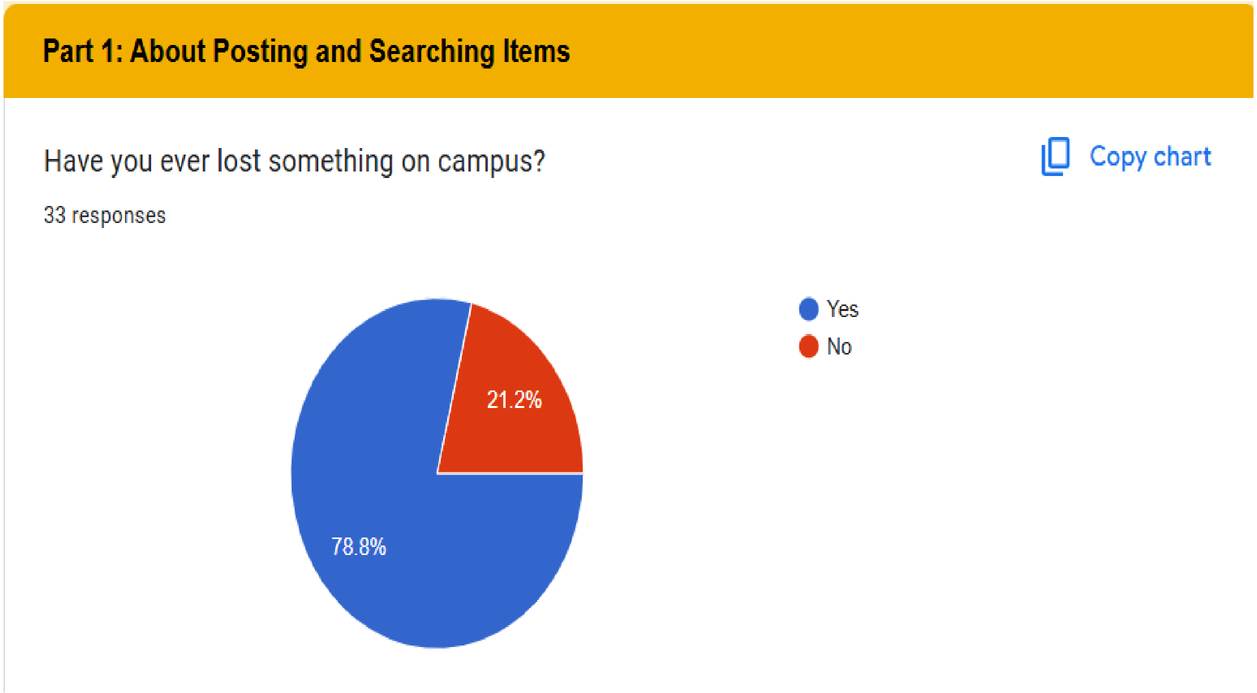
TeknnecT

Part I. Data Collection - survey/interview questionnaire with statistics

TeknnecT Campus Exchange Survey

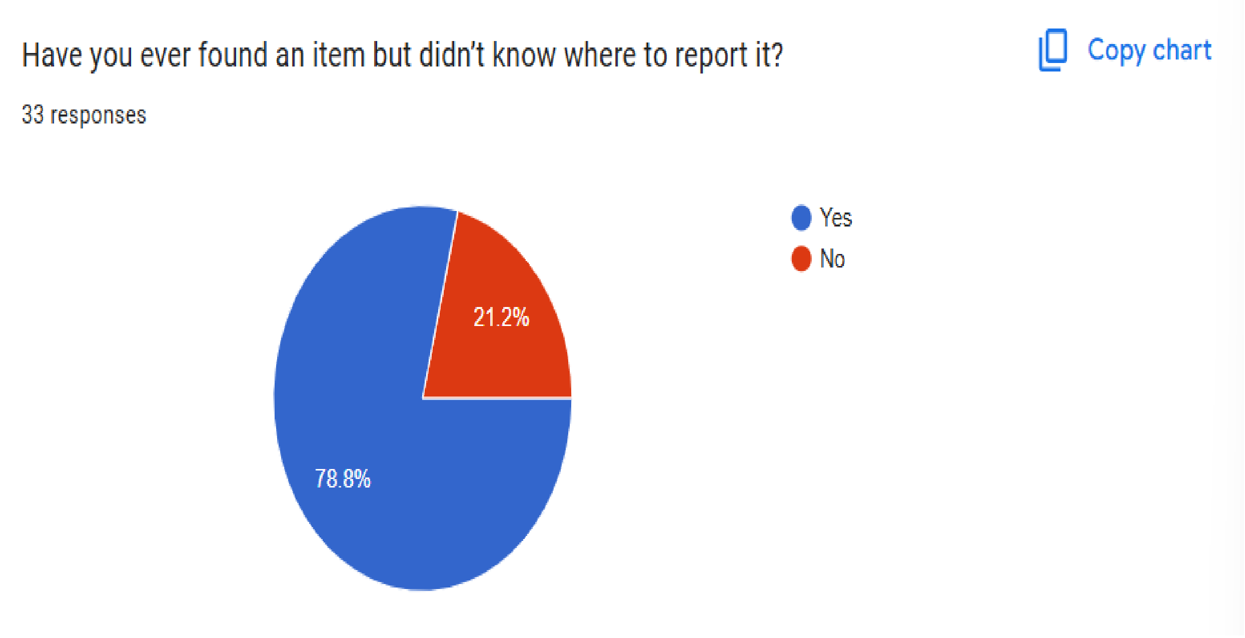
33 Respondents



Part 1: About Posting and Searching Items

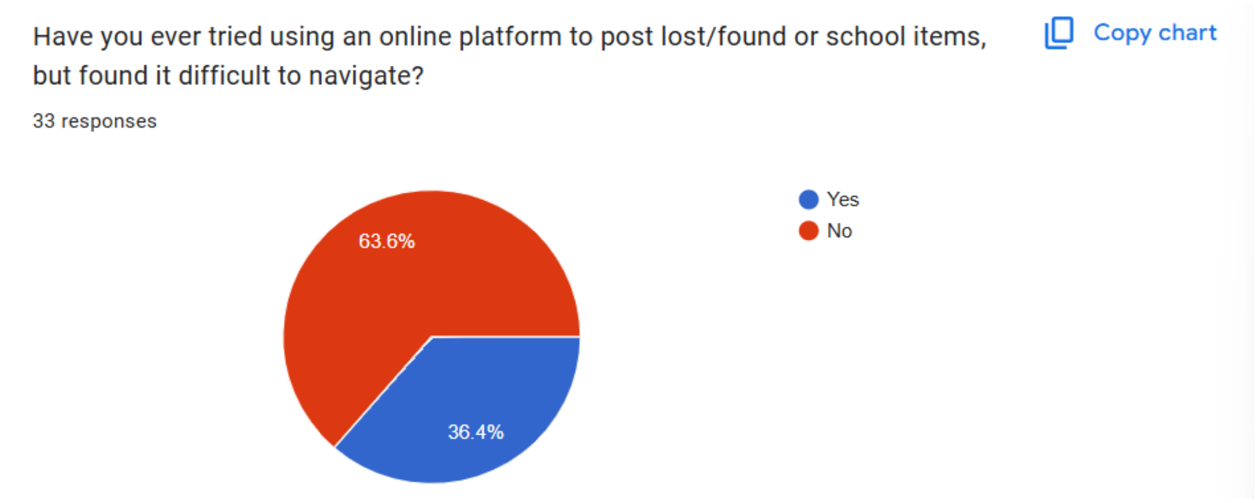
Have you ever lost something on campus?

