Project Notebook

Team Name: P2P Marketeers
Team Members(GTID): Chris Cheung (ccheung49), Somtochukwu Nwagbata (snwagbata3), Varun Patel (vpatel394), Srikar Viswanantha(sviswanatha6)

Team Agreement:

- Will meet biweekly either in person(CULC) or on a remote call(Discord/Zoom) on Tuesdays and Thursdays and weekends if need be
- Let other members know if you cannot make it to the meeting or will be late
- Respond promptly to texts within the group chat

Problem Overview

Based on initial customer discovery interviews and research, a major pain point for college students is the inability to easily and efficiently buy and sell used goods amongst their peers on campus. Key challenges uncovered include:

- Extremely low textbook buyback rates from campus bookstores (<25%) compared to potential direct peer resale value (50%+)
- Difficulties finding interested buyers for niche secondhand student gear via fragmented platforms
- Lack of tailored protections addressing fraud risks transacting high-value goods anonymously
- No existing solution purposefully aggregating all student buyers/sellers for alignment

These shortcomings stem from current generalized marketplace offerings failing to cater to the unique student use case specifically. Mainstream apps do facilitate some peer transactions but without functionality serving core university populations needs around verification, notifications, item cataloging/pricing, on-campus logistics, etc.
Significant value is being stranded between students owing to obscured visibility and stifled platform optimization for their precise goals. Specialization could better support this meaningful yet often constrained ecommerce.

Competitive Analysis

Top existing marketplace platforms students currently use include Facebook Groups/Marketplace, GroupMe chats, and Craigslist. However, as outlined these contain no tailored mechanisms for identity confirmation or inventory consolidation exclusive to campus populations. Niche players like textbook swap sites and college classifieds have fragmentation challenges. Major platforms also offer little specialization around peer transaction safety, logistics, and pricing dynamics.
This presents a strategic opening to address direct university stakeholder problems through an integrated, compliant mobile ecommerce solution leveraging verifications, analytics, and community activation to unlock more value. Another potential competition that we might see is in

the form of traditional social networking. Platforms used mainly for social media such as Instagram have started to turn into miniature marketplaces as sellers create a “post” of their item with contact information below. The biggest reason that this would be a competition is due to the fact that platforms like instagram have a high amount of users. This is something that we will struggle with at the initial stages of our approach.

Solution Approach

Approach 1: Traditional Marketplace App

- Dedicated mobile app allowing students to list items for sale and browse listings posted by other students
- Verification process checks enrollment status via university system integration
- Ratings systems provides feedback on buyers/sellers to build trust
- Custom push notifications for new relevant listings
- Integrated campus logistics coordination and payments

Pros:

- Tailored specifically to college use cases
- Addresses inventory fragmentation and verification issues
- Enables feature optimization around notifications, item cataloging, etc.

Cons:

- Significant dev work for custom built app
- Getting user base initial traction and network effects
- Manual verifications don’t scale

Differentiation: Specialized mandate solely focused on serving students vs mainstream platforms. Builds trust and unlocks existing supply chains between classmates.

Approach 2: Social Driven Marketplace

- Develop a marketplace mobile app with integrated social elements to enhance community engagement.
- Introduce features like comments, likes, and direct messaging to facilitate communication and interaction.
- Emphasize user connections and shared interests within the marketplace community.

Pros:

- Enhanced User Engagement.
- Personalization to individual users

Cons:

- Dependency on External Platforms for social-drive
- Privacy Concerns

Differentiation: Hyperlocal Community Focus: Fosters strong social connections and targets a specific audience, but limited reach and potential exclusivity are limitations.

Approach 3: Gamified Marketplace

- Integrate gamification elements to enhance user engagement on the marketplace mobile app.
- Incorporate features such as user levels, achievements, badges, challenges, and special events to create a dynamic and rewarding experience.

Pros:
- Increased Engagement
- Incentivized Actions

Cons:
- Potential for Addiction is somewhat high if done wrongly
- Inauthentic Interactions with the users

Differentiation: Leveraging gamification mechanics to drive desired peer reuse behaviors rather than pure exchange value.

Use Cases

Approach 1: Traditional Marketplace

Buyer: Use the app to purchase items of need. Upload data such as payment information, address for tax purposes / shipping. Create account login to save information and recent purchases.

Seller: Use the app for listing items for sale. Upload item description, price, condition, location, images. Have a seller profile page to display all item information.

Browser: Use the app to look for items for sale. have login in order to remember previous searches. Have a saved item feature to favorite what to buy.

Approach 2: Social Driven Marketplace

Buyer: Someone might use this app to sell their unwanted items. They would give input data like item description, images, price, condition, etc.

Seller: Someone might use this app to sell the items they don't want anymore. They would use data like payment information, address, and name.

Influencer: With the social media aspect, someone might use this app to gain a large following. With this, they could potentially earn a brand sponsorship to sell certain items on the store. They might attain this using information such as number of followers, number of following, as well as a verification badge for verified people.

Approach 3: Gamified Marketplace

Buyer: Someone might use this app to sell their unwanted items. They would give input data like item description, images, price, condition, etc.

Seller: Someone might use this app to sell the items they don't want anymore. They would use data like payment information, address, and name.

Gamer: this person might use the app because of the special discounts they would get for the "gamification" of the store. It might bring them coupons which they can use in order to get free items. This would use data such as user screen time, user preferences and location services to better understand the users wants and needs

---

Learning Prototype

Approach 1: Traditional Marketplace

- User Preferences: Specific preferences users have for a traditional marketplace experience.
- Impact of Simplicity: How much simplicity and lack of additional features impact user satisfaction.
- Optimal User Flow: The most efficient and user-friendly flow for browsing and purchasing products.

Approach 2: Social Driven Marketplace

- User Privacy Concerns: Specific user concerns related to privacy when integrating social features.
- Impact of Social Interactions: How much social interactions contribute to user engagement and trust.
- Effective Personalization: The extent to which personalized recommendations based on social connections are valued.

Approach 3: Gamified Marketplace

- User Reaction to Gamification: How users will react to gamification elements in a marketplace context.
- Long-Term User Engagement: The long-term impact of gamification on user engagement and retention.
- Preferred Gamification Elements: Which gamification elements users find most

enjoyable and motivating.

Learning Prototype Plans

Approach 1: Traditional Marketplace

- Evaluate user experience with standard marketplace features like browsing, searching, and listing items.
- Assess the efficiency of the transaction process and ease of communication between buyers and sellers.
- Understand user preferences regarding traditional marketplace functionality.

Approach 2: Social Driven Marketplace

- Assess user engagement with social features, including comments, likes, and direct messaging.
- Evaluate the impact of social interactions on item discovery and user trust.
- Understand the role of shared interests and community connections in driving marketplace activity.

Approach 3: Gamified Marketplace

- Evaluate user interaction and engagement with gamified elements within the app.
- Understand how users respond to challenges, quests, and special events.
- Assess the impact of gamification on item discovery, user ratings, and social interactions.

Biggest Concerns

- **User adoption**: Our biggest worry is struggling to gain initial traction among college students and not reaching sufficient scale early on. To de-risk, we plan to partner closely with student groups/influencers from the start and explore creative marketing engagement tactics leveraging on-campus events/activities.

- **Team coordination**: As an early-stage project with limited initial resources, effective collaboration will be essential. We plan to over-communicate, leverage streamlined tools like Notion, and meet both synchronously and asynchronously to align despite other commitments.

- **Domain expertise**: Building trust and safety mechanisms like ID verification for a marketplace involves solving complex edge cases. Consulting school administrators early while designing pragmatic anti-fraud systems will help overcome our group's limited firsthand institutional knowledge.
- **Consistency:** Getting users is already an existing issue that we might face. Another issue that might come along with this, is to have users that consistently use this app. We don't want to have an app that users download and then stop using. In order to ensure consistent use, we have to have frequent motivational marketing strategies so that people are inclined to still use the app.

Interview Analysis

I had the brief time to interview the founders of Candor. This startup is a simple social media application that allows users to report incidents. The reason that this interview was helpful to our project was because it highlights student relations with SGA. A student can report an issue and see how SGA is acting on this issue through status updates. Because we are trying to make a student-run marketplace, I was able to communicate with the founders to understand their approach of getting users. One of the biggest things that I learned from talking to them was that getting consistent users is one of the most important things towards having a successful app. The best way to do this is by getting the word out on the street. One of the tactics that they recommended was ways to make the company name associated with a lot of things. The best way to do this is with a lot of paper advertising. By utilizing the printing service at Georgia Tech, it is possible to create various flyers and promotional items. Moreover, I noticed how they personalized the colors of the app (for whatever university they were working with at the time) to match the colors of the university. They mentioned that small things like this resonate with the user as it makes the application seem more official and real. This builds trust and faith in the user in the sense that the application won't fail.

Sprint 2 Updates

Following the feedback we got from Sprint 1, we'll be updating our approaches, use cases, etc (Use cases and learning prototype have been completely changed, for the sections that had minimal change what was changed is written in blue)

Team Agreement:

- Will meet biweekly either in person(CULC) or on a remote call(Discord/Zoom) on Tuesdays and Thursdays and weekends if need be
- Let other members know if you cannot make it to the meeting or will be late
- Respond promptly to texts within the group chat
- As an individual update other members on the progress you are making throughout the sprint
- Let other members know if you are struggling to complete a section so that they can help

Problem Overview

Based on initial customer discovery interviews and research, a major pain point for college students is the inability to easily and efficiently buy and sell used goods amongst their peers on campus. Key challenges uncovered include:

- Extremely low textbook buyback rates from campus bookstores (<25%) compared to potential direct peer resale value (50%+)
- Difficulties finding interested buyers for niche secondhand student gear via fragmented platforms
- Lack of tailored protections addressing fraud risks transacting high-value goods anonymously
- No existing solution purposefully aggregating all student buyers/sellers for alignment

These shortcomings stem from current generalized marketplace offerings failing to cater to the unique student use case specifically. Mainstream apps do facilitate some peer transactions but without functionality serving core university populations needs around verification, notifications, item cataloging/pricing, on-campus logistics, etc. ~~There is not an online thrift store dedicated just for college students. There are plenty of pop-up thrift markets and other thrift sites but nothing that is generalized towards specific college audiences.~~ College students lack a secure and user-friendly online platform tailored specifically for their thrift shopping needs. While there are numerous pop-up thrift markets and general thrift websites available, none cater specifically to the unique preferences and requirements of college students. This complicates their search for affordable and trendy clothing and items. Additionally, significant value is being stranded between students owing to obscured visibility and stifled platform optimization for their precise goals. Specialization could better support this meaningful yet often constrained ecommerce.

Competitive Analysis

Top existing marketplace platforms students currently use include Facebook Groups/Marketplace, GroupMe chats, and Craigslist. However, as outlined these contain no tailored mechanisms for identity confirmation or inventory consolidation exclusive to campus populations. Niche players like textbook swap sites and college classifieds have fragmentation challenges. Major platforms also offer little specialization around peer transaction safety, logistics, and pricing dynamics.

This presents a strategic opening to address direct university stakeholder problems through an integrated, compliant mobile ecommerce solution leveraging verifications, analytics, and community activation to unlock more value. Another potential competition that we might see is in the form of traditional social networking. Platforms used mainly for social media such as Instagram have started to turn into miniature marketplaces as sellers create a “post” of their item with contact information below. The biggest reason that this would be a competition is due to the fact that platforms like instagram have a high amount of users. This is something that we will struggle with at the initial stages of our approach.

Another competitor we have looked at is TikTok and how their platform manages to keep users online for hours on end. The scrolling feature the app provides is the main drawpoint of the app as it specifically tailors the ‘For You’ page of the application for the user and that the scrolling feature of the app is what makes it addicting to keep on using. The easy to use functionalities and tailored feeds are what keeps TikTok as one of the most used social media apps in the App Store.

Solution Approach:

Approach 1: Peer-to-Peer Mobile Marketplace App

- Dedicated mobile app allowing students to list items for sale and browse listings posted by other students
- Verification process checks enrollment status via university system integration
- Ratings systems provides feedback on buyers/sellers to build trust
- Custom push notifications for new relevant listings
- Integrated campus logistics coordination and payments

Pros:

- Tailored specifically to college use cases
- Addresses inventory fragmentation and verification issues
- Enables feature optimization around notifications, item cataloging, etc.

Cons:

- Significant dev work for custom built app
- Getting user base initial traction and network effects
- Manual verifications don’t scale

Differentiation:

Specialized mandate solely focused on serving students vs mainstream platforms. Builds trust and unlocks existing supply chains between classmates.

Approach 2: Digital Kiosk Marketplaces

- Network of digital kiosks located around campus where students can list/browse listings and arrange exchanges
- Kiosks manage inventory, coordinate scheduling for testing, provide storage, etc.
- Integrated identity verification and payments

Pros:

- Brick-and-mortar outlet creates additional trust
- Logistics handled through central depot

Cons:

- Accessibility limits flexibility
- Higher overhead than pure software solutions

Differentiation:
Physical retail presence plus digitally-powered exchange creates unique centralized hybrid.

Approach 3: University Thrift Store Model

- Main campus storefront exclusively buys/sells secondhand student goods
- Handles inventory, valuations, merchandising
- Students consign their items and get payouts when sold

Pros:

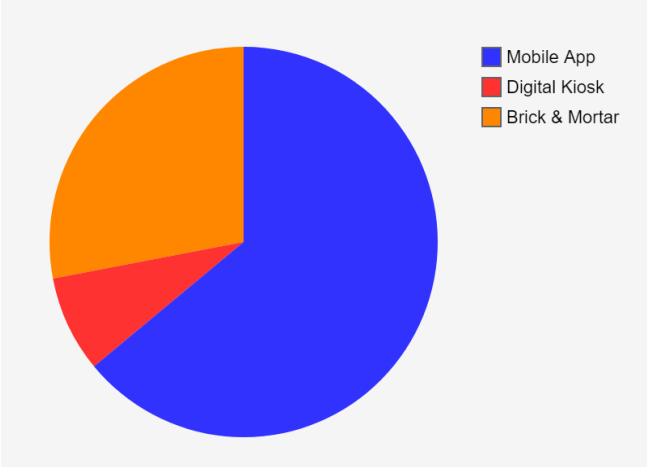
- No direct peer-to-peer risks
- Creates retail experience

Cons:

- Operational overhead with managed model
- Narrower selection than open peer sources

Differentiation:
Instead of unfettered peer exchange, intermediated by a fixed central authority.

Solution Chosen: Approach 1
Based on a total of 25 random user interviews, an overwhelming number of users stated that they would like a mobile application developed for this problem rather than a brick & mortar store or digital kiosk layout. The percentage are as follows: Mobile App - 17/25(68%), Digital Kiosk - 2/25(8%), and Brick & Mortar - 6/25(24%).



Use Cases:

Approach 1: Peer-to-Peer Mobile App

Student Seller:

1. Mark just purchased a new laptop and wants to sell his old one to another student
2. He downloads the app, verifies his student credentials, and lists his laptop for sale including photos, specs, and condition notes
3. Mark hopes to use the app for the ease of adding an item with minimal details needed
4. An interested buyer messages Mark to ask some additional questions and confirm functionality
5. They arrange a meetup on campus where the buyer can test everything before completing the sale securely in the app

Student Buyer:

1. Sarah needs a specific textbook for her engineering course that is quite expensive new
2. She opens the app and filters textbook listings to find several used options from other students
3. Sarah wants to have a tailored shopping experience where he sees what he likes on his feed
4. Sarah wants to be able to save the items she likes to look at for a later time
5. Sarah messages one seller to get more information about textbook edition, highlights, etc.
6. After confirming it meets her needs, she schedules a quick meet in the engineering building to pick it up and pay securely via the app.

Approach 2: Digital Kiosk Marketplaces

Seller:

1. James finished his semester and wants to list his microeconomics textbook for sale locally rather than deal with shipping
2. He visits the campus resale kiosk, verifies as a student, and lists the book for sale including all relevant details
3. James hands the textbook to the kiosk attendant to catalog and store for the listing period and gets a receipt
4. When a buy order comes through, he receives an alert to collect his payout from the kiosk

Buyer:

1. Lidia sees a film studies textbook she needs listed on the resale kiosk platform from another student
2. She claims the listing on the kiosk and pays for the book, entering her student ID
3. The kiosk prints a receipt confirming her digital purchase
4. Lidia shows the QR code on her receipt when she swings back by later to collect the textbook

Approach 3: Managed University Thrift Store
Student Consigner:

1. AJ is moving out of his apartment near campus and looks to sell possessions he no longer needs
2. He brings decorative furniture, small appliances and kitchenware to the student thrift store
3. The store manager assists AJ in cataloging his items and assessing possible resale pricing/fees
4. AJ formally consigns the agreed upon goods to be displayed, stored and sold by the store manager
5. Every month AJ receives his consignment profit share from sold items as payments

Student Shopper:

1. Christine browses the racks and shelves of the university resale store for dorm room decor
2. She finds some fun looking lamps, wall hangings and pillows in great shape from previous students
3. As a student, Christine receives a discount at checkout on top of the already affordable used prices
4. She shows her active student ID and pays for the thrifted decor items, helping sustain the store nonprofit mandate

Updated Domain Research Interviews:

- Interviewed BGThrifts and GTShopt
    - These are thrift organizations on campus at Georgia Tech. They host thrifting events once a month where they bring a bunch of unwanted materials to tech green to thrift
    - They mentioned that they had issues in terms of:
        - 1. People want more thrifting events
        - 2. People want to sell their own individual items but these clubs can't handle it
    - They mentioned that an online marketplace would really help solve these issues
        - People can have their own apartment act as their own warehouse and sell from their apartment
        - People can buy/sell whenever they want by just using their app.

Learning Prototype:

Peer-to-Peer Mobile App (Approach 1)
Unknowns:

- Will students actually adopt and regularly utilize such a specialized exchange platform?
- What inventory visibility threshold needs to be reached for necessary liquidity?
- Can we craft reliable identity confirmations and fraud protections?

Learning Prototypes:

- Landing page with explainer video to gauge interest

- Wizard of Oz MVP with manual verification and limited inventory
- User surveys evaluating various value prop messaging

Kiosk Marketplaces (Approach 2)
Unknowns:
- Will students take time to visit physical kiosk locations for transactions?
- How much logistics support responsibility should fall to kiosks?
- What categories beyond textbooks have highest exchange potential?

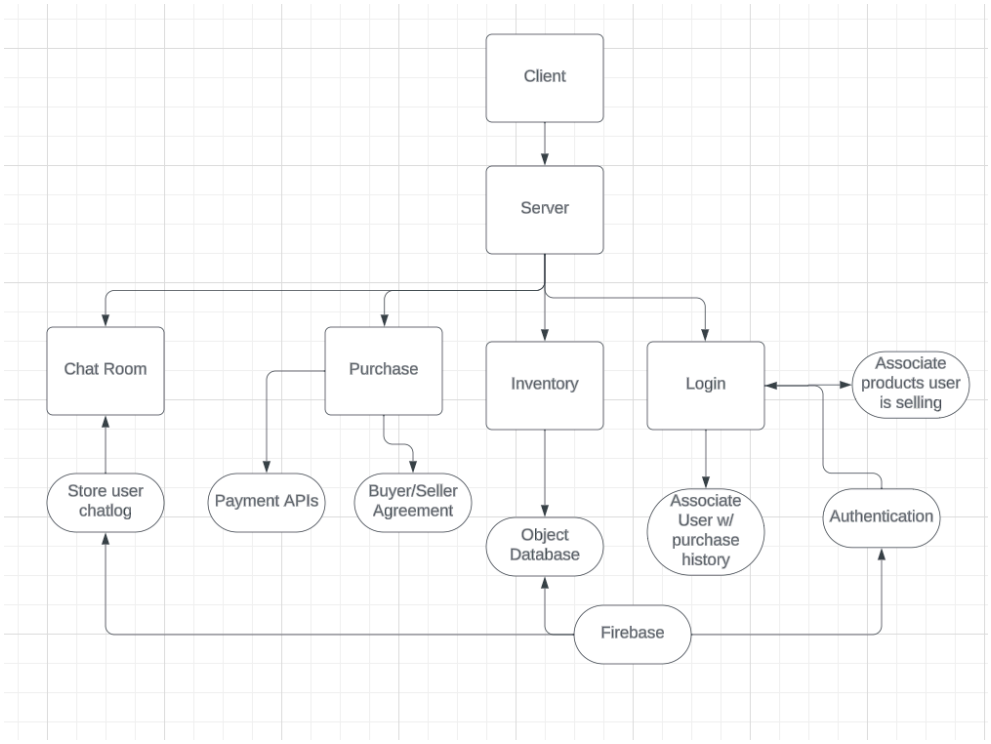
Learning Prototypes:
- Campus intercept interviews on theoretical concept
- Tablets demonstrating kiosk UX for ease-of-use testing
- Focus groups evaluating potential inventory types

University Thrift Store (Approach 3)
Unknowns:
- Would students prefer consigning vs direct peer exchanges?
- Can suitable retail space exist on prime campus real estate?
- Would administration sponsor store given bureaucratic hurdles?

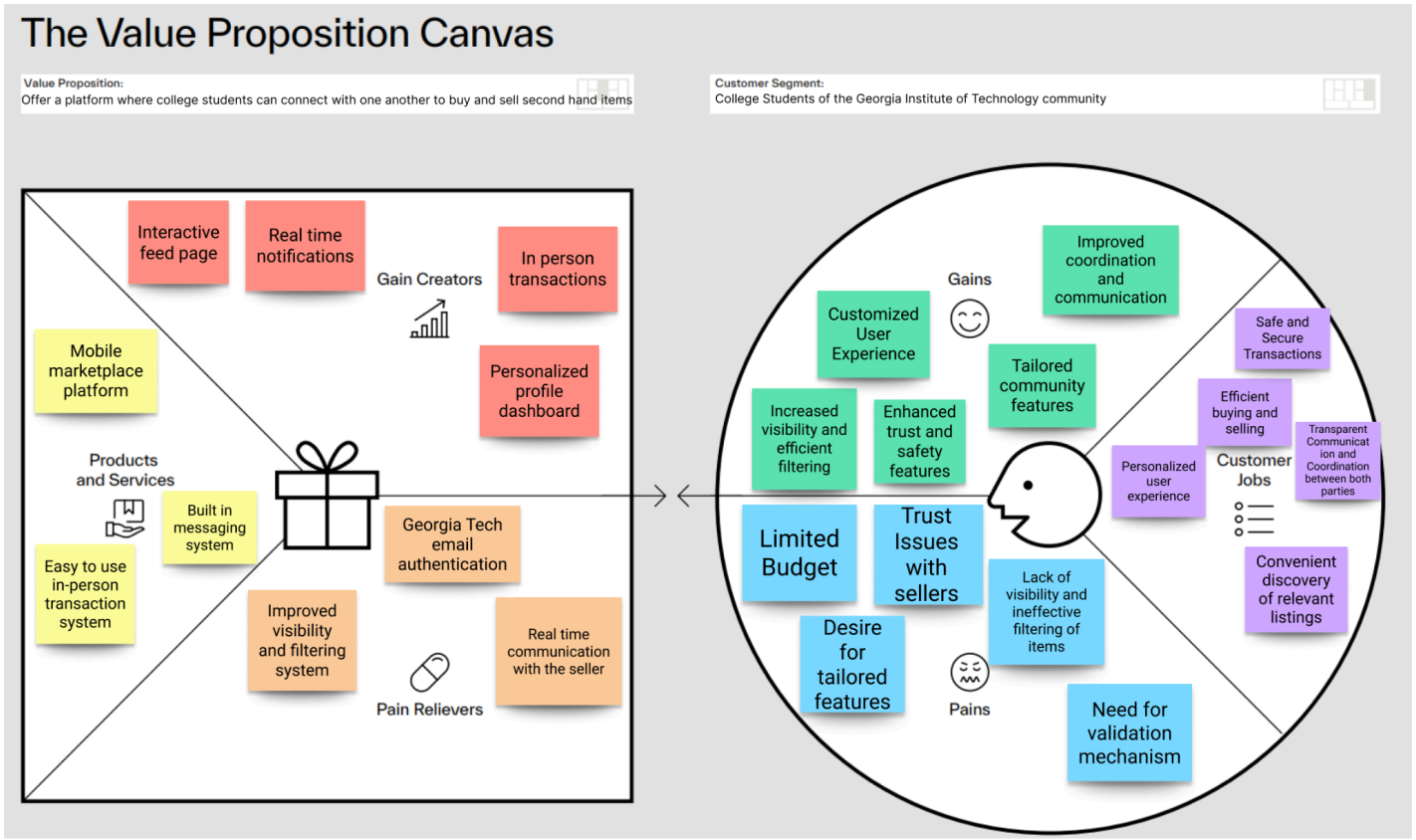
Learning Prototypes:
- Student surveys gauging consignment model appeal
- University decision maker discussions on endorsement openness
- Thrift pop-up concept to validate assortment and shopping experience

Technical Discussion:

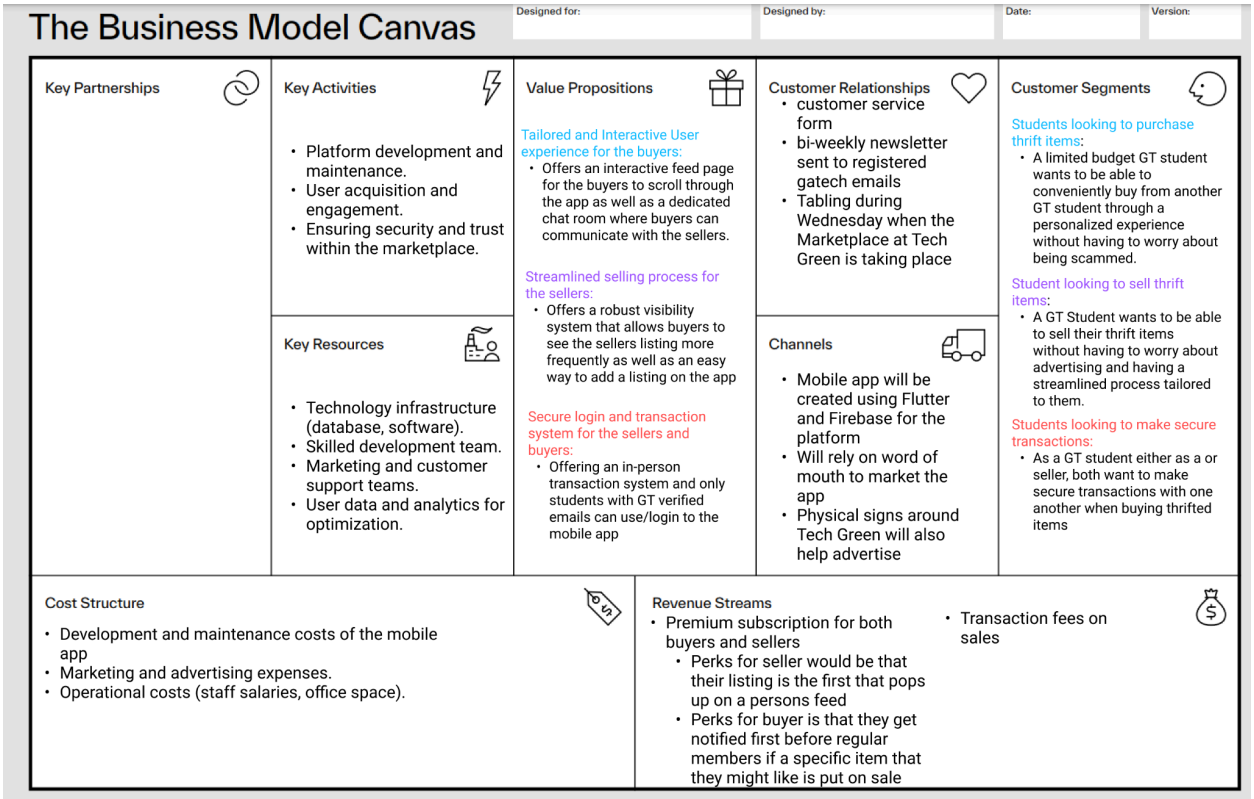
- In regards to the platforms, data flows, and the architecture of this app, we see there to be 4 distinct concerns for the app to work properly.
    - Product Inventory: We plan on utilizing Firebase Firestore as our database to store products, and we will query from Firebase in our Flutter code in order to pull the products. This is to be implemented, as we focused on sprint 2 to be the basis for our app. Thus we built the product page display logic, and have some hardcoded products for now. (https://firebase.google.com/docs/firestore)
    - Login Page: This is mostly finished, except for some elements of the UI that we have in mind to change. We allow new users to sign up with an email and password, or their google account. The authentication is also supported by Firebase.
    - Purchasing: For our payment APIs, we plan on utilizing the Apple Pay and Google Pay APIs, as well as a conventional payment API in which users can write their credit card details.
    - Below is a high level architectural design of our app.



Value Proposition Canvas:



BMC (Includes VPC):



The Business Model Canvas

Designed for: | Designed by: | Date: | Version:

Key Partnerships

Key Activities

- Platform development and maintenance.
- User acquisition and engagement.
- Ensuring security and trust within the marketplace.

Key Resources

- Technology infrastructure (database, software).
- Skilled development team.
- Marketing and customer support teams.
- User data and analytics for optimization.

Value Propositions

Tailored and Interactive User experience for the buyers:

- Offers an interactive feed page for the buyers to scroll through the app as well as a dedicated chat room where buyers can communicate with the sellers.

Streamlined selling process for the sellers:

- Offers a robust visibility system that allows buyers to see the sellers listing more frequently as well as an easy way to add a listing on the app

Secure login and transaction system for the sellers and buyers:

- Offering an in-person transaction system and only students with GT verified emails can use/login to the mobile app

Customer Relationships

- customer service form
- bi-weekly newsletter sent to registered gatech emails
- Tabling during Wednesday when the Marketplace at Tech Green is taking place

Channels

- Mobile app will be created using Flutter and Firebase for the platform
- Will rely on word of mouth to market the app
- Physical signs around Tech Green will also help advertise

Customer Segments

Students looking to purchase thrift items:

- A limited budget GT student wants to be able to conveniently buy from another GT student through a personalized experience without having to worry about being scammed.

Student looking to sell thrift items:

- A GT Student wants to be able to sell their thrift items without having to worry about advertising and having a streamlined process tailored to them.

Students looking to make secure transactions:

- As a GT student either as a or seller, both want to make secure transactions with one another when buying thrifted items

Cost Structure

- Development and maintenance costs of the mobile app
- Marketing and advertising expenses.
- Operational costs (staff salaries, office space).

Revenue Streams

- Premium subscription for both buyers and sellers
  - Perks for seller would be that their listing is the first that pops up on a persons feed
  - Perks for buyer is that they get notified first before regular members if a specific item that they might like is put on sale
- Transaction fees on sales

Feature Analysis:

Features that we would like to add to Emporia Include:

- Favorites Section: Users can like certain products from the marketplace. This would allow them easy access to the items they like the most, along with the contact of the seller. This is very important as it allows users another way to interact with products on the app and allows them to save items which they may show to other individuals, which can increase our user base.
- Search Bar: Users would be able to search for an item, and a list of related items would appear from the searchbar. This is also very important, so that users have the ability to quickly see and search the app for what they are looking for.
- Chat Room: Have a private chat room between the user and seller so that they can talk more about the product. This is extremely important as it allows the buyer and seller to talk to each other privately regarding any negotiations or where to meet for the purchase.
- User Agreement Logic: What we intend to happen is that the buyer and seller meet in person, and they can verify each other through some sort of authentication process (could be through a 4 digit code like Uber, or scanning personal QR codes), and from this only then will the payment option open up on the app. From there, the buyer and seller can complete the transaction. This would be very important as it makes sure that the product is real, the buyer saw it in person, and that neither side of the transaction is fraudulent.

- Payment API: Develop a safe payment scheme through secure payment APIs (Venmo/Apple Pay/Google Pay) - This is very important as this is how we see ourselves making money.

Learning prototype results:

Prototype Description:
We created a clickable mobile app prototype demonstrating the core peer-to-peer product listing flows. This enabled basic capability testing around key actions like posting an item for sale, browsing listings, and contacting sellers.

Methodology:
We conducted 6 usability test sessions with students matching our target demographics. Sessions lasted 30-45 minutes and involved free exploration of flows as well as defined tasks. We asked testers to verbalize thoughts while using the prototype and followed up with specific questions on areas needing input.

Key Feedback & Learnings:

- 4 of 6 participants easily completed posting an item for sale without assistance
- 3 users tried unsuccessfully to enlarge product photos
- "Connect" button expected to trigger in-platform messaging vs phone/email

Takeaways:

- Core listing task flows demonstrate adequate usability fit
- Additional tutorials may further boost user performance
- Messaging capabilities should integrate platform inbox for streamlined discussions

By isolating key interactions around our consignment model in this prototype, we can confirm foundational elements while collecting pointed improvement areas to address in subsequent versions. We plan to expand future tests to transaction and account management flows with this initial validation complete.

Learning prototype plans - Explain the purpose of your *next learning prototype*. Discuss what features you have considered, which are most important, which are not, and why you have come to this conclusion. Explain what questions you intend to answer with this prototype. **This section should be updated for all Sprints.**

Learning prototype plans:

- Chat Room
    - We plan on creating a chatroom where users can login and communicate with another user in regards to a product.
    - They can negotiate over the product
        - Login and talk about the product and save their conversations
- Georgia Tech Student Authentication Login
    - Login with GT email so that it is specifically catered towards GT students with @gatech.edu at the end of their emails.

