App design challenges

Group member: Siyi Du, Sicheng Chen

a) Emergency response app.

Every year at BU incoming freshman are overwhelmed by the
city and occasionally get themselves into dangerous situations. What are some of your ideas for an App that would enable someone to know where it is safe to go and, if in trouble, quickly and easily notify others.

i) Consider the different sensors on an Android Handset.

ii) Also consider the possibility of crowdsourcing real-time and archived data

Solution: Emergency Response App

Features:

1. Safety Map:

Utilize the phone's GPS to display a map of the user's surroundings.

Mark safe zones (like police stations, hospitals) and high-risk zones based on historical and real-time data.

Crowdsourced data from users and local authorities can keep the map updated.

Enable route suggestions to nearest safe zones.

2. SOS Button:

A prominently displayed SOS button that, when pressed, sends the user's location and a distress message to pre-set contacts, local authorities, and nearby app users.

Optionally activate the phone's camera and microphone to record the situation, which can be automatically sent along with the distress message.

Auto-dial local emergency number.

3. Safety Tips and Information:

Provide safety tips and information based on the user's current location.

Link to local laws, rights, and emergency procedures.

4. Real-Time Alerts:

Send real-time alerts about nearby incidents or areas to avoid.

Crowdsourcing and integration with local authorities' systems for up-to-date information.

How it Works:

1. User Registration:

Users register and set up their profile with emergency contacts and other necessary details.

2. Map Navigation:

On launching the app, users see a map interface with their location, safe and unsafe zones, and route suggestions to safe zones.

3. Emergency Situations:

In an emergency, users press the SOS button.

The app sends alerts with the user's location and other details to the contacts and authorities.

The app auto-dials the emergency number and starts recording audio and video.

4. Receiving Alerts:

Users receive real-time alerts about their surroundings.

5. Information Access:

Users can access safety information and tips based on their location.

Sensors/Tools Used:

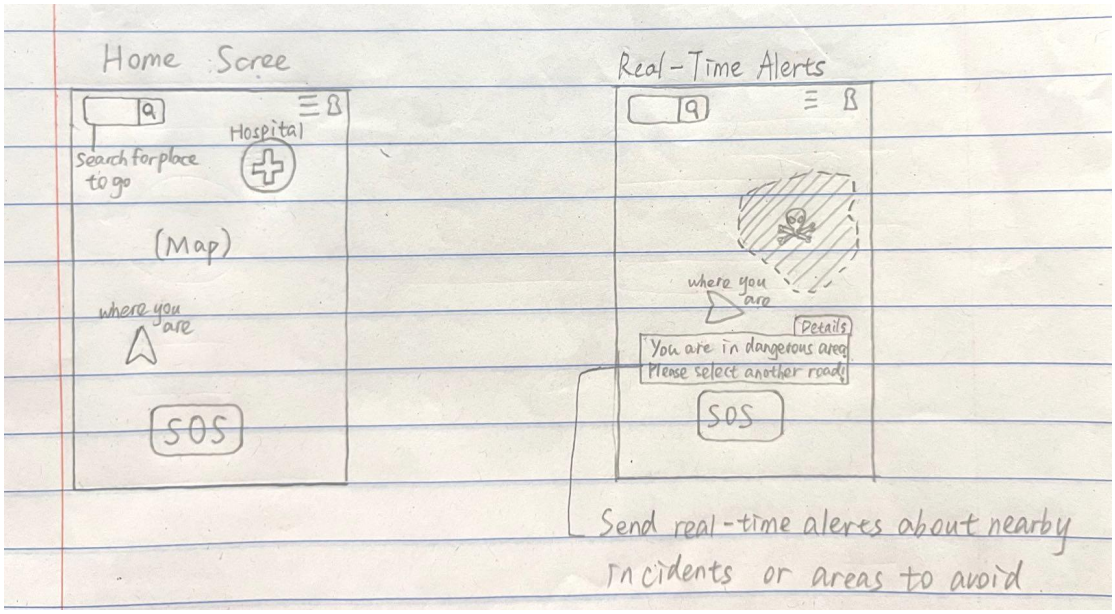
GPS: For location tracking and mapping.

Camera: For recording incidents when SOS is activated.

Microphone: For recording audio in emergencies.

Internet Connection: For sending alerts and updating maps and data.

SMS/Calls: For sending distress messages and making emergency calls.



b) Contractor for you.