33 responses



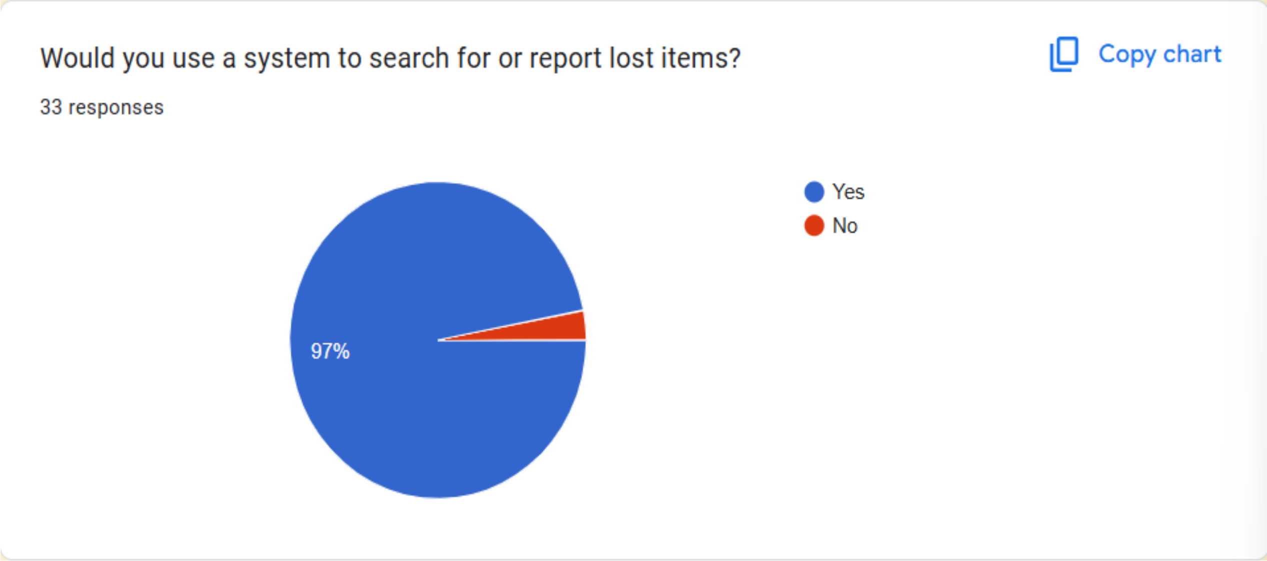
Have you ever found an item but didn't know where to report it?

33 responses



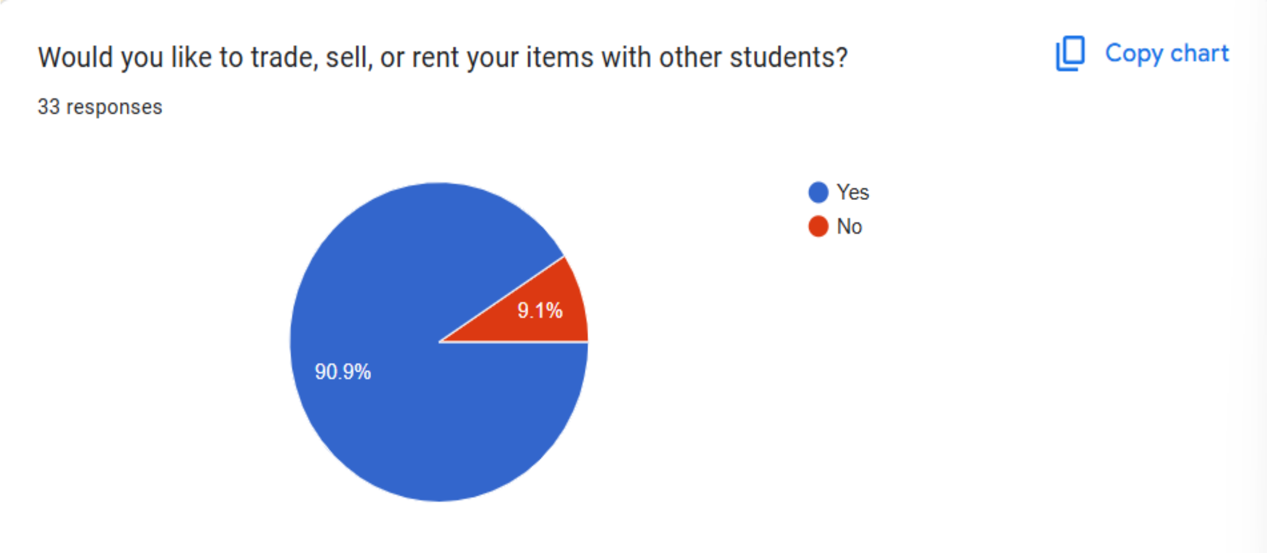
Have you ever tried using an online platform to post lost/found or school items, but found it difficult to navigate?

33 responses



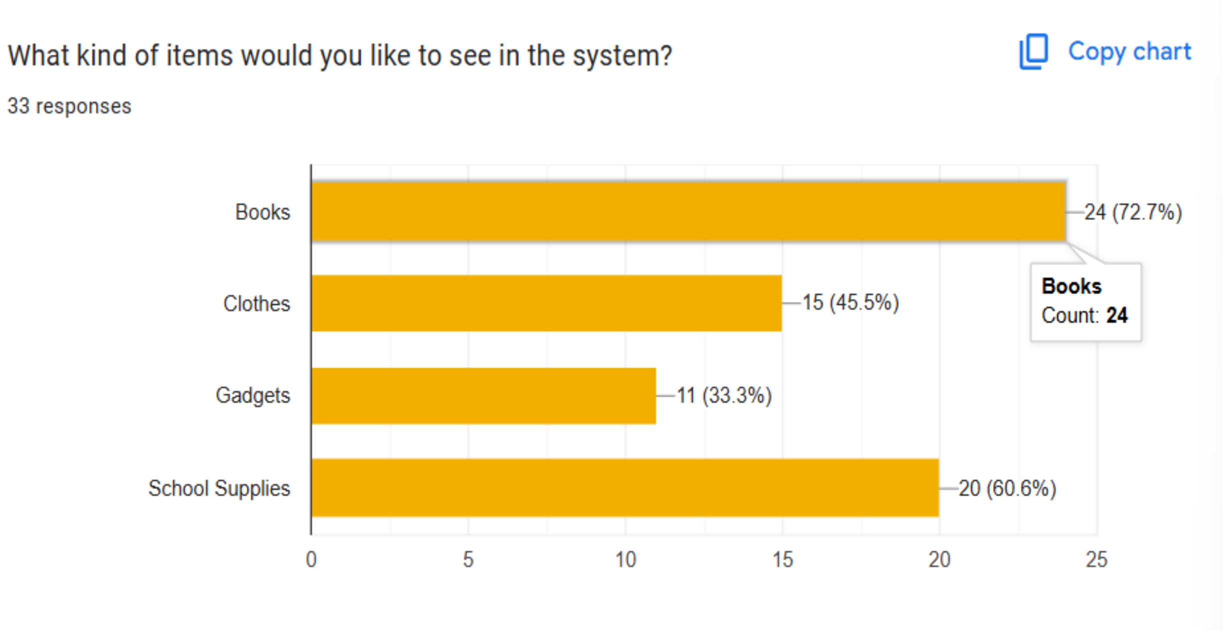
Would you use a system to search for or report lost items?

33 responses



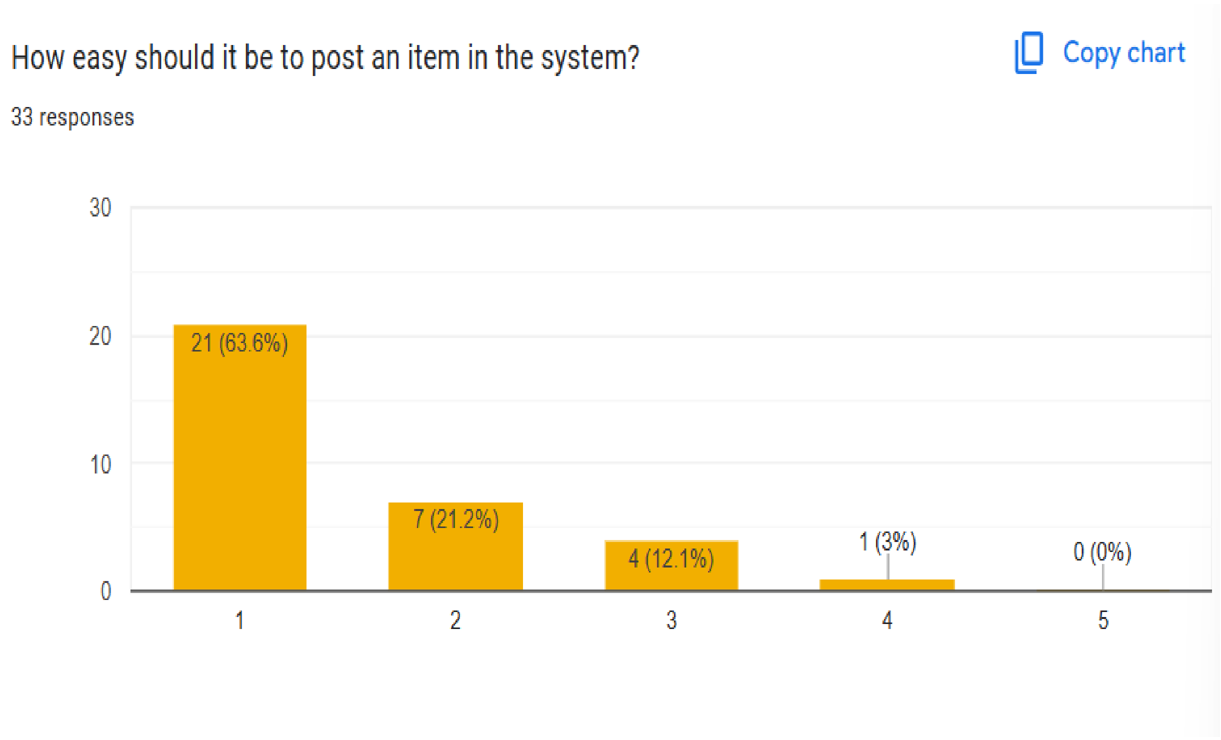
Would you like to trade, sell, or rent your items with other students?

