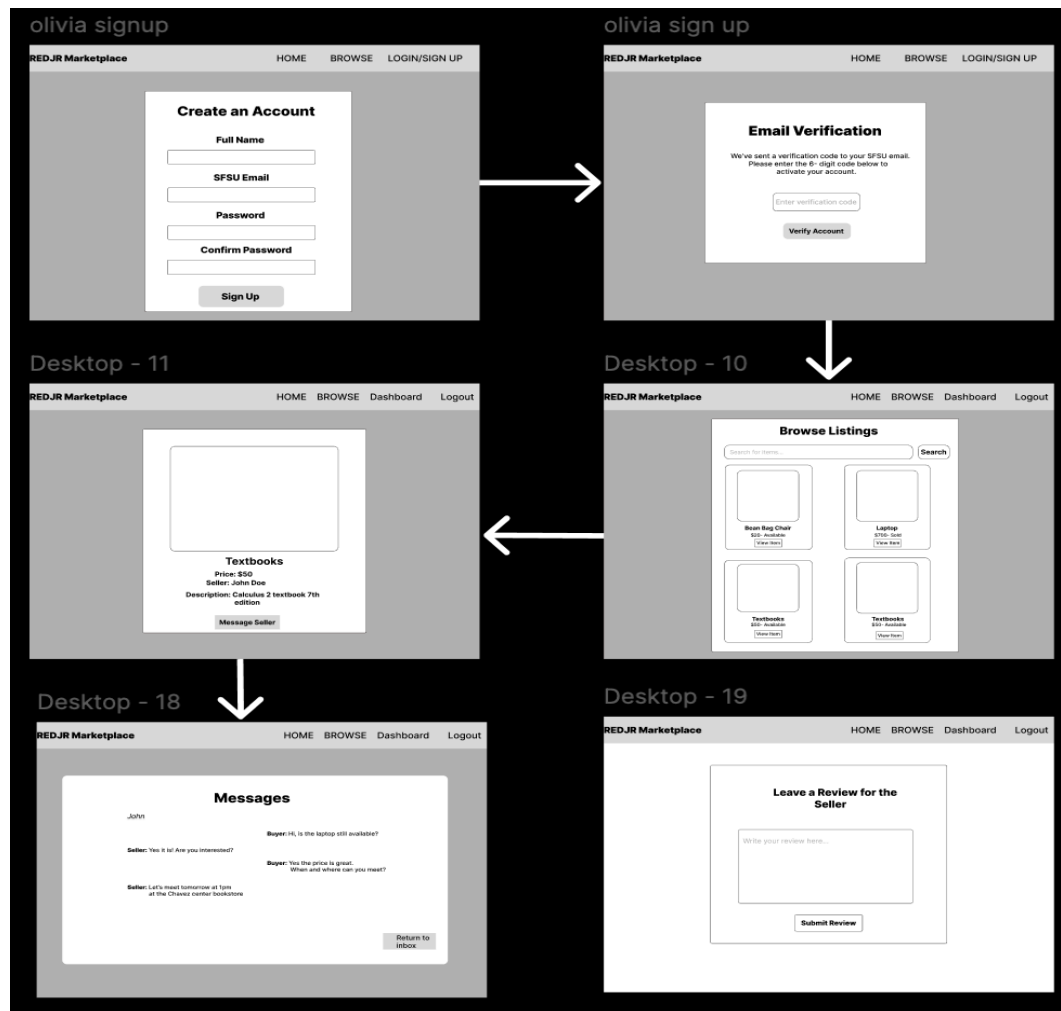


In this mockup, Griffey accesses the site and is brought to the homepage which then he proceeds to login since he already has an account and existing listings. After he logs in he views his

current listings. He also creates a new listing and finalizes it. Somebody messages him about an item and they agree to meet. After they meet and he sells the item, then marks the item as sold.

Olivia's Signup Flow

Use Case 2: Buyer registers, browses, and communicates with sellers.