This is an app that connects professional contractors with individuals
who need work done on their home. If someone were to pay you to design this app,
what are some of the things you would need to consider? What would some of the
requirements be in terms of device hardware/somware/back-end storage, etc?

Design Considerations:

1. User Profiles:

Separate profiles for contractors and homeowners.

Verification features for contractors' credentials, licenses, and references.

2. Project Listings:

Homeowners can post detailed descriptions of the work they need to be done, including photos, expected completion time, and budget.

Contractors can search and respond to listings.

3. Bidding System:

Contractors can bid on projects, and homeowners can review and accept bids.

4. Communication:

In-app messaging system for users to discuss project details securely.

5. Ratings and Reviews:

Both parties can leave ratings and reviews upon project completion.

6. Payment System:

Secure in-app payment system, including escrow features to protect both parties.

7. Location Services:

Use GPS to find local contractors and projects.

Technical Requirements:

Device Hardware/Software:

1. GPS:

For location-based searches and services.

2. Camera:

Allows homeowners to upload pictures of the work area, and contractors to upload portfolios or completed work.

3. Notifications:

Push notifications for project updates, bids, messages, etc.

4. Internet Connection:

Continuous internet access for real-time updates.

Back-End Storage/Infrastructure:

1. Database:

Store user profiles, project listings, bids, messages, and reviews.

Consider scalable solutions like AWS RDS or Google Cloud SQL.

2. Server:

Handle requests from the app, such as user registration, project listing, and bid submission.

Scalable cloud solutions like AWS EC2 or Google Compute Engine could be used.

3. Storage:

Store images and documents uploaded by users.

Services like Amazon S3 or Google Cloud Storage can be used.

4. Payment Gateway:

Integrate a secure payment gateway for handling transactions.

Examples include Stripe, PayPal, or Square.

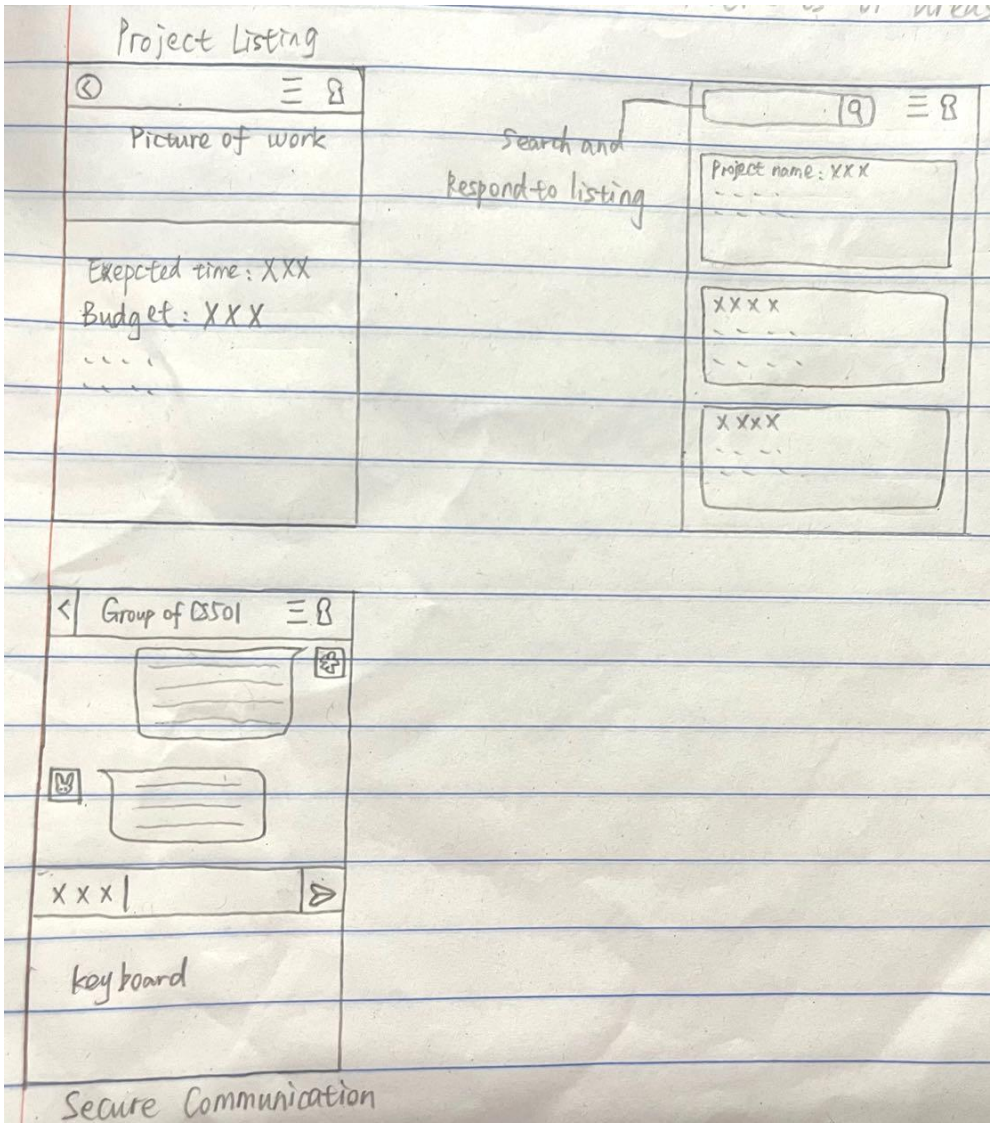
5. Security:

Implement robust security measures to protect user data and payments.

Use HTTPS, data encryption, and secure authentication methods.

6. APIs:

Develop or integrate APIs for mapping, payment, and other services.



c) Don’t flfleece me dude.

Quite a few users of credit cards do not regularly check their statements. Or, when they do, they check them long amer making a charge. Unscrupulous vendors might take advantage of this laxness. Let's focus on one specifific area, Vpping at restaurants.

Design an app that would enable a restaurant patron to validate that the Vp they lem is the same as the Vp that was charged. For example, when you go to a restaurant, a hold is placed on your credit card and a Vp is added amer you leave. What if an unscrupulous waiter charged you a diffffferent amount, then you had wriUen in? How could you automaVcally be noVfified that this occurred?

We design an app called “TipGuard”. The "TipGuard" app is designed to help users ensure that the tip amount they leave at restaurants is the same amount that gets charged to their credit card. The app will use OCR (Optical Character Recognition) technology to read the receipt amount and tip, and connect to the user’s credit card transaction data to verify the charged amount.

Features:

1. Receipt Scanner:

Use the phone's camera to scan the restaurant receipt.

Extract total amount and tip using OCR technology.

2. Transaction Monitoring:

Link the app to the user's credit card account to monitor transactions.

Verify the charged amount with the scanned receipt amount and tip.

3. Notification:

Notify the user if the charged amount does not match the receipt.

Provide options to dispute the charge or mark it as correct.

4. History:

Keep a history of scanned receipts and corresponding transactions for reference.

5. Dispute Assistance:

Assist the user in raising a dispute by providing the necessary details and steps.

How it Works:

1. User Registration:

Users register and link their credit card accounts to the app.

2. Scanning the Receipt:

After paying at a restaurant, the user scans the receipt using the app.

The app uses OCR to extract the total amount and tip from the receipt.

The app stores this information and continuously monitors the linked credit card transactions.

3. Transaction Verification:

When the transaction appears on the user's credit card, the app compares the charged amount with the stored receipt information.

If there is a mismatch, especially in the tip amount, the app immediately notifies the user.

4. Dispute Handling:

If there is a discrepancy, the app provides the option to dispute the charge.

The app can auto-fill dispute forms with relevant information, making it easier for the user to report the issue.

Device Hardware/Software Requirements:

Camera:

For scanning the restaurant receipt.

Internet Connection:

For monitoring credit card transactions and sending notifications.

OCR Software:

To extract text from the scanned receipts.

Secure Authentication:

Secure methods to link and verify the user's credit card account with the app.

Notifications:

To alert the user about any discrepancies.

Security Considerations:

1. Data Encryption:

All data, especially credit card information and transactions, should be encrypted.

2. User Authentication:

Implement robust user authentication methods to ensure only the authorized user has access to the app.

3. Privacy Compliance:

Ensure the app complies with data protection laws and regulations.

Example User Interface:

1. Home Screen:

Options to Scan Receipt, View Transaction History, and Settings.