33 responses



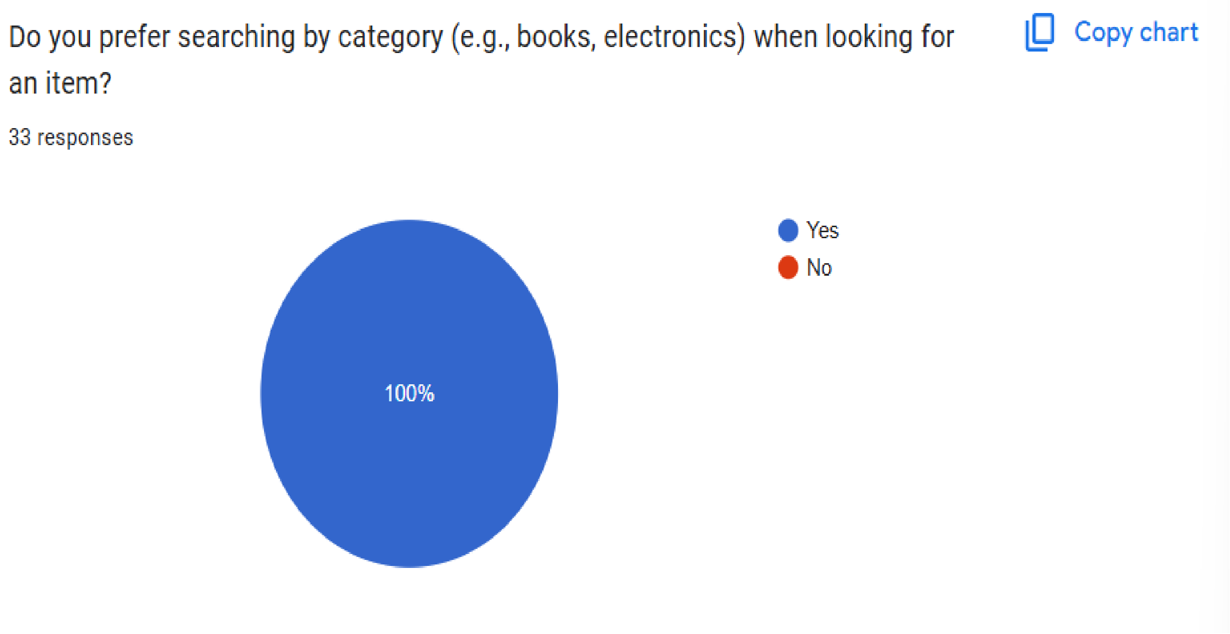
What kind of items would you like to see in the system?

33 responses



How easy should it be to post an item in the system?

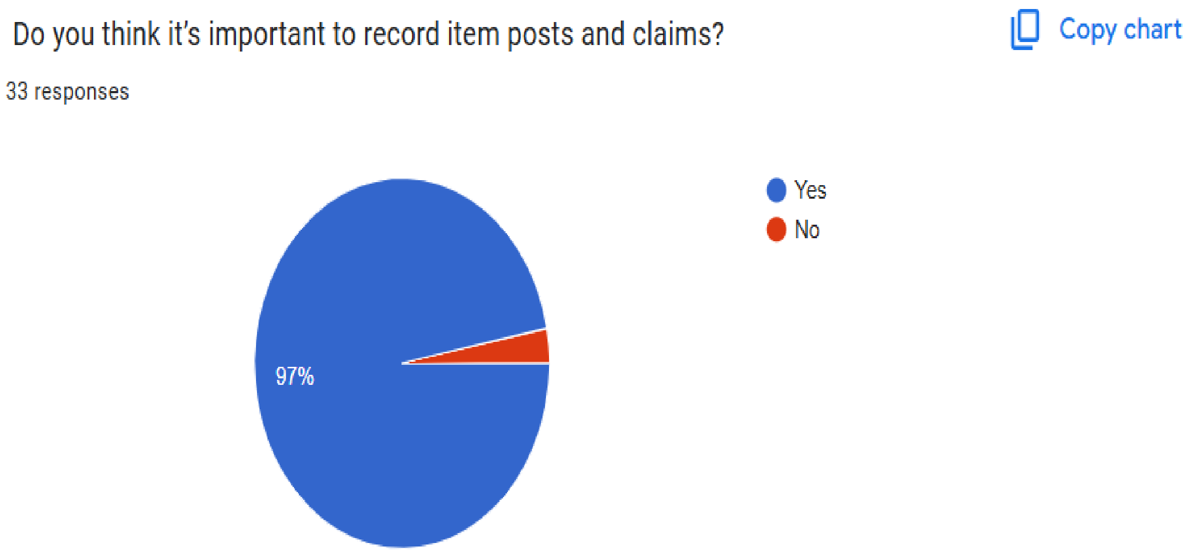
33 responses



Do you prefer searching by category (e.g., books, electronics) when looking for an item?