        - By logging in with GT emails, people will be guaranteed to Georgia Tech students
        - Might need to find a way to further differentiate between a student and professor as professors also have @gatech.edu in their email.
- Scrollable dashboard
    - A scrollable dashboard that has all relevant items related to a user's search.
        - Save the spot in the search so that a user can see the items that they have already looked at and be updated with new relevant listings that release
        - Have it so that the listing will pop up towards the user to make it a interactive experience
- Shopping Cart system
    - A user can like a product listing which will add it to a shopping cart for them
        - From there the user can look at what they liked and decide whether or not to contact the buyer through the dedicated chat system
- Have an in-person transaction system
    - Once users meet up in person they should be able to exchange money and goods with one another
    - Thinking of implementing a 4 digit code, qr code, or personalized chimes system to allow for this feature

Note: The purpose of our next learning prototype is to create a prototype with a user friendly UI that has the capabilities of allowing a buyer and seller to be able to communicate with one another via integrated chat room. Above are the features we have considered for our next prototype. The features that we believe are most important to be implemented is a scrollable dashboard, chat room and GT email authentication. This is because those features would answer questions such as how would the buyer see listings from sellers, how would the app ensure that only GT affiliated people are using the app, and how would the two parties be able to communicate with one another. We came to this conclusion because while having a shopping cart and like system is a feature that we are looking to also implement later on they are not necessary for the main functionalities of our app which is to allow GT students to buy and sell from one another through a streamlined platform with ease.

Biggest concerns updated:

- Some of our biggest concerns are regarding the actual payment between the buyer and the seller.
- How to develop an interactive UI that is tailored specifically toward the Georgia Tech student population
- How do we ensure data privacy? A stalker might buy a product from someone just so they can meet them in a weird location. We need to ensure that we have enough stability to prevent this.
- Scalability. We need to ensure that there is some way that we can expand to other universities later down the line. We cant put all of our eggs into one basket
- We need to consider the environmental impact of this application in the sense that people shouldn’t just buy things and then thrift them to rip people off.

- How to verify that the person using the App is a GT student and not a random person on the internet or around campus.
- ~~State Management within the App - making sure that states are properly handled so that input and changes are reflected upon the entire system, not just a momentary instance of it~~

Link to Figma Mockups and Value Proposition Canvas:
https://www.figma.com/file/dt2k7IiqvLixZUzhw5ru3x/Mockup-1?type=design&node-id=23%3A107&mode=design&t=BUFtMKEGqeuGeGnk-1

Link to Github:
https://github.com/sviswanatha5/EmporiaApp

Queastion

Sprint 3

Team Agreement:

- Will meet biweekly either in person(CULC) or on a remote call(Discord/Zoom) on Tuesdays and Thursdays and weekends if need be
- Let other members know if you cannot make it to the meeting or will be late
- Respond promptly to texts within the group chat
- As an individual update other members on the progress you are making throughout the sprint
- Let other members know if you are struggling to complete a section so that they can help

Problem Overview

Based on initial customer discovery interviews and research, a major pain point for college students is the inability to easily and efficiently buy and sell used goods amongst their peers on campus. Key challenges uncovered include:

- Extremely low textbook buyback rates from campus bookstores (<25%) compared to potential direct peer resale value (50%+)
- Difficulties finding interested buyers for niche secondhand student gear via fragmented platforms
- Lack of tailored protections addressing fraud risks transacting high-value goods anonymously
- No existing solution purposefully aggregating all student buyers/sellers for alignment

These shortcomings stem from current generalized marketplace offerings failing to cater to the unique student use case specifically. Mainstream apps do facilitate some peer transactions but without functionality serving core university populations needs around verification, notifications, item cataloging/pricing, on-campus logistics, etc. ~~There is not an online thrift store dedicated just for college students. There are plenty of pop up thrift markets and other thrift sites but nothing that is generalized towards specific college audiences.~~ College students lack a secure and user-friendly online platform tailored specifically for their thrift shopping needs. While there are numerous pop-up thrift markets and general thrift websites available, none cater specifically to the unique preferences and requirements of college students. This complicates their search for affordable and trendy clothing and items. Additionally, significant value is being stranded between students owing to obscured visibility and stifled platform optimization for their precise goals. Specialization could better support this meaningful yet often constrained ecommerce.

User Persona: Michael
Background: Michael is a 21 year old male student at Georgia Tech. He is interested in selling some of his T-shirts and buying used ones from other GT students. Michael does not have a lot of time to go to in person thrift shops and has a limited budget. He is also weary of scammers.
Why Michael will choose to use Emporia:

- Wants a quick way to see what other GT students are selling
- Has a dedicated chatroom for buyers and sellers to setup a time to meet
- Will make use of the dedicated “For You Page” in the app

- Won't be concerned about scammers since the only people using the app are GT verified
- Able to sell and buy more secondhand items than just shirts

Competitive Analysis

Top existing marketplace platforms students currently use include Facebook Groups/Marketplace, GroupMe chats, and Craigslist. However, as outlined these contain no tailored mechanisms for identity confirmation or inventory consolidation exclusive to campus populations. Niche players like textbook swap sites and college classifieds have fragmentation challenges. Major platforms also offer little specialization around peer transaction safety, logistics, and pricing dynamics.

This presents a strategic opening to address direct university stakeholder problems through an integrated, compliant mobile ecommerce solution leveraging verifications, analytics, and community activation to unlock more value. Another potential competition that we might see is in the form of traditional social networking. Platforms used mainly for social media such as Instagram have started to turn into miniature marketplaces as sellers create a "post" of their item with contact information below. The biggest reason that this would be a competition is due to the fact that platforms like instagram have a high amount of users. This is something that we will struggle with at the initial stages of our approach.

Another competitor we have looked at is TikTok and how their platform manages to keep users online for hours on end. The scrolling feature the app provides is the main drawpoint of the app as it specifically tailors the 'For You' page of the application for the user and that the scrolling feature of the app is what makes it addicting to keep on using. The easy to use functionalities and tailored feeds are what keeps TikTok as one of the most used social media apps in the App Store.

Additionally, Instagram Reels also has a similar for you page functionality where it shows users videos that they want based on their preferences. It is as simple as liking a post or another reel that would cause a shift in the content that instagram shows to the end user. To this end the app manages to keep users engaged for hours on end.

Solution Approach:

Approach 1: Peer-to-Peer Mobile Marketplace App

- Dedicated mobile app allowing students to list items for sale and browse listings posted by other students
- Verification process checks enrollment status via university system integration
- Ratings systems provides feedback on buyers/sellers to build trust
- Custom push notifications for new relevant listings
- Integrated campus logistics coordination and payments

Pros:

- Tailored specifically to college use cases
- Addresses inventory fragmentation and verification issues
- Enables feature optimization around notifications, item cataloging, etc.

Cons:

- Significant dev work for custom built app
- Getting user base initial traction and network effects
- Manual verifications don't scale

Differentiation:
Specialized mandate solely focused on serving students vs mainstream platforms. Builds trust and unlocks existing supply chains between classmates.

Approach 2: Digital Kiosk Marketplaces

- Network of digital kiosks located around campus where students can list/browse listings and arrange exchanges
- Kiosks manage inventory, coordinate scheduling for testing, provide storage, etc.
- Integrated identity verification and payments

Pros:

- Brick-and-mortar outlet creates additional trust
- Logistics handled through central depot

Cons:

- Accessibility limits flexibility
- Higher overhead than pure software solutions

Differentiation:
Physical retail presence plus digitally-powered exchange creates unique centralized hybrid.

Approach 3: University Thrift Store Model

- Main campus storefront exclusively buys/sells secondhand student goods
- Handles inventory, valuations, merchandising
- Students consign their items and get payouts when sold

Pros:

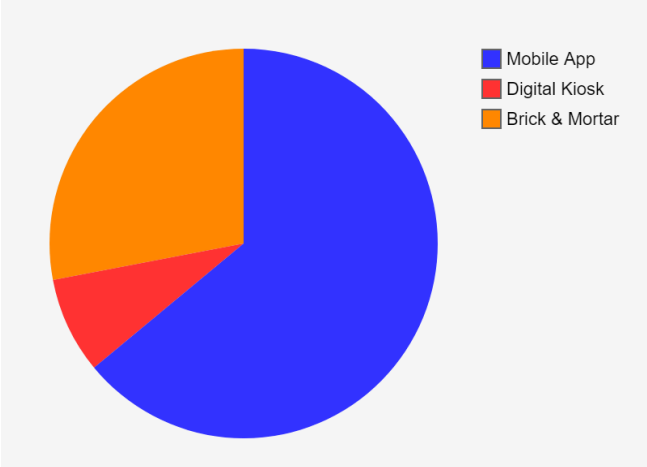
- No direct peer-to-peer risks
- Creates retail experience

Cons:

- Operational overhead with managed model
- Narrower selection than open peer sources

Differentiation:
Instead of unfettered peer exchange, intermediated by a fixed central authority.

Solution Chosen: Approach 1
Based on a total of 25 random user interviews, an overwhelming number of users stated that they would like a mobile application developed for this problem rather than a brick & mortar store or digital kiosk layout. The percentage are as follows: Mobile App - 17/25(68%), Digital Kiosk - 2/25(8%), and Brick & Mortar - 6/25(24%).
How this solution differs with others on the market is that it is an interactive



Use Cases:

Approach 1: Peer-to-Peer Mobile App

Student Seller:

6. Mark just purchased a new laptop and wants to sell his old one to another student
7. He downloads the app, verifies his student credentials, and lists his laptop for sale including photos, specs, and condition notes
8. Mark hopes to use the app for the ease of adding an item with minimal details needed
9. An interested buyer messages Mark to ask some additional questions and confirm functionality
10. They arrange a meetup on campus where the buyer can test everything before completing the sale securely in the app

Student Buyer:

7. Sarah needs a specific textbook for her engineering course that is quite expensive new
8. She opens the app and filters textbook listings to find several used options from other students
9. Sarah wants to have a tailored shopping experience where he sees what he likes on his feed
10. Sarah wants to be able to save the items she likes to look at for a later time
11. Sarah messages one seller to get more information about textbook edition, highlights, etc.
12. After confirming it meets her needs, she schedules a quick meet in the engineering building to pick it up and pay securely via the app.

Michael is looking at t-shirts he wants to get rid of	Mike is looking through his for you page to find a t-shirt to buy	Mike is looking through his for you page to find a t-shirt to buy	Mike schedules a meeting with the buyer through Emporia's chat feature	Michael and Mike meet with each other to make the exchange

Approach 2: Digital Kiosk Marketplaces

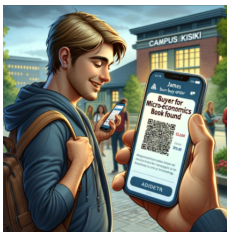
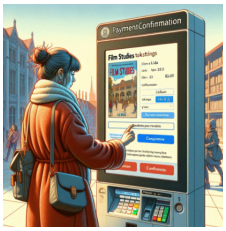
Seller:

5. James finished his semester and wants to list his microeconomics textbook for sale locally rather than deal with shipping
6. He visits the campus resale kiosk, verifies as a student, and lists the book for sale including all relevant details
7. James hands the textbook to the kiosk attendant to catalog and store for the listing period and gets a receipt
8. When a buy order comes through, he receives an alert to collect his payout from the kiosk

Buyer:

5. Lidia sees a film studies textbook she needs listed on the resale kiosk platform from another student
6. She claims the listing on the kiosk and pays for the book, entering her student ID
7. The kiosk prints a receipt confirming her digital purchase
8. Lidia shows the QR code on her receipt when she swings back by later to collect the textbook

Seller	1.James finished his semester and wants to list his microeconomics textbook for sale locally rather	2.He visits the campus resale kiosk, verifies as a student, and lists the book for sale including all relevant details	3.James hands the textbook to the kiosk attendant to catalog and store for the listing period	4.When a buy order comes through, he receives an alert to collect his payout from the kiosk

	than deal with shipping		and gets a receipt	
Seller				
Buyer	1.Lidia sees a film studies textbook she needs listed on the resale kiosk platform from another student	2.She claims the listing on the kiosk and pays for the book, entering her student ID	3.The kiosk prints a receipt confirming her digital purchase	4.Lidia shows the QR code on her receipt when she swings back by later to collect the textbook
Buyer				

<u>Approach 3</u>: Managed University Thrift Store

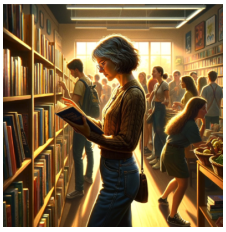
Student Consigner:

6. AJ is moving out of his apartment near campus and looks to sell possessions he no longer needs
7. He brings decorative furniture, small appliances and kitchenware to the student thrift store
8. The store manager assists AJ in cataloging his items and assessing possible resale pricing/fees
9. AJ formally consigns the agreed upon goods to be displayed, stored and sold by the store manager
10. Every month AJ receives his consignment profit share from sold items as payments

Student Shopper:

5. Christine browses the racks and shelves of the university resale store for dorm room decor
6. She finds some fun looking lamps, wall hangings and pillows in great shape from previous students

7. As a student, Christine receives a discount at checkout on top of the already affordable used prices
8. She shows her active student ID and pays for the thrifted decor items, helping sustain the store nonprofit mandate
9.

Student Consigner:	1.AJ is moving out of his apartment near campus and looks to sell possessions he no longer needs	2.He brings decorative furniture, small appliances and kitchenware to the student thrift store	3.The store manager assists AJ in cataloging his items and assessing possible resale pricing/fees	4.AJ formally consigns the agreed upon goods to be displayed, stored and sold by the store manager
Student Consigner:				
Student Shopper:	1.Christine browses the racks and shelves of the university resale store for dorm room decor	2.She finds some fun looking lamps, wall hangings and pillows in great shape from previous students	3.As a student, Christine receives a discount at checkout on top of the already affordable used prices	4.She shows her active student ID and pays for the thrifted decor items, helping sustain the store nonprofit mandate
Student Shopper:				

Updated Domain Research Interviews:

- Interviewed BGThrifts and GTShopt
    - These are thrift organizations on campus at Georgia Tech. They host thrifting events once a month where they bring a bunch of unwanted materials to tech green to thrift
    - They mentioned that they had issues in terms of:
        - 1. People want more thrifting events

        - 2. People want to sell their own individual items but these clubs can't handle it
    - They mentioned that an online marketplace would really help solve these issues
        - People can have their own apartment act as their own warehouse and sell from their apartment
        - People can buy/sell whenever they want by just using their app.
        -
- Interviewed Salvation Army
    - From your experience, what are the main reasons people donate items to the Salvation Army, and how might a college-focused thrift app impact these donations?
        - People want to do charity, this app might cause a reduction for charity
    - Can you discuss any logistical challenges related to managing donated items, and how might these apply to a college thrift platform?
        - Larger items during peak moving periods might be a concern because people might not handle it correctly / cut corners
    - What are the main challenges you face in ensuring donor and recipient privacy and security, and how could these issues manifest in a college-focused platform?
        - Some people don't come to a thrift store because they are scared of people judging what they buy
            - Thrift store app might help with this because people do solo transactions
        - **MAIN TAKEAWAY: Focus on this by highlighting privacy in the application**
- Interviewed Brittain Dining Hall
    - How might a college-centered social selling app impact the dining hall's operations or student behavior within the dining hall?
        - There could be concerns about students selling or trading meal plan items
            - This isn't outlined in the dining hall contract, should look intro contract outlines so that when the seller and buyer meet there are no more negotiations.
    - What measures does the dining hall currently have in place to prevent theft or misuse of meal plan items, and how might these measures need to adapt in response to the introduction of a social selling app?
        - Talked about monitoring systems in place to track meal plan usage and prevent abuse. She liked the idea for GT SSO login

        - In your experience, what are some common tactics or behaviors students engage in to circumvent dining hall rules or policies, and how might these translate to the use of a social selling app?
            - Students may attempt to share or sell meal plan items to others, which could lead to misuse of dining hall resources.
                - People might pose as other people/ use others accounts
                    - **MAIN TAKEAWAY: Should have a picture verification after items are traded**
-

Questions Answered:

Spent time interviewing random people in Tech square regarding our idea and asked 3 different individuals each question:

- How will the app handle disputes between buyers and sellers, such as issues with item condition, no-shows, or other conflicts?
    - Allow users to report issues directly within the app. This could include problems with item condition, disputes over transactions, or other concerns.
    - A team that would be responsible for gathering information from both parties, reviewing evidence (e.g., chat logs, item descriptions, photos), and making fair decisions based on the app's policies and guidelines.
- Do you plan to monetize the app, and if so, what revenue models are you considering? How will you balance generating revenue with providing a service that is accessible and valuable to college students?
    - Offer the app for free with basic features that fulfill the core needs of buying and selling among students. Then, introduce premium features that enhance the user experience, such as advanced search filters, increased listing visibility, or additional security measures.
    - Give coupons (operate at a loss) in the beginning so that we can users and then switch
- How do you plan to market the app to GT students and encourage its adoption? What channels and strategies will you use to reach your target audience?
    - Collaborate with campus organizations, clubs, and student government to promote the app.
    - We can host launch events, pop-up booths, or informational sessions at campus busy spots, such as the student center, dining halls, and major events like FASET or football games

Learning Prototype:

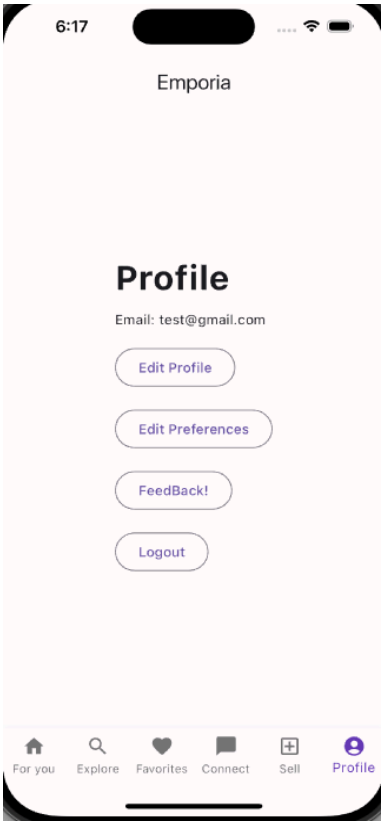
Link to demo: https://youtu.be/eOhB86eKfKw?si=XMEX76gCTjt46sU4

With this Sprint, we have created a mobile app concept as our learning prototype. The main question that we tried to answer in this learning prototype was how to keep users on the app. App retention is extremely important for us as it becomes more likely that a user will find a product that they like and will buy.

Thus, after understanding our users more, who are mainly college students, we decided to base our additions of this LP to be somewhat like Instagram. With Instagram, users can follow each other to see their posts specifically, and there's an explore section, which shows posts from everyone on the platform. Additionally, instagram has a very good retention rate of users, so we thought understanding and somewhat mimicking their model would hopefully increase our user retention as well. Thus, we decided that creating a "For you" page, in which users can subscribe to what they are interested in most, and having an explore page that would display the general market, would be the best approach to this problem of user retention.

The "things" users subscribe to in Emporia are product genres, of which we currently have 9 - Vintage, Tops, Bottoms, Tech, Jewelry, Accessories, Books, Shoes, and Decor. We got these 9 categories from previous user interviews at multiple thrift shops, in which we interviewed both buyers and thrift shop owners for the most common goods sold.

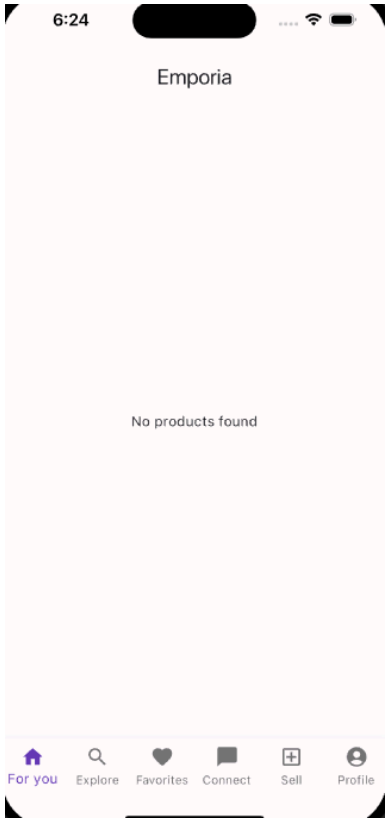
Users can update their preferences in the profile page as shown in the pictures (pic 1). Originally, this user did not have any preferences (pic 4), so they saw nothing in their for-you page. After updating it to include tops, bottoms, and accessories (pic 5), they see a sweater, hat, and shades for sale. :

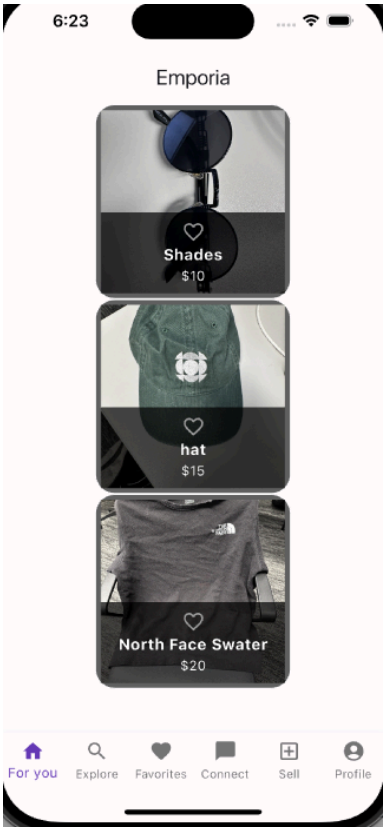

6:17
Emporia
Profile
Email: test@gmail.com
Edit Profile
Edit Preferences
FeedBack!
Logout
For you
Explore
Favorites
Connect
Sell
Profile


6:18
Pick Your Preferences
Vintage
Tops
Bottoms
Tech
Jewlery
Accessories
Books
Shoes
Decor
Update

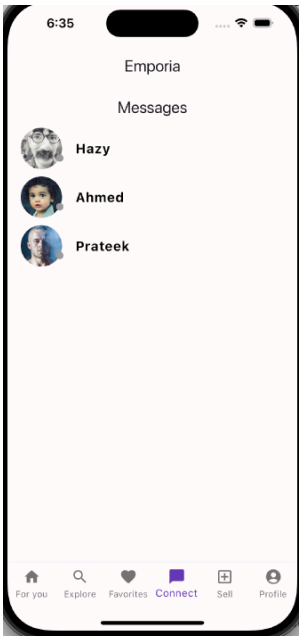

6:23
Pick Your Preferences
Vintage
Tops
Bottoms
Tech
Jewlery
Accessories
Books
Shoes
Decor
Update

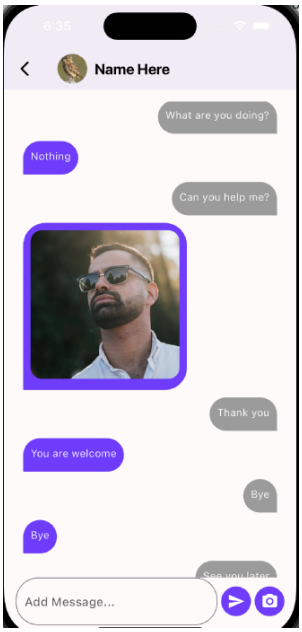
-


6:24
Emporia
No products found
For you
Explore
Favorites
Connect
Sell
Profile

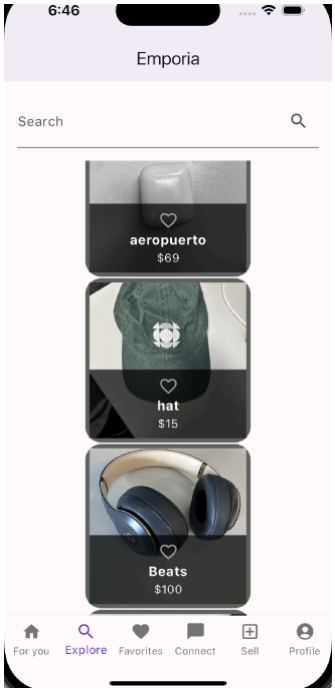

6:23
Emporia
Shades
$10
hat
$15
North Face Swater
$20
For you
Explore
Favorites
Connect
Sell
Profile

In addition, to keep users on, we discovered from our previous sprint that talking to the vendor about not only the product, but other things that they may be selling, can increase user retention on our app. Thus, we implemented a message feature on our app where users can connect with vendors for discussing about a product.



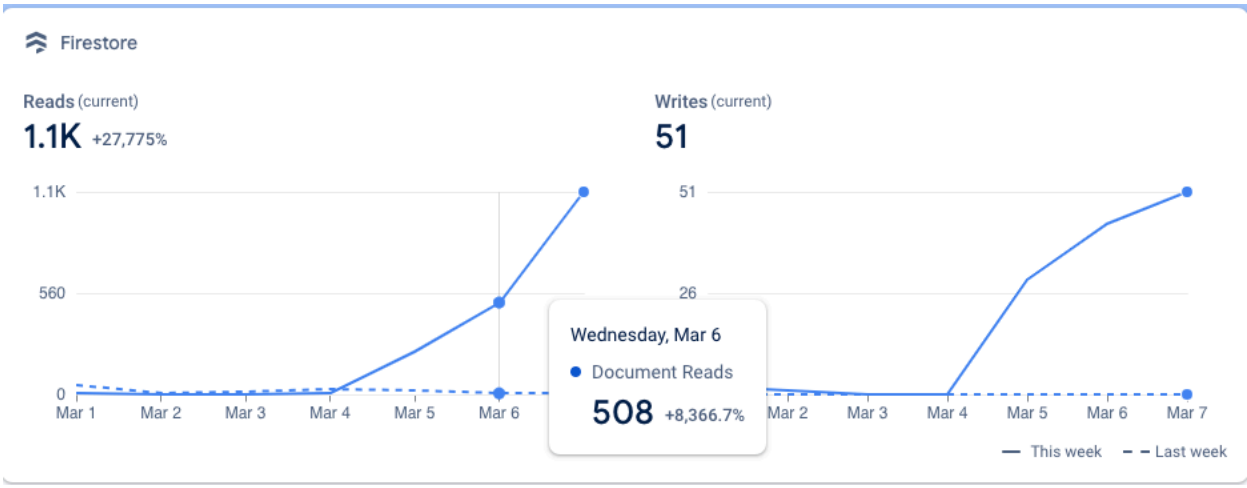


Next, for user retention we implemented a new explore feature. This will show all the marketplace products, even the ones that the user does not have their preferences set for. Additionally, users will be able to search for products and even vendors in the future. Thus, this is another way that we can keep users on the app, - with an explore page, they can scroll freely while also being able to see new products and new vendors.



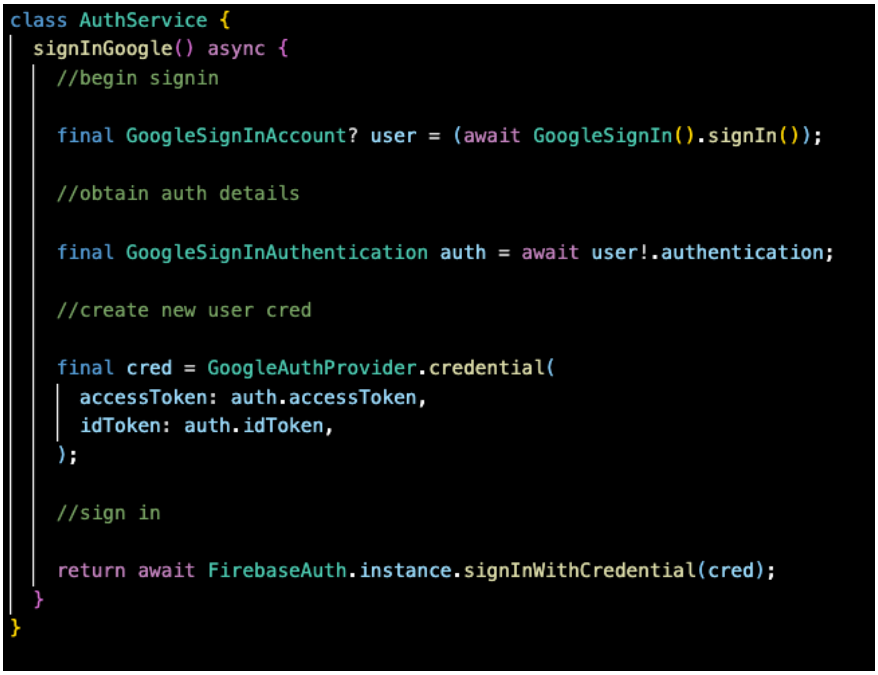
Technical Discussion:

- In regards to the platforms, data flows, and the architecture of this app, we see there to be 4 distinct concerns for the app to work properly.
    - Product Inventory: We have stored our product objects, including features such as its images and genres into cloud FireStore. We utilize consistent calls to Firebase in order to write and read the data from the database, making sure that our app updates in real-time.



- Login Page: The authentication is provided by Firebase Authentication. We also provide users the option of google authentication for ease, but we plan to remove this in the upcoming sprint. In order to make the app only available for Georgia Tech students, we will be verifying emails and making sure that they're .gatech@edu only.
- Purchasing: We plan on utilizing the Stripe API for payment, as it is an easy addition to Flutter and makes payment security very simple.
- Messaging: Being able to message vendors is crucial for our app. We currently have the foundation for the messages functionality. We need to store and link messages to cloud FireStore, and to each user. After doing this in the next sprint, we will be able to support messaging completely.
- Below is all of the instances in our code in which we call upon Firebase Auth and cloud Firestore to update our UI

This is for Google Authentication



```
class AuthService {
  signInGoogle() async {
    //begin signin

    final GoogleSignInAccount? user = (await GoogleSignIn().signIn());

    //obtain auth details

    final GoogleSignInAuthentication auth = await user!.authentication;

    //create new user cred

    final cred = GoogleAuthProvider.credential(
      accessToken: auth.accessToken,
      idToken: auth.idToken,
    );

    //sign in

    return await FirebaseAuth.instance.signInWithCredential(cred);
  }
}
```

This is for user sign-in for the login page.

```
try {
  await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: emailController.text, password: passwordController.text);

  FirebaseAuth user = FirebaseAuth.instance;

  List<bool> preferences = List.generate(9, (index) => false);

  FirebaseFirestore.instance
      .collection('users')
      .doc(user.currentUser?.uid)
      .set({
    'uid': user.currentUser?.uid,
    'email': user.currentUser!.email,
    'preferences': preferences,
  }, SetOptions(merge: true));
```

Adding a product to the database after the user creates it

```
void buttonLogic() async {
  await Future.delayed(const Duration(seconds: 2));

  Reference reference = FirebaseStorage.instance.ref();
  Reference refImages = reference.child('images');

  String imageFileName = DateTime.now().millisecondsSinceEpoch.toString();

  Reference uploadImage = refImages.child(imageFileName);

  try {
    await uploadImage.putFile(File(image!.path));
    imageUrl = await uploadImage.getDownloadURL();
  } catch (error) {}

  //debugPrint(priceController.toString());
  Product newProduct = Product(
      name: nameController.text.trim(),
      price: double.parse(priceController.text.trim()),
      description: descriptionController.text.trim(),
      vendor: FirebaseAuth.instance.currentUser!.email.toString(),
      isLiked: false,
      images: [imageUrl],
      id: "product${counter++}",
      productGenre: selectedGenres);

  addProduct(newProduct);
```

This is an example of retrieving user data in order to figure out which products to add on their “for-you” page

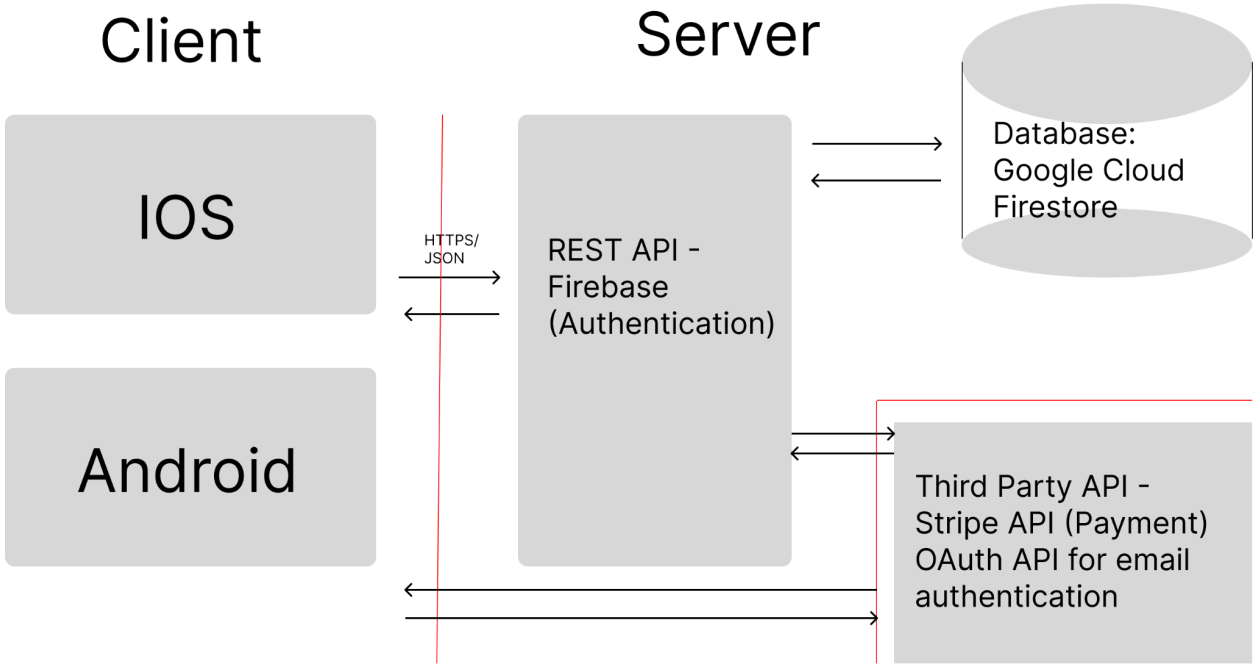
```dart
Future<List<bool>> getPreferences() async {
  String currentUserUid = FirebaseAuth.instance.currentUser!.uid;

  try {
    DocumentSnapshot userSnapshot = await FirebaseFirestore.instance
        .collection('users')
        .doc(currentUserUid)
        .get();

    if (userSnapshot.exists) {
      List<bool> preferences =
          List<bool>.from(userSnapshot['preferences'] ?? []);
      return preferences;
    } else {
      print('User document does not exist');
      return [];
    }
  } catch (e) {
    print('Error getting preferences: $e');
    return [];
  }
}
```

This is code registering a new user, storing their email and their default preferences.