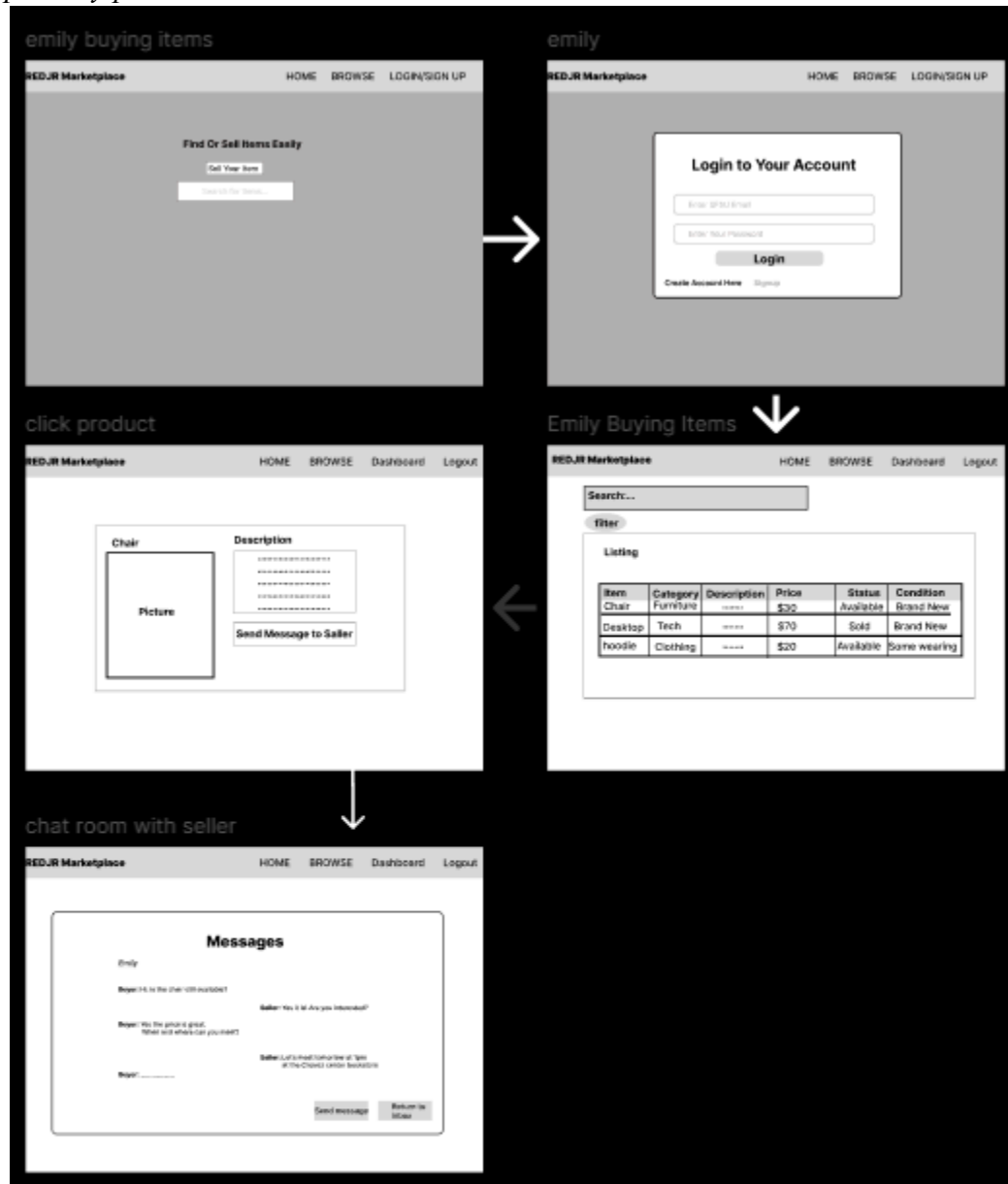


In this mockup, Olivia signs up to REDJR marketplace with her SFSU email (the homepage is the same as griffey mockup) and then she is brought to a page to verify her email. Once she does she officially has an account and access. She then browses for items and clicks on the textbook listing. She messages the seller for availability and then proceeds to make the exchange. After the exchange she leaves a review for the seller.