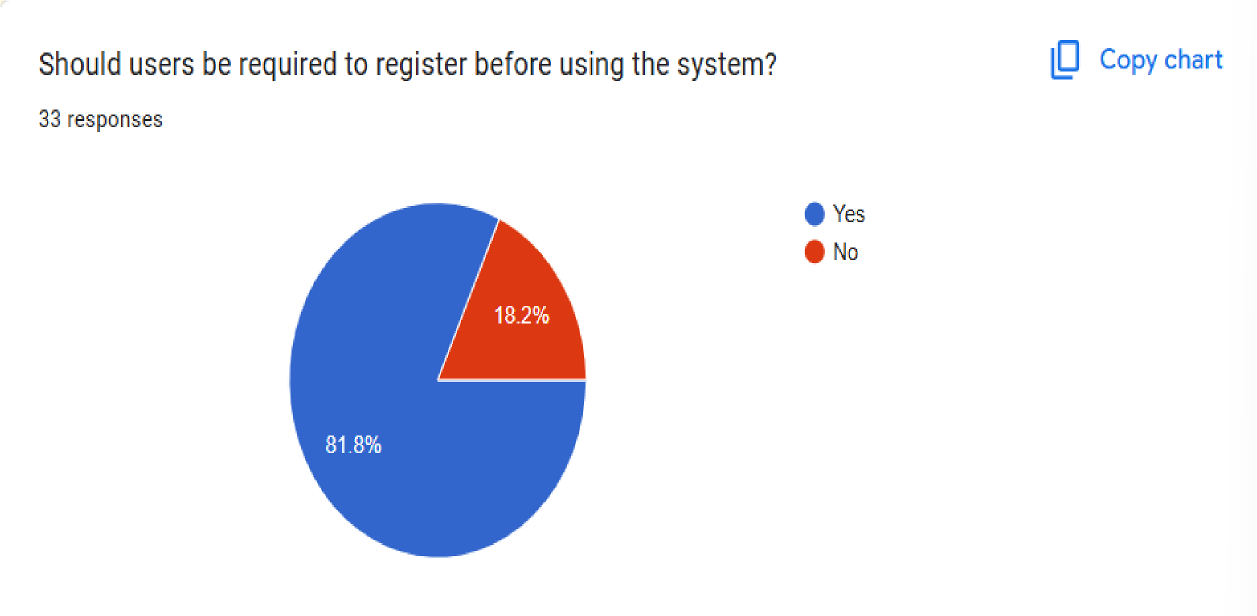
33 responses

Part 2: About Safe Transactions and Messaging



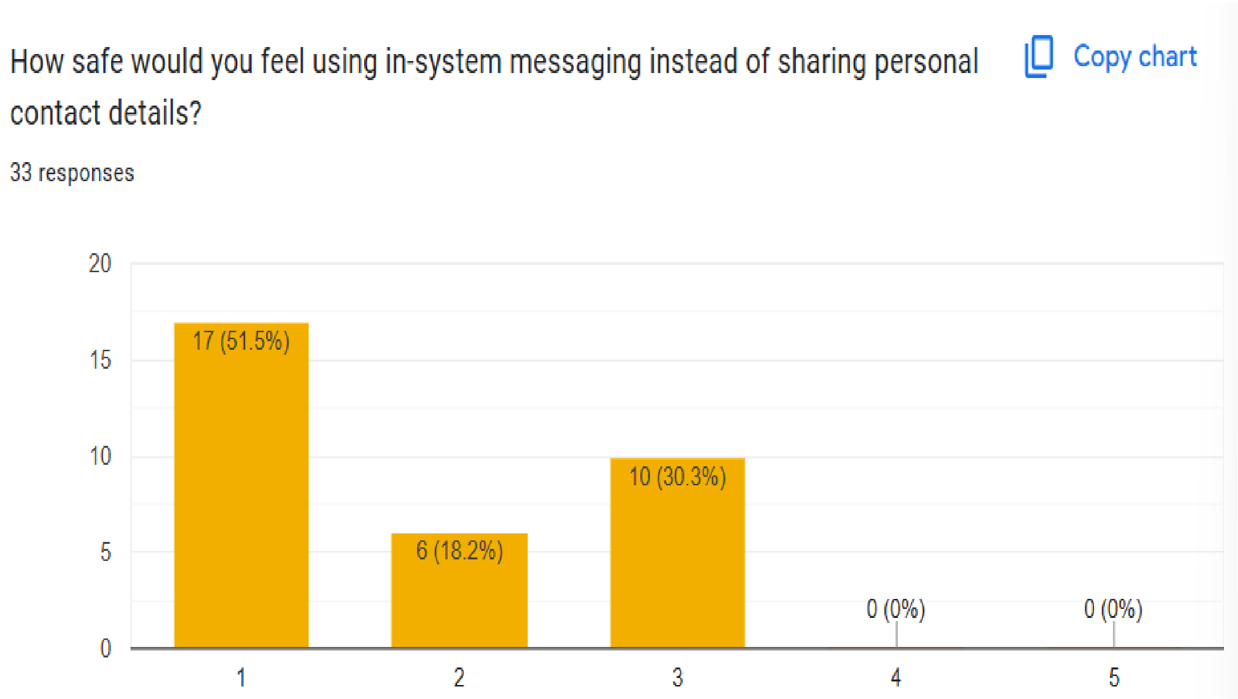
Do you think it’s important to record item posts and claims?

33 responses



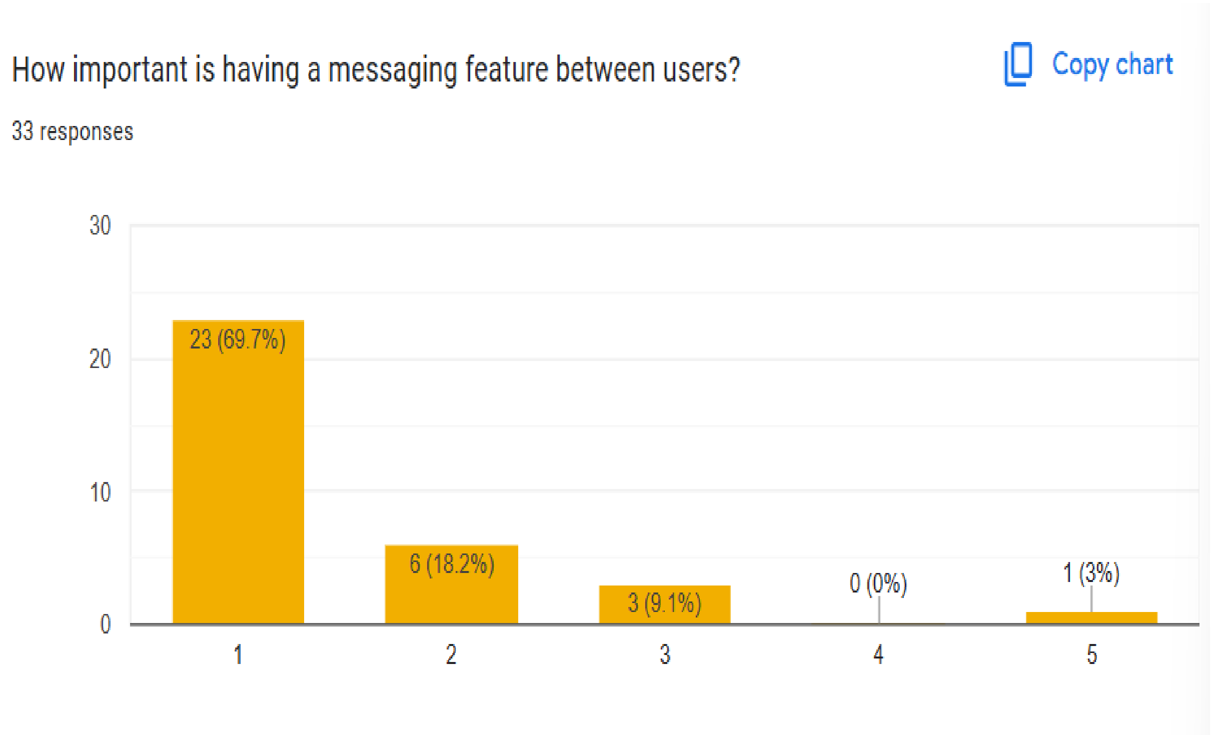
Should users be required to register before using the system?

33 responses



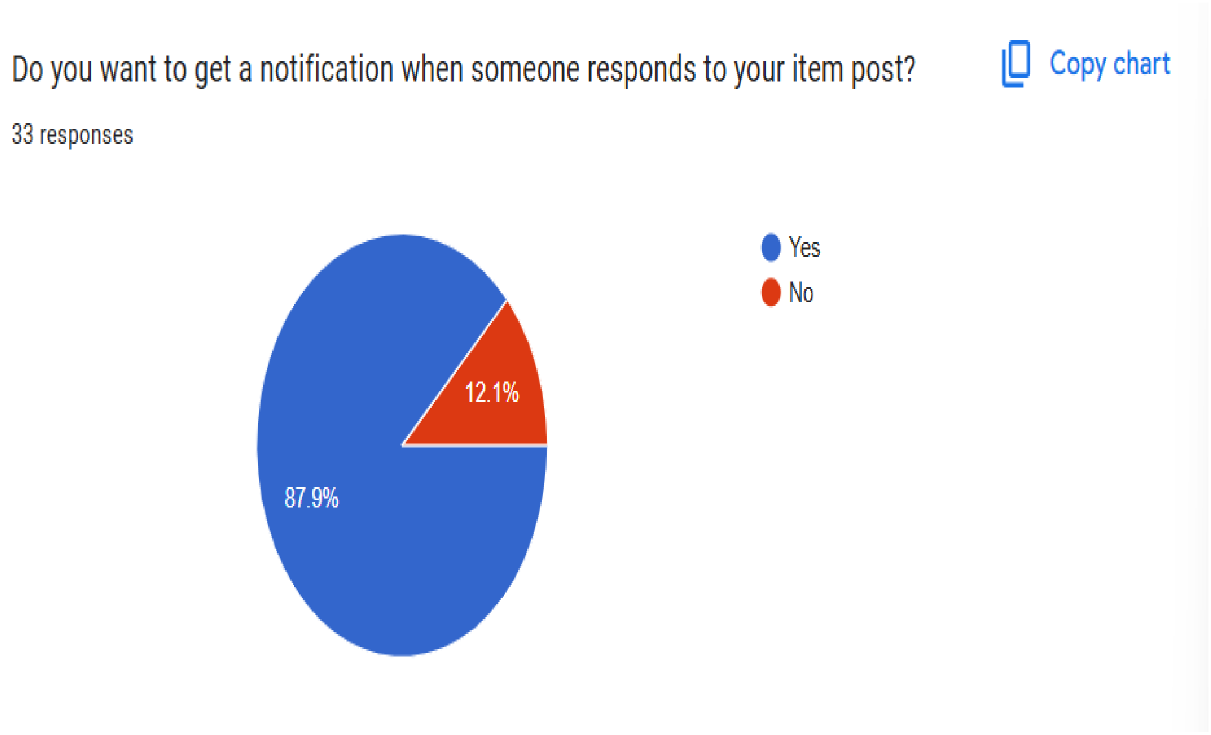
How safe would you feel using in-system messaging instead of sharing personal contact details?