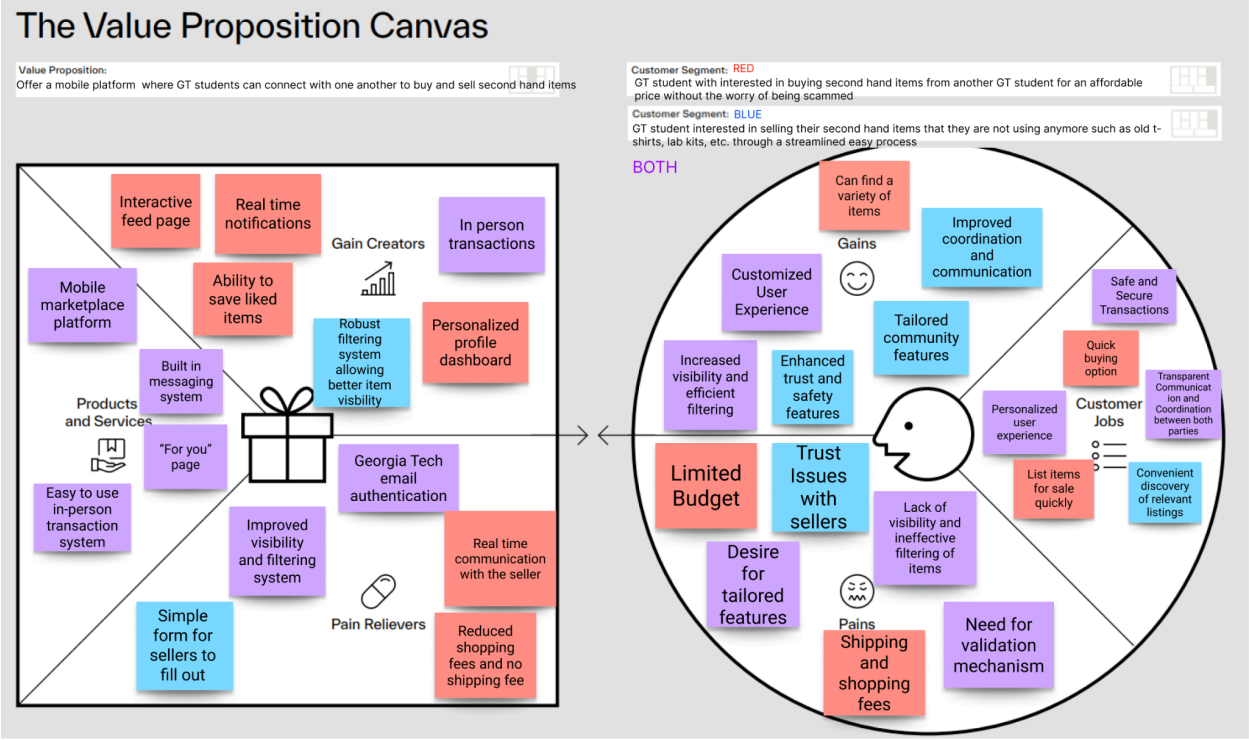
```dart
try {
  if (passwordController.text == confirmPasswordController.text) {
    await FirebaseAuth.instance.createUserWithEmailAndPassword(
        email: emailController.text.trim(), password: passwordController.text.trim());

    FirebaseAuth user = FirebaseAuth.instance;

    List<bool> preferences = List.generate(9, (index) => false);

    FirebaseFirestore.instance
        .collection('users')
        .doc(user.currentUser?.uid)
        .set({
      'uid': user.currentUser?.uid,
      'email': user.currentUser!.email,
      'preferences': preferences,
    });
  } else {
    //show error message that passwords aren't the same
    wrongInputMessage("Passwords don't match");
  }
```

- Below is a high-level architectural design of our app.

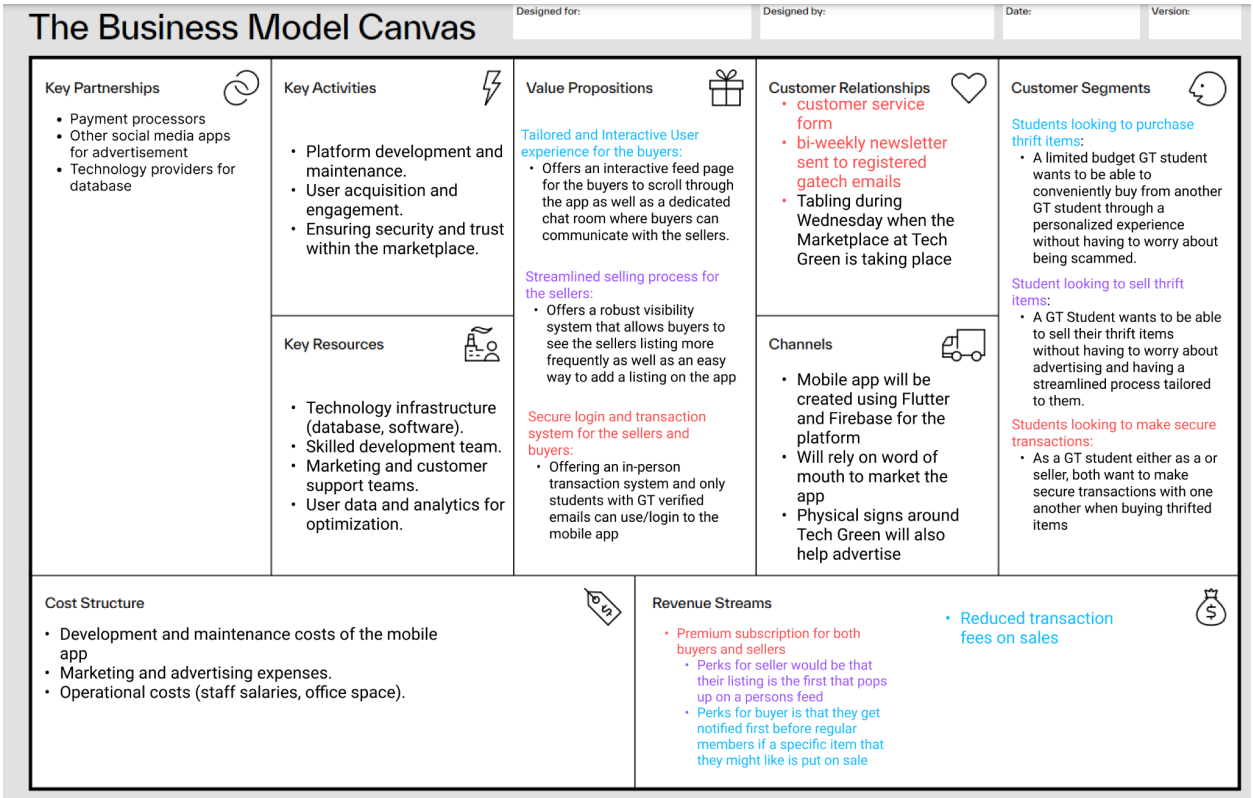
New architecture diagram:



Value Proposition Canvas: Updated to reflect customer segments color coded



BMC (Includes VPC):



Feature Analysis:

Feature analysis (Ranked):

1. Search Bar: We have ignored the implementation of the search bar for this sprint in order to develop the for you page. Now, we want a user to be able to search for the specific products that they want. Additionally, they would be able to search for certain vendors as well. In our research, we realized that the majority of vendors on our app would be from thrift stores or students with thrift business, as they have the most inventory.
2. Email Verification: Though we had this planned for this sprint, due to the length of the other tasks we have done, we were not able to complete this. This remains a high priority to us and will be completed in the next sprint, as we want to ensure that only GT students use this app.
3. New product showcase design: Currently, we display our products in a singular column. However, after hearing user feedback, we realized that this is not the best design for this app. Some of the ideas that we currently are looking into, from user feedback, would be either to display products on 2 columns, or to display them in a “flashcard”-esque way. This “flashcard” way would be like Tinder, in which a user would go to their for you page, see one item at a time, and have the ability to either pass upon the item or like it and connect with the vendor. The proper layout could lead to an increase in user retention, thus this is why this feature is so high up.
4. Favorites: We have ignored this feature for this sprint, and have to push it to a later sprint. It is implemented in our app, but not properly, meaning that the favorite button does not save state or save to a user. We need to connect the database to this in order for it to be fully functional.
5. Messages: We have the foundation for this feature, and we just have to connect it to Firestore. We can implement a contract feature where a special type of a message acts as an agreement between the seller and the buyer for a specific trade.
6. User Agreement Logic: What we intend to happen is that the buyer and seller meet in person, and they can verify each other through some sort of authentication process (could be through a 4 digit code like Uber, or scanning personal QR codes), and from this only then will the payment option open up on the app. From there, the buyer and seller can complete the transaction. This would be very important as it makes sure that the product is real, the buyer saw it in person, and that neither side of the transaction is fraudulent.
7. Payment API: Develop a safe payment scheme through secure payment APIs (Venmo/Apple Pay/Google Pay) - This is very important as this is how we see ourselves making money.

Learning prototype results:

Prototype Description:

We created a clickable mobile app prototype demonstrating the core peer-to-peer product listing flows. This enabled basic capability testing around key actions like posting an item for sale, browsing listings, and contacting sellers.

Methodology:
We conducted 30 usability test sessions with students on the GT campus, as well as individuals who work in GT Thrifting organizations. Sessions lasted 5-10 minutes and involved free exploration of flows as well as defined tasks. We asked testers to verbalize thoughts while using the prototype and followed up with specific questions on areas needing input. We took down these comments, while additionally asking questions regarding design, and what users would expect in a marketplace app.

Key Feedback & Learnings:

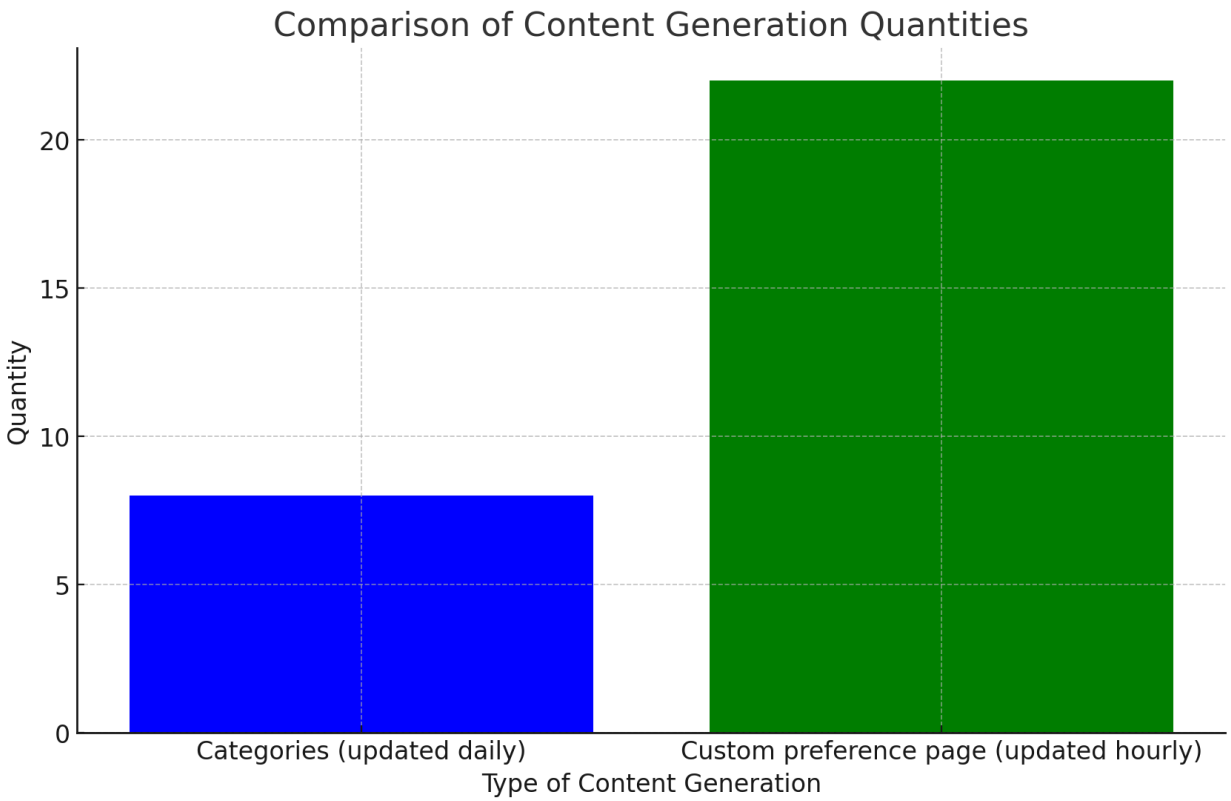
- Almost 90% of individuals thought that the UI can be better for the product icons
- 9 users instinctively swiped on photos, assuming to see more
- 5 users asked for more specific categories that they could list the product in
- 3 users commented on the time it takes to confirm that the product was added

Takeaways:

- Need to address UI issues
- Can try another design for the explore and the for-you page
- Messaging capabilities should integrate platform inbox for streamlined discussions
- Allow users to swipe to see more pictures on a product, and maybe even video
- New way to update preferences, instead of buttons maybe introduce the checklist

By isolating key interactions around our consignment model in this prototype, we can confirm foundational elements while collecting pointed improvement areas to address in subsequent versions. We plan to expand future tests to transaction and account management flows with this initial validation complete.

Research Data:

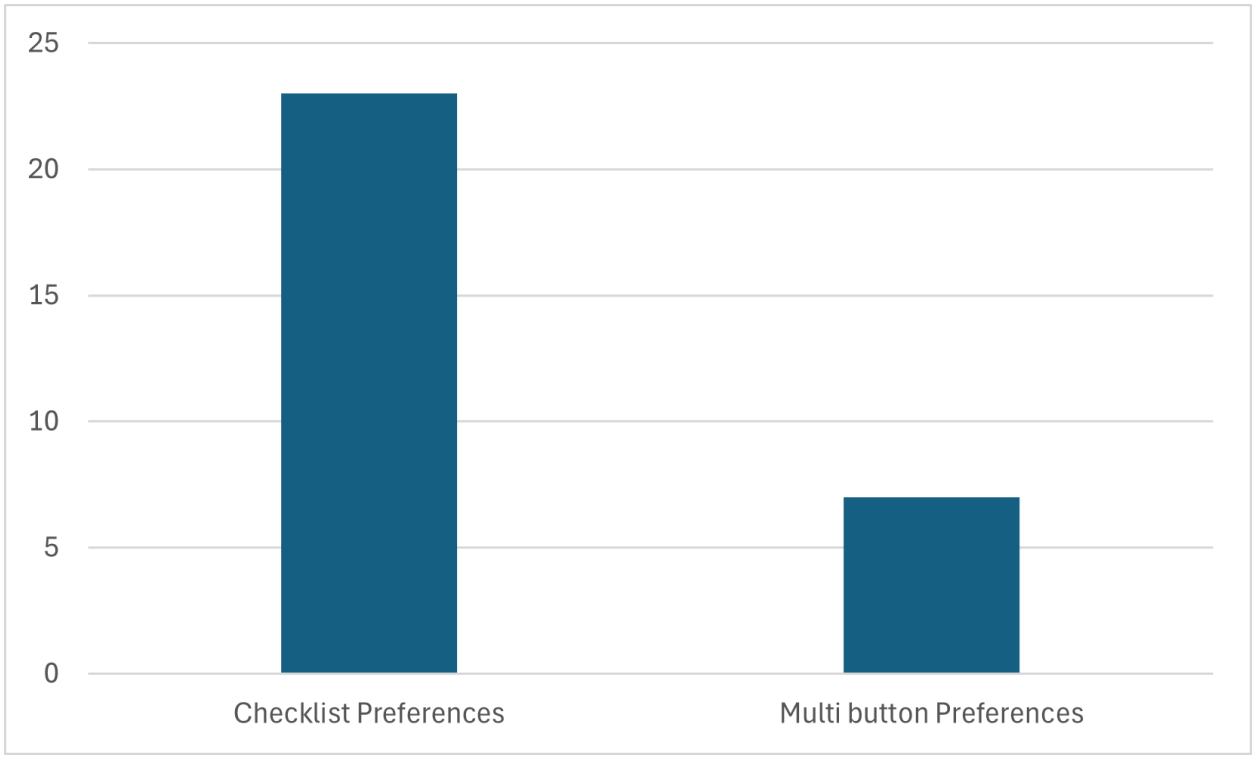


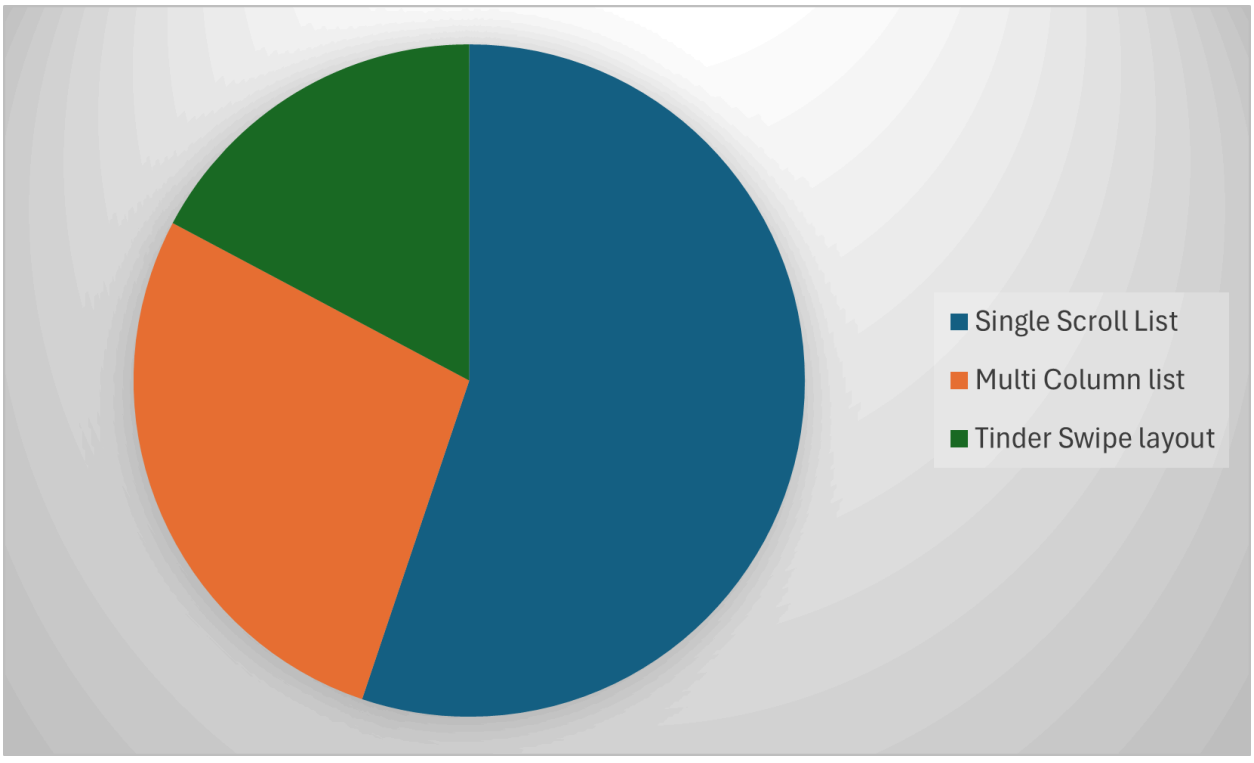
- Research was done with sample size 30 regarding the home page. The 2 options were a homepage that was customized directly with the users preferences. This would be updated hourly. The other option was a basic homepage that simply had a bunch of categories. This would be updated daily. After analyzing the data with this graph, it was obvious that many people preferred the custom preference page over the categorical page. For this reason, we made it our focus to work on the customizable for you page.

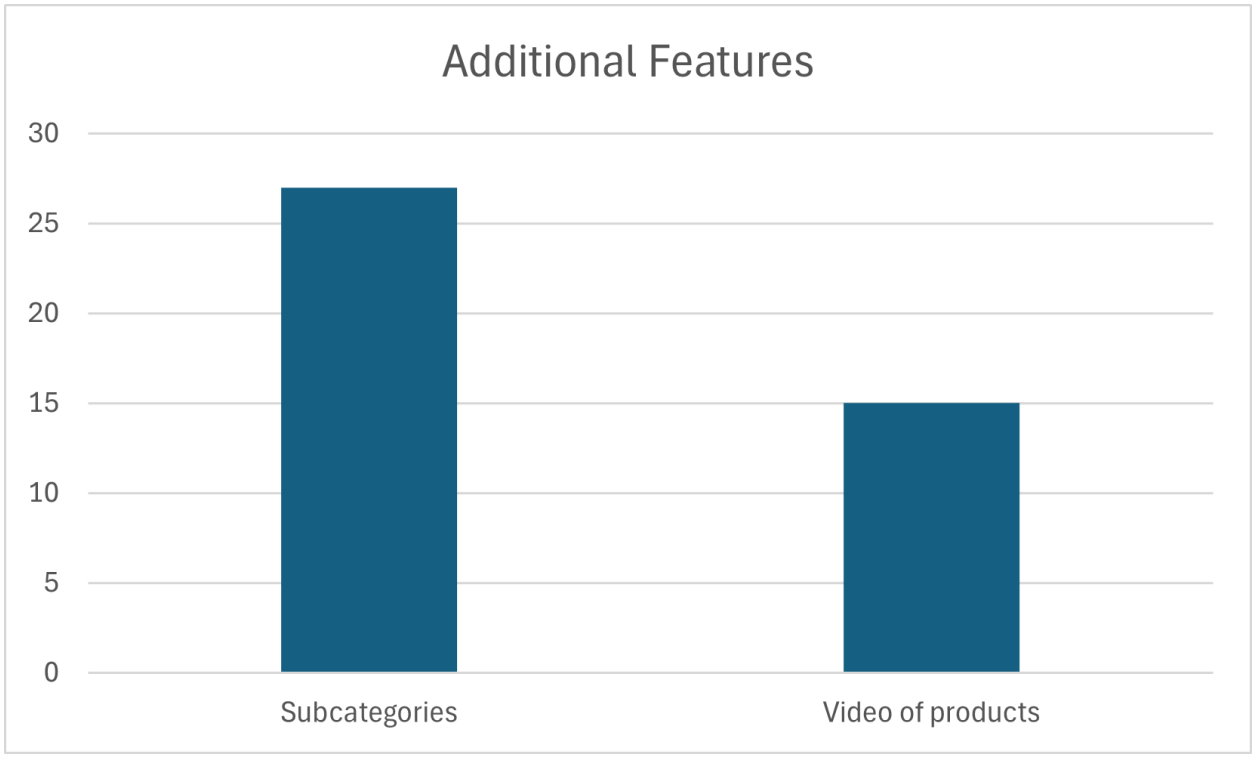
jn

- Furthermore, we did more research in terms of determining the exact age of our audience. We stood outside of Publix in midtown for 50 minutes and talked to every person that left the store. We got their age and asked them when the last time was when they went thrifting. Like we suspected, thrifting is something that appeals more to those under the age of 25 (college students). Another question that we asked that was quantifiable was whether an online alternative for thrifting would change their answer. Every single person said yes -which was something

that suprised us.





- 

Learning prototype plans For next sprint:

- Georgia Tech Student Authentication Login
  - Login with GT email so that it is specifically catered towards GT students with @gatech.edu at the end of their emails.
    - By logging in with GT emails, people will be guaranteed to Georgia Tech students
    - Might need to find a way to further differentiate between a student and professor as professors also have @gatech.edu in their email.
- Trying a new UI for displaying Products
  - After conducting research, there is a good amount of users who think using another design aside from the singular column stream for displaying products may be better. We will try to see another system
- SearchBar
  - Implementing a good search bar will make the app more useable for users. This will be a main priority for this sprint.
- Updating user preferences
  - From our research, we see that we need a different design to update user preferences than buttons. Maybe a checklist would be better
- Shopping Cart system

        - A user can like a product listing which will add it to a shopping cart for them
            - From there the user can look at what they liked and decide whether or not to contact the buyer through the dedicated chat system
- Have an in-person transaction system
    - Once users meet up in person they should be able to exchange money and goods with one another
    - Thinking of implementing a 4 digit code, qr code, or personalized chimes system to allow for this feature

Biggest concerns updated:

- Some of our biggest concerns are regarding the actual payment between the buyer and the seller.
- How to develop an interactive UI that is tailored specifically toward the Georgia Tech student population
- How do we ensure data privacy? A stalker might buy a product from someone just so they can meet them in a weird location. We need to ensure that we have enough stability to prevent this.
- Scalability. We need to ensure that there is some way that we can expand to other universities later down the line. We cant put all of our eggs into one basket
- We need to consider the environmental impact of this application in the sense that people shouldn't just buy things and then thrift them to rip people off.
- Quality Control: Ensuring that the quality of the items is exactly as promised in the contract. There needs to be some type of management in order to ensure that new items are actually in "new" condition and soforth.
- Dispute Resolution: If there is a dispute between the seller and the buyer, there needs to be a resolution that satisfies both parties to ensure customer satisfaction.
- Scams: Some products might look like real products, for expensive items we should have a authentication service that ensures that these products are as they are marketed.
- Ensuring that only GT students can use the app requires a verification system. There's the challenge of keeping the platform secure from non-GT users and impersonators. The risk of personal data breaches or identity theft is dangerous because they could scam the other party

Other Questions Left to Answer:

- How will we monetize the market and incentivize users to use our application for a fee
- How will we start the application (need items for sellers to buy first)
- How will we provide security for the application in terms of payment (cash is unsecure)
- How will we account for missed meetings (seller/buyer forgets to come to agreed location at agreed time)

Link to Figma Mockups and Value Proposition Canvas:
https://www.figma.com/file/dt2k7liqvLixZUzhw5ru3x/Mockup-1?type=design&node-id=23%3A107&mode=design&t=BUFtMKEGqeuGeGnk-1

Link to Github:
https://github.com/sviswanatha5/EmporiaApp

Feedback: metrics need to be collected. Automated collection. Shoe size feature. Might take too long to search, time item was added, probability of user clicking item that has almost perfect preference matches

- Timestamps for when the item is posted(Chris)
- UI Overhaul (Anyone)
- Changes to BMC
- Problem description and summary can mostly stay the same
- Search bar(Srikar)
- Figure out how to use Google Firebase/Analytics to conduct A/B testing
- Need quantified user tests not user interviews
- Convert the item preference boxes to a checklist format (Could maybe use this for A/B testing)=

Sprint 4

Main Goals:

- Develop the UI more
- Add functionality to search bar
- Add email authentication
- Add timestamps to item descriptions to show how recent they were posted
- Add any changes to the BMC and VPC accordingly from Sprint 3 feedback
- Conduct A/B testing
- Change preference selection from squares to a checklist

Questions answered from last sprint:

- How should the UI be improved?
    - Conducted A/B testing on the UI focusing on color contrast, product square placement, and preview description of product
- How will the email authentication system be implemented?
    - Used Google Firebase Authentication to parse through an email during account registration to check if the email is a gatech email and sent an email verification if it was
- How can we monetize the product while also incentivising people to keep using it?
    - Plan on charging a small fee during transactions that occur through our app
    - Plan on allowing ads throughout the app. These ads will be catered towards GT organizations or stores endorsed by GT.
    - Considering implementing a monthly premium subscription system that would benefit both buyers and sellers
- How would we automate data collection?
    - Decided to use Google Firebase for A/B testing
    - Have a link to an external google form that users can use to express their opinions on the product

Team Agreement:

- Will meet biweekly either in person(CULC) or on a remote call(Discord/Zoom) on Tuesdays and Thursdays and weekends if need be
- Let other members know if you cannot make it to the meeting or will be late
- Respond promptly to texts within the group chat
- As an individual update other members on the progress you are making throughout the sprint
- Let other members know if you are struggling to complete a section so that they can help

Problem Overview

Based on initial customer discovery interviews and research, a major pain point for college students is the inability to easily and efficiently buy and sell used goods amongst their peers on campus. Key challenges uncovered include:

- Extremely low textbook buyback rates from campus bookstores (<25%) compared to potential direct peer resale value (50%+)
- Difficulties finding interested buyers for niche secondhand student gear via fragmented platforms
- Lack of tailored protections addressing fraud risks transacting high-value goods anonymously
- No existing solution purposefully aggregating all student buyers/sellers for alignment

These shortcomings stem from current generalized marketplace offerings failing to cater to the unique student use case specifically. Mainstream apps do facilitate some peer transactions but without functionality serving core university populations needs around verification, notifications, item cataloging/pricing, on-campus logistics, etc. ~~There is not an online thrift store dedicated just for college students. There are plenty of pop up thrift markets and other thrift sites but nothing that is generalized towards specific college audiences.~~ College students lack a secure and user-friendly online platform tailored specifically for their thrift shopping needs. While there are numerous pop-up thrift markets and general thrift websites available, none cater specifically to the unique preferences and requirements of college students. This complicates their search for affordable and trendy clothing and items. Additionally, significant value is being stranded between students owing to obscured visibility and stifled platform optimization for their precise goals. Specialization could better support this meaningful yet often constrained ecommerce.

User Persona: Michael
Background: Michael is a 21 year old male student at Georgia Tech. He is interested in selling some of his T-shirts and buying used ones from other GT students. Michael does not have a lot of time to go to in person thrift shops and has a limited budget. He is also weary of scammers.
Why Michael will choose to use Emporia:

- Wants a quick way to see what other GT students are selling
- Has a dedicated chatroom for buyers and sellers to setup a time to meet
- Will make use of the dedicated “For You Page” in the app
- Won’t be concerned about scammers since the only people using the app are GT verified
- Able to sell and buy more secondhand items than just shirts
- Wants to be able to like items and see them in a separate list from the FYP
- Wants to be able to search for specific items using keywords

Competitive Analysis

Top existing marketplace platforms students currently use include Facebook Groups/Marketplace, GroupMe chats, and Craigslist. However, as outlined these contain no

tailored mechanisms for identity confirmation or inventory consolidation exclusive to campus populations. Niche players like textbook swap sites and college classifieds have fragmentation challenges. Major platforms also offer little specialization around peer transaction safety, logistics, and pricing dynamics.

This presents a strategic opening to address direct university stakeholder problems through an integrated, compliant mobile ecommerce solution leveraging verifications, analytics, and community activation to unlock more value. Another potential competition that we might see is in the form of traditional social networking. Platforms used mainly for social media such as Instagram have started to turn into miniature marketplaces as sellers create a “post” of their item with contact information below. The biggest reason that this would be a competition is due to the fact that platforms like instagram have a high amount of users. This is something that we will struggle with at the initial stages of our approach.

Another competitor we have looked at is TikTok and how their platform manages to keep users online for hours on end. The scrolling feature the app provides is the main drawpoint of the app as it specifically tailors the ‘For You’ page of the application for the user and that the scrolling feature of the app is what makes it addicting to keep on using. The easy to use functionalities and tailored feeds are what keeps TikTok as one of the most used social media apps in the App Store.

Additionally, Instagram Reels also has a similar for you page functionality where it shows users videos that they want based on their preferences. It is as simple as liking a post or another reel that would cause a shift in the content that instagram shows to the end user. To this end the app manages to keep users engaged for hours on end.

Solution Approach:

Approach 1: Peer-to-Peer Mobile Marketplace App

- Dedicated mobile app allowing students to list items for sale and browse listings posted by other students
- Verification process checks enrollment status via university system integration
- Ratings systems provides feedback on buyers/sellers to build trust
- Custom push notifications for new relevant listings
- Integrated campus logistics coordination and payments

Pros:

- Tailored specifically to college use cases
- Addresses inventory fragmentation and verification issues
- Enables feature optimization around notifications, item cataloging, etc.