2. Scan Receipt Screen:

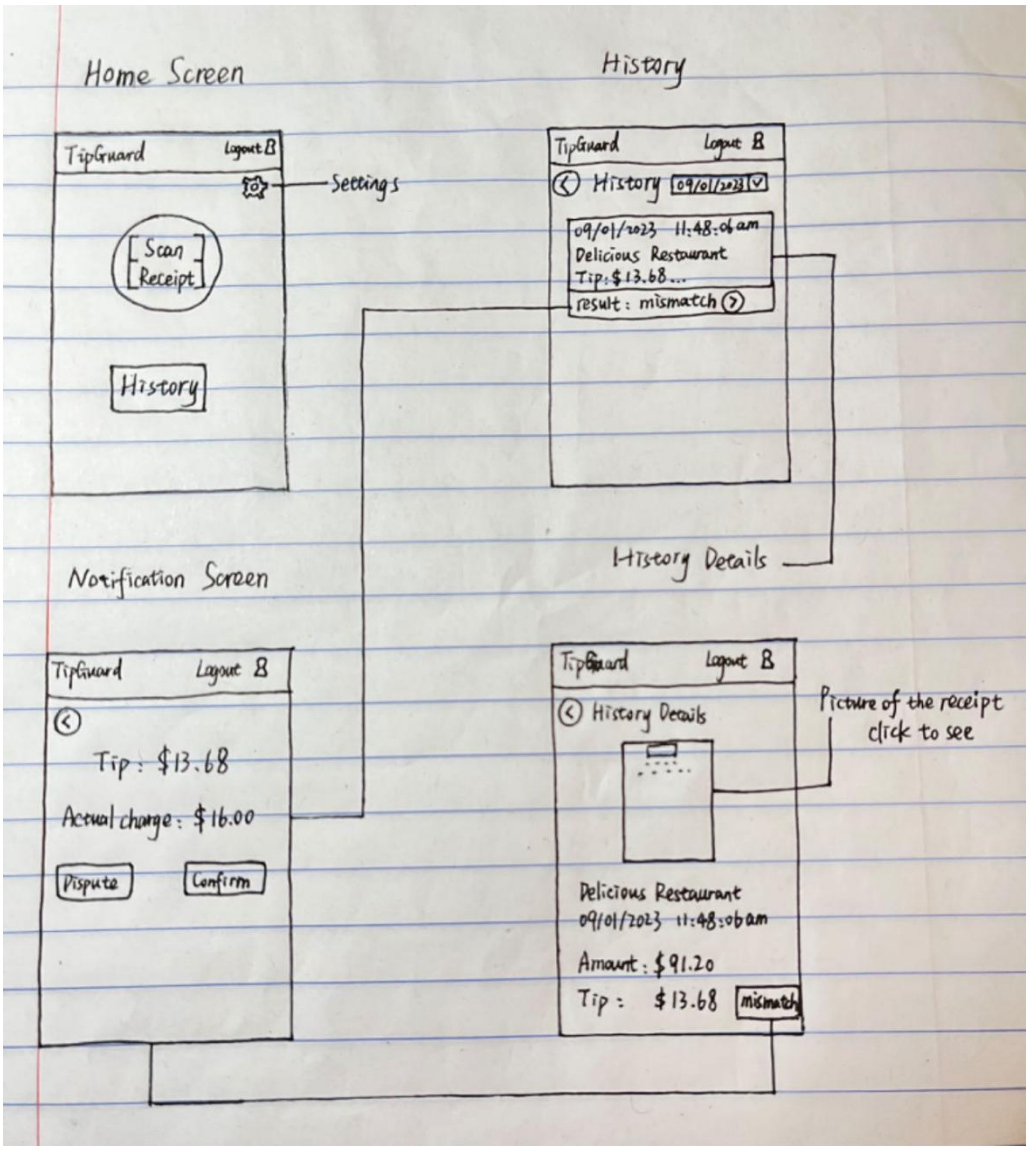
Access to the phone's camera to scan the receipt and automatically extract and store the amount and tip.

3. Transaction History:

List of scanned receipts and their corresponding credit card transactions, with a status indicating a match or mismatch.

4. Notification Screen:

Details of the mismatch and options to dispute or confirm the charge.


Home Screen
History
TipGuard
Logout
Settings
Scan Receipt
History
TipGuard
Logout
History
09/01/2023
09/01/2023 11:48:06 am
Delicious Restaurant
Tip: $13.68...
result: mismatch
History Details
Notification Screen
TipGuard
Logout
Tip: $13.68
Actual charge: $16.00
Dispute
Confirm
TipGuard
Logout
History Details
Picture of the receipt click to see
Delicious Restaurant
09/01/2023 11:48:06am
Amount: $91.20
Tip: $13.68
mismatch