33 responses



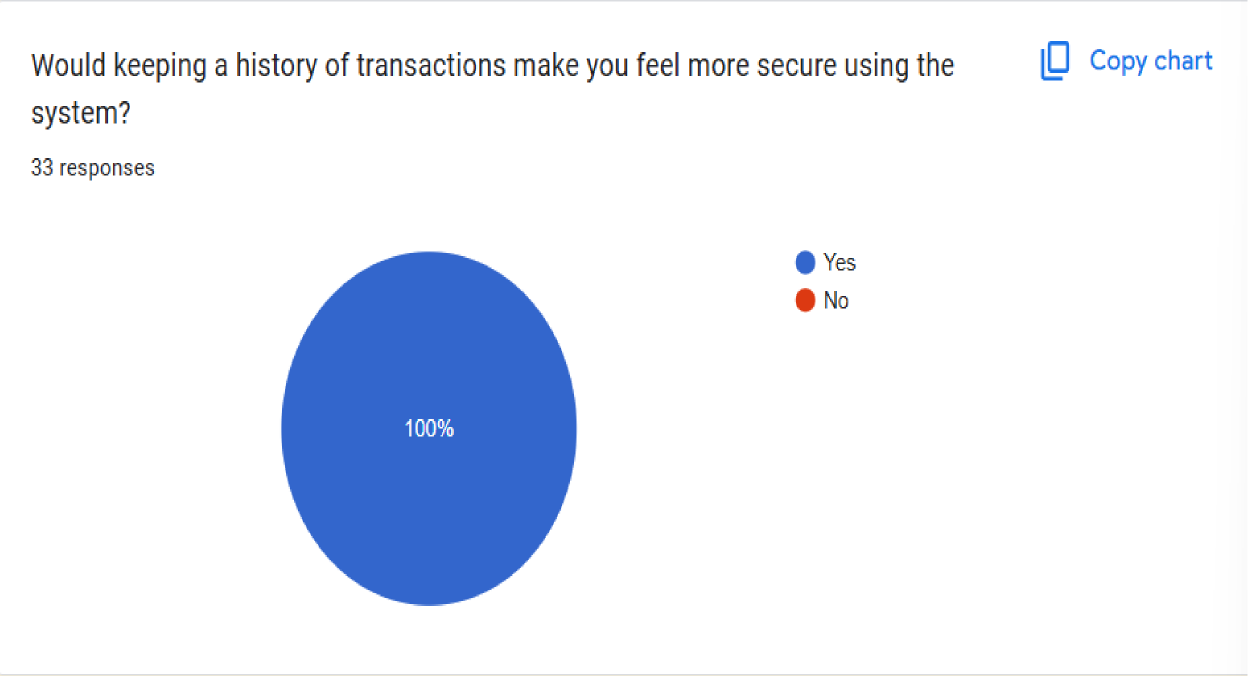
How important is having a messaging feature between users?

33 responses



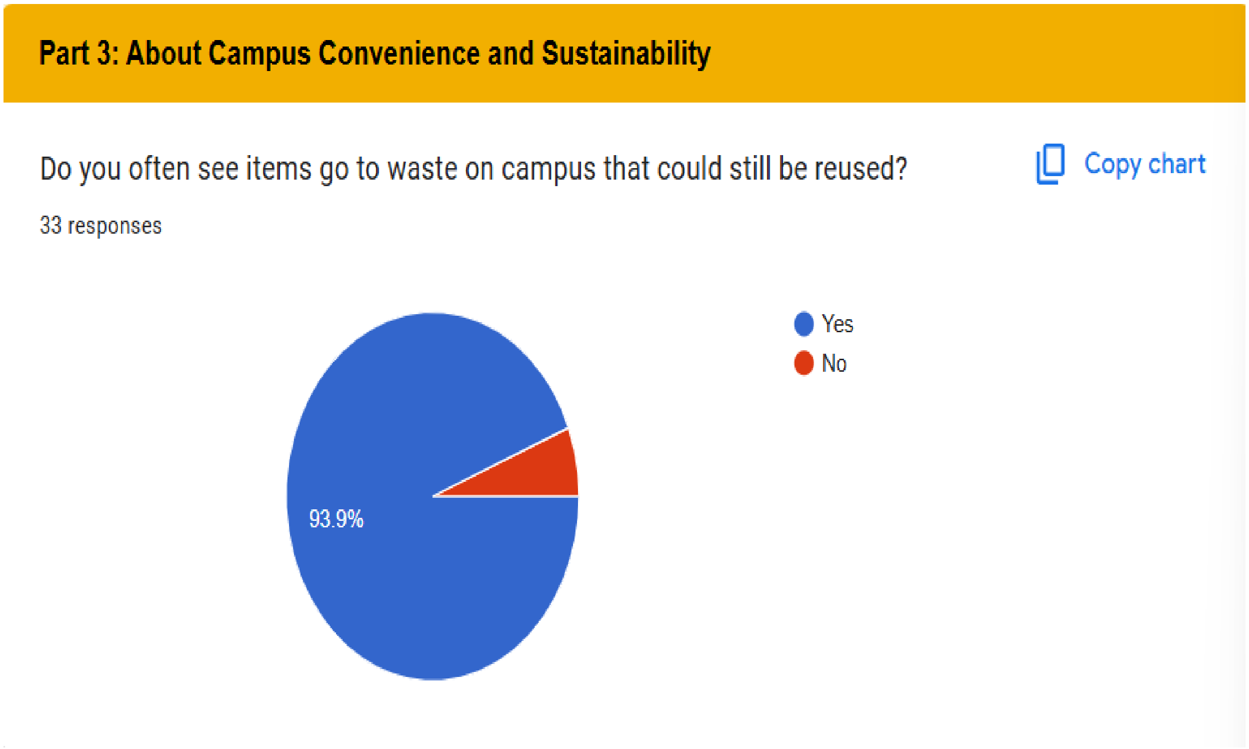
Do you want to get a notification when someone responds to your item post?

33 responses



Would keeping a history of transactions make you feel more secure using the system?