Cons:

- Significant dev work for custom built app
- Getting user base initial traction and network effects
- Manual verifications don’t scale

Differentiation:

Specialized mandate solely focused on serving students vs mainstream platforms. Builds trust and unlocks existing supply chains between classmates.

Approach 2: Digital Kiosk Marketplaces

- Network of digital kiosks located around campus where students can list/browse listings and arrange exchanges
- Kiosks manage inventory, coordinate scheduling for testing, provide storage, etc.
- Integrated identity verification and payments

Pros:

- Brick-and-mortar outlet creates additional trust
- Logistics handled through central depot

Cons:

- Accessibility limits flexibility
- Higher overhead than pure software solutions

Differentiation:
Physical retail presence plus digitally-powered exchange creates unique centralized hybrid.

Approach 3: University Thrift Store Model

- Main campus storefront exclusively buys/sells secondhand student goods
- Handles inventory, valuations, merchandising
- Students consign their items and get payouts when sold

Pros:

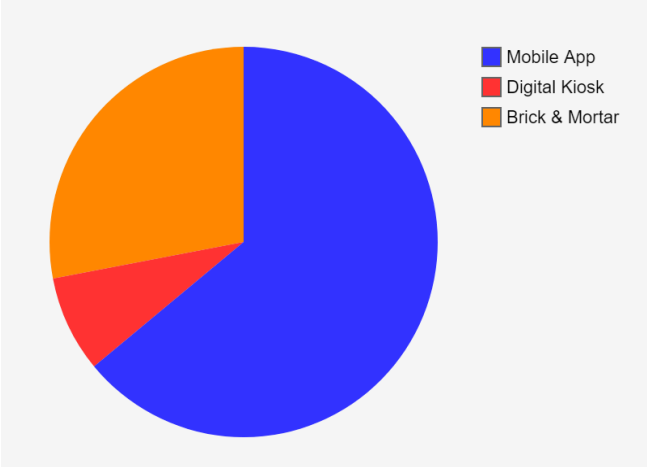
- No direct peer-to-peer risks
- Creates retail experience

Cons:

- Operational overhead with managed model
- Narrower selection than open peer sources

Differentiation:
Instead of unfettered peer exchange, intermediated by a fixed central authority.

Solution Chosen: Approach 1
Based on a total of 25 random user interviews, an overwhelming number of users stated that they would like a mobile application developed for this problem rather than a brick & mortar store or digital kiosk layout. The percentage are as follows: Mobile App - 17/25(68%), Digital Kiosk - 2/25(8%), and Brick & Mortar - 6/25(24%).



Competitive Advantage:
How this solution differs from others on the market is that it is an interactive promise that only GT associated emails can login/use the app to ensure security for its users. It will also offer a robust way for buyers and sellers to connect, as well as buyers having an easy time finding the items they want to buy and sellers posting their items to the app with ease. We also aim to make our app targeted towards GT students by running weekly ads that would be catered towards creating a sense of community amongst buyers, sellers, product developers, and stakeholders.

Use Cases:

Approach 1: Peer-to-Peer Mobile App

Student Seller:

11. Mark just purchased a new laptop and wants to sell his old one to another student
12. He downloads the app, verifies his student credentials, and lists his laptop for sale including photos, specs, and condition notes
13. Mark hopes to use the app for the ease of adding an item with minimal details needed
14. An interested buyer messages Mark to ask some additional questions and confirm functionality
15. They arrange a meetup on campus where the buyer can test everything before completing the sale securely in the app

Student Buyer:

13. Sarah needs a specific textbook for her engineering course that is quite expensive new
14. She opens the app and filters textbook listings to find several used options from other students
15. Sarah wants to have a tailored shopping experience where he sees what he likes on his feed
16. Sarah wants to be able to save the items she likes to look at for a later time
17. Sarah messages one seller to get more information about textbook edition, highlights, etc.
18. After confirming it meets her needs, she schedules a quick meet in the engineering building to pick it up and pay securely via the app.

				
Michael is looking at t-shirts he	Mike is looking through his for you page	Mike is looking through his for you page	Mike schedules a meeting with the buyer	Michael and Mike meet with each

wants to get rid of	to find a t-shirt to buy	to find a t-shirt to buy	through Emporia's chat feature	other to make the exchange

Approach 2: Digital Kiosk Marketplaces

Seller:

9. James finished his semester and wants to list his microeconomics textbook for sale locally rather than deal with shipping
10. He visits the campus resale kiosk, verifies as a student, and lists the book for sale including all relevant details
11. James hands the textbook to the kiosk attendant to catalog and store for the listing period and gets a receipt
12. When a buy order comes through, he receives an alert to collect his payout from the kiosk

Buyer:

9. Lidia sees a film studies textbook she needs listed on the resale kiosk platform from another student
10. She claims the listing on the kiosk and pays for the book, entering her student ID
11. The kiosk prints a receipt confirming her digital purchase
12. Lidia shows the QR code on her receipt when she swings back by later to collect the textbook

Approach 3: Managed University Thrift Store

Student Consigner:

11. AJ is moving out of his apartment near campus and looks to sell possessions he no longer needs
12. He brings decorative furniture, small appliances and kitchenware to the student thrift store
13. The store manager assists AJ in cataloging his items and assessing possible resale pricing/fees
14. AJ formally consigns the agreed upon goods to be displayed, stored and sold by the store manager
15. Every month AJ receives his consignment profit share from sold items as payments

Student Shopper:

10. Christine browses the racks and shelves of the university resale store for dorm room decor
11. She finds some fun looking lamps, wall hangings and pillows in great shape from previous students
12. As a student, Christine receives a discount at checkout on top of the already affordable used prices
13. She shows her active student ID and pays for the thrifted decor items, helping sustain the store nonprofit mandate

Updated Domain Research Interviews:

- Interviewed BGThrifts and GTShopt
    - These are thrift organizations on campus at Georgia Tech. They host thrifting events once a month where they bring a bunch of unwanted materials to tech green to thrift
    - They mentioned that they had issues in terms of:
        - 1. People want more thrifting events
        - 2. People want to sell their own individual items but these clubs can't handle it
    - They mentioned that an online marketplace would really help solve these issues
        - People can have their own apartment act as their own warehouse and sell from their apartment
        - People can buy/sell whenever they want by just using their app.
        -
- Interviewed Salvation Army
    - From your experience, what are the main reasons people donate items to the Salvation Army, and how might a college-focused thrift app impact these donations?
        - People want to do charity, this app might cause a reduction for charity
    - Can you discuss any logistical challenges related to managing donated items, and how might these apply to a college thrift platform?
        - Larger items during peak moving periods might be a concern because people might not handle it correctly / cut corners
    - What are the main challenges you face in ensuring donor and recipient privacy and security, and how could these issues manifest in a college-focused platform?
        - Some people don’t come to a thrift store because they are scared of people judging what they buy
            - Thrift store app might help with this because people do solo transactions
        - **MAIN TAKEAWAY: Focus on this by highlighting privacy in the application**
- Interviewed Brittain Dining Hall
    - How might a college-centered social selling app impact the dining hall's operations or student behavior within the dining hall?
        - There could be concerns about students selling or trading meal plan items

                - This isn't outlined in the dining hall contract, should look intro contract outlines so that when the seller and buyer meet there are no more negotiations.
        - What measures does the dining hall currently have in place to prevent theft or misuse of meal plan items, and how might these measures need to adapt in response to the introduction of a social selling app?
            - Talked about monitoring systems in place to track meal plan usage and prevent abuse. She liked the idea for GT SSO login
        - In your experience, what are some common tactics or behaviors students engage in to circumvent dining hall rules or policies, and how might these translate to the use of a social selling app?
            - Students may attempt to share or sell meal plan items to others, which could lead to misuse of dining hall resources.
                - People might pose as other people/ use others accounts
                    - **MAIN TAKEAWAY: Should have a picture verification after items are traded**
- For this sprint, we spent a lot of time reworking the UI and making the application more conducive to users. In order to do this, I spent a lot of time working with Varun from Startup Exchange. I was showing him our prototype and the designs that we used. From his startup, one of the best pieces of advice that he had to give was to choose a color palette. This palette should be at max 4-5 colors and these colors should define the entire application. He went through our application and told us about instead of finding colors that look good, we should change the UI to work around with our chosen palette of colors and make those colors look good.

Questions Answered:

Spent time interviewing random people in Tech square regarding our idea and asked 3 different individuals each question:

- How will the app handle disputes between buyers and sellers, such as issues with item condition, no-shows, or other conflicts?
    - Allow users to report issues directly within the app. This could include problems with item condition, disputes over transactions, or other concerns.
    - A team that would be responsible for gathering information from both parties, reviewing evidence (e.g., chat logs, item descriptions, photos), and making fair decisions based on the app's policies and guidelines.

- Do you plan to monetize the app, and if so, what revenue models are you considering? How will you balance generating revenue with providing a service that is accessible and valuable to college students?
    - Offer the app for free with basic features that fulfill the core needs of buying and selling among students. Then, introduce premium features that enhance the user experience, such as ~~advanced search filters,~~ increased listing visibility, or additional security measures.
    - ~~Give coupons (operate at a loss) in the beginning so that we can users and then switch~~
- How do you plan to market the app to GT students and encourage its adoption? What channels and strategies will you use to reach your target audience?
    - Collaborate with campus organizations, clubs, and student government to promote the app.
    - We can host launch events, pop-up booths, or informational sessions at campus busy spots, such as the student center, dining halls, and major events like FASET or football games
-

Learning Prototype:

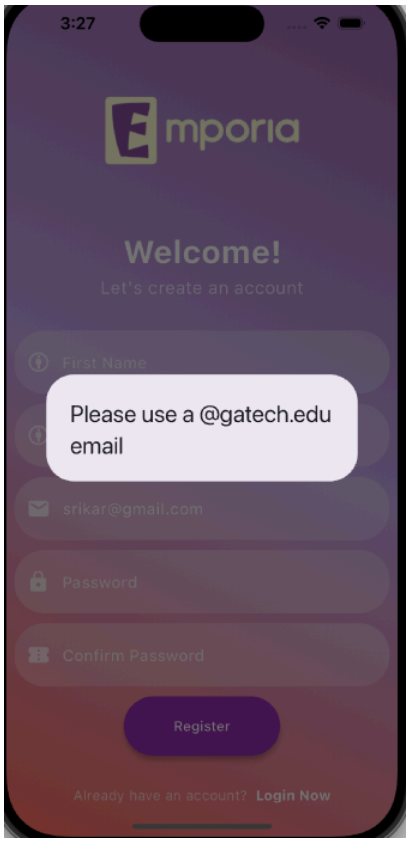
Link to Demo: https://youtu.be/EcRAy8qaBEE

With this sprint, we have updated our LP from Sprint 3 by encompassing user feedback along with our goals for this sprint in order to create a new learning prototype. Encompassing user reviews from the last sprint, specifically UI-based, along with implementing some features that we wanted to get, which included GT email verification, search bar functionality, and improving the favorites page.

For Email verification, we were able to use Firebase email Authentication, while checking if it was a @gatech.edu email using a regex statement (Image 1 and Image 2). For the favorites functionality, we fixed it majorly by linking it to users and storing favorited products as an attribute to the user in the database, with the new favoriteProducts attribute (Images 13 and 14). Additionally, we made the search bar functional, so that it can filter based on product names (Image 5).

For UI updates, I have posted both a current and previous picture, with the previous picture being the left and the current being the right to display the differences. After taking in previous user feedback about the UI, we included major updates, specifically to the For you and Explore pages (Image 3 and 4). Instead of a singular column, it now has 2 columns, which was something our users thought would look better. For the product's UI display, we improved the box that it was in to better highlight

the image including where the name, price, and like button was displayed (Images 8 and 9). Another thing that users wanted last sprint was a timestamp for the product list, to see when it was added. We performed A/B testing on whether to add the date when the product was posted, or post the difference in time between the current time and when the product was posted, which the latter was more popular (Image 10). In regards to the add_product page, users complained that the button took too long to add a product, so we updated out add_product logic so that it compresses images to download them, which sped up the average listing process by 50%. Aside from this, we also added a “take image now” functionality, so that the user has either the option to take a picture of the product right then or choose an image from their camera roll (Images 11 and 12). Lastly, in regards to listening to user feedback, we made the user preferences into a checklist, replacing the old button utilization (Images 6 and 7).



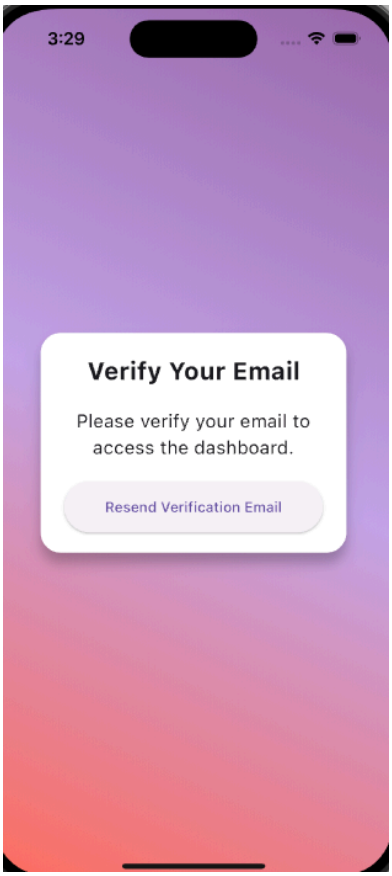
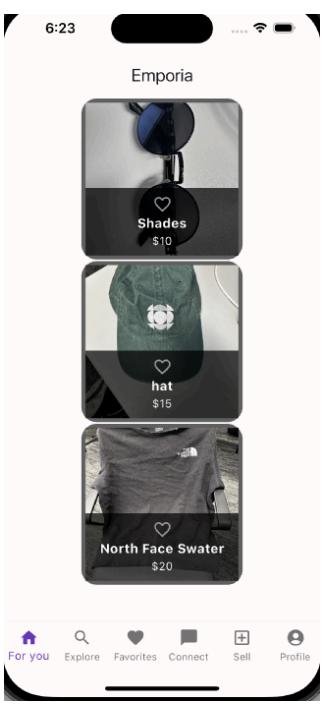
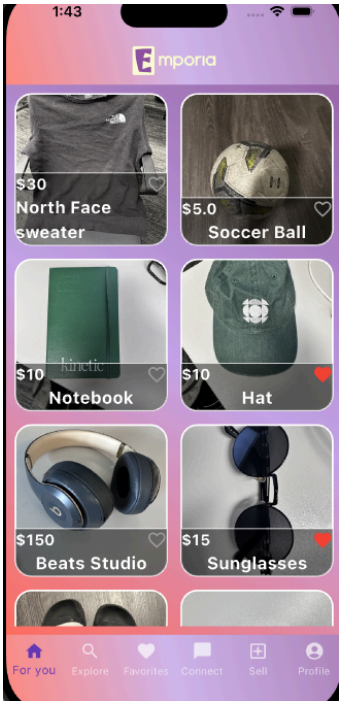


Image 1 and 2





Images 3 and 4

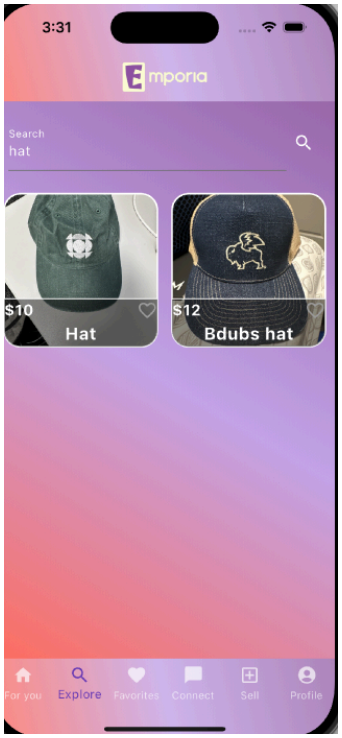


Image 5

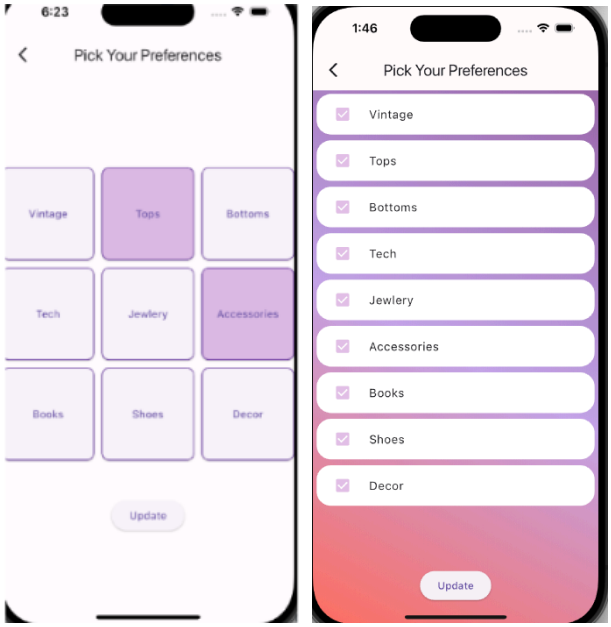


Image 6 and 7

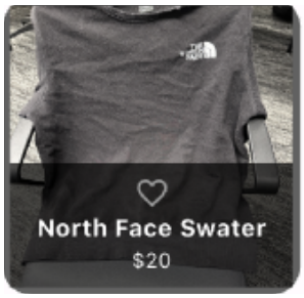


Image 8 and 9

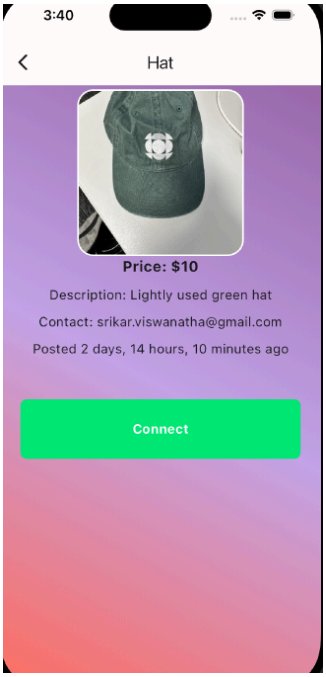


Image 10

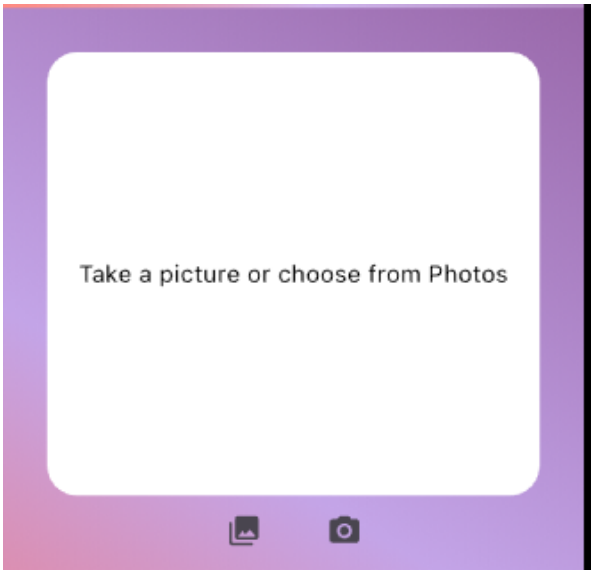
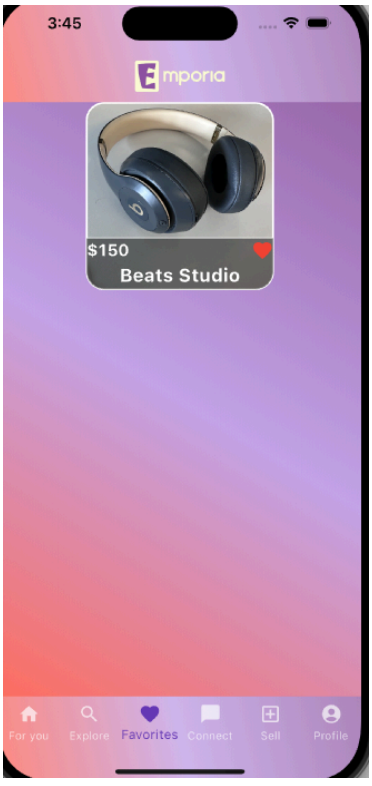
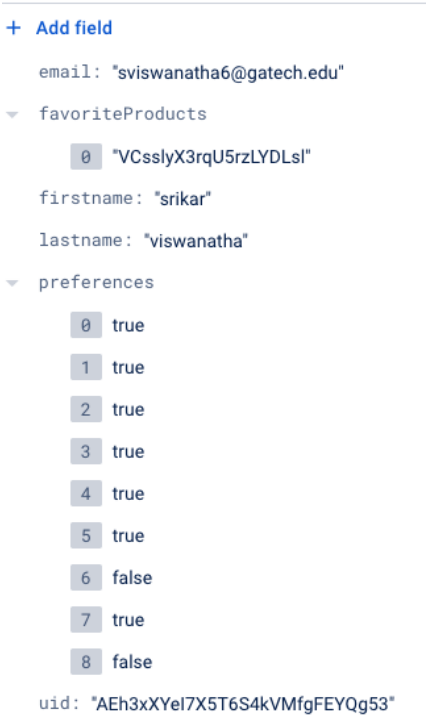


Image 11 and 12





-

Image 13 and 14

-

Technical Discussion:

- In regards to the platforms, data flows, and the architecture of this app, we see there to be 4 distinct concerns for the app to work properly.
    - Product Inventory: We have stored our product objects, including features such as its images and genres into cloud FireStore. We utilize consistent calls to Firebase in order to write and read the data from the database, making sure that our app updates in real-time. In Addition to products, we store users as well using cloud FireStore, to store their names, emails, preferences, and favorite Products.

    - Login Page: The authentication is provided by Firebase Authentication. We also provide users the option of google authentication for ease, but we plan to remove this in the upcoming sprint. In order to make the app only available for Georgia Tech students, we will be verifying emails and making sure that they're .gatech@edu only. This has been fixed in this Sprint. We use regex to make sure

that emails are .gatech@edu only, and we use Firebase Authentication for email verification.

- Purchasing: We plan on utilizing the Stripe API for payment, as it is an easy addition to Flutter and makes payment security very simple.
- Messaging: Being able to message vendors is crucial for our app. We currently have the foundation for the messages functionality. We need to store and link messages to cloud FireStore, and to each user. We are close to finishing this feature, and it will be completed in the upcoming sprint
- Below is all of the instances in our code in which we call upon Firebase Auth and cloud Firestore to update our UI

Firebase Auth Email Verification

Logic to make sure only gatech emails sign up

```
try {
  RegExp gatechEmailRegex = RegExp(r'@gatech\.edu$');
  bool endsWithGatechEmail =
      gatechEmailRegex.hasMatch(emailController.text.trim());

  if (!endsWithGatechEmail) {
    wrongInputMessage("Please use a @gatech.edu email");
  } else {
```

This is for user sign-in for the login page.

```
try {
  await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: emailController.text, password: passwordController.text);

  FirebaseAuth user = FirebaseAuth.instance;

  List<bool> preferences = List.generate(9, (index) => false);

  FirebaseFirestore.instance
      .collection('users')
      .doc(user.currentUser?.uid)
      .set({
    'uid': user.currentUser?.uid,
    'email': user.currentUser!.email,
    'preferences': preferences,
  }, SetOptions(merge: true));
```

Adding a product to the database after the user creates it

```
void buttonLogic() async {
  await Future.delayed(const Duration(seconds: 2));

  Reference reference = FirebaseStorage.instance.ref();
  Reference refImages = reference.child('images');

  String imageFileName = DateTime.now().millisecondsSinceEpoch.toString();

  Reference uploadImage = refImages.child(imageFileName);

  try {
    await uploadImage.putFile(File(image!.path));
    imageUrl = await uploadImage.getDownloadURL();
  } catch (error) {}

  //debugPrint(priceController.toString());
  Product newProduct = Product(
      name: nameController.text.trim(),
      price: double.parse(priceController.text.trim()),
      description: descriptionController.text.trim(),
      vendor: FirebaseAuth.instance.currentUser!.email.toString(),
      isLiked: false,
      images: [imageUrl],
      id: "product${counter++}",
      productGenre: selectedGenres);

  addProduct(newProduct);
```

Searchbar database retrieval logic

```
List<Product> items = documents.map((data) {
  return Product(
      id: data['id'],
      name: data['name'],
      description: data['description'],
      price: data['price']
          .toDouble(), // Assuming 'price' is stored as a double
      images: (data['images'] as List<dynamic>)
          .map((image) => image.toString())
          .toList(),
      isLiked: data['isLiked'],
      vendor: data['vendor'],
      timeAdded: data['timeAdded'],
      productGenre: List<bool>.from(data['productGenre'])); // Product
}).toList();

void filterProducts(String query) {
  setState(() {
    if (query.isNotEmpty) {
      // Otherwise, filter products based on the search query
      filteredProductList = items.where((product) {
        // Check if the product name contains the search query
        return product.name
            .toLowerCase()
            .contains(query.toLowerCase());
      }).toList();

      print(query);
    } else {
      filteredProductList = items;
    }
  });
}
```

This is an example of retrieving user data in order to figure out which products to add on their “for-you” page

```
Future<List<bool>> getPreferences() async {
  String currentUserUid = FirebaseAuth.instance.currentUser!.uid;

  try {
    DocumentSnapshot userSnapshot = await FirebaseFirestore.instance
        .collection('users')
        .doc(currentUserUid)
        .get();

    if (userSnapshot.exists) {
      List<bool> preferences =
          List<bool>.from(userSnapshot['preferences'] ?? []);
      return preferences;
    } else {
      print('User document does not exist');
      return [];
    }
  } catch (e) {
    print('Error getting preferences: $e');
    return [];
  }
}
```

Updated register code storing user email, name, preferences, and favorite Products

```
FirebaseAuth user = FirebaseAuth.instance;

List<bool> preferences = List.generate(9, (index) => false);

List<String> favoriteProducts = [];

FirebaseFirestore.instance
    .collection('users')
    .doc(user.currentUser?.uid)
    .set({
  'uid': user.currentUser?.uid,
  'email': user.currentUser!.email,
  'preferences': preferences,
  'firstname': firstNameController.text.trim(),
  'lastname': lastNameController.text.trim(),
  'favoriteProducts' : favoriteProducts
});
} else {
//show error message that passwords aren't the same
wrongInputMessage("Passwords don't match");
}
```

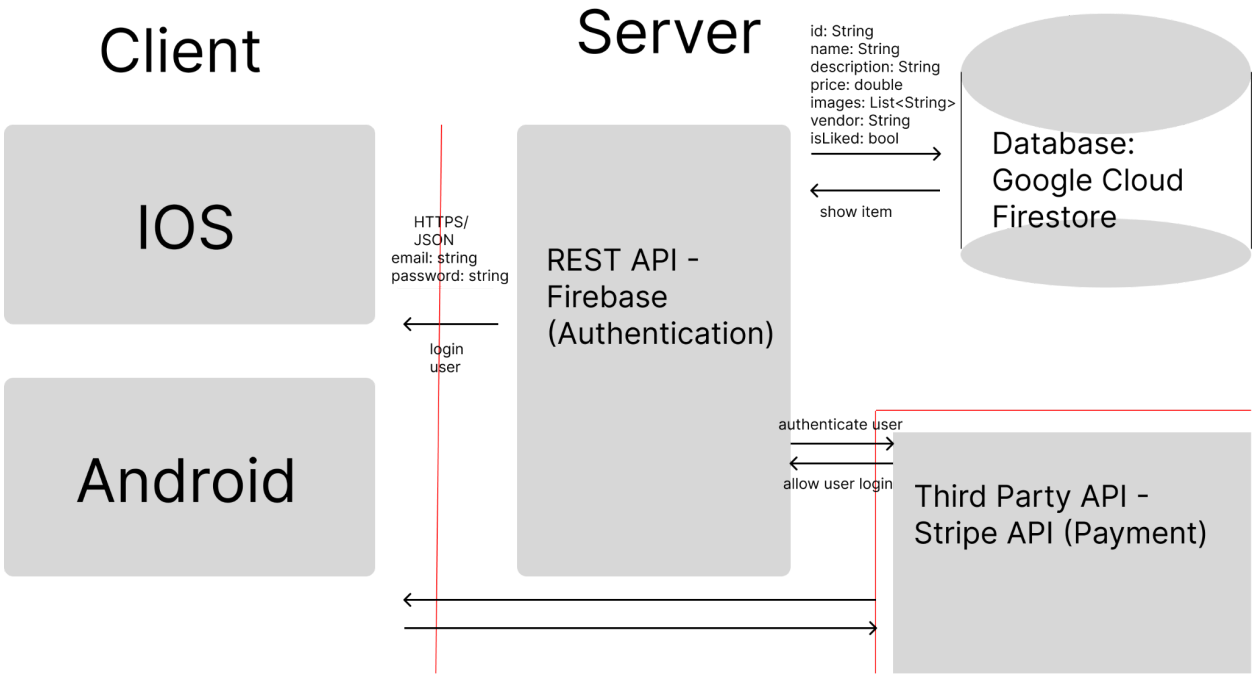
Database updates for if a user favorites or un-favorites a product

```dart
void addFavorite(DocumentReference userRef, List<String> favoriteProductsID,
    Map<String, String> productIDMappings, Product product) {
  if (!favoriteProductsID.contains(productIDMappings[product.id])) {
    favoriteProductsID.add(productIDMappings[product.id]!);
    updateUserFavorites(userRef, favoriteProductsID);
  }
}

void removeFavorite(DocumentReference userRef, List<String> favoriteProductsID,
    Map<String, String> productIDMappings, Product product) {
  favoriteProductsID.remove(productIDMappings[product.id]);
  updateUserFavorites(userRef, favoriteProductsID);
}

void updateUserFavorites(
    DocumentReference userRef, List<String> newFavorites) async {
  try {
    await userRef.update({'favoriteProducts': newFavorites});
    print('User favorites updated successfully');
  } catch (e) {
    print('Error updating user favorites: $e');
    // Handle error appropriately, such as showing a snackbar or dialog to the user
  }
}
```

Call to database to load user's favorite Products

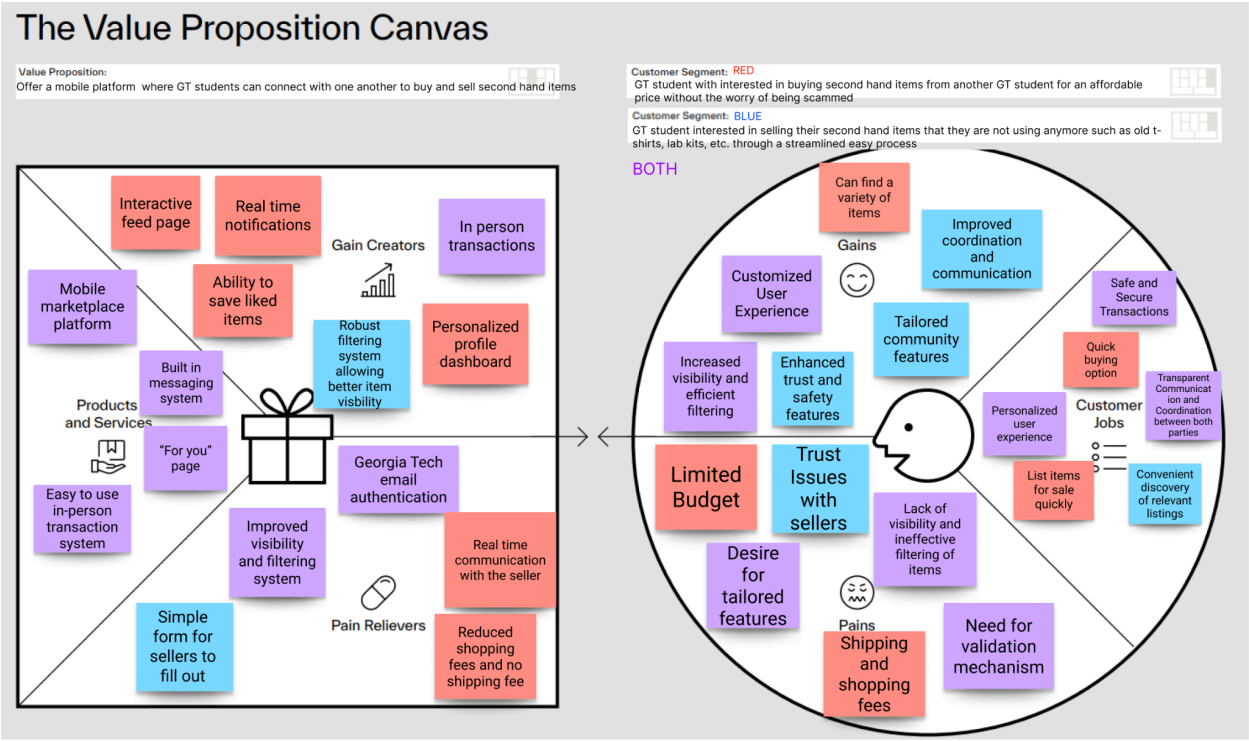
```
Future<List<Product>> getFavoriteProducts() async {
  QuerySnapshot querySnapshot = await products.get();

  List<Product> productList = [];

  for (QueryDocumentSnapshot documentSnapshot in querySnapshot.docs) {
    if (favoriteProductsID.contains(documentSnapshot.id)) {
      Map<String, dynamic> data =
          documentSnapshot.data() as Map<String, dynamic>;
      productList.add(Product(
        id: data['id'],
        name: data['name'],
        description: data['description'],
        price: data['price'].toDouble(),
        images: List<String>.from(data['images']),
        isLiked: data['isLiked'],
        vendor: data['vendor'],
        timeAdded: data['timeAdded'],
        productGenre: List<bool>.from(data['productGenre']),
      )); // Product
    }
  }

  return productList;
}
```

New architecture diagram:



Reduced the amount of Third Party APIs to be just Stripe API for payment as Google Firebase is what we are using for GT email authentication.