Emily's Buying Flow

Use Case 3: Buyer logs in, searches, and buys an item.

This mockup is slightly similar to Olivia's but in this one Emily already has an account. Emily accesses the website and then proceeds to login. She then browses for items and sees one to her liking. She then messages the seller and agrees to meet at a given location to view the item and possibly purchase.



05. High Level Architecture Design

DB Organization:

Tables:

1. Categories table

Stores various categories of items available for rent or sale.

Table Name : categories

Columns:

Column name	Data Type	Description
category_id	INT	Primary key, auto-increment
category_name	VARCHAR(100)	Unique name of the category

2. Items table

Stores item details uploaded by users.

Table name : items

Columns:

Column name	Data Type	Description
item_id	INT	Primary key, auto-incremented.
title	VARCHAR(255)	Title or name of the item.
category_id	INT	Foreign key referencing categories(category_id)
item_description	TEXT	Detailed description of the item.
item_condition	ENUM('Brand New', 'Used - Like New', 'Used - Good', 'Used - Acceptable')	Condition of the item.
price	DECIMAL(10,2)	Price of the item.

created_at	TIMESTAMP	Automatically stores the timestamp of when the item was created.
------------	-----------	--

3. ItemImages Table (Using BLOB Storage)

Stores images of items as binary data (BLOBs).

Table Name : ItemImages

Columns:

Column name	Data Type	Description
image_id	INT	Primary key, auto-increment
item_id	INT	Foreign key referencing items(item_id)
image_data	LONGBLOB	Stores the thumbnails.
mime_type	VARCHAR(50)	MIME type of the image (e.g., 'image/jpeg', 'image/png').
uploaded_at	TIMESTAMP	Automatically stores the timestamp of when the image was uploaded.

Media Storage :

Choice : Using LONGBLOB for Storing Images

- Images are stored directly in the database as thumbnails using the LONGBLOB data type in the ItemImages table. The original uploaded image is resized on the server before storage to reduce size and improve performance.
- The mime_type column helps ensure proper rendering on the client side. Images are served via a separate /image/<image_id> route with caching to improve load times and reduce database queries.

Why BLOB Storage?

1. **Easier Data Integrity:** Images and item details are stored together, ensuring consistency.
2. **Simpler Backup and Restore:** Everything is stored within the database.
3. **Avoids File Handling Issues:** No need to manage file systems or AWS S3 for image storage.

Search/Filter:

In the navigation bar, there will be a text box for searching items listed on the website. To the left of the search bar will be a drop down menu showing a list of the main item categories on the website.

The main categories are:

- Textbooks/School Supplies
- Electronics
- Furniture
- Clothing
- Miscellaneous

When searching through items by category, the default sorting is by newest listings first. Users can also sort by oldest listing if they want. Sorting items by condition (brand new, like new, some wear, and so on) is also an option, as well as listing items by seller rating from highest rating to lowest. Lastly, a buyer can sort by the asking price of the item, either lowest to highest or highest to lowest.

For searching on keywords, we will compile a list of items first by searching the item table for its title attribute, first selecting all the items that match the search exactly, then we'll query the database from items that match the first three words of search (if applicable), then the first two words, and then the first word. The items will be listed to the user in respective order. For sorting the list by default, users have the same filter features as with doing categories: newest to oldest, oldest to newest, seller rating, price by highest to lowest and vice versa, and condition of the item. When users use filters, the list will no longer be sectioned by closest matching to least, but will be mixed. If the sql query does not return any results, the site will show that no items match the search.

If the user does not type anything in the search box but clicks the search button, the site will list all items sorted according to how recently the item was listed. When filtering for items, if none of the listed items fit the conditions, the website will give a no results message. Users can also filter by if the item has an image or not. This filter will reduce the search based on the other filters that are applied.