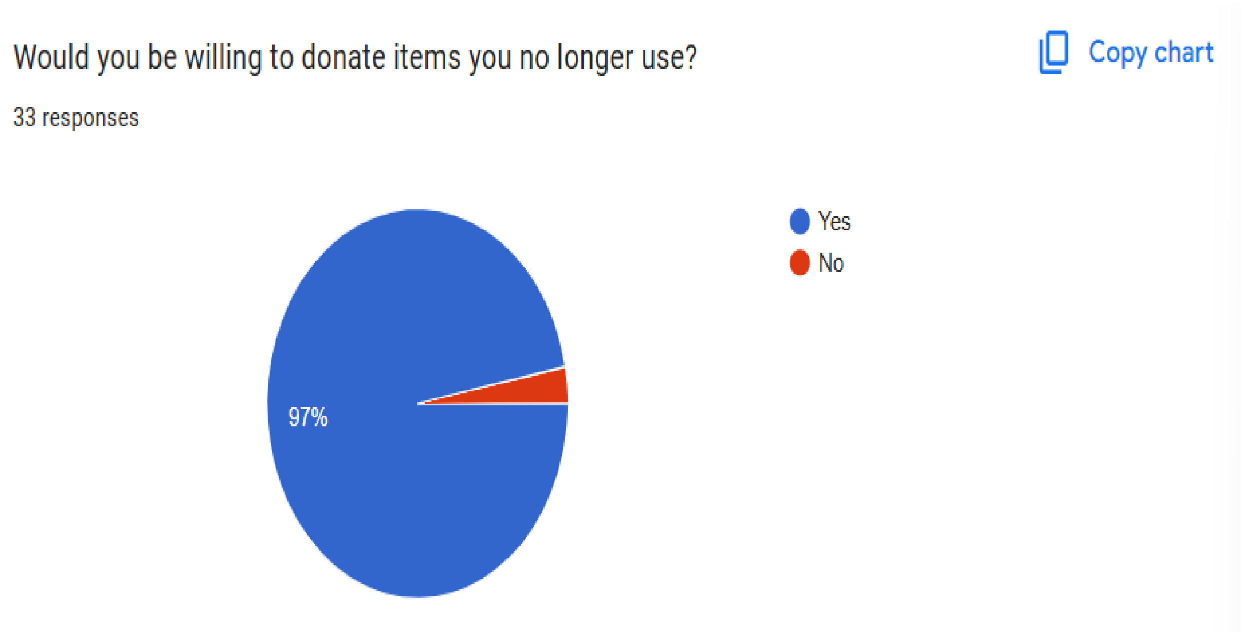
33 responses



Part 3: About Campus Convenience and Sustainability

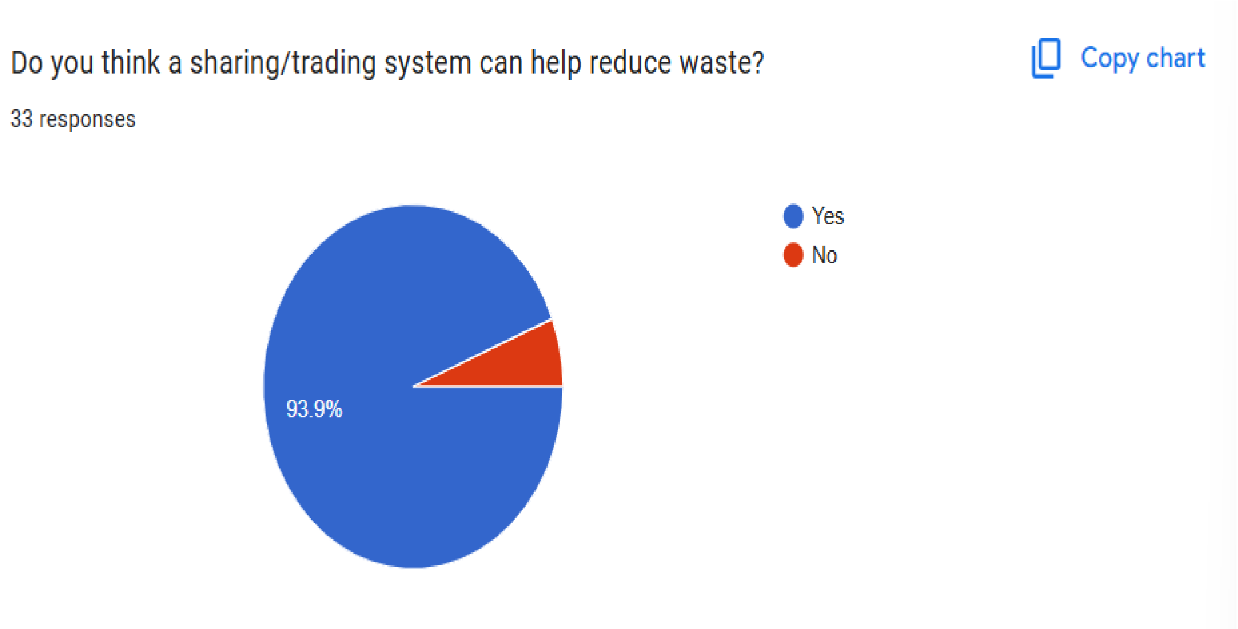
Do you often see items go to waste on campus that could still be reused?

33 responses



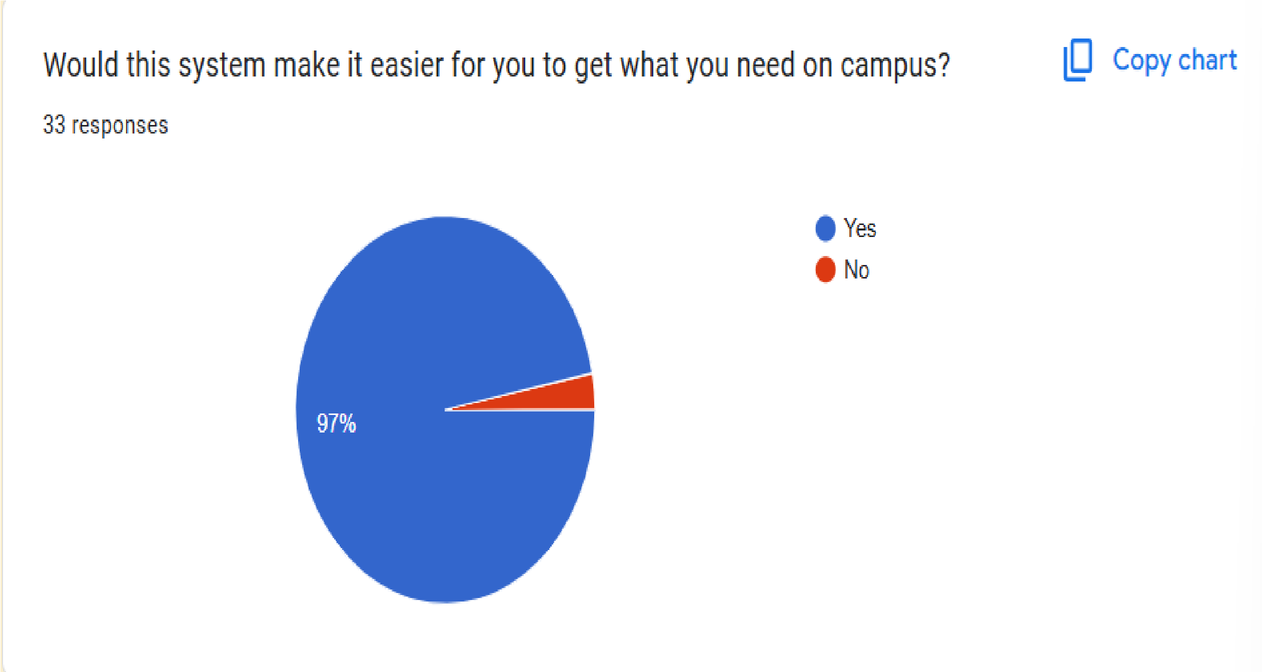
Would you be willing to donate items you no longer use?

33 responses



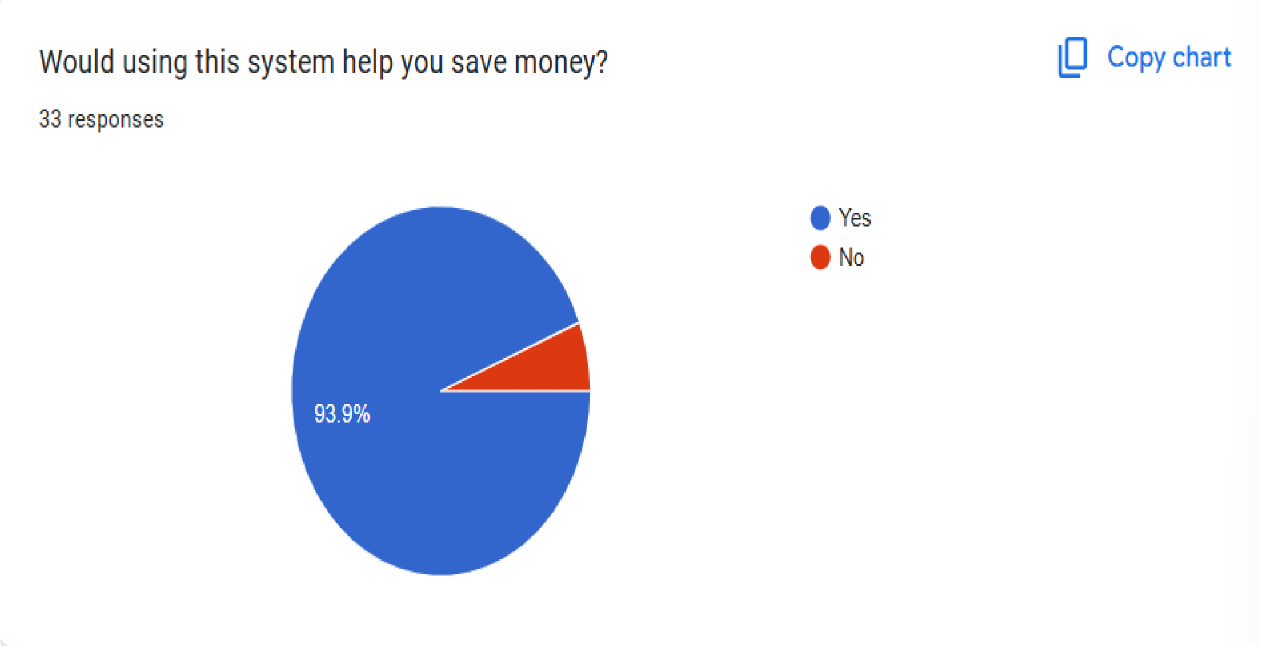
Do you think a sharing/trading system can help reduce waste?