Value Proposition Canvas: Updated to reflect customer segments color coded



BMC (Includes VPC): Updated BMC for Sprint 4 in the revenue stream section and indicates which features are in development for Sprint 5 and our future MVP plans

The Business Model Canvas

Designed for: | Designed by: | Date: | Version:

Key Partnerships

- Student vendors
- Campus orgs
- GT Thrifting group chats
- GT Officials (GTPD, CRC, Dean of Students & GT President endorsement)

Key Activities

- Platform development and maintenance.
- User acquisition and engagement.
- Ensuring security and trust within the marketplace.

Key Resources

- Technology infrastructure (database, software).
- Skilled development team.
- Marketing and customer support teams.
- User data and analytics for optimization.

Value Propositions

Tailored and Interactive User experience for the buyers:

- Offers an interactive feed page for the buyers to scroll through the app as well as a dedicated chat room where buyers can communicate with the sellers.

Streamlined selling process for the sellers:

- Offers a robust visibility system that allows buyers to see the sellers listing more frequently as well as an easy way to add a listing on the app

Secure login and transaction system for the sellers and buyers:

- Offering an in-person transaction system (in development for S5) and only students with GT verified emails can use/ login to the mobile app

Customer Relationships

- customer service form
- bi-weekly newsletter sent to registered gatech emails (in future mvp plans)
- Tabling during Wednesday when the Marketplace at Tech Green is taking place (in future mvp plans)

Channels

- Mobile app will be created using Flutter and Firebase for the platform
- Will rely on word of mouth to market the app
- Physical signs around Tech Green will also help advertise(in future mvp plans)

Customer Segments

Students looking to purchase thrift items:

- A limited budget GT student wants to be able to conveniently buy from another GT student through a personalized experience without having to worry about being scammed.

Student looking to sell thrift items:

- A GT Student wants to be able to sell their thrift items without having to worry about advertising and having a streamlined process tailored to them.

Students looking to make secure transactions:

- As a GT student either as a or seller, both want to make secure transactions with one another when buying thrifted items

Cost Structure

- Development and maintenance costs of the mobile app
- Marketing and advertising expenses.
- Operational costs (staff salaries, office space).

Revenue Streams (All revenue features currently in future MVP plans)

- Premium subscription for both buyers and sellers
  - Perks for seller would be that their listing is the first that pops up on a persons feed
  - Perks for buyer is that they get notified first before regular members if a specific item that they might like is put on sale
- Reduced transaction fees on sales
- Will have a listing fee for the seller if they wish to keep their item up for more than 2 weeks in the catalog
- Will have weekly ads catered towards GT orgs and GT endorsed stores

Feature analysis (Ranked):

1. Search Bar: We have ignored the implementation of the search bar for this sprint in order to develop the for you page. Now, we want a user to be able to search for the specific products that they want. Additionally, they would be able to search for certain vendors as well. In our research, we realized that the majority of vendors on our app would be from thrift stores or students with thrift business, as they have the most inventory.
This feature has been implemented in Sprint 4 with the basic functionality to search for an item by keywords in the explore page.
2. Email Verification: Though we had this planned for this sprint, due to the length of the other tasks we have done, we were not able to complete this. This remains a high priority to us and will be completed in the next sprint, as we want to ensure that only GT students use this app.
This feature has been implemented in Sprint 4 using Google Firebase Authentication to only let GT emails register with the app. Additionally a confirmation/verification email is sent to the GT user email attempting to sign up to reduce botting.
3. New product showcase design: Currently, we display our products in a singular column. However, after hearing user feedback, we realized that this is not the best design for this app. Some of the ideas that we currently are looking into, from user feedback, would be either to display products on 2 columns, or to display them in a “flashcard”-esque way. This “flashcard” way would be like Tinder, in which a user would go to their for you page, see one item at a time, and have the ability to either pass upon the item or like it and connect with the vendor. The proper layout could lead to an increase in user retention, thus this is why this feature is so high up.
This feature has been implemented in Sprint 4. Through A/B testing it seems that users prefer the double column rather than the single one so it will be kept in the product.
4. Favorites: We have ignored this feature for this sprint, and have to push it to a later sprint. It is implemented in our app, but not properly, meaning that the favorite button does not save state or save to a user. We need to connect the database to this in order for it to be fully functional.
The main functionality for this feature has been implemented in Sprint 4 where the product which is liked appears in a separate tab labeled favorites. However, there is a small bug that does not update the favorites in real-time, meaning that the user has to reload the page to see their changes. We will fix this in a future spring through implementing the page with a StreamBuilder for real-time updates.
5. Messages: We have the foundation for this feature, and we just have to connect it to Firestore. We can implement a contract feature where a special type of a message acts as an agreement between the seller and the buyer for a specific trade.
This logic is almost complete, but not present in our demo. Will be completed in the next sprint.

6. User Agreement Logic: What we intend to happen is that the buyer and seller meet in person, and they can verify each other through some sort of authentication process

(could be through a 4 digit code like Uber, or scanning personal QR codes), and from this only then will the payment option open up on the app. From there, the buyer and seller can complete the transaction. This would be very important as it makes sure that the product is real, the buyer saw it in person, and that neither side of the transaction is fraudulent.
This feature will be implemented in Sprint 5 as time was an issue and it felt appropriate to implement along with the payment system.

7. Payment API: Develop a safe payment scheme through secure payment APIs (Venmo/Apple Pay/Google Pay) - This is very important as this is how we see ourselves making money.
This feature will be implemented in Sprint 5 as it is the last function that our app would need to be a minimum viable prototype. Our app can operate separate from a payment system if it was not implemented.

S4 Feature Analysis:

1. Messages: This is in the final works, and will be completed next sprint.
2. User Agreement Logic: What we intend to happen is that the buyer and seller meet in person, and they can verify each other through some sort of authentication process (could be through a 4 digit code like Uber, or scanning personal QR codes), and from this only then will the payment option open up on the app. From there, the buyer and seller can complete the transaction. This would be very important as it makes sure that the product is real, the buyer saw it in person, and that neither side of the transaction is fraudulent.
3. Payment API: Develop a safe payment scheme through secure payment APIs (Venmo/Apple Pay/Google Pay) - This is very important as this is how we see ourselves making money.
4. User Profile Page Update: Big updates are needed in the user profile page, such as a way for them to edit their profile, and for them to see and take down their listings.

Learning prototype results:

Model: As you can observe in the Learning Prototype section of the notebook, we have conducted A/B testing, and the images above show the differences in what we tested.

Methodology:
We conducted 20 usability test sessions with students on the GT campus, as well as individuals who work in GT Thrifting organizations. Sessions lasted 5-10 minutes and involved free exploration of flows as well as defined tasks. We asked testers to verbalize thoughts while using the prototype and followed up with specific questions on areas needing input. We also

conducted A/B testing by showing the previous app, versus the new app, then asked each person about the individual changes to the design and the UI.

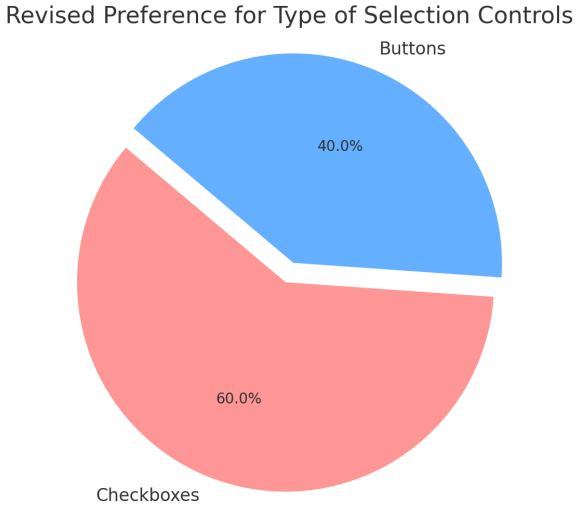
Key Feedback & Learnings:

- Almost 90% of individuals thought liked the UI changes and preferred it to the old
- 4 users instinctively swiped on photos, assuming to see more
- 5 users asked for more specific categories that they could list the product in
- 3 users commented on the difference in adding product speed between the old and new app version

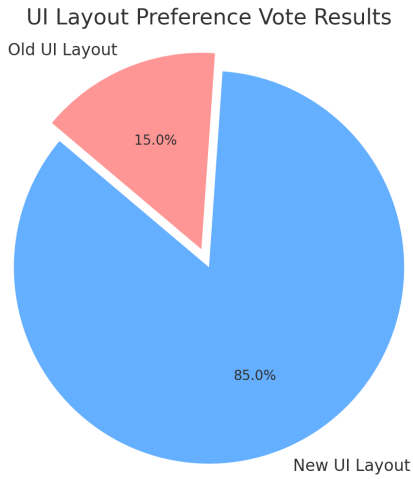
Takeaways:

- UI was improved very much and users appreciate this, need to ask for more suggestions
- Messaging capabilities should integrate platform inbox for streamlined discussions
- Allow users to swipe to see more pictures on a product, and maybe even video
- Implement more categories in the checklist

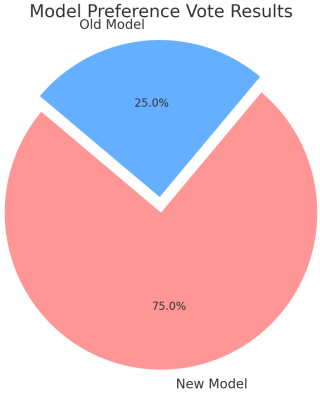
A/B Individual Component Testing Data



- First and foremost, we had done some research with 20 people regarding whether they liked buttons or checkboxes. This was for the selection of their preferred categories. Some people like checkboxes where you select a checkbox for which categories you want and some people like choosing a button with the name of the category on top. We noticed that 12 from the group preferred the checkboxes and 8 preferred the buttons.

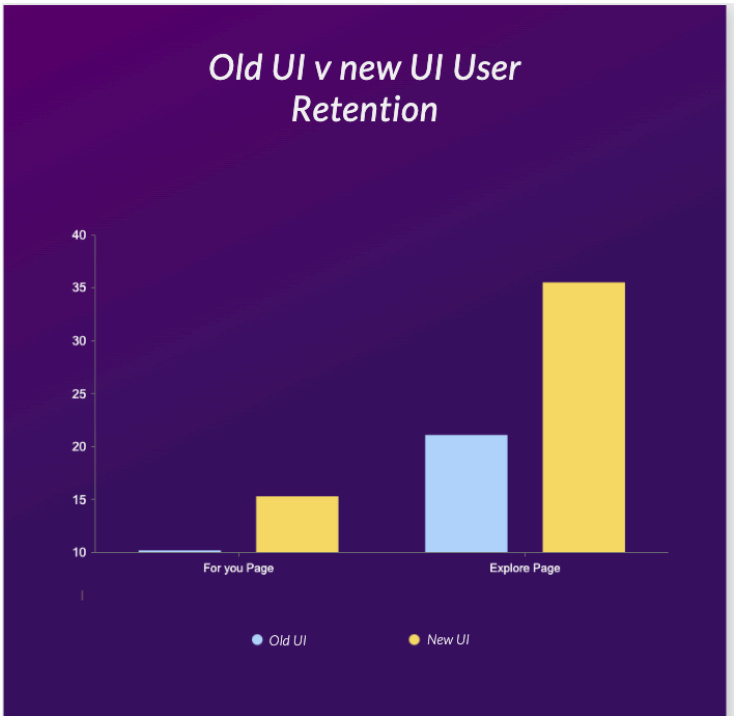


- Secondly, we interviewed about the design of the UI. We interviewed 20 people. We had introduced a new gradient design and a new animation for some of the new features. We had done a survey for which layout was better and 17 people said that the new one was better. The people that said it wasn't better said it wasn’t because they preferred standard white and black and simple colors rather than complex gradients. We noticed that 3 people did not like the new UI and liked the old one more.

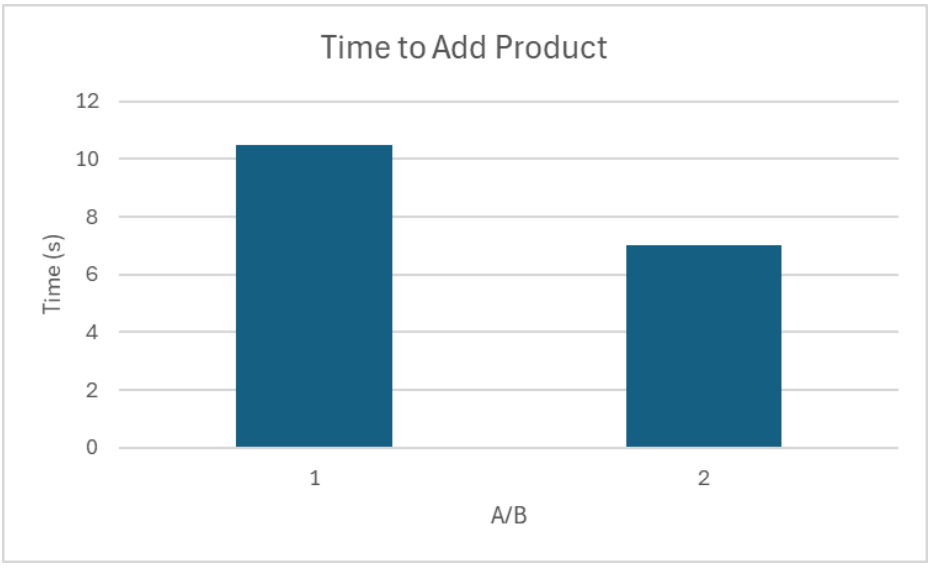


- We then did another study on the new UI vs the old UI for the product listings. We interviewed 20 people regarding whether they liked the new way that products were listed or the old one. We learned that 15 of them liked the new product listing and that 5 of them liked the old one. They were mentioning that the new listing was a bit small and complicated for their phones (android).

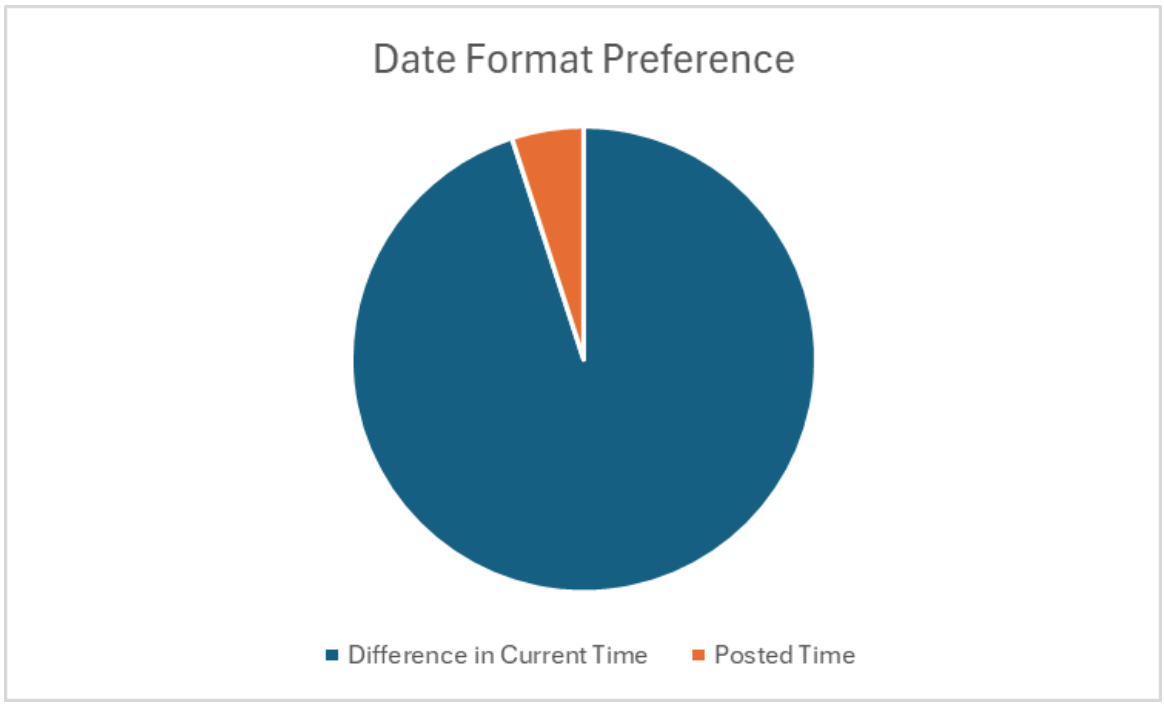
Old UI vs New UI user retention rates on the For you Page and the Explore Page (average retention time measured in seconds)



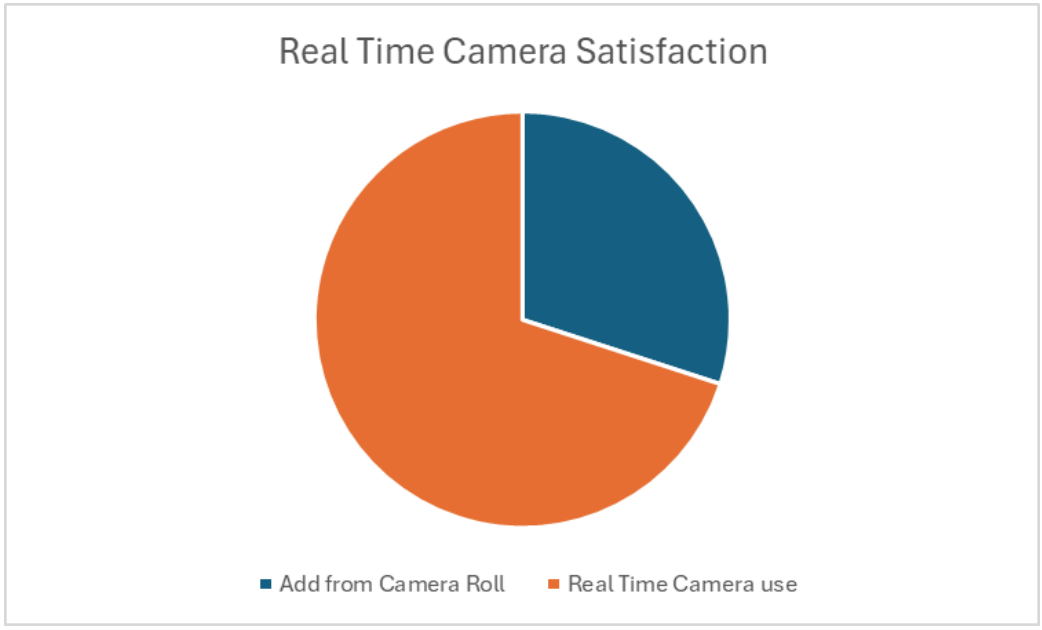
- We also asked users to use the explore and for you pages of both old and new UIs, and measured their time on each page. We saw an 100% increase in retention rate between the old and new UI for the for you page, and over 70% increase in retention rate for the explore page.



Additionally, we tracked the time it would take to add a product with our old and new add_product methods. The improved version was significantly faster on average, up to 50% faster than before.



We also queried which date format preference users liked better. They liked to see the difference between current time and the time the product was posted.



A lot of our users previously asked for real-time camera usage for adding a product, so we added this feature.

Analytics and Data Collection.

Google Analytics was integrated into the app to track and analyze user behavior and app performance.It gathered data on user engagement metrics, such as session duration, bounce rates, and screen flow. It also collected data related to user activities within the app, such as items listed for sale, purchases made, and search queries. As we have implemented this feature recently, there is not enough data to make significant claim, but as we move on into our next sprint, this data collection will provide good insights into what we have to improve upon.

Event Tracking

- Track specific user interactions as events (e.g., listings viewed, purchases, sign-ups)
- Use event parameters to capture additional details (e.g., item category, price)

```
analytics.logViewItem(
    currency: 'usd',
    value: userItems[index].price,
    parameters: <String, dynamic>{
      'name': userItems[index].name,
      'id': userItems[index].id,
      'vendor': userItems[index].vendor,
      'productGenre':
          userItems[index].productGenre.toString()
    });
```

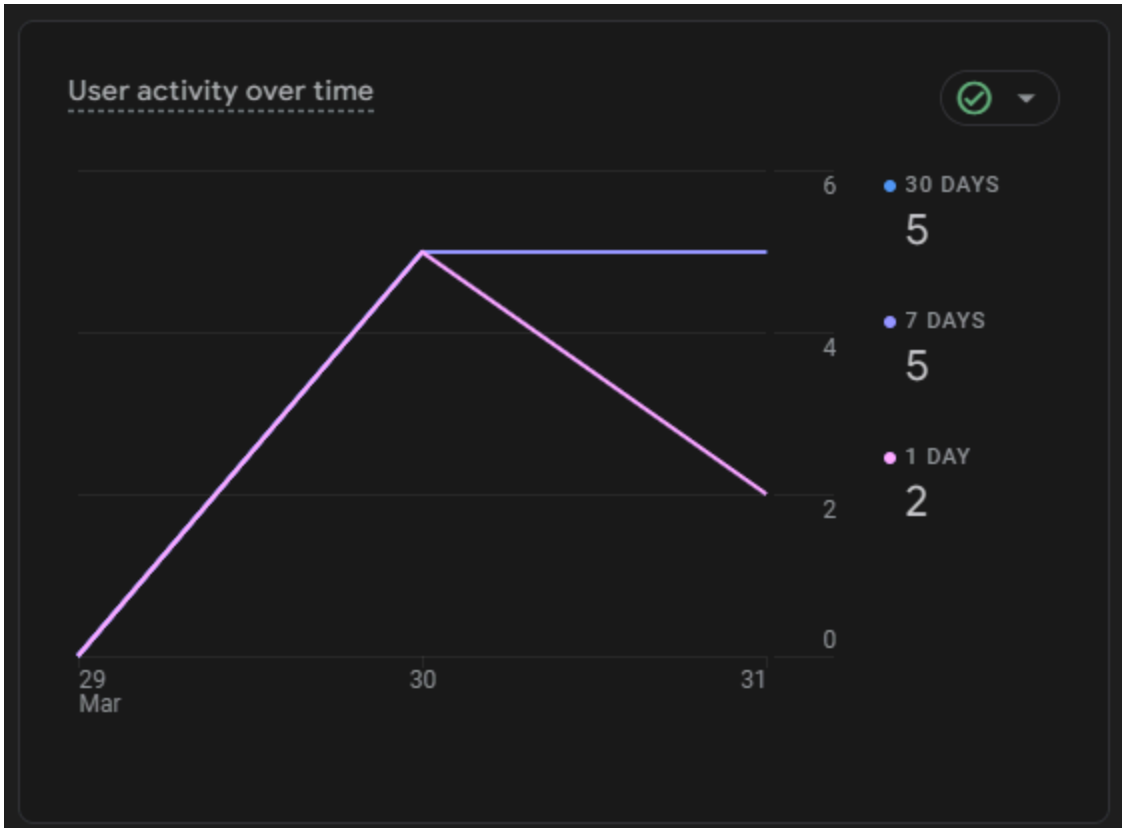
```
await analytics.logSearch(
    searchTerm: _searchController.text);
```

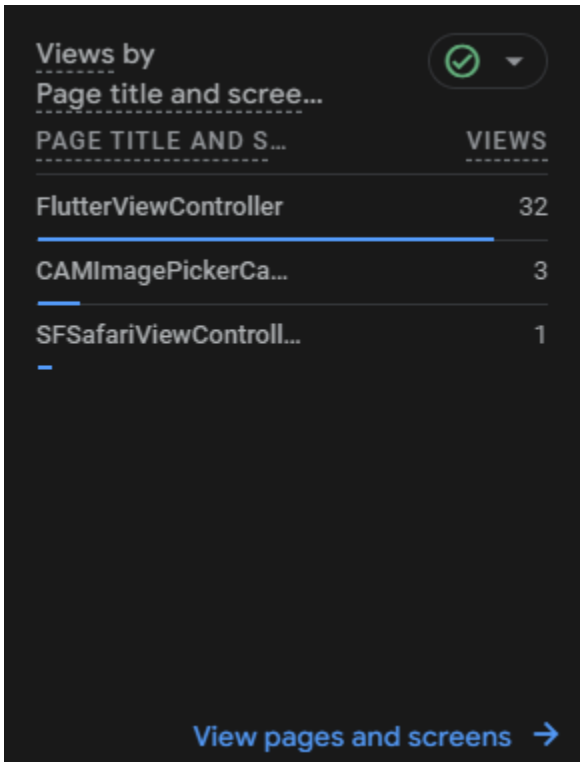
```
analytics.logEvent(name: "vendor_contact"),
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ChatScreen(
      userId: product.vendor,
    ), // ChatScreen
  ), // MaterialPageRoute
)
```

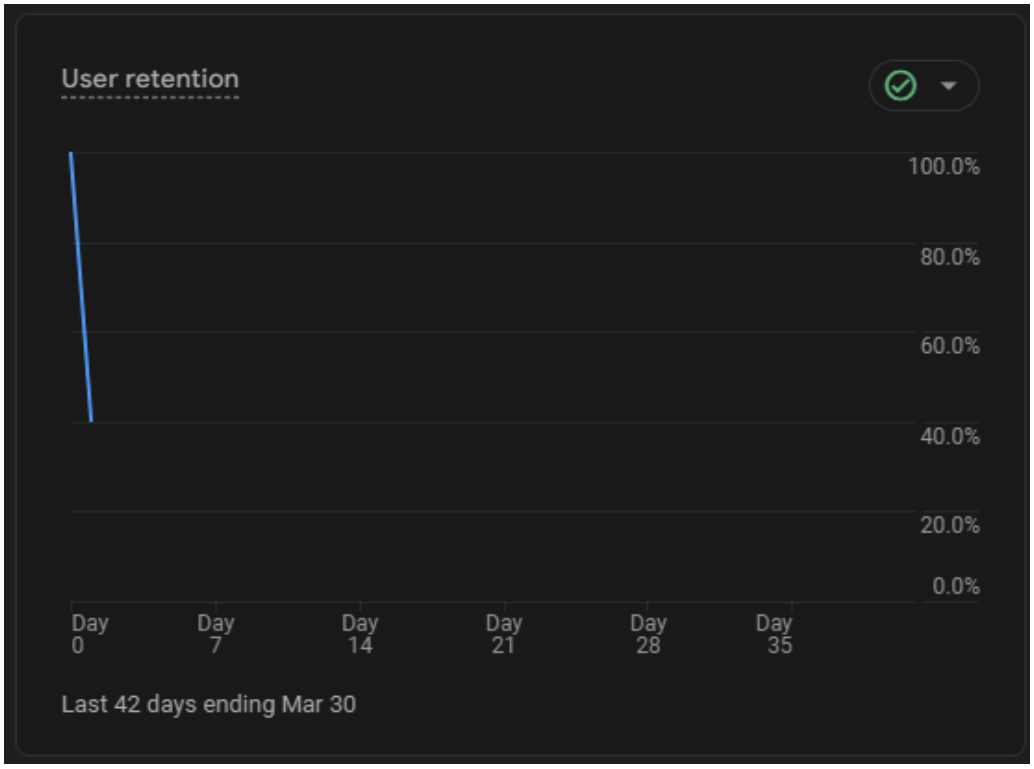
```
await analytics.logSignUp(signUpMethod: "GaTech Email");
```

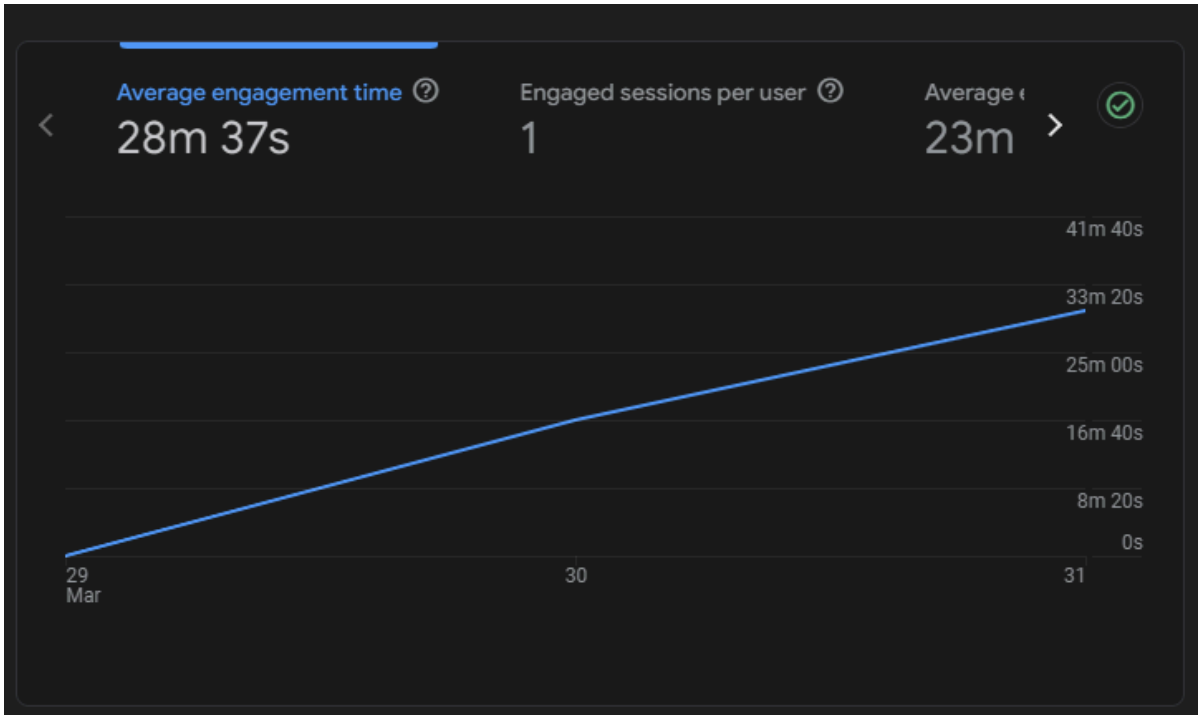
```
FirebaseAnalytics.instance.logAddToWishlist(
    currency: 'usd',
    value: widget.product.price,
    parameters: <String, dynamic>{
      'name': widget.product.name,
      'id': widget.product.id,
      'vendor': widget.product.vendor,
      'productGenre': widget.product.productGenre.toString()
    });
```

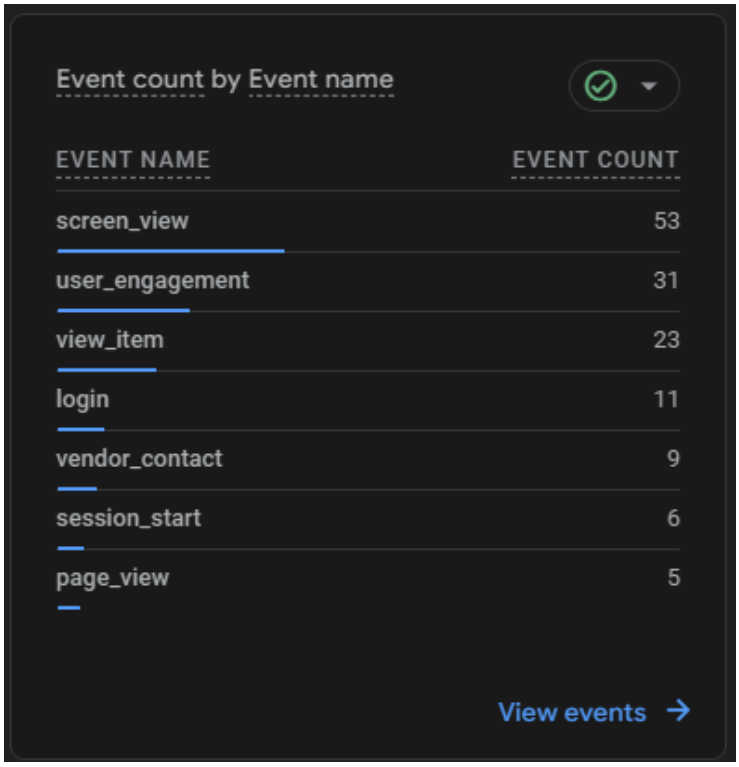
The four images above are from several sections in our code where we track user activity such as product views, search queries, seller messaging, and favorites feature usage across our app.