Data items used for search:

Item Table

Title: for keywords search

Category: For listing items that only belong to a specific category

Condition: For filtering the search list on condition of item

Price: For sorting the search item list based on price

Created_at: For sorting the items from when it was posted

The Seller Table

Seller Rating: For sorting based on rating score(if seller rating implemented)

The Image Table

Item_id(FK): used for relating an image with an item so to only show listings that have images

06. Key Risks

Skills risks: Some members of the group have lower experience in software engineering / with the Flask framework for backend. We will mitigate this using the following techniques: Task breakdown to keep things manageable, checkpoints with team lead, reading up on documentation & learning how to use Flask / asking for help from knowledgeable teammates.

07. Project Management

Project management is done with the help of **GitHub projects** linked to the group's main repository. The project is organized so each member has a column of tasks as well as a column for team-wide tasks and tasks that have been completed. Tasks are broken down by Team Lead (Ria) and assigned to the relevant team member's column; once tasks are finished, they are moved to the "Done" column. Code-related tasks are put in as GitHub issues and pushed to the repository. Team Lead checks in at the halfway point of the business week to assess progress and see if there are any blockers/items that team members need support with.

This method was used for Milestones 1 and 2 and will be utilized going forward.

08. Use of GenAI tools

Used ChatGPT in doing research in how %like functions in SQL

- Got an understanding for how it works and asked if I could use %like to find items if the search was slightly off (fuzzy search), and explained that %like is used for pattern matching a sequence of characters so it will not bring up items if search spelled incorrectly.

Question - for % like can I do it where it lists items that match the spelling correctly then list items where the search description is slightly off from item name

ChatGPT response - Great question — you're essentially asking if you can **prioritize exact or close matches first**, then show **partial or fuzzy matches** after. With standard SQL **LIKE**, you can only do **basic pattern matching**, but if you want **ranked results** based on how closely they match the search term, you've got a few solid options depending on your database system.

- Also ask ChatGPT if possible to stack %like where it gives a list of items that match the search exactly and then items who have the first 3 words, then first 2 words and so on.

Question - Can I do it where it searches the whole phrase then it searches first three words, then first two words, then the first word?

ChatGPT response - Yes, you absolutely can do that! That kind of **progressive fallback search** is a smart way to return the most relevant results first and still show something useful if no exact match is found.

✓ The Strategy:

Assume the user searches for: **"MacBook Pro 13-inch 2020"**

You want to:

1. Try full phrase match first
2. Then match first 3 words: **"MacBook Pro 13-inch"**
3. Then match first 2 words: **"MacBook Pro"**
4. Then match just **"MacBook"**

```
SELECT * FROM items
WHERE item_name LIKE '%MacBook Pro 13-inch 2020%'
  OR item_name LIKE '%MacBook Pro 13-inch%'
  OR item_name LIKE '%MacBook Pro%'
  OR item_name LIKE '%MacBook%'
ORDER BY
CASE
  WHEN item_name LIKE '%MacBook Pro 13-inch 2020%' THEN 1
  WHEN item_name LIKE '%MacBook Pro 13-inch%' THEN 2
  WHEN item_name LIKE '%MacBook Pro%' THEN 3
  WHEN item_name LIKE '%MacBook%' THEN 4
```

ELSE 5

END;

Analysis - I found it pretty useful and gave me a good understanding of how the %like operator works, and I believe in a quicker time than if I would have tried to Google or YouTube search about it. A thing I like about ChatGPT is that I could ask follow up questions so I could make sure my understanding of a topic is correct

09. Team Lead Checklist

- So far all team members are fully engaged and attending team sessions when required
 - **ON TRACK**
- Team ready and able to use the chosen back and front end frameworks and those who need to learn are working on learning and practicing
 - **ON TRACK**
- Team reviewed suggested resources before drafting Milestone 2
 - **DONE**
- Team lead checked Milestone 2 document for quality, completeness, formatting and compliance with instructions before the submission
 - **DONE**
- Team lead ensured that all team members read the final Milestone 2 document and agree/understand it before submission
 - **DONE**
- Team shared and discussed experience with GenAI tools among themselves
 - **DONE**