33 responses



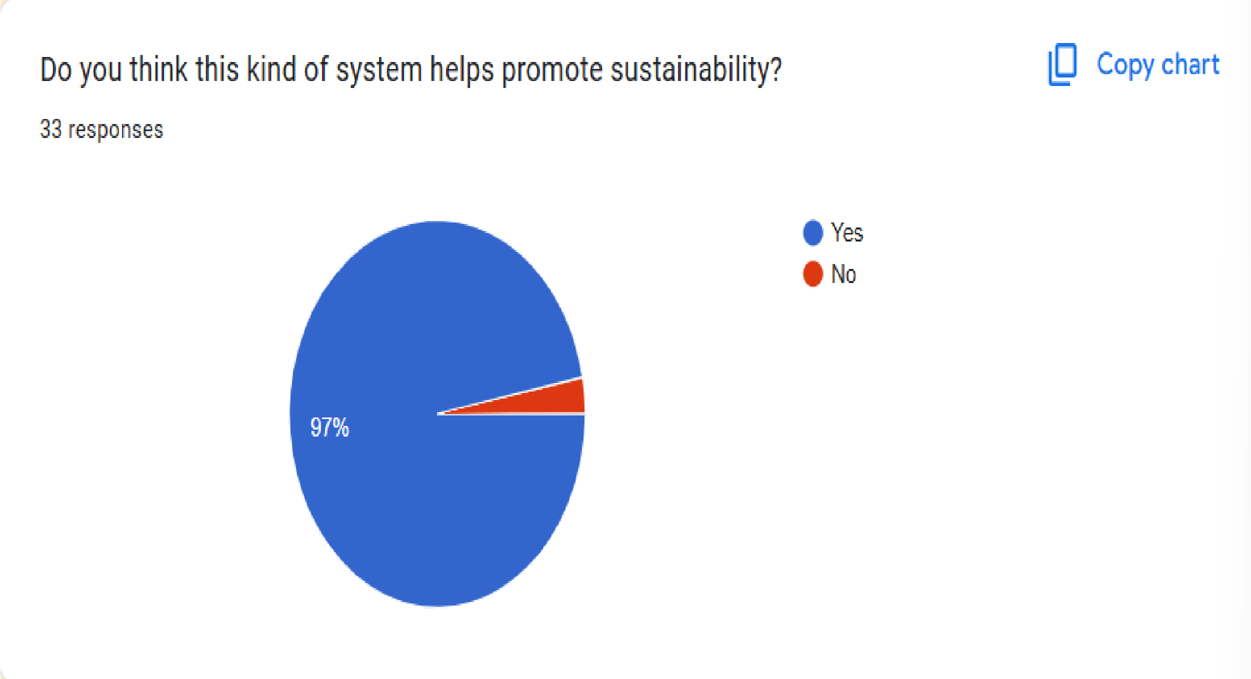
Would this system make it easier for you to get what you need on campus?

33 responses



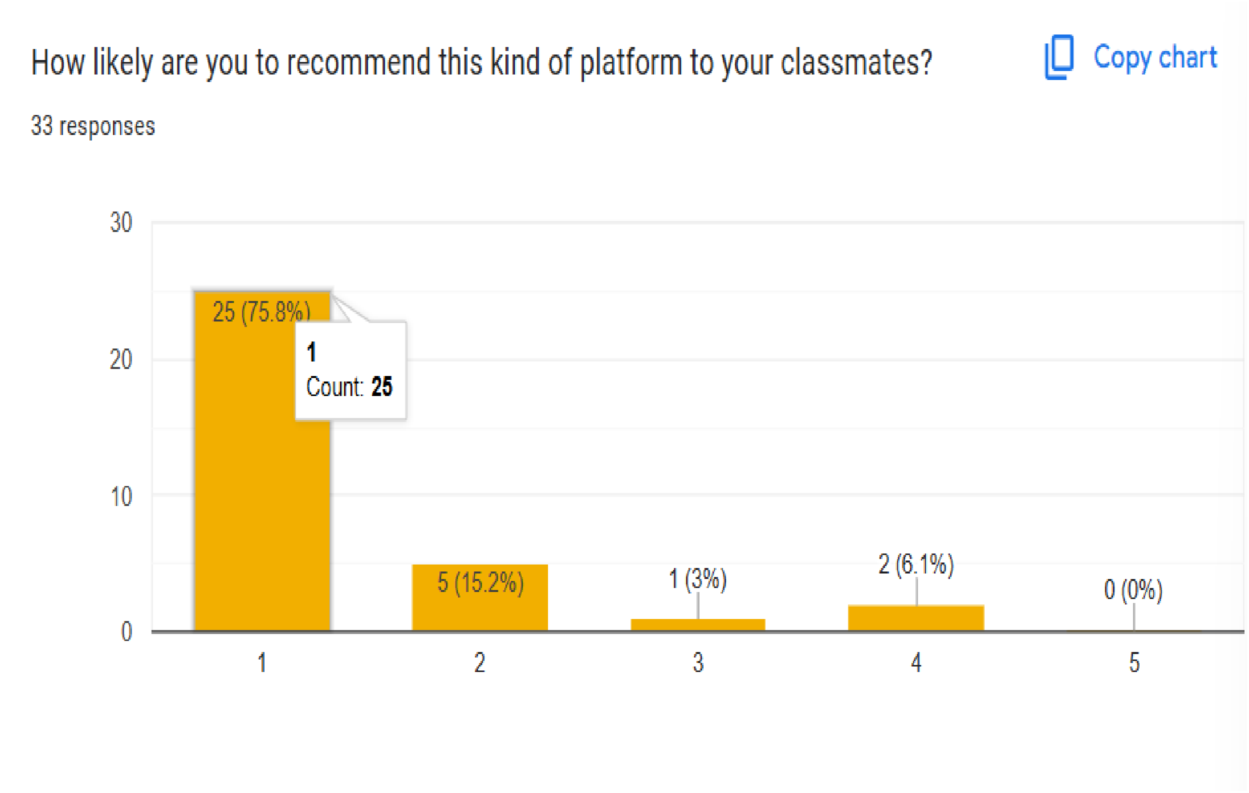
Would using this system help you save money?

33 responses



Do you think this kind of system helps promote sustainability?

33 responses

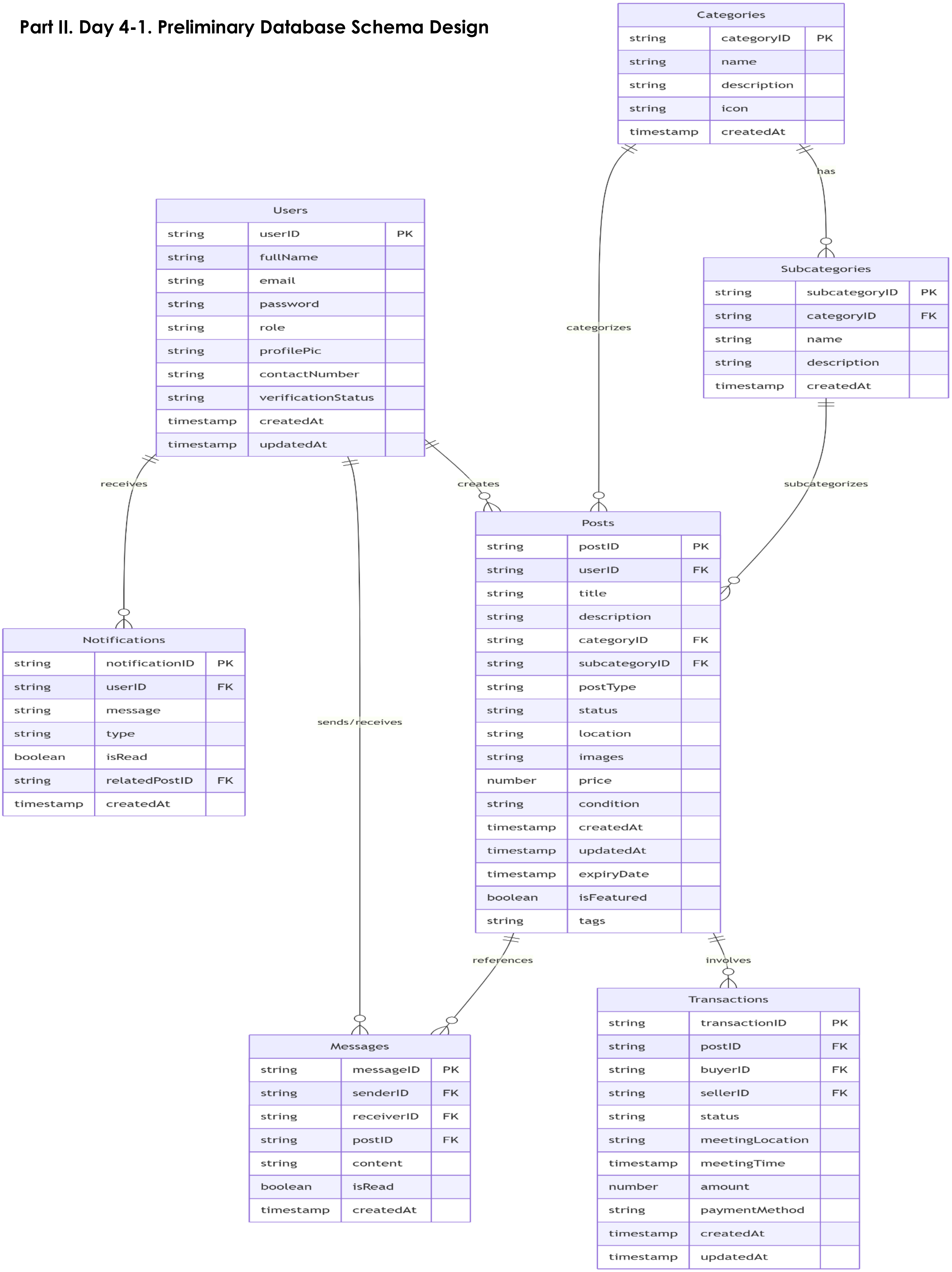


How likely are you to recommend this kind of platform to your classmates?