User activity over time
6
4
2
0
29
Mar
30
31
30 DAYS
5
7 DAYS
5
1 DAY
2


Views by
Page title and scree...
PAGE TITLE AND S...
VIEWS
FlutterViewController
32
CAMImagePickerCa...
3
SFSafariViewControll...
1
View pages and screens


User retention
100.0%
80.0%
60.0%
40.0%
20.0%
0.0%
Day 0
Day 7
Day 14
Day 21
Day 28
Day 35
Last 42 days ending Mar 30


Average engagement time
28m 37s
Engaged sessions per user
1
Average
23m
41m 40s
33m 20s
25m 00s
16m 40s
8m 20s
0s
29 Mar
30
31



The images above are from our firebase analytics dashboard logging several metrics like user retention, engagement, and events.

Learning prototype plans For next sprint:

- Shopping Cart System
    - A user can like a product listing which will add it to a shopping cart for them
        - From there the user can look at what they liked and decide whether or not to contact the buyer through the dedicated chat system
- Have an in-person transaction system
    - Once users meet up in person they should be able to exchange money and goods with one another
    - Thinking of implementing a 4 digit code, qr code, or personalized chimes system to allow for this feature
    - A
- Payment API implementation
    - Will most likely use Stripe API for secure payment feature
- User Profile Page Update
    - Big updates to the profile page, including fixing the edit profile button, new button to see users' personal listings, place to put an image of the user
- Search Bar Update

    - Currently our search bar will return the search results based on product names only. We want to include product description, and user's profiles as things that it can search also.
- Favorite feature update
    - Favorites feature works, but does not happen in real-time, meaning that users have to reload the page to see their changes. We will make this feature occur in real time next sprint.

Biggest concerns updated:

- Some of our biggest concerns are regarding the actual payment between the buyer and the seller.
- How to develop an interactive UI that is tailored specifically toward the Georgia Tech student population
- How do we ensure data privacy? A stalker might buy a product from someone just so they can meet them in a weird location. We need to ensure that we have enough stability to prevent this.
- Scalability. We need to ensure that there is some way that we can expand to other universities later down the line. We cant put all of our eggs into one basket
- We need to consider the environmental impact of this application in the sense that people shouldn't just buy things and then thrift them to rip people off.
- Quality Control: Ensuring that the quality of the items is exactly as promised in the contract. There needs to be some type of management in order to ensure that new items are actually in "new" condition and so forth.
- Dispute Resolution: If there is a dispute between the seller and the buyer, there needs to be a resolution that satisfies both parties to ensure customer satisfaction.
- Scams: Some products might look like real products, for expensive items we should have an authentication service that ensures that these products are as they are marketed.
- Ensuring the app is accessible to all college students, inclusive of people with disabilities. This includes compliance with ADA standards and making sure the UI/UX is navigable for people with quite a number bodily and cognitive competencies.
- Beyond the preliminary charge worries, there's the chance of fraudulent transactions wherein stolen credit playing cards or faux financial institution bills are used.
- Making sure the authenticity of all products bought at the platform is crucial to hold accept as true with integrity.
- Protecting the platform against cyber threats, consisting of hacking, phishing assaults, and malware, to protect personal facts and transactions.
- Implementing systems to monitor user behavior for capability harassment, unlawful behaviors, or violations of the platform's policies and moderating content material as essential.
- Ensuring the platform complies with all relevant legal guidelines and policies, which include tax legal guidelines and client protection legal guidelines.

- Integration with Campus Systems: For verification functions and to enhance capability, you may want to integrate with Georgia Tech's information structures securely and responsibly.

Other Questions Left to Answer:

- How will we monetize the market and incentivize users to use our application for a fee
- How will we start the application (need items for sellers to buy first)
- How will we provide security for the application in terms of payment (cash is unsecure)
- How will we account for missed meetings (seller/buyer forgets to come to agreed location at agreed time)
- How are we going to regulate people that make a lot of money on the app in terms of extra fees?
- How are we going to maintain the authenticity of products?

Sprint 5

Main Goals:

- Complete the message functionality
- Add payment functionality
- Add featured users functionality for subscription model
- Updated User Profile Page
- Conduct A/B testing

Questions answered from last sprint:

- How will we provide security for the application in terms of payment (cash is unsecure)?
    - Use Stripe API to develop an integrated payment system within the chatroom. This chat room will only allow the vendor to request payment from the buyer. If the buyer does not want to pay they do not need to do anything.
- How are we going to maintain the authenticity of products if they are sold?
    - The item will be deleted from the database to ensure user security and privacy.
- How to better engage users and plan for premium subscription in the future?
    - Implemented a feature where it says to check out a specific user's items. For future installments, when a premium subscription is implemented, vendor's can pay for the subscription which will propagate their names to the top of the feed of other user's.

Team Agreement:

- Will meet biweekly either in person(CULC) or on a remote call(Discord/Zoom) on Tuesdays and Thursdays and weekends if need be
- Let other members know if you cannot make it to the meeting or will be late
- Respond promptly to texts within the group chat
- As an individual update other members on the progress you are making throughout the sprint
- Let other members know if you are struggling to complete a section so that they can help

Problem Overview

Based on initial customer discovery interviews and research, a major pain point for college students is the inability to easily and efficiently buy and sell used goods amongst their peers on campus. Key challenges uncovered include:

- Extremely low textbook buyback rates from campus bookstores (<25%) compared to potential direct peer resale value (50%+)

- Difficulties finding interested buyers for niche secondhand student gear via fragmented platforms
- Lack of tailored protections addressing fraud risks transacting high-value goods anonymously
- No existing solution purposefully aggregating all student buyers/sellers for alignment

These shortcomings stem from current generalized marketplace offerings failing to cater to the unique student use case specifically. Mainstream apps do facilitate some peer transactions but without functionality serving core university populations needs around verification, notifications, item cataloging/pricing, on-campus logistics, etc. ~~There is not an online thrift store dedicated just for college students. There are plenty of pop-up thrift markets and other thrift sites but nothing that is generalized towards specific college audiences.~~ College students lack a secure and user-friendly online platform tailored specifically for their thrift shopping needs. While there are numerous pop-up thrift markets and general thrift websites available, none cater specifically to the unique preferences and requirements of college students. This complicates their search for affordable and trendy clothing and items. Additionally, significant value is being stranded between students owing to obscured visibility and stifled platform optimization for their precise goals. Specialization could better support this meaningful yet often constrained ecommerce.

User Persona: Michael
Background: Michael is a 21 year old male student at Georgia Tech. He is interested in selling some of his T-shirts and buying used ones from other GT students. Michael does not have a lot of time to go to in person thrift shops and has a limited budget. He is also weary of scammers.
Why Michael will choose to use Emporia:

- Wants a quick way to see what other GT students are selling
- Has a dedicated chatroom for buyers and sellers to setup a time to meet
- Will make use of the dedicated “For You Page” in the app
- Won’t be concerned about scammers since the only people using the app are GT verified
- Able to sell and buy more secondhand items than just shirts
- Wants to be able to like items and see them in a separate list from the FYP
- Wants to be able to search for specific items using keywords

Competitive Analysis

Top existing marketplace platforms students currently use include Facebook Groups/Marketplace, GroupMe chats, and Craigslist. However, as outlined these contain no tailored mechanisms for identity confirmation or inventory consolidation exclusive to campus populations. Niche players like textbook swap sites and college classifieds have fragmentation challenges. Major platforms also offer little specialization around peer transaction safety, logistics, and pricing dynamics.
This presents a strategic opening to address direct university stakeholder problems through an integrated, compliant mobile ecommerce solution leveraging verifications, analytics, and

community activation to unlock more value. Another potential competition that we might see is in the form of traditional social networking. Platforms used mainly for social media such as Instagram have started to turn into miniature marketplaces as sellers create a “post” of their item with contact information below. The biggest reason that this would be a competition is due to the fact that platforms like instagram have a high amount of users. This is something that we will struggle with at the initial stages of our approach.

Another competitor we have looked at is TikTok and how their platform manages to keep users online for hours on end. The scrolling feature the app provides is the main drawpoint of the app as it specifically tailors the ‘For You’ page of the application for the user and that the scrolling feature of the app is what makes it addicting to keep on using. The easy to use functionalities and tailored feeds are what keeps TikTok as one of the most used social media apps in the App Store.

Additionally, Instagram Reels also has a similar for you page functionality where it shows users videos that they want based on their preferences. It is as simple as liking a post or another reel that would cause a shift in the content that instagram shows to the end user. To this end the app manages to keep users engaged for hours on end.

Solution Approach:

Approach 1: Peer-to-Peer Mobile Marketplace App

- Dedicated mobile app allowing students to list items for sale and browse listings posted by other students
- Verification process checks enrollment status via university system integration
- Ratings systems provides feedback on buyers/sellers to build trust
- Custom push notifications for new relevant listings
- Integrated campus logistics coordination and payments

Pros:

- Tailored specifically to college use cases
- Addresses inventory fragmentation and verification issues
- Enables feature optimization around notifications, item cataloging, etc.

Cons:

- Significant dev work for custom built app
- Getting user base initial traction and network effects
- Manual verifications don’t scale

Differentiation:

Specialized mandate solely focused on serving students vs mainstream platforms. Builds trust and unlocks existing supply chains between classmates.

Approach 2: Digital Kiosk Marketplaces

- Network of digital kiosks located around campus where students can list/browse listings and arrange exchanges
- Kiosks manage inventory, coordinate scheduling for testing, provide storage, etc.
- Integrated identity verification and payments

Pros:

- Brick-and-mortar outlet creates additional trust
- Logistics handled through central depot

Cons:

- Accessibility limits flexibility
- Higher overhead than pure software solutions

Differentiation:
Physical retail presence plus digitally-powered exchange creates unique centralized hybrid.

Approach 3: University Thrift Store Model

- Main campus storefront exclusively buys/sells secondhand student goods
- Handles inventory, valuations, merchandising
- Students consign their items and get payouts when sold

Pros:

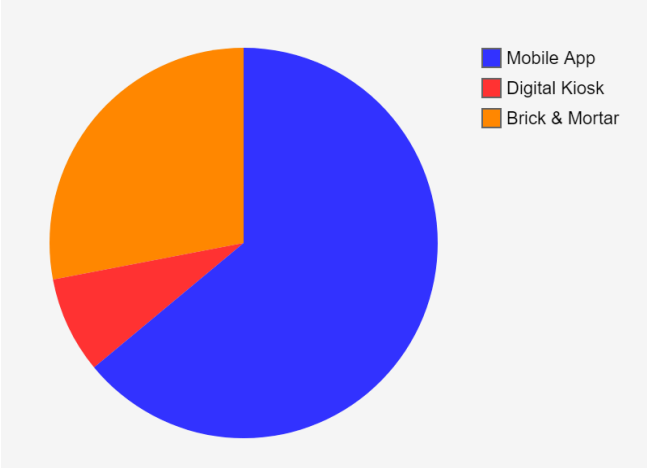
- No direct peer-to-peer risks
- Creates retail experience

Cons:

- Operational overhead with managed model
- Narrower selection than open peer sources

Differentiation:
Instead of unfettered peer exchange, intermediated by a fixed central authority.

Solution Chosen: Approach 1
Based on a total of 25 random user interviews, an overwhelming number of users stated that they would like a mobile application developed for this problem rather than a brick & mortar store or digital kiosk layout. The percentage are as follows: Mobile App - 17/25(68%), Digital Kiosk - 2/25(8%), and Brick & Mortar - 6/25(24%).



Competitive Advantage:
How this solution differs from others on the market is that it is an interactive promise that only GT associated emails can login/use the app to ensure security for its users. It will also offer a robust way for buyers and sellers to connect, as well as buyers having an easy time finding the items they want to buy and sellers posting their items to the app with ease. We also aim to make our app targeted towards GT students by running weekly ads that would be catered towards creating a sense of community amongst buyers, sellers, product developers, and stakeholders.

Use Cases:

Approach 1: Peer-to-Peer Mobile App

Student Seller:

16. Mark just purchased a new laptop and wants to sell his old one to another student
17. He downloads the app, verifies his student credentials, and lists his laptop for sale including photos, specs, and condition notes
18. Mark hopes to use the app for the ease of adding an item with minimal details needed
19. An interested buyer messages Mark to ask some additional questions and confirm functionality
20. They arrange a meetup on campus where the buyer can test everything before completing the sale securely in the app

Student Buyer:

19. Sarah needs a specific textbook for her engineering course that is quite expensive new
20. She opens the app and filters textbook listings to find several used options from other students
21. Sarah wants to have a tailored shopping experience where he sees what he likes on his feed
22. Sarah wants to be able to save the items she likes to look at for a later time
23. Sarah messages one seller to get more information about textbook edition, highlights, etc.
24. After confirming it meets her needs, she schedules a quick meet in the engineering building to pick it up and pay securely via the app.

				
Michael is looking at t-shirts he wants to get rid of	Mike is looking through his for you page to find a t-shirt to buy	Mike is looking through his for you page to find a t-shirt to buy	Mike schedules a meeting with the buyer through Emporia's chat feature	Michael and Mike meet with each other to make the exchange

Approach 2: Digital Kiosk Marketplaces

Seller:

13. James finished his semester and wants to list his microeconomics textbook for sale locally rather than deal with shipping
14. He visits the campus resale kiosk, verifies as a student, and lists the book for sale including all relevant details
15. James hands the textbook to the kiosk attendant to catalog and store for the listing period and gets a receipt
16. When a buy order comes through, he receives an alert to collect his payout from the kiosk

Buyer:

13. Lidia sees a film studies textbook she needs listed on the resale kiosk platform from another student
14. She claims the listing on the kiosk and pays for the book, entering her student ID
15. The kiosk prints a receipt confirming her digital purchase
16. Lidia shows the QR code on her receipt when she swings back by later to collect the textbook

Approach 3: Managed University Thrift Store

Student Consigner:

16. AJ is moving out of his apartment near campus and looks to sell possessions he no longer needs
17. He brings decorative furniture, small appliances and kitchenware to the student thrift store
18. The store manager assists AJ in cataloging his items and assessing possible resale pricing/fees
19. AJ formally consigns the agreed upon goods to be displayed, stored and sold by the store manager
20. Every month AJ receives his consignment profit share from sold items as payments

Student Shopper:

14. Christine browses the racks and shelves of the university resale store for dorm room decor
15. She finds some fun looking lamps, wall hangings and pillows in great shape from previous students
16. As a student, Christine receives a discount at checkout on top of the already affordable used prices
17. She shows her active student ID and pays for the thrifted decor items, helping sustain the store nonprofit mandate

Updated Domain Research Interviews:

- Interviewed BGThrifts and GTShopt
    - These are thrift organizations on campus at Georgia Tech. They host thrifting events once a month where they bring a bunch of unwanted materials to tech green to thrift
    - They mentioned that they had issues in terms of:
        - 1. People want more thrifting events

        - 2. People want to sell their own individual items but these clubs can't handle it
    - They mentioned that an online marketplace would really help solve these issues
        - People can have their own apartment act as their own warehouse and sell from their apartment
        - People can buy/sell whenever they want by just using their app.
        -
- Interviewed Salvation Army
    - From your experience, what are the main reasons people donate items to the Salvation Army, and how might a college-focused thrift app impact these donations?
        - People want to do charity, this app might cause a reduction for charity
    - Can you discuss any logistical challenges related to managing donated items, and how might these apply to a college thrift platform?
        - Larger items during peak moving periods might be a concern because people might not handle it correctly / cut corners
    - What are the main challenges you face in ensuring donor and recipient privacy and security, and how could these issues manifest in a college-focused platform?
        - Some people don't come to a thrift store because they are scared of people judging what they buy
            - Thrift store app might help with this because people do solo transactions
        - **MAIN TAKEAWAY: Focus on this by highlighting privacy in the application**
- Interviewed Brittain Dining Hall
    - How might a college-centered social selling app impact the dining hall's operations or student behavior within the dining hall?
        - There could be concerns about students selling or trading meal plan items
            - This isn't outlined in the dining hall contract, should look intro contract outlines so that when the seller and buyer meet there are no more negotiations.
    - What measures does the dining hall currently have in place to prevent theft or misuse of meal plan items, and how might these measures need to adapt in response to the introduction of a social selling app?
        - Talked about monitoring systems in place to track meal plan usage and prevent abuse. She liked the idea for GT SSO login

        - In your experience, what are some common tactics or behaviors students engage in to circumvent dining hall rules or policies, and how might these translate to the use of a social selling app?
            - Students may attempt to share or sell meal plan items to others, which could lead to misuse of dining hall resources.
                - People might pose as other people/ use others accounts
                    - **MAIN TAKEAWAY: Should have a picture verification after items are traded**
- For this sprint, we spent a lot of time reworking the UI and making the application more conducive to users. In order to do this, I spent a lot of time working with Varun from Startup Exchange. I was showing him our prototype and the designs that we used. From his startup, one of the best pieces of advice that he had to give was to choose a color palette. This palette should be at max 4-5 colors and these colors should define the entire application. He went through our application and told us about instead of finding colors that look good, we should change the UI to work around with our chosen palette of colors and make those colors look good.

Questions Answered:

Spent time interviewing random people in Tech square regarding our idea and asked 3 different individuals each question:

- How will the app handle disputes between buyers and sellers, such as issues with item condition, no-shows, or other conflicts?
    - Allow users to report issues directly within the app. This could include problems with item condition, disputes over transactions, or other concerns.
    - A team that would be responsible for gathering information from both parties, reviewing evidence (e.g., chat logs, item descriptions, photos), and making fair decisions based on the app's policies and guidelines.
- Do you plan to monetize the app, and if so, what revenue models are you considering? How will you balance generating revenue with providing a service that is accessible and valuable to college students?
    - Offer the app for free with basic features that fulfill the core needs of buying and selling among students. Then, introduce premium features that enhance the user experience, such as ~~advanced search filters,~~ increased listing visibility, or additional security measures.
    - ~~Give coupons (operate at a loss) in the beginning so that we can users and then switch~~

- How do you plan to market the app to GT students and encourage its adoption? What channels and strategies will you use to reach your target audience?
    - Collaborate with campus organizations, clubs, and student government to promote the app.
    - We can host launch events, pop-up booths, or informational sessions at campus busy spots, such as the student center, dining halls, and major events like FASET or football games
-

Learning Prototype:

Link to Demo: https://youtu.be/B6mvdTQ-ZrA

Being the last sprint, we wanted to encompass user feedback from previous sprints, as well as finish the majority of features needed to be implemented from our feature analysis to make a MVP. This learning prototype is an evolution of the previous prototype, as it includes more critical features that were not previously implemented, while also taking into account user feedback through the form of A/B testing.

The main thing that we have finished implementing this sprint is the messaging feature, which allows users to connect with vendors to discuss and pay for the product that they are interested in. With this, users click the connect button on a product, and from there can negotiate and ask more questions about the product with the seller. Users can ask for more pictures from the seller, as we have a “send image” function, and discuss about where they want to meet up to deliver the product. After the exchange is done, the seller can initiate a payment request for the amount they negotiated with the buyer for, and send that request to the user through the messaging system. Then, the user only needs to click on the payment request, and They will be able to enter their payment details, and pay the buyer swiftly and easily. Some things that we have changed in the messaging page due to user testing includes the colors of the texts (Gray/Purple scheme vs Blue/Grey), the arrangement of the buttons (Send at the end versus send in the beginning), and the inclusion of a message timestamp (Images 3 and 4), and the payment request either saying “Pay Amount” or just the amount itself (Images 8 and 9)

Another big feature that was completed this sprint was the inclusion of a payment system. The system that we decided to use was the Stripe Connect API, which allows buyers and vendors to seamlessly pay each other through a safe and reliable means. The way that the payment is done is that the vendor initiates a payment intent through the messaging platform, which is then sent to the user. The user can click on this intent, and using stripe can pay the user. The payment gets processed, and the system will

generate a unique “Successful Payment” message from the buyer’s end, so that the seller knows that the buyer has not ripped them off or tried to deceive them. This message is sent only if the payment is completed, which adds extra assurance for both sides. This money is then deposited into a remote Stripe Account, and with the Stripe Connect feature, automatically dispenses the money into the vendor’s Stripe account. (From this point on, the discussion for the payment has not been fully implemented yet. We have struggled with implementing the connect portion of Stripe, as it is very difficult to achieve P2P payment. Due to time constraints, we were not able to complete the other half of the process, but have a solid plan of action to do so). A new feature that we had to add is to ensure that users are paid properly, if a user tries to add an product to the app, they have to have a Stripe Connect account. They can do this in the user profile page, and when clicked upon provides a secure website in which they fill their banking and personal details with to create the account. Only then will they be able to post items on the marketplace. With the Stripe API, Emporia is theoretically able to take a small percentage of the transaction, which adds to our revenue streams as well.

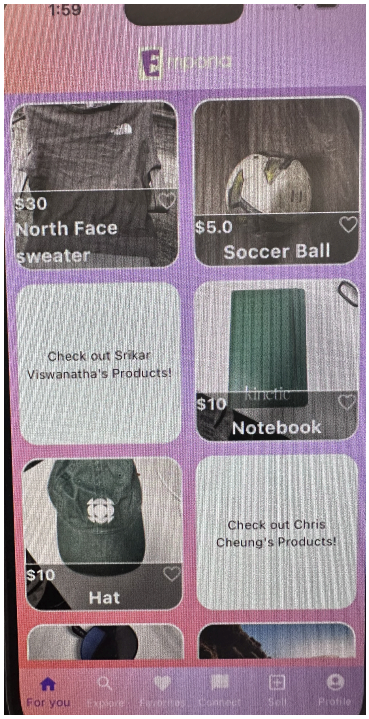
In our BMC, we expressed the ability to generate revenue through a user subscription model, which would promise to highlight subscribed user’s products over others. In this sprint, we created a basic “featured users’” feature, which would display some users to check out on one’s for-you page. These users are currently ranked bases on how many products they have posted, but this sorting algorithm is very easy to change and thus can include other factors to update rankings, such as subscription tier status, and overall user product popularity, in the future. If you click on the widget, it will bring the user to a list of the featured users’ products (Image 1 and 2).

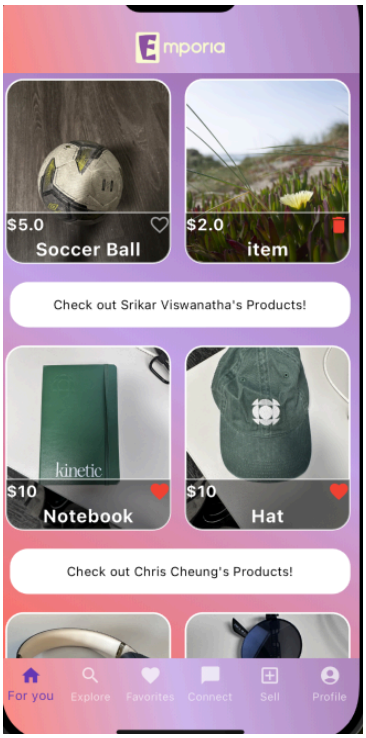
Additionally, we have updated the user’s profile page a good amount. One thing that we have implemented is the user listings page, which allows users to see their listings, and safely delete them. This feature has not been implemented so far, due to our negligence and other features having higher priorities, but this listings page is essential for our MVP (Image 7 and 8). Finally, we have added a new button to the users page which allows them to obtain a Stripe connect account in order to get paid - this functionality will be improved upon to successfully complete account onboarding and get payments sent ot them.

In regards to the searchbar, we have made it so that it can search up the vendor along with it being able to search the product description for key words entered in. Previously, it only searched through the product names, so this new implementation allows our users to search a broader range of words to find what they’re looking for.

Additionally, we have fixed the problem with the Favorite’s page not being able to update in real-time. This was fixed by using a StreamBuilder instead of a FutureBuilder to render the products. Now, when you dislike an item, it will be immediately removed instead of having to refresh the screen to see the change. Unfortunately, due to time

constraints and severe UI complications occuring after trying to change it, we were not able to make the same changes in the Explore and For You pages.





Images 1 and 2

Images 3 and 4

Images 5 and 6

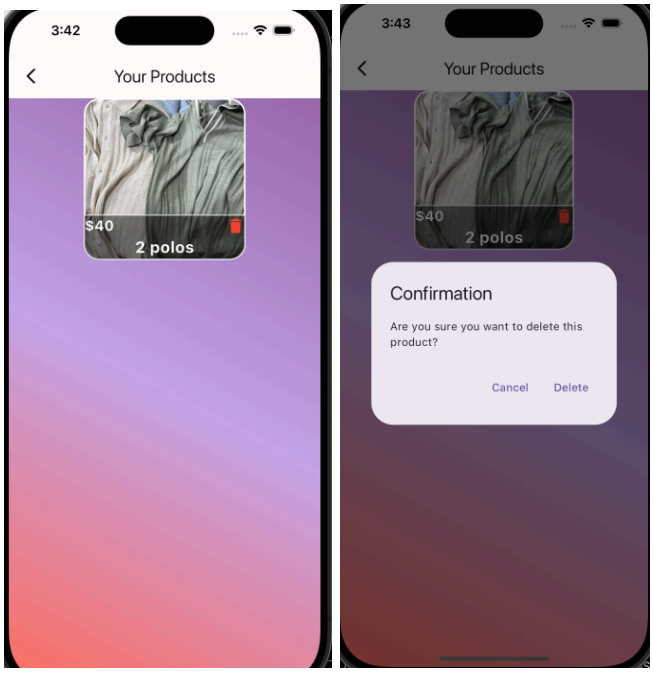
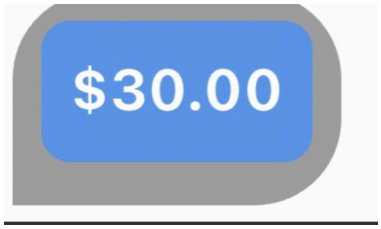
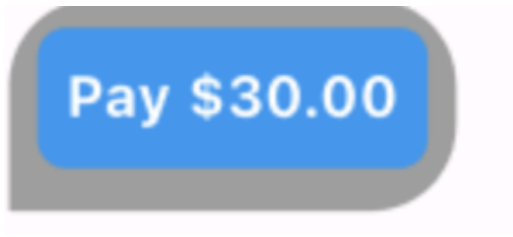


Image 7 and 8



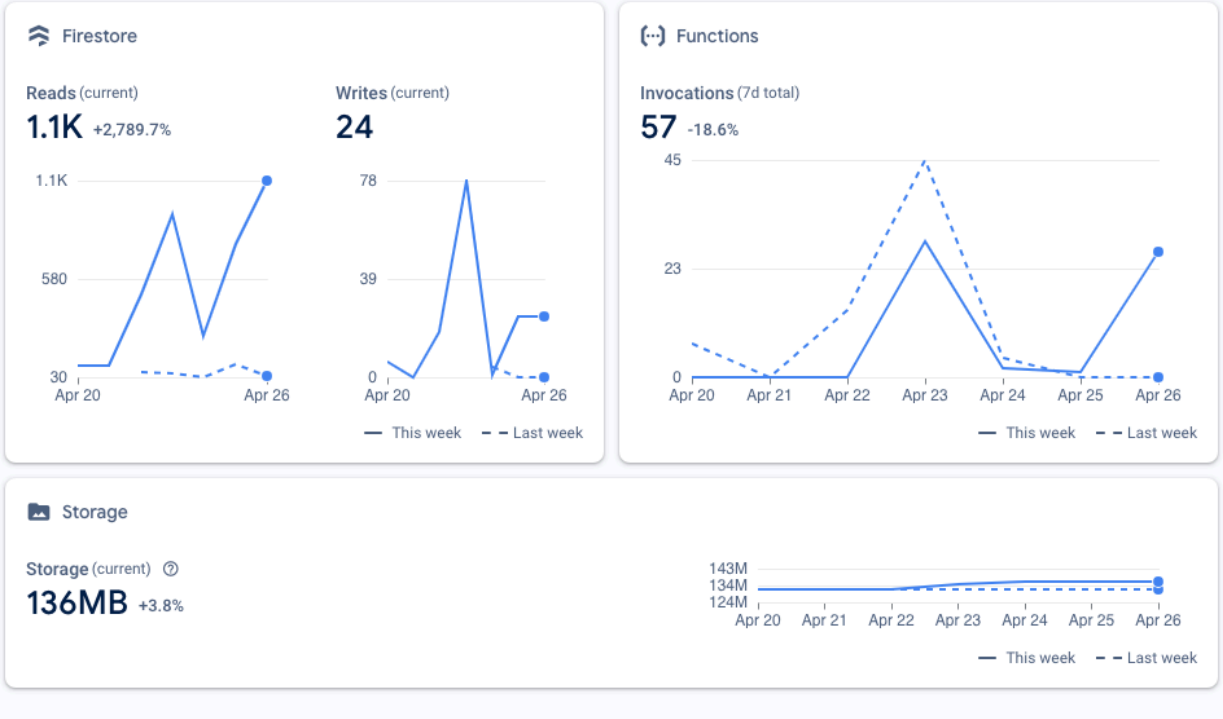


Images 8 and 9

Technical Discussion:

- In regards to the platforms, data flows, and the architecture of this app, we see there to be 4 distinct concerns for the app to work properly.
    - Product Inventory: We have stored our product objects, including features such as its images and genres into cloud FireStore. We utilize consistent calls to Firebase in order to write and read the data from the database, making sure that our app updates in real-time. In Addition to products, we store users as well using cloud FireStore, to store their names, emails, preferences, and favorite Products. For the final sprint, we have also allowed users to delete their own products, and have a page seeing their currently listed products. (The reads/writes are down

from previous sprints due to the need of having to purge the database of bad/usless data)



- Login Page: The authentication is provided by Firebase Authentication. We also provide users the option of google authentication for ease, but we plan to remove this in the upcoming sprint. In order to make the app only available for Georgia Tech students, we will be verifying emails and making sure that they're .gatech@edu only. This has been fixed in this Sprint. We use regex to make sure that emails are .gatech@edu only, and we use Firebase Authentication for email verification.
- Purchasing: We plan on utilizing the Stripe API for payment, as it is an easy addition to Flutter and makes payment security very simple. In this sprint, we have implemented payments using the Stripe Connect service, along with the Stripe API. We have 2 cloud functions, hosted by Firebase, which serves as a local backend to manage and request the API calls. The functions are written in JavaScript. One of them handles the payment request initialization, associating the user and amount that they are sending to Stripe. The other function handles creatting an express Stripe Connect account for vendors, and associating their email with their Stripe accountID, and the last function handles payouts to the vendors(These last 2 functions still don't entirely work but is in our future plans)
- Messaging: Being able to message vendors is crucial for our app. We currently have the foundation for the messages functionality. We need to store and link messages to cloud FireStore, and to each user. We are close to finishing this feature, and it will be completed in the upcoming sprint. We have finished this

feature, and messages are stored in cloud Firestore. Each message session is known from the buyer Email, seller Email, and the productID they are discussing.

- Below is all of the instances in our code in which we call upon Firebase Auth and cloud Firestore to update our UI, along with Stripe API calls.

Firebase Auth Email Verification

Logic to make sure only gatech emails sign up

```
try {
  RegExp gatechEmailRegex = RegExp(r'@gatech\.edu$');
  bool endsWithGatechEmail =
      gatechEmailRegex.hasMatch(emailController.text.trim());

  if (!endsWithGatechEmail) {
    wrongInputMessage("Please use a @gatech.edu email");
  } else {
```

This is for user sign-in for the login page.

```
try {
  await FirebaseAuth.instance.signInWithEmailAndPassword(
      email: emailController.text, password: passwordController.text);

  FirebaseAuth user = FirebaseAuth.instance;

  List<bool> preferences = List.generate(9, (index) => false);

  FirebaseFirestore.instance
      .collection('users')
      .doc(user.currentUser?.uid)
      .set({
    'uid': user.currentUser?.uid,
    'email': user.currentUser!.email,
    'preferences': preferences,
  }, SetOptions(merge: true));
```

Adding a product to the database after the user creates it

```dart
void buttonLogic() async {
  await Future.delayed(const Duration(seconds: 2));

  Reference reference = FirebaseStorage.instance.ref();
  Reference refImages = reference.child('images');

  String imageFileName = DateTime.now().millisecondsSinceEpoch.toString();

  Reference uploadImage = refImages.child(imageFileName);

  try {
    await uploadImage.putFile(File(image!.path));
    imageUrl = await uploadImage.getDownloadURL();
  } catch (error) {}

  //debugPrint(priceController.toString());
  Product newProduct = Product(
      name: nameController.text.trim(),
      price: double.parse(priceController.text.trim()),
      description: descriptionController.text.trim(),
      vendor: FirebaseAuth.instance.currentUser!.email.toString(),
      isLiked: false,
      images: [imageUrl],
      id: "product${counter++}",
      productGenre: selectedGenres);

  addProduct(newProduct);
```

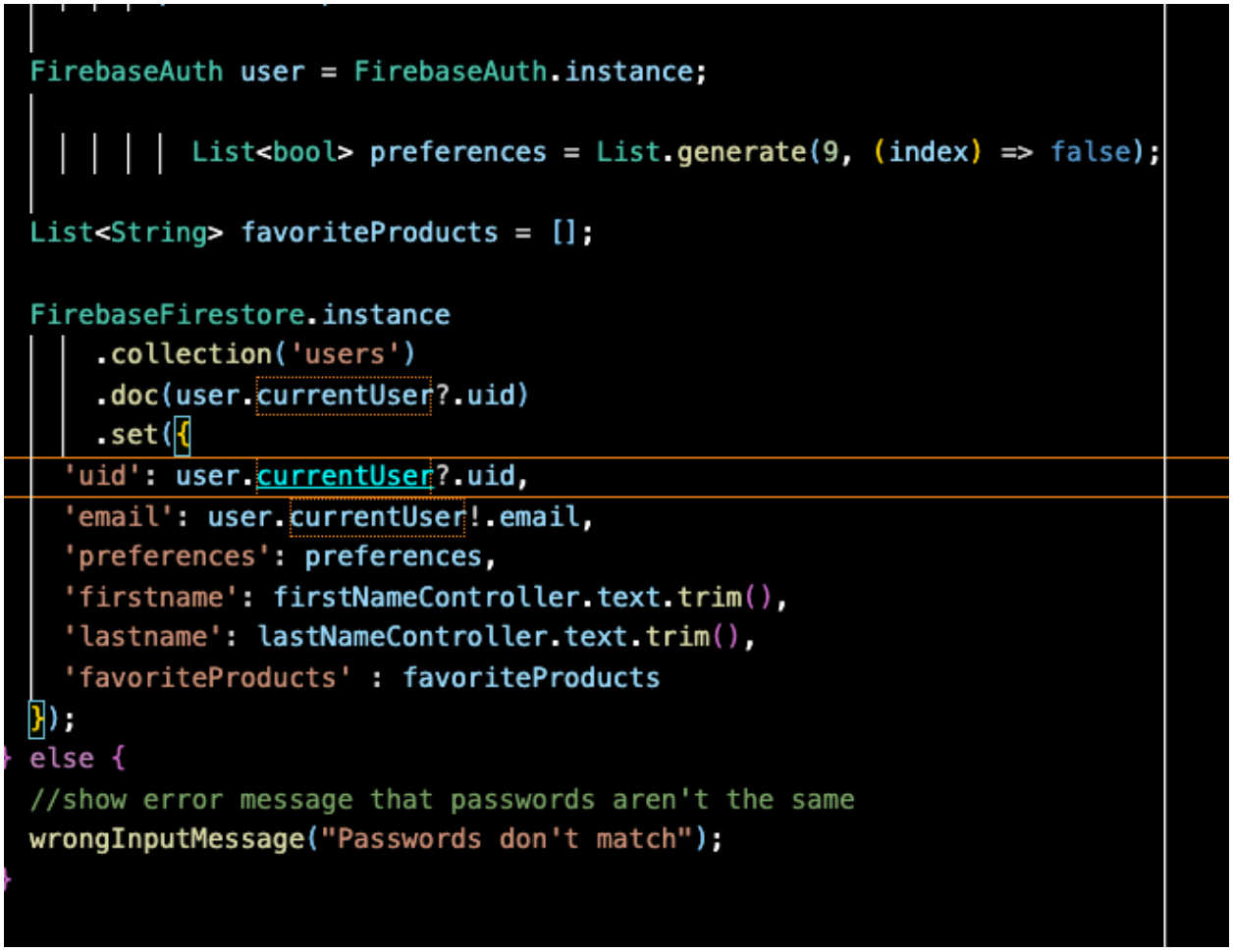
Updated Seachbar Retrieval Logic

```
void filterProducts(String query) {
  setState(() {
    if (query.isNotEmpty) {
      // Otherwise, filter products based on the search query
      filteredProductList = items.where((product) {
        // Check if the product name contains the search query
        return product.name
                .toLowerCase()
                .contains(query.toLowerCase()) ||
            product.vendor
                .toLowerCase()
                .contains(query.toLowerCase()) ||
            product.description
                .toLowerCase()
                .contains(query.toLowerCase());
      }).toList();

      print(query);
    } else {
      filteredProductList = items;
    }
  });
}
```

This is an example of retrieving user data in order to figure out which products to add on their “for-you” page

```dart
Future<List<bool>> getPreferences() async {
  String currentUserUid = FirebaseAuth.instance.currentUser!.uid;

  try {
    DocumentSnapshot userSnapshot = await FirebaseFirestore.instance
        .collection('users')
        .doc(currentUserUid)
        .get();

    if (userSnapshot.exists) {
      List<bool> preferences =
          List<bool>.from(userSnapshot['preferences'] ?? []);
      return preferences;
    } else {
      print('User document does not exist');
      return [];
    }
  } catch (e) {
    print('Error getting preferences: $e');
    return [];
  }
}
```

Updated register code storing user email, name, preferences, and favorite Products



```
FirebaseAuth user = FirebaseAuth.instance;

List<bool> preferences = List.generate(9, (index) => false);

List<String> favoriteProducts = [];

FirebaseFirestore.instance
    .collection('users')
    .doc(user.currentUser?.uid)
    .set({
  'uid': user.currentUser?.uid,
  'email': user.currentUser!.email,
  'preferences': preferences,
  'firstname': firstNameController.text.trim(),
  'lastname': lastNameController.text.trim(),
  'favoriteProducts' : favoriteProducts
});
} else {
//show error message that passwords aren't the same
wrongInputMessage("Passwords don't match");
}
```

Database updates for if a user favorites or un-favorites a product

```
void addFavorite(DocumentReference userRef, List<String> favoriteProductsID,
    Map<String, String> productIDMappings, Product product) {
  if (!favoriteProductsID.contains(productIDMappings[product.id])) {
    favoriteProductsID.add(productIDMappings[product.id]!);
    updateUserFavorites(userRef, favoriteProductsID);
  }
}

void removeFavorite(DocumentReference userRef, List<String> favoriteProductsID,
    Map<String, String> productIDMappings, Product product) {
  favoriteProductsID.remove(productIDMappings[product.id]);
  updateUserFavorites(userRef, favoriteProductsID);
}

void updateUserFavorites(
    DocumentReference userRef, List<String> newFavorites) async {
  try {
    await userRef.update({'favoriteProducts': newFavorites});
    print('User favorites updated successfully');
  } catch (e) {
    print('Error updating user favorites: $e');
    // Handle error appropriately, such as showing a snackbar or dialog to the user
  }
}
```

Call to database to load user's favorite Products

```dart
Future<List<Product>> getFavoriteProducts() async {
  QuerySnapshot querySnapshot = await products.get();

  List<Product> productList = [];

  for (QueryDocumentSnapshot documentSnapshot in querySnapshot.docs) {
    if (favoriteProductsID.contains(documentSnapshot.id)) {
      Map<String, dynamic> data =
          documentSnapshot.data() as Map<String, dynamic>;
      productList.add(Product(
        id: data['id'],
        name: data['name'],
        description: data['description'],
        price: data['price'].toDouble(),
        images: List<String>.from(data['images']),
        isLiked: data['isLiked'],
        vendor: data['vendor'],
        timeAdded: data['timeAdded'],
        productGenre: List<bool>.from(data['productGenre']),
      )); // Product
    }
  }

  return productList;
}
```

Stripe API call to send payment

```
Future<void> sendPaymentRequest(
    String? email, double price, BuildContext context) async {
  try {
    print("Function entered");
    final response = await http.post(
        Uri.parse(
            "https://us-central1-cs4261assignment1.cloudfunctions.net/stripePaymentIntentRequest"),
        body: {
          'email': email,
          'amount': (price * 100).toString(),
        });

    final jsonResponse = jsonDecode(response.body);
    print(jsonResponse.toString());

    await Stripe.instance.initPaymentSheet(
        paymentSheetParameters: SetupPaymentSheetParameters(
      paymentIntentClientSecret: jsonResponse['paymentIntent'],
      merchantDisplayName: 'Emporia',
      customerId: jsonResponse['customer'],
      customerEphemeralKeySecret: jsonResponse['ephemeralKey'],
    ));
    print("Made payment sheet");

    await Stripe.instance.presentPaymentSheet();
    print("done");

    ScaffoldMessenger.of(context)
        .showSnackBar(const SnackBar(content: Text('Payment is successful')));
  } catch (error) {
    if (error is StripeException) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text('Error occured: ${error.error.localizedMessage}'))); // SnackBar
    }
```

Stripe Cloud Functions (Hosts as a local backend for the HTTP requests)

The below function creates the payment request and submits the payment to the system

```
exports.stripePaymentIntentRequest = functions.https.onRequest(async (req, res) => {
    try {
        let customerId;
        const customerList = await stripe.customers.list({
            email: req.body.email,
            limit: 1
        })

        if (customerList.data.length != 0) {
            customerId = customerList.data[0].id;
        }
        else {
            const customer = await stripe.customers.create({
                email: req.body.email
            });
            customerId = customer.data.id;
        }



        const ephemeralKey = await stripe.ephemeralKeys.create(
            { customer: customerId },
            {apiVersion: '2020-08-27'}
        )

        const paymentIntent = await stripe.paymentIntents.create({
            amount: parseInt(req.body.amount),
            currency: 'usd',
            customer: customerId,
        })

        res.status(200).send({
            paymentIntent: paymentIntent.client_secret,
            ephemeralKey: ephemeralKey.secret,
            customer: customerId,
            success: true,

        })

    }
    catch (error) {
        res.status(404).send({ success: false, error: error.messsage });
```

This function creates a Stripe account ID, which is important for users to create a Stripe account and get paid

```
exports.createStripeAccount = functions.https.onRequest(async (req, res) => {
  const { method } = req;
  //const host = 'https://your-host-url'; // Replace 'your-host-url' with your actual host URL


    try {
      const account = await stripe.accounts.create({
        type: 'custom',
        country: 'US',
        email: req.body.email,
        capabilities: {
            card_payments: {
            requested: true,
            },
            transfers: {
            requested: true,
            },
        },
        });


        // In case of request generated from the flutter app, return a json response
        res.status(200).json({ success: true, accountId: account.id });

    } catch (error) {
      console.error('Error creating Stripe account:', error);
      res.status(500).send('Error creating Stripe account');
    }

});
```

This function is still in development but creates the payout so that the vendor gets paid

```
exports.createPayout = functions.https.onCall(async (data, context) => {
  // Authenticate the user (ensure the request is from a valid user)
  if (!context.auth) {
    throw new functions.https.HttpsError('failed-precondition', 'The function must be called whi
  }

  try {
    const amount = data.amount; // Amount in cents
    const stripeAccount = data.stripeAccount; // Connected Stripe account ID

    const payout = await stripe.payouts.create({
      amount: amount,
      currency: 'usd',
    }, {
      stripeAccount: stripeAccount,
    });

    return { success: true, payoutId: payout.id };
  } catch (error) {
    console.error('Payout creation failed:', error);
    throw new functions.https.HttpsError('unknown', 'Payout creation failed', error);
  }
});
```

Message Sending Logic, Updating the database whenever users send messages

```
Future<void> _sendText(
    BuildContext context, bool isPayment, String content) async {
  if (docRef == "none") {
    final checksAndBalances = await FirebaseFirestore.instance
        .collection("chat")
        .where(Filter.or(
            Filter("buyer",
                isEqualTo: FirebaseAuth.instance.currentUser!.email),
            Filter("vendor",
                isEqualTo: FirebaseAuth.instance.currentUser!.email)))
        .get();
    if (checksAndBalances.docs.isEmpty) {
      final newChatRef =
          await FirebaseFirestore.instance.collection('chat').add({
        "productId": widget.productId,
        "vendor": widget.vendor,
        "buyer": FirebaseAuth.instance.currentUser!.email
      });
      setState(() {
        docRef = newChatRef.path;
      });
      await FirebaseFirestore.instance
          .collection("${newChatRef.path}/messages")
          .add({
        "content": content,
        "messageType": isPayment ? "payment" : "text",
        "senderId": FirebaseAuth.instance.currentUser!.email,
        "sentTime": DateTime.now(),
      });
    }
```

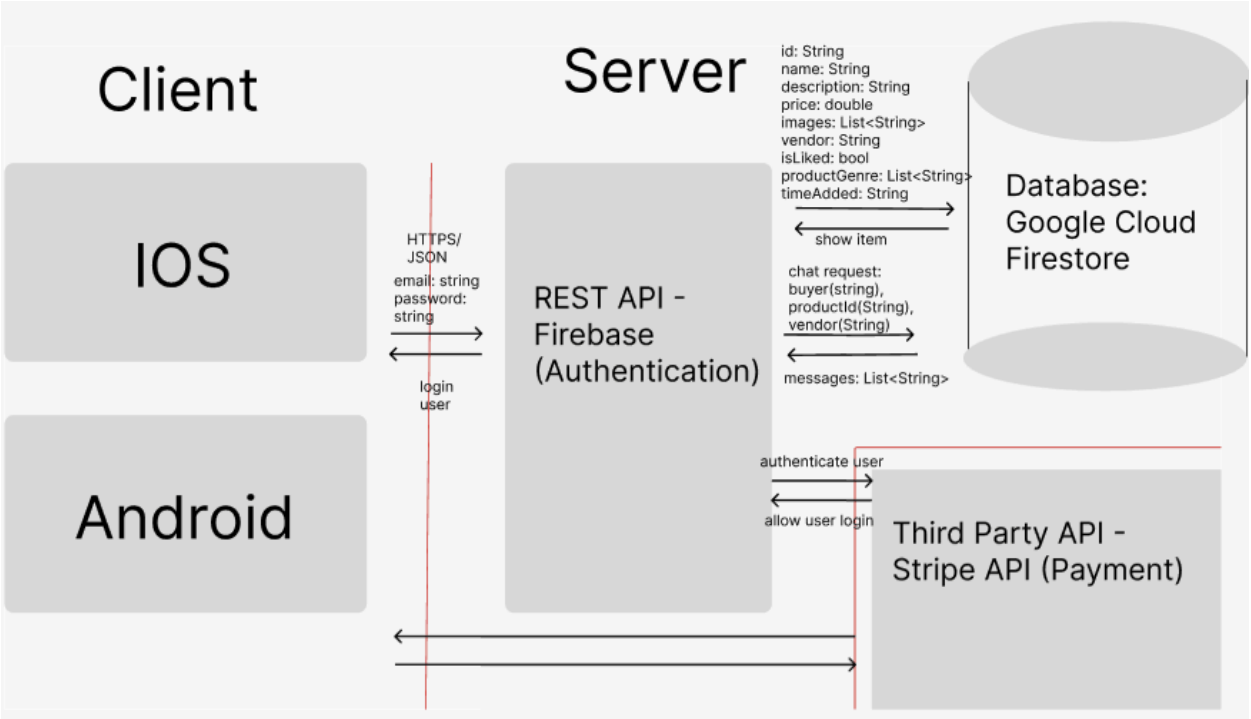
Logic to get a User's Listings

```dart
Future<List<String>> getUserListings() async {
  try {
    // Retrieve the document snapshot from Firestore
    DocumentSnapshot<Map<String, dynamic>> snapshot = await FirebaseFirestore
        .instance
        .collection('users')
        .doc(FirebaseAuth.instance.currentUser!.uid)
        .get();

    // Check if the document exists
    if (!snapshot.exists) {
      throw Exception('User not found');
    }

    // Extract user listings from the document data
    List<String>? userListings = snapshot.data()?['userListings'] != null
        ? List<String>.from(snapshot.data()?['userListings'])
        : [];

    return userListings;
  } catch (e) {
    // Handle errors
    print('Error retrieving user listings: $e');
    rethrow; // Rethrow the exception to be caught by the caller
  }
}
```

Logic to delete a user's listings

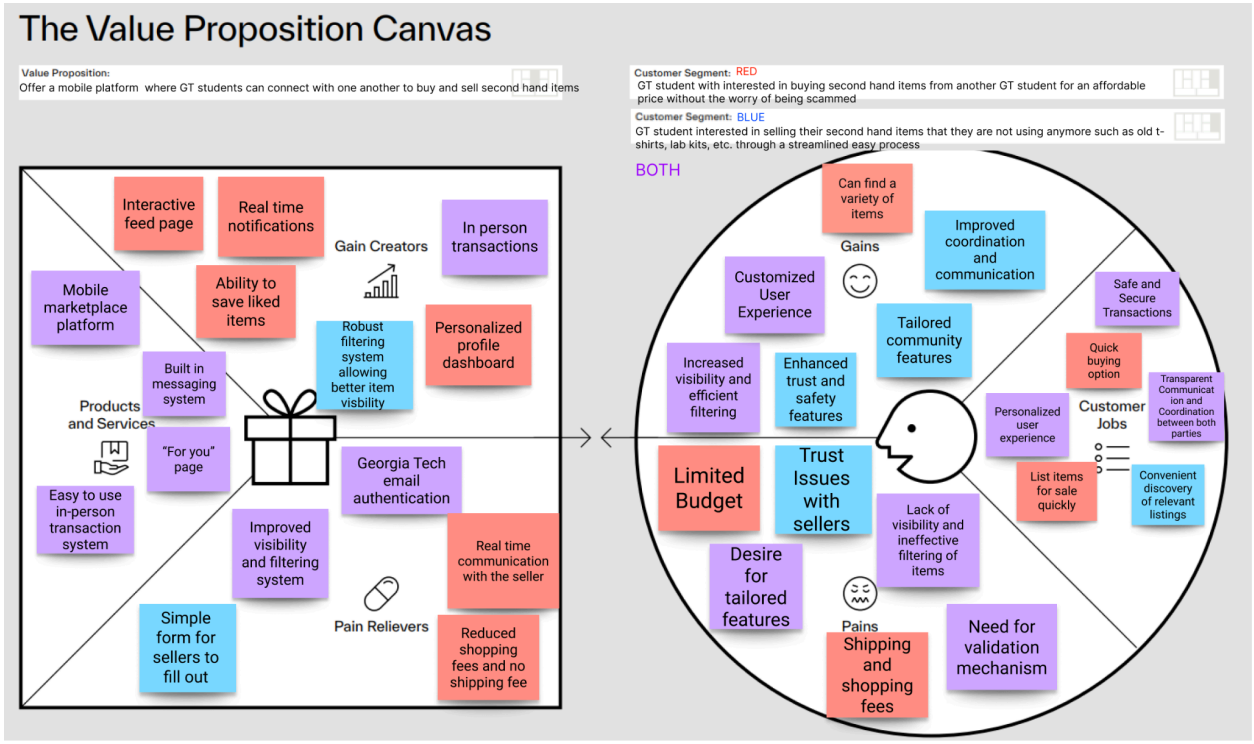
```
Future<void> removeUserListing(
    Map<String, String> productIDMappings, String productId) async {
  try {
    // Get the current user reference
    User? currentUser = FirebaseAuth.instance.currentUser;
    if (currentUser == null) {
      // Handle the case when no user is signed in
      throw Exception('No user signed in');
    }

    DocumentReference userRef =
        FirebaseFirestore.instance.collection('users').doc(currentUser.uid);

    List<String> userListings = await getUserListings();

    userListings.remove(productIDMappings[productId]!);
    updateUserListings(userRef, userListings);
  } catch (error) {
    // Handle errors here
    print('Error adding user listing: $error');
  }
}
```

New architecture diagram:

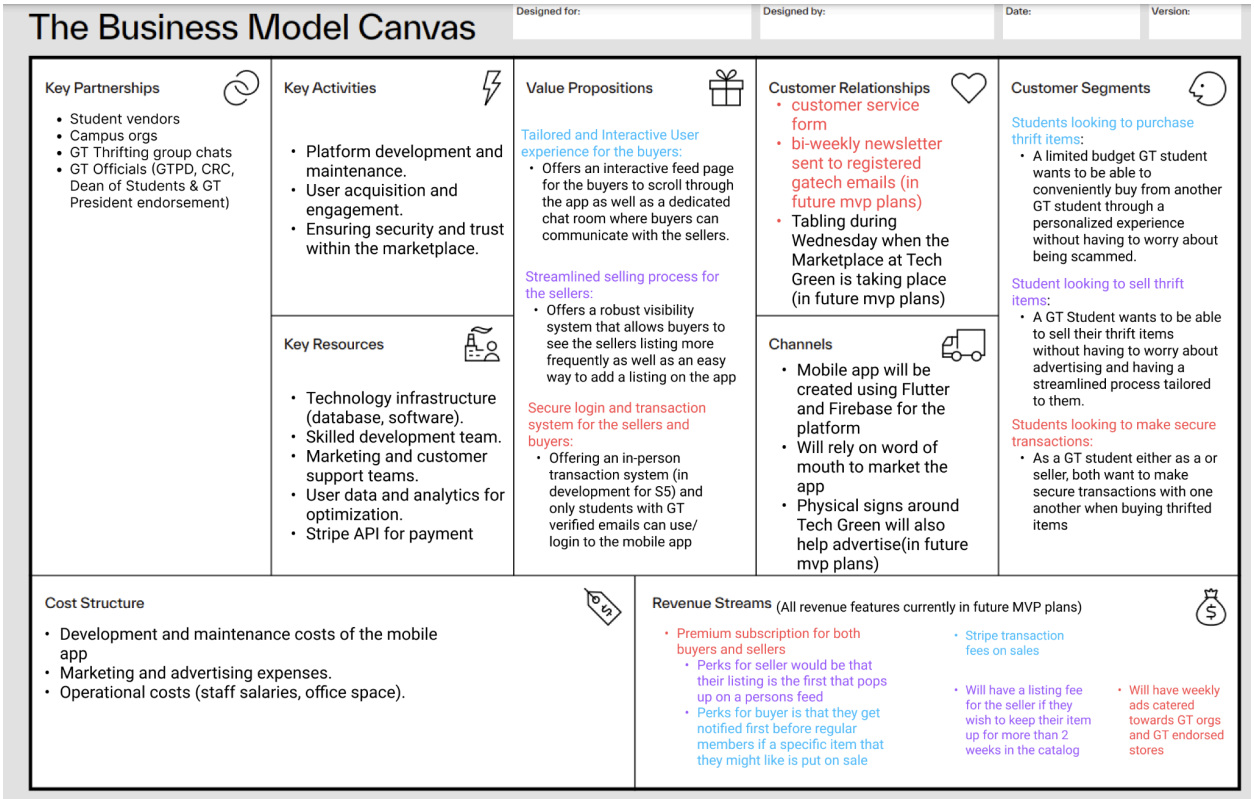


Updated system architecture diagram to account for the product genre and time added when listing a product. Also added how the API requests for a chat room using the buyer, productId, and vendor as inputs. Firestore then sets up a chatroom between the vendor and buyer.

Value Proposition Canvas: Updated to reflect current learning prototype



BMC (Includes VPC): Updated BMC for Sprint 5 in the key resources section and kept the same future MVP plans



The Business Model Canvas

Designed for: | Designed by: | Date: | Version:

Key Partnerships
- Student vendors
- Campus orgs
- GT Thrifting group chats
- GT Officials (GTPD, CRC, Dean of Students & GT President endorsement)

Key Activities
- Platform development and maintenance.
- User acquisition and engagement.
- Ensuring security and trust within the marketplace.

Key Resources
- Technology infrastructure (database, software).
- Skilled development team.
- Marketing and customer support teams.
- User data and analytics for optimization.
- Stripe API for payment

Value Propositions

Tailored and Interactive User experience for the buyers:
- Offers an interactive feed page for the buyers to scroll through the app as well as a dedicated chat room where buyers can communicate with the sellers.

Streamlined selling process for the sellers:
- Offers a robust visibility system that allows buyers to see the sellers listing more frequently as well as an easy way to add a listing on the app

Secure login and transaction system for the sellers and buyers:
- Offering an in-person transaction system (in development for S5) and only students with GT verified emails can use/ login to the mobile app

Customer Relationships
- customer service form
- bi-weekly newsletter sent to registered gatech emails (in future mvp plans)
- Tabling during Wednesday when the Marketplace at Tech Green is taking place (in future mvp plans)

Channels
- Mobile app will be created using Flutter and Firebase for the platform
- Will rely on word of mouth to market the app
- Physical signs around Tech Green will also help advertise(in future mvp plans)

Customer Segments

Students looking to purchase thrift items:
- A limited budget GT student wants to be able to conveniently buy from another GT student through a personalized experience without having to worry about being scammed.

Student looking to sell thrift items:
- A GT Student wants to be able to sell their thrift items without having to worry about advertising and having a streamlined process tailored to them.

Students looking to make secure transactions:
- As a GT student either as a or seller, both want to make secure transactions with one another when buying thrifted items

Cost Structure
- Development and maintenance costs of the mobile app
- Marketing and advertising expenses.
- Operational costs (staff salaries, office space).

Revenue Streams (All revenue features currently in future MVP plans)
- Premium subscription for both buyers and sellers
  - Perks for seller would be that their listing is the first that pops up on a persons feed
  - Perks for buyer is that they get notified first before regular members if a specific item that they might like is put on sale
- Stripe transaction fees on sales
- Will have a listing fee for the seller if they wish to keep their item up for more than 2 weeks in the catalog
- Will have weekly ads catered towards GT orgs and GT endorsed stores

S5 Feature Analysis:

1. Messages: This is in the final works, and will be completed next sprint.
   This has been completed this sprint, and users can text, send messages, and send payment requests through our messages feature
2. User Agreement Logic: What we intend to happen is that the buyer and seller meet in person, and they can verify each other through some sort of authentication process (could be through a 4 digit code like Uber, or scanning personal QR codes), and from this only then will the payment option open up on the app. From there, the buyer and seller can complete the transaction. This would be very important as it makes sure that the product is real, the buyer saw it in person, and that neither side of the transaction is fraudulent.
   This has not been completed, as from user research through A/B testing, users found it safe enough to talk to the vendors on the app, and for them to meet up in person to assess the quality of the product, as all vendors are GT students.
3. Payment API: Develop a safe payment scheme through secure payment APIs (Venmo/Apple Pay/Google Pay) - This is very important as this is how we see ourselves making money.

This has been completed. Using the Stripe Connect API, we are able to accept payments from buyers and send those to the sellers in theory.

4. User Profile Page Update: Big updates are needed in the user profile page, such as a way for them to edit their profile, and for them to see and take down their listings. Minor updates have been made, which include allowing users to see their listings and take them down, as well as a new profile image which will show up when conversing.

Updated Feature Analysis

1. Update the For-you and Explore Pages: Both of these pages are not updated in real-time, which makes it hard for users to see how they interacted with these pages without having to refresh the page. This issue was hard for us to deal with this sprint, as when trying to deal with it, the UI would always get compromised.
2. A “Popularity Feature”: Items which are being clicked on multiple times and being viewed by many buyers would be put on this page. This page would represent whats currently trending in the marketplace.
3. Better Payment Logic: Our current payment logic is that the buyers will pay to the central Stripe account, and the vendors would get paid out from this Stripe account. This is the closest we have gotten to achieving P2P payment system with the time permitted. However, this method is very complex and can most likely be done in another easier way through other mediums. Additionally, there is not much help online regarding integrating P2P payments in Stripe Connect through flutter, which makes it more difficult to make changes and updates in the future if we were to continue with this system.

Learning prototype results:

Model:
As mentioned in the learning prototype and the A/B section of the notebook, we did a lot of research and trial and error regarding various features. The images show the differences in what we have tested.

Methodology:
Essentially, what we did was that we conducted 20 test sessions with students on campus at GT. We also talked to those that work with various thrifting organizations at GT. The 20 people were randomly sampled from the street and took about 5 minutes. Here we talked about various features and showed these students the 2 options (A and B) that we were trying to test between. This time, we asked the testers what they preferred and also asked them to rank the features from 1 to 5 based on how they liked it. By doing this A/B testing and domain research, we were able to identify what features are important and truly understand what users are looking to get out of this application. In particular, we got a lot of insight in regards to what they are looking for from the UI.

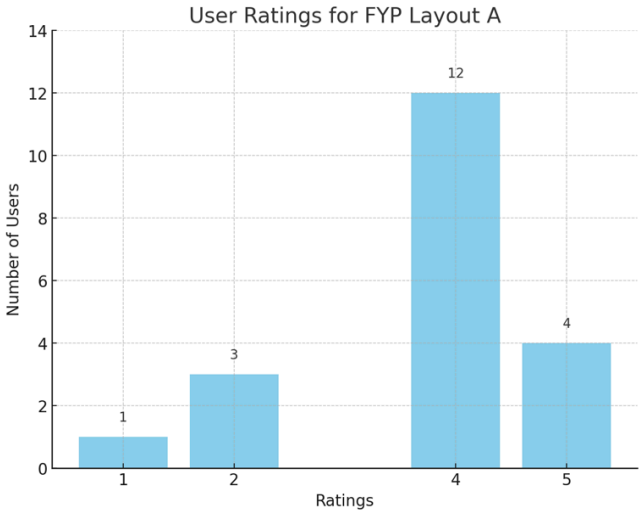
Key Feedback & Learnings:

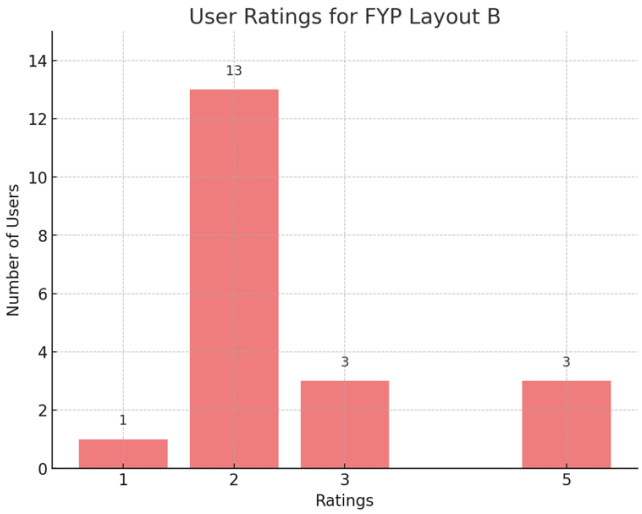
- For all of the features that we were doing A/B testing with
- Almost 90% of individuals thought liked the UI changes and preferred it to the old
- Most of the users preferred saving their information rather than it getting deleted for their own personal records
- Users enjoyed seeing exactly how their data is being used rather than it being hidden

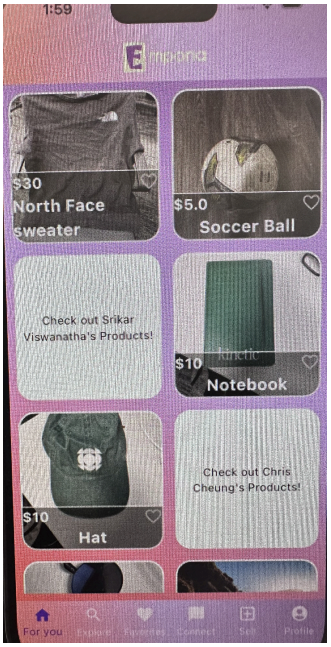
Takeaways:

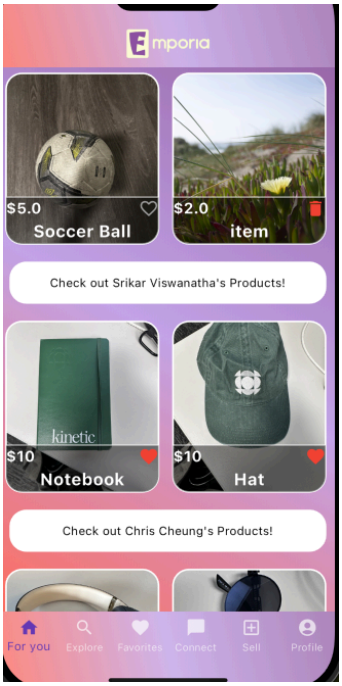
- UI was improved very much and users appreciate this, need to ask for more suggestions
- Users like the new messaging dashboard as it looks much more fluid and the data is kept for retention.
- The users like to see things that are more personalized for them
    - With the way that the A/B testing went for the FYP layout, users like it more when there is more personalization towards them so that they see exactly what they prefer.