33 responses

Link: https://docs.google.com/forms/d/15OAFsojDWAByvI-vzWpTv1nZaW7wricvob5NHQckOGQ/edit?fbclid=IwY2xjawLGN8ZleHRuA2FlbQIxMABicmlkETE0ODFhNDRBcm5DVXM3dU5VAR44KQ1r3CXaBuo3yEOWvAcAWa0ahIU6VH0gcszjfGi2-XyYtSnZaygQ7_MzZg_aem_4DB58KqEZav0wGkghVFQxg#responses

Part II. Day 4-1. Preliminary Database Schema Design



Categories		
string	categoryID	PK
string	name	
string	description	
string	icon	
timestamp	createdAt	

Users		
string	userID	PK
string	fullName	
string	email	
string	password	
string	role	
string	profilePic	
string	contactNumber	
string	verificationStatus	
timestamp	createdAt	
timestamp	updatedAt	

Subcategories		
string	subcategoryID	PK
string	categoryID	FK
string	name	
string	description	
timestamp	createdAt	

Posts		
string	postID	PK
string	userID	FK
string	title	
string	description	
string	categoryID	FK
string	subcategoryID	FK
string	postType	
string	status	
string	location	
string	images	
number	price	
string	condition	
timestamp	createdAt	
timestamp	updatedAt	
timestamp	expiryDate	
boolean	isFeatured	
string	tags	

Notifications		
string	notificationID	PK
string	userID	FK
string	message	
string	type	
boolean	isRead	
string	relatedPostID	FK
timestamp	createdAt	

Messages		
string	messageID	PK
string	senderID	FK
string	receiverID	FK
string	postID	FK
string	content	
boolean	isRead	
timestamp	createdAt	

Transactions		
string	transactionID	PK
string	postID	FK
string	buyerID	FK
string	sellerID	FK
string	status	
string	meetingLocation	
timestamp	meetingTime	
number	amount	
string	paymentMethod	
timestamp	createdAt	
timestamp	updatedAt	

Database: Firebase

Note: All relationships (like foreign keys) are stored as document IDs (strings).

User Table

Attribute Name	Data Type	Description
userID	String (PK)	Unique identifier from Firebase Authentication
fullName	String	User's full name
email	String	User's email address
password	String	Encrypted password (handled by Firebase Auth)
role	String	'student', 'employee', or 'admin'
profilePic	String	URL to profile picture
contactNumber	String	Optional contact number
verificationStatus	String	'verified' or 'unverified'
createdAt	Timestamp	When user account was created
updatedAt	Timestamp	When user profile was last updated

Posts Table

Attribute Name	Data Type	Description
postID	String (PK)	Unique identifier for the post
userID	String (FK)	Reference to Users/userID
title	String	Title of the post
description	String	Detailed description of the post
categoryID	String (FK)	Reference to Categories/categoryID
subcategoryID	String (FK)	Optional reference to Subcategories
postType	String	'lost', 'found', 'sell', 'rent', 'donate'
status	String	'available', 'pending', 'completed', 'expired'
location	String	Simple location description
images	Array[String]	Array of image URLs
price	Number	Price for items being sold/rented
condition	String	'new', 'like new', 'good', 'fair', 'poor'
createdAt	Timestamp	When post was created
updatedAt	Timestamp	When post was last updated
expiryDate	Timestamp	For temporary posts like lost/found

isFeatured	Boolean	For admin-featured posts
tags	Array[String]	Search tags for the post

Categories Table

Attribute Name	Data Type	Description
categoryID	String (PK)	Unique identifier for category
name	String	Category name (e.g., 'Lost & Found')
description	String	Description of the category
icon	String	Icon class or URL
createdAt	Timestamp	When category was created

Subcategories Table

Attribute Name	Data Type	Description
subcategoryID	String (PK)	Unique identifier for subcategory
categoryID	String (FK)	Reference to parent category
name	String	Subcategory name (e.g., 'Textbooks')
description	String	Description of the subcategory
createdAt	Timestamp	When subcategory was created

Transaction Table

Attribute Name	Data Type	Description
transactionID	String (PK)	Unique identifier for transaction
postID	String (FK)	Reference to the post involved
buyerID	String (FK)	Reference to buyer's userID
sellerID	String (FK)	Reference to seller's userID
status	String	'pending', 'completed', 'cancelled', 'disputed'
meetingLocation	String	Agreed meeting location
meetingTime	Timestamp	Agreed meeting time
amount	Number	Transaction amount (if applicable)
paymentMethod	String	Payment method used
createdAt	Timestamp	When transaction was initiated
updatedAt	Timestamp	When transaction was last updated

Messages Table

Attribute Name	Data Type	Description
messageID	String (PK)	Unique identifier for message
senderID	String (FK)	Reference to sender's userID
receiverID	String (FK)	Reference to receiver's userID
postID	String (FK)	Reference to related post (optional)
content	String	Message content
isRead	Boolean	Whether message has been read
createdAt	Timestamp	When message was sent

Notification Table

Attribute Name	Data Type	Description
notificationID	String (PK)	Unique identifier for notification
userID	String (FK)	Reference to recipient's userID
message	String	Notification content
type	String	'message', 'transaction', 'system'
isRead	Boolean	Whether notification has been read
relatedPostID	String (FK)	Reference to related post (optional)
createdAt	Timestamp	When notification was created

Key Relationships:

1. One User can have Many Posts (1:*)
2. One Category can have Many Subcategories (1:*)
3. One Post belongs to One Category (1:1)
4. One Post can have One Subcategory (1:1)
5. One Post can have Many Transactions (1:*)
6. One User can send/receive Many Messages (1:*)
7. One User can have Many Notifications (1:*)
8. One Transaction involves One Buyer and One Seller (1:1 for each)

Part III. D4-2: System Type Decision

For our project **TeknnecT**, we have decided to build a **browser-based system**. This decision is based on the need for accessibility, ease of deployment, and compatibility with different user devices within the CIT-U community. Since our target users include students, university employees, and admins using different platforms (laptops, phones, or school computers), a web-based solution ensures universal access without requiring installation or special system requirements.

Browser-Based System

Pros:

- Accessible from any device with an internet connection (PC, phone, tablet).
- No need for software installation or updates by the user.
- Easier to maintain and deploy changes or bug fixes centrally.
- Platform-independent — works on Windows, macOS, Linux, and mobile browsers.

Cons:

- Requires a stable internet connection to function fully.
- May have limited offline capabilities unless specifically developed.
- Security must be carefully managed across shared/public networks.

Desktop System

Pros:

- Can run faster and work offline once installed.
- Better control over local resources like storage and file system.
- More secure in environments with strict internal access.

Cons:

- Requires users to download and install the software.
- Updates must be rolled out manually or with an installer.
- Limited accessibility — harder to use on mobile devices or public computers.

In conclusion, the **browser-based approach** aligns better with the nature of our platform, which is intended to be used anytime, anywhere on campus. It allows for smoother collaboration, faster updates, and a frictionless experience for all Teknoys, regardless of their device.

Part IV. D5-1: Programming Language and Environment

For the development of our system **TeknnecT**, we decided to use **React JavaScript** as our main programming language and frontend framework. React is widely used for creating **Single-Page Applications (SPAs)**—web apps that dynamically update content without refreshing the entire page. This allows for a smooth, responsive, and fast user experience, which is ideal for a platform like TeknnecT that involves item posting, searching, and messaging. React's component-based architecture also helps us build reusable and maintainable UI parts, making development more organized and scalable. One of our instructors recommended React due to its strong community support, flexibility, and compatibility with modern web development tools.

For our development environment, we are using **Visual Studio Code (VS Code)**. It is lightweight, fast, and highly extensible with support for React, JavaScript, Firebase, and Git. It also integrates well with live server extensions, syntax highlighting, and project structure tools that help improve developer productivity. For styling the interface, we use **CSS** to ensure the platform looks clean, responsive, and user-friendly across different screen sizes.

Initially, we explored using **MongoDB** (a NoSQL document-based database) and **Supabase** (a backend-as-a-service platform built on PostgreSQL) for handling the backend and data storage. Both options have their strengths, MongoDB for its flexibility with unstructured data, and Supabase for its built-in authentication and real-time features. However, after considering ease of use, documentation, community recommendations, and direct compatibility with React, we ultimately chose **Firebase**. Firebase offers a range of powerful tools such as **Realtime Database**, **Authentication**, **Cloud Storage**, and easy integration with frontend technologies. It allows us to build a functioning backend without the complexity of server setup, making it especially ideal for a student-led project.

Overall, our chosen tech stack, **React**, **Firebase**, **VS Code**, and **CSS**, provides a reliable, efficient, and scalable environment for building a modern browser-based platform for the CIT-U community.