A/B Individual Component Testing Data

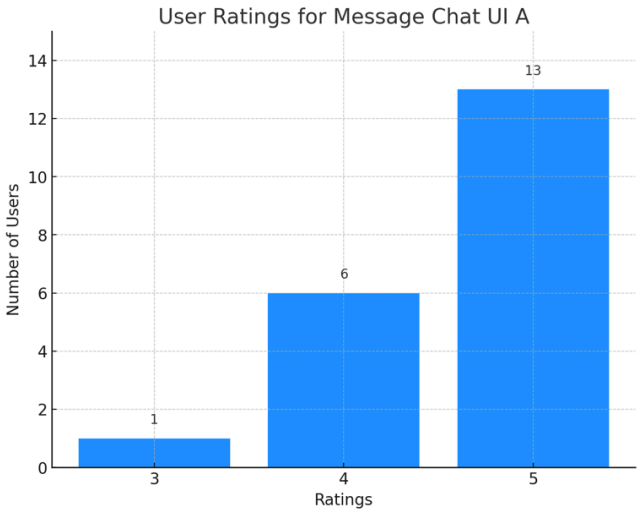


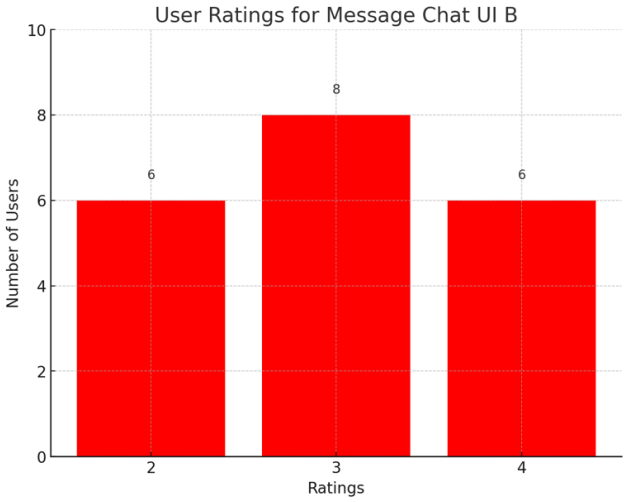


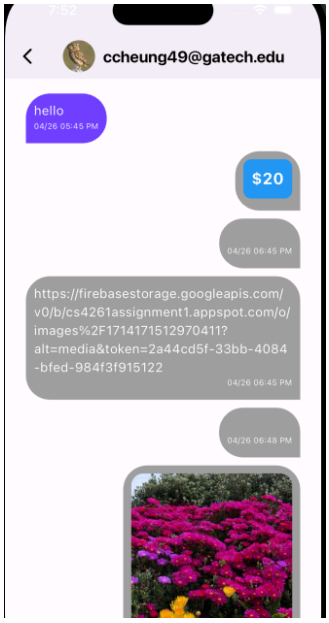


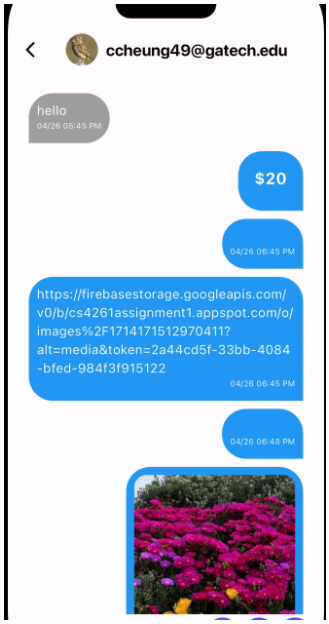


Layout A is represented on the right while layout B is represented on the left.
In evaluating the FYP layouts, Layout A was the clear favorite, with an average rating of 4.05 compared to Layout B's 2.78. Users appreciated Layout A for its user-friendly interface and modern aesthetics, which made navigation smooth and visually appealing. This positive feedback, along with strong ratings, led us to choose Layout A for our final design, ensuring it met both the functional and stylistic preferences of our users.

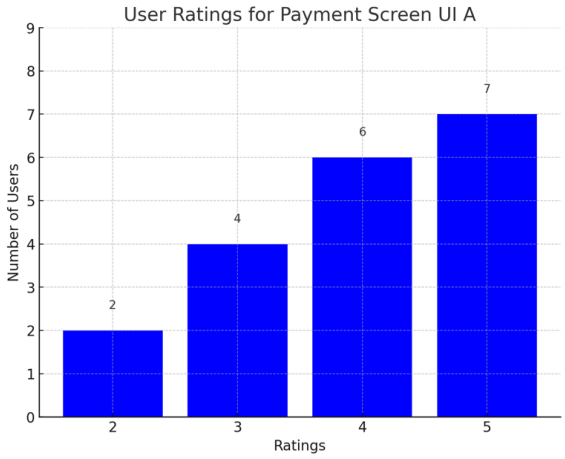


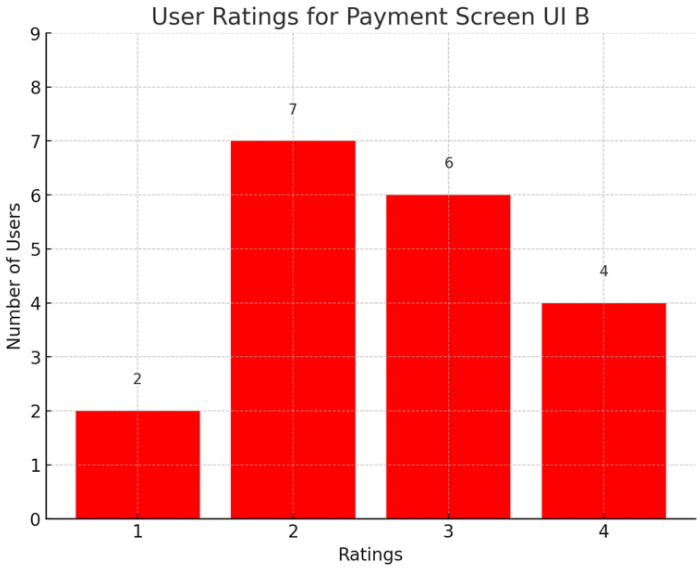


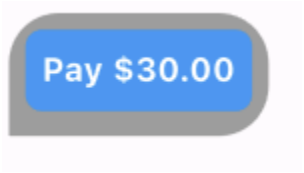


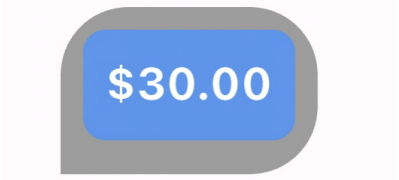


UI A is represented on the right and UI B is represented on the left.
In our review of the Message Chat UI options, UI A stood out with an average rating of 4.5, significantly higher than UI B's 3.17. Users preferred UI A for its simplicity and efficiency in facilitating communication, noting how easy it was to use during daily activities. The positive reception and high ratings for UI A influenced our decision to adopt it as the final choice, aiming to enhance user satisfaction and engagement through its streamlined design.

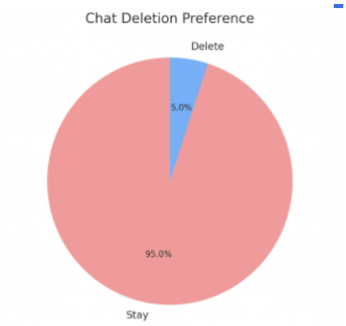


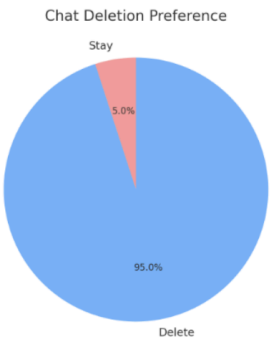


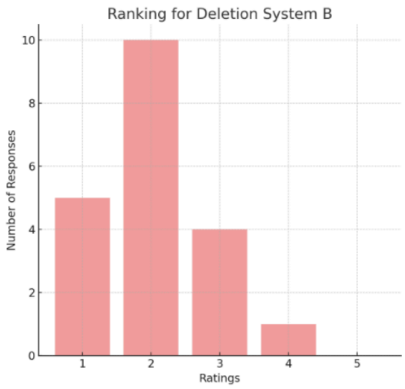


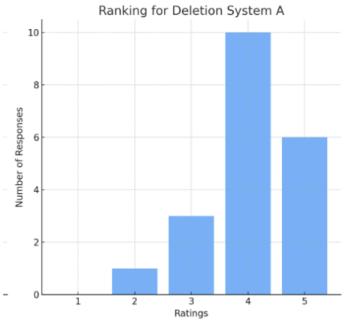


UI B is the top screenshot that says “Pay $30.00” while UI A is the bottom screenshot saying “$30.00”. For the Payment Screen UIs, UI A emerged as the preferred option with an average rating of 4.05, outperforming UI B, which had an average of 2.63. Users favored UI A for its clear layout and secure features, which made transactions straightforward and trustworthy. This positive feedback guided our decision to implement UI A, ensuring our payment system is both user-friendly and reliable, enhancing the overall user experience.







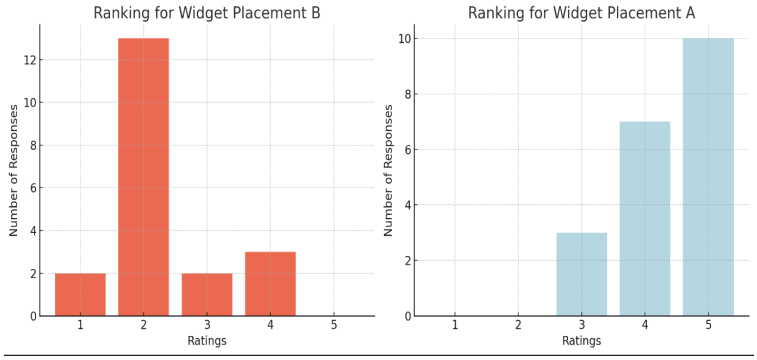


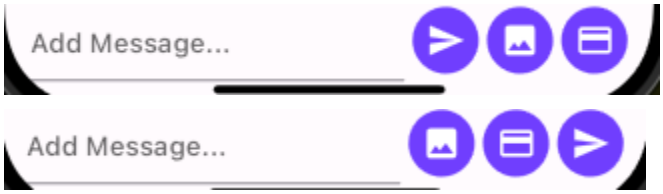
While collecting data, we found that a majority of users (95%) had a preference for having their chat history kept with the deletion of a product. This shows that for some people, saving chat history is important as they might want it for their records. On the other hand, a small sample (5%) preferred to have their chat history deleted on product deletion. This highlights a significant difference towards maintaining privacy and ensuring that no data is left behind.

All in all, the Feedback for Deletion System B was less favorable. While calculating the average rating, we got Deletion System B a score of 1.95 out of 5. This low score suggests that users are not satisfied with this feature and are looking for more to come out of it. This was the feature where the chat history was deleted and was not stored in the system memory.

On the other hand, the Feedback for Deletion System A was much more favorable. The average rating for Deletion System A is 3.95 out of 5 which was 2 whole points higher than system B. This version was able to meet user expectations better as it was providing functionality and user experience that was more favorable than that of System B. Seems like users like having their records stored for future reference.

Overall, the feedback shows a strong preference for systems that not only align with their privacy concerns. System A came out as the winning option as it offered more usability and resources for the user. There is a clear difference between System A and System B and it is evident that the reason one was better than the other was due to privacy concerns.





A is the top widget screenshot and b is the bottom widget screen shot.We did more A/B testing regarding the layout of the widget buttons. For B, the layout was upload picture, send payout, send message. For A, the layout was send message, upload picture, send payment. The average ranking for option B was 2.3 out of 5 and the average ranking for option A was 4.35.

Analytics and Data Collection.

Google Analytics was integrated into the app to track and analyze user behavior and app performance.It gathered data on user engagement metrics, such as session duration, bounce rates, and screen flow. It also collected data related to user activities within the app, such as items listed for sale, purchases made, and search queries. As we have implemented this feature recently, there is not enough data to make significant claim, but as we move on into our next sprint, this data collection will provide good insights into what we have to improve upon.

Event Tracking

- Track specific user interactions as events (e.g., listings viewed, purchases, sign-ups)
- Use event parameters to capture additional details (e.g., item category, price)

```
analytics.logViewItem(
    currency: 'usd',
    value: userItems[index].price,
    parameters: <String, dynamic>{
      'name': userItems[index].name,
      'id': userItems[index].id,
      'vendor': userItems[index].vendor,
      'productGenre':
          userItems[index].productGenre.toString()
    });
```

```
await analytics.logSearch(
    searchTerm: _searchController.text);
```

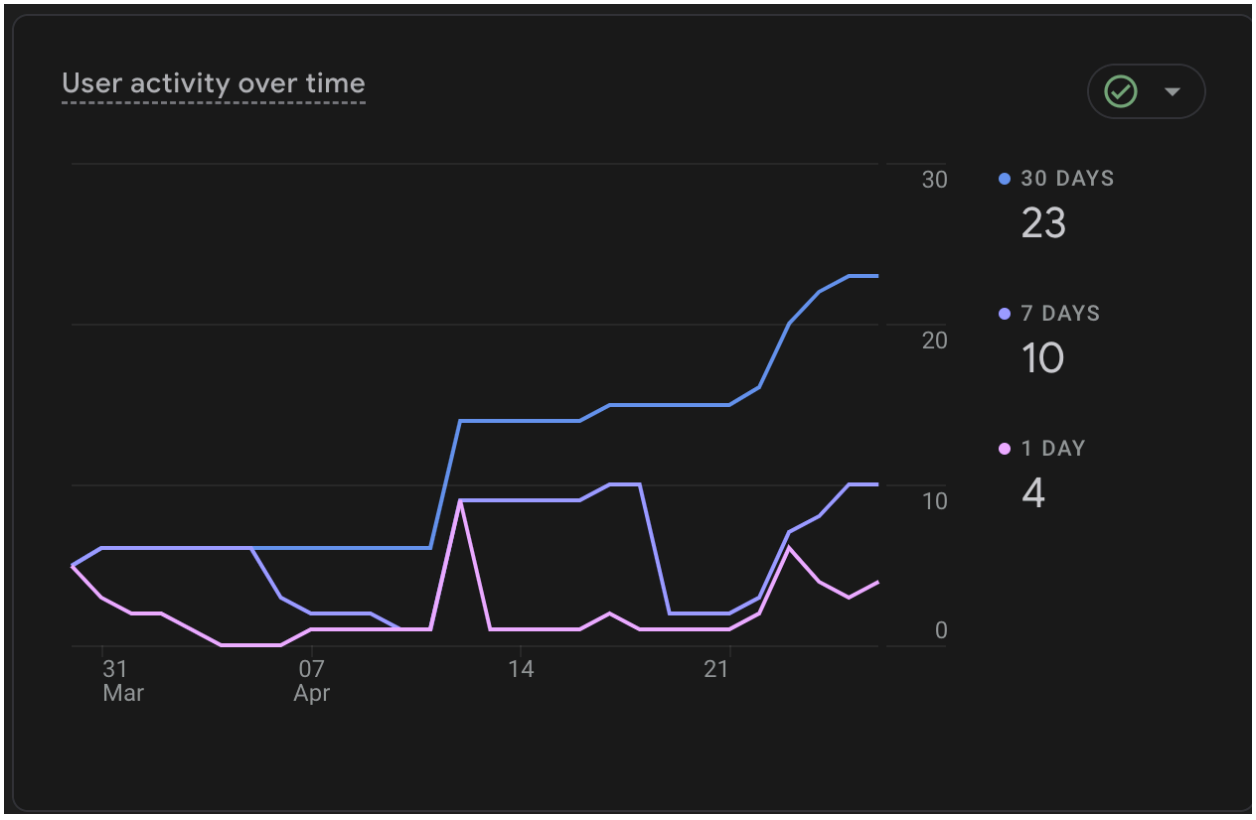
```
analytics.logEvent(name: "vendor_contact"),
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => ChatScreen(
      userId: product.vendor,
    ), // ChatScreen
  ), // MaterialPageRoute
)
```

```
await analytics.logSignUp(signUpMethod: "GaTech Email");
```

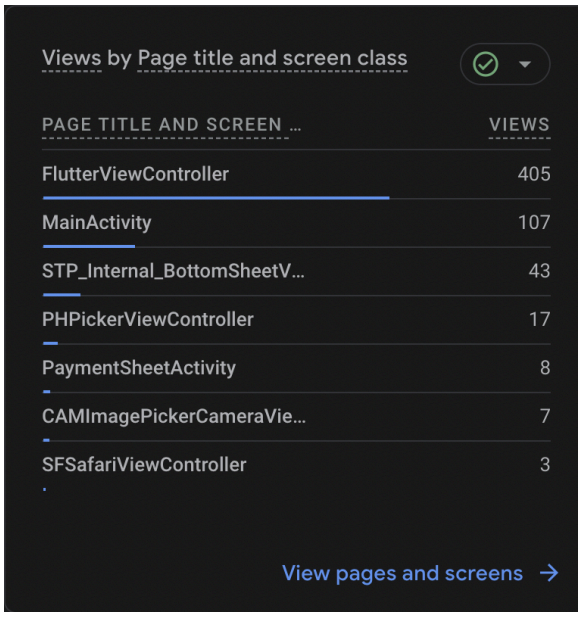
```
FirebaseAnalytics.instance.logAddToWishlist(
    currency: 'usd',
    value: widget.product.price,
    parameters: <String, dynamic>{
      'name': widget.product.name,
      'id': widget.product.id,
      'vendor': widget.product.vendor,
      'productGenre': widget.product.productGenre.toString()
    });
```

The four images above are from several sections in our code where we track user activity such as product views, search queries, seller messaging, and favorites feature usage across our app.

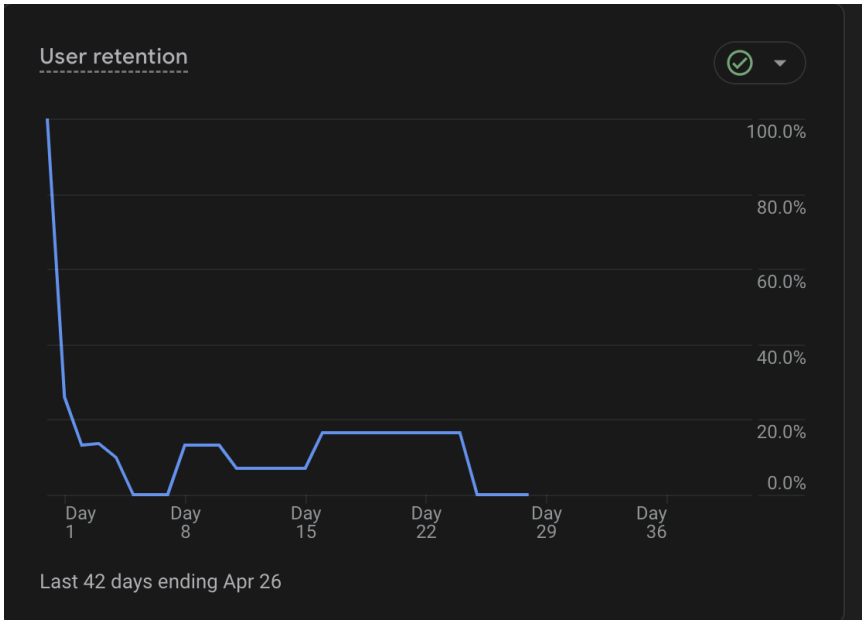
Sprint 5 Update: As a result of our user testing, several metrics in our analytics dashboard wwent up as can be show in the user activity over time graph below.



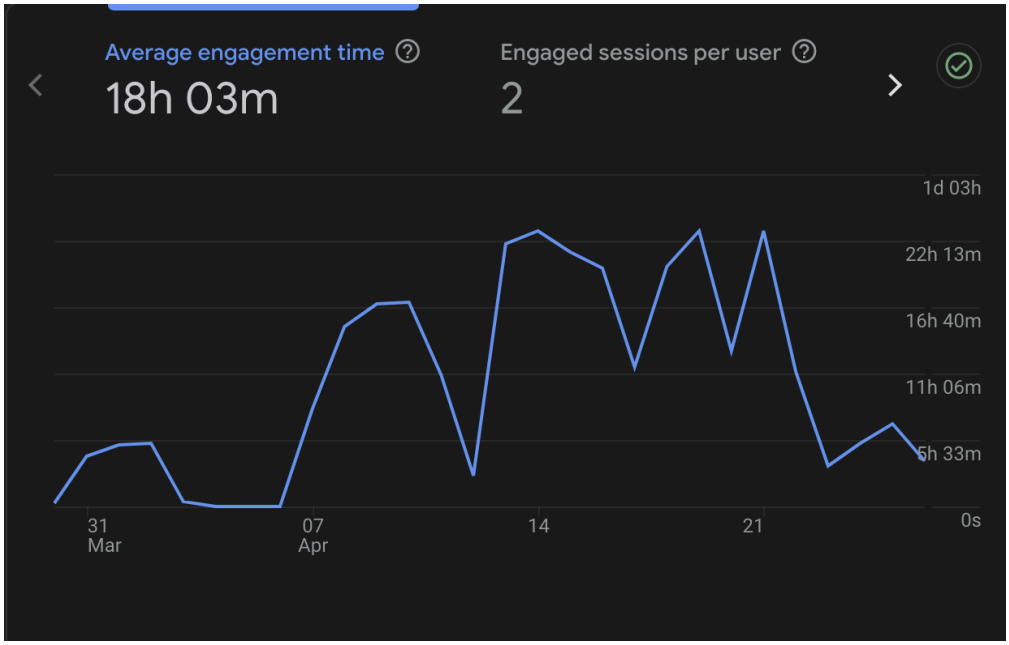
Sprint 5 Update: we tracked what page title the user was In the FlutterViewController and see what pages were frequently used.

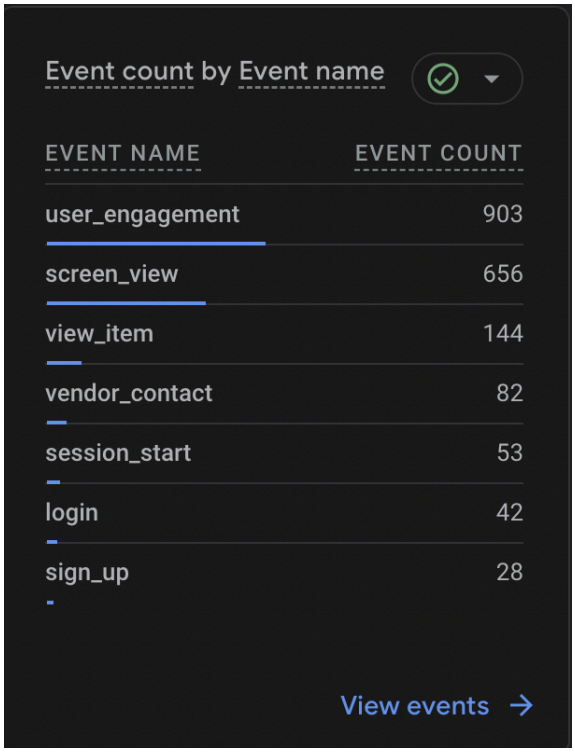


Sprint 5 Update: We also have a graph for tracking user retention. We expect retention to be low for now since we're just testing the app with limited users.



Sprint 5 Update: Tracking user engagement and activities performed in the app let's know where the users spend the most time on in the app.





The images above are from our firebase analytics dashboard logging several metrics like user retention, engagement, and events.

Future Direction:

Many of our intended features have been implemented throughout these 5 sprints, but we believe that more can be done in the future to make Emporia better. At the heart of this course, as we have learned, is the user experience. We have seen this extensively through extensive user interviewing, developing the use case, and talking about A/B testing. These all are centered towards users, and that is the direction we want to move on in the future - making sure that we take into account the user's experience and opinions as we develop. We have some additional features in plan, which were not able to make our current prototype due to time constraints.

Some features that we want to implement in the future include better payment options and revenue streams.

- Showing a user authentication code is something that we are looking into. Like other mobile apps, such as Uber, having an authentication code would help improve the security of the users and make transactions even more secure.
- A feature that we are interested in implementing is a subscription system. The way we would implement this is by using the new UI that we decided to implement for the for you

page. If a vendor decides to pay a premium subscription they can have their name appear close to the top of the feed making their items more likely to be clicked on.

We are also interested in implementing AI within our app. This would include having an AI assistant that would help the user lsit their items automatically instead of having to manually input the information everytime. This would help expedite the time it takes for vendor's to list their items if they have a lot to sell.

We are also interested in implementing an edit item functionality as it will allow the sellers to change their item listings without having to delete them entirely.

Biggest concerns updated:

- Some of our biggest concerns are regarding the actual payment between the buyer and the seller.
- Quality Control: Ensuring that the quality of the items is exactly as promised in the contract. There needs to be some type of management in order to ensure that new items are actually in "new" condition and so forth.
- Dispute Resolution: If there is a dispute between the seller and the buyer, there needs to be a resolution that satisfies both parties to ensure customer satisfaction.
- Scams: Some products might look like real products, for expensive items we should have an authentication service that ensures that these products are as they are marketed.
- Ensuring the app is accessible to all college students, inclusive of people with disabilities. This includes compliance with ADA standards and making sure the UI/UX is navigable for people with quite a number bodily and cognitive competencies.
- Beyond the preliminary charge worries, there's the chance of fraudulent transactions wherein stolen credit playing cards or faux financial institution bills are used.
- Protecting the platform against cyber threats, consisting of hacking, phishing assaults, and malware, to protect personal facts and transactions.
- Ensuring the platform complies with all relevant legal guidelines and policies, which include tax legal guidelines and client protection legal guidelines. (GDPR)
- Updating the backend so that things develop in real time and render in real time without severely impacting the UI
- Find out better ways to facilitate P2P transactions

Other Questions Left to Answer:

- How are we going to ensure fake items are not being sold
- How are we going to regulate that there are no scams being done
- How are we going to regulate any disputes between the buyer and seller
- How are we going to manage missed meet up sessions?

- How are we going to ensure that the quality of the items are up to standard that they are described
- How are we going to ensure that we follow GDPR guidelines

Code Review:
In our efforts to deliver features quickly, we prioritized getting the core functionality working over implementing all the bells and whistles. This approach, while effective in getting the product out the door, has resulted in a buildup of technical debt. Moving forward, we need to address this tech debt to ensure a more reliable, performant, and secure system in the long run.

For the code organization our app is split up into 6 different sections: dashboard, chat, products, profile, screens, and services.
After running main the user is authenticated by logging in with their email and password. They are then taken to the dashboard where they can use the app to the full extent.

```
Future<List<Product>> userPreferenceProducts(
    List<Product> allProducts) async {
  List<Product> userPreferredProducts = [];

  List<bool> userPreferences = await getPreferences();

  for (int i = 0; i < allProducts.length; i++) {
    bool isMatch = false;
    List<bool> productGenre = allProducts[i].productGenre;

    for (int j = 0; j < 9; j++) {
      if (userPreferences[j] == true &&
          productGenre[j] == userPreferences[j]) {
        isMatch = true;
        break;
      }
    }

    if (isMatch) {
      userPreferredProducts.add(allProducts[i]);
    }
  }

  return userPreferredProducts;
}
```

This function is a fundamental piece of our platform's strategy to provide personalized product recommendations to users, which is essential for maximizing engagement. The function starts by taking in a comprehensive list of all the products in the database. Then, it dynamically fetches the user's preferences, translating them into a series of boolean values representing their likes and dislikes across different product genres. Using this information, the function compares each product with its genre and the user's preferences. Whenever it finds a match, indicating that the user has expressed interest in that specific genre and the product aligns with it, it adds that product to a special list. Finally, it returns a list of products that match with the user's interests.

```
Future<void> sendPaymentRequest(
    String? email, double price, BuildContext context) async {
  try {
    print("Function entered");
    final response = await http.post(
        Uri.parse(
            "https://us-central1-cs4261assignment1.cloudfunctions.net/stripePaymentIntentRequest"),
        body: {
          'email': email,
          'amount': (price * 100).toString(),
        });

    final jsonResponse = jsonDecode(response.body);
    print(jsonResponse.toString());

    await Stripe.instance.initPaymentSheet(
        paymentSheetParameters: SetupPaymentSheetParameters(
      paymentIntentClientSecret: jsonResponse['paymentIntent'],
      merchantDisplayName: 'Emporia',
      customerId: jsonResponse['customer'],
      customerEphemeralKeySecret: jsonResponse['ephemeralKey'],
    ));
    print("Made payment sheet");

    await Stripe.instance.presentPaymentSheet();
    print("done");

    ScaffoldMessenger.of(context)
        .showSnackBar(const SnackBar(content: Text('Payment is successful')));
  } catch (error) {
    if (error is StripeException) {
      ScaffoldMessenger.of(context).showSnackBar(SnackBar(
          content: Text('Error occured: ${error.error.localizedMessage}'))); // SnackBar
    }
  }
}
```

Payment request functionality that sends a payment request to a Stripe API that allows the vendor to pay the system.

```
Future sendVerificationEmail() async {
  try {
    final user = FirebaseAuth.instance.currentUser!;
    await user.sendEmailVerification();
  } catch (e) {}
}

Future checkEmailVerified() async {
  await FirebaseAuth.instance.currentUser!.reload();
  setState(() {
    isEmailVerified = FirebaseAuth.instance.currentUser!.emailVerified;
  });

  if (isEmailVerified) {
    timer?.cancel();
  }
}
```

Email Authentication that checks if the user is using a gatech email using regex. Much

```dart
Future<void> _sendText(BuildContext context, bool isPayment, String content,
    bool isImage) async {
  if (docRef == "none") {
    print("new docref");
    final checksAndBalances = await FirebaseFirestore.instance
        .collection("chat")
        .where(Filter.and(
            Filter("buyer",
                isEqualTo: FirebaseAuth.instance.currentUser!.email),
            Filter("vendor", isEqualTo: widget.vendor),
            Filter("productId", isEqualTo: widget.productId)))
        .get();
    if (checksAndBalances.docs.isEmpty) {
      final newChatRef =
          await FirebaseFirestore.instance.collection('chat').add({
        "productId": widget.productId,
        "vendor": widget.vendor,
        "buyer": FirebaseAuth.instance.currentUser!.email
      });
      setState(() {
        docRef = newChatRef.path;
      });
      await FirebaseFirestore.instance
          .collection("${newChatRef.path}/messages")
          .add({
        "content": content,
        "messageType": isPayment
            ? "payment"
            : isImage
                ? "image"
                : "text",
        "senderId": FirebaseAuth.instance.currentUser!.email,
        "sentTime": DateTime.now(),
      });
    }
  } else {
    await FirebaseFirestore.instance.collection("$docRef/messages").add({
      "content": content,
      "messageType": isPayment
          ? "payment"
          : isImage
              ? "image"
              : "text",
      "senderId": FirebaseAuth.instance.currentUser!.email,
      "sentTime": DateTime.now()
    });
  }
  controller.clear();
  FocusScope.of(context).unfocus();
}
```

This code (`_sendText`) handles sending text messages within a chat context. It efficiently checks for an existing chat document and creates a new one if needed. Messages are stored in subcollections for organization. While the logic is clear and avoids unnecessary checks, there are areas for improvement.

One key consideration is error handling. Currently, the code doesn't handle potential issues like failing to create a chat document or add a message. Implementing proper error handling mechanisms would provide a better user experience by displaying informative messages in case of failures.

Another point to consider is performance optimization. For scenarios with a large number of messages, pagination for retrieving messages could be explored. Additionally, security might need further attention. While user email is used as sender ID, a more robust authorization system could be implemented if sensitive information is being sent.

These potential improvements are primarily due to the presence of "tech debt." This means that the current implementation prioritizes functionality over more robust error handling, performance optimization, and security features. While this approach might have been necessary to deliver a working feature quickly, addressing the tech debt in the future would lead to a more reliable, performant, and secure chat system.

```
print(userItems);
return ListView.separated(
  itemBuilder: (context, index) {
    print("out: $index");
    final rowIndex = index ~/ 2; // Calculate row index
    final firstItemIndex = index * 2;
    final secondItemIndex = firstItemIndex + 1;

    print("in: $index");

    if(secondItemIndex < userItems.length && firstItemIndex < userItems.length){
    return Row(
      children: [
        Expanded(
          child: Padding(
            padding: const EdgeInsets.only(
                top: 10,
                bottom: 10,
                left: 5,
                right:
                    10), // Add padding of 5 pixels on all sides // EdgeInsets.only
            child: SquareTileProduct(
              product: userItems[firstItemIndex],
              onTap: () {
                analytics.logViewItem(
                    currency: 'usd',
                    value: userItems[firstItemIndex].price,
                    parameters: <String, dynamic>{
                      'name': userItems[firstItemIndex].name,
                      'id': userItems[firstItemIndex].id,
                      'vendor':
                          userItems[firstItemIndex].vendor,
                      'productGenre': userItems[firstItemIndex]
                          .productGenre
                          .toString()
                    });
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (context) => ProductDetailScreen(
                        userItems[firstItemIndex]), // ProductDetailScreen
```

Ln 648, Col 1 Spaces: 2 UTF

This is the main code for our UI layout in the for-you page. This was especially hard to do, as changing the index of the regular builder is nearly impossible. After multiple different attempts to try getting the 2-product 1 -user screen, we were able to use the ListView separated class to iterate over the products and the users at the same time.
As we loaded data form the database dynamically, it was difficult to

```dart
Future<void> initializeUserData() async {
  final currentUserUid = FirebaseAuth.instance.currentUser!.uid;
  final userRef =
      FirebaseFirestore.instance.collection('users').doc(currentUserUid);

  final userSnapshot = await userRef.get();

  if (userSnapshot.exists) {
    favoriteProductsID =
        List<String>.from(userSnapshot['favoriteProducts'] ?? []);
    products = FirebaseFirestore.instance.collection('products');
  } else {
    print('User document does not exist');
    favoriteProductsID = [];
  }
}

Future<List<Product>> getFavoriteProducts(List<String> favoriteProducts) async {
  List<Product> userFavoriteProducts = [];

  for (final id in favoriteProducts) {
    final snapshot =
        await FirebaseFirestore.instance.collection('products').doc(id).get();
    if (snapshot.exists) {
      final data = snapshot.data() as Map<String, dynamic>;
      final product = Product(
        id: data['id'],
        name: data['name'],
        description: data['description'],
        price: data['price'].toDouble(),
        images: List<String>.from(data['images']),
        vendor: data['vendor'],
        isLiked: data['isLiked'],
        timeAdded: data['timeAdded'],
        productGenre: List<bool>.from(data['productGenre']),
      ); // Product
      userFavoriteProducts.add(product);
    }
  }

  return userFavoriteProducts;
}
```

This is our favorites page. The database loads the user's favorite images, then filters them out of the products.

Link to Figma Mockups and Value Proposition Canvas:
https://www.figma.com/file/dt2k7liqvLixZUzhw5ru3x/Mockup-1?type=design&node-id=23%3A107&mode=design&t=BUFtMKEGqeuGeGnk-1

Link to Github:
https://github.com/sviswanatha5/EmporiaApp