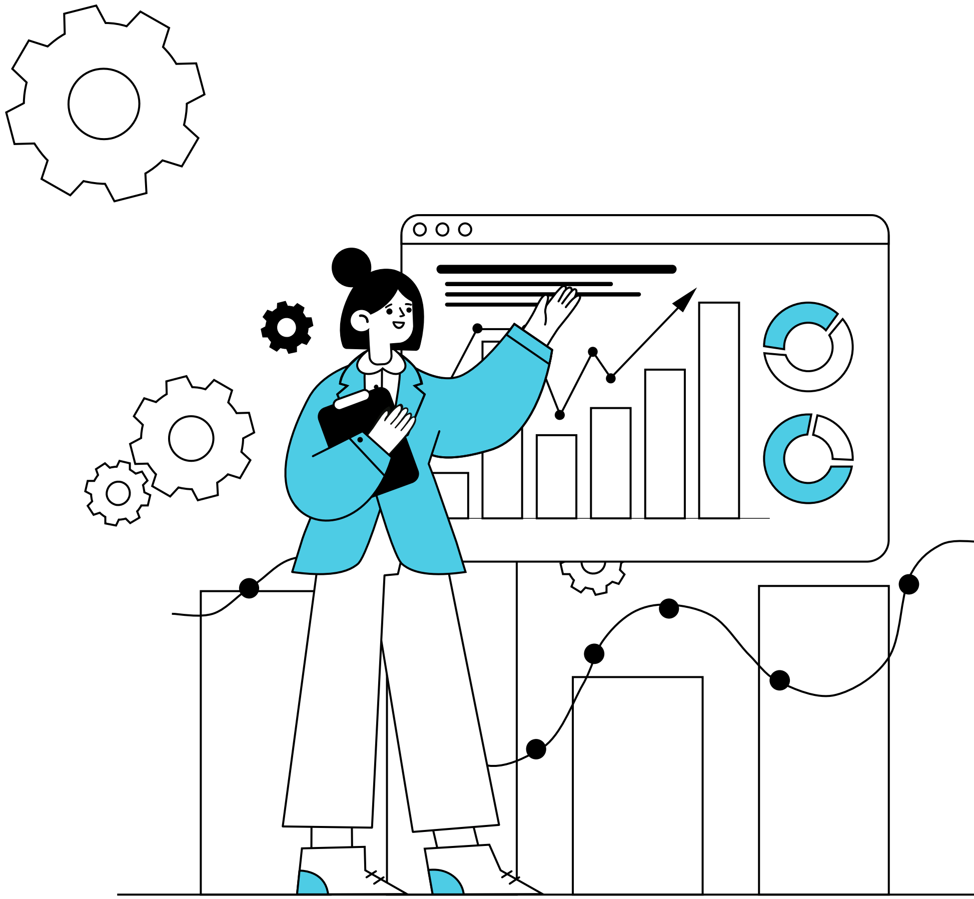
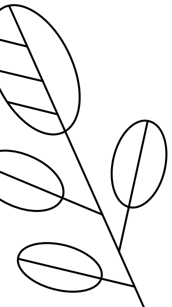
Geournal

Map-based journal app

ARI WILFORD, ARYANA MOHAMMADI,
HEZZY SEGAL, KEVIN LI





OUR TEAM

ARI

CS B.A., Class of ‘25

Firebase & backend integration

HEZZY

CS B.A., Class of ‘25

Map views, location tracking and updating

ARYANA

CS B.A., Class of '24

Home page, expanded map page, settings page, tab bar navigation, HealthKit integration

KEVIN

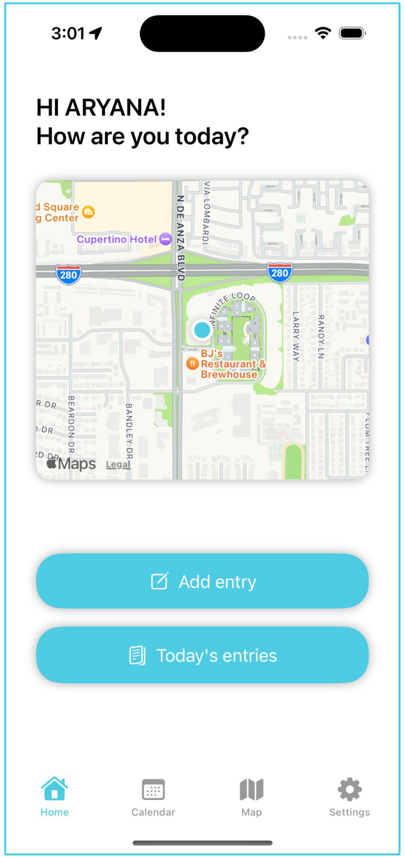
CS M.S., Class of ‘24

Project idea, UI design, logo design, Geournal entry interfaces (calendar, day, detailed entry, image and location pickers)

Geournal

Geournal is a geo-based diary app that enables users to create diary entries enriched with images tied to specific locations. Users can visually explore their memories on a map, offering a personalized and location-centric way to document and relive their experiences. Use cases include everyday use, traveling, and day trips.

ARYANA

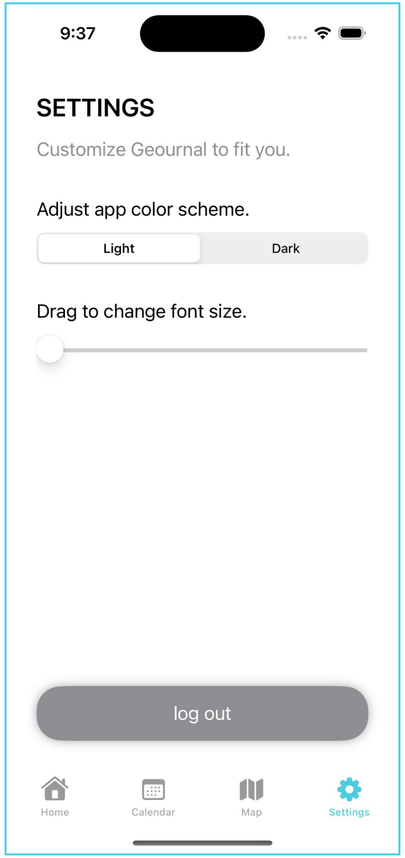

3:01

HI ARYANA!
How are you today?

Add entry

Today's entries

Home Calendar Map Settings


9:37

SETTINGS

Customize Geournal to fit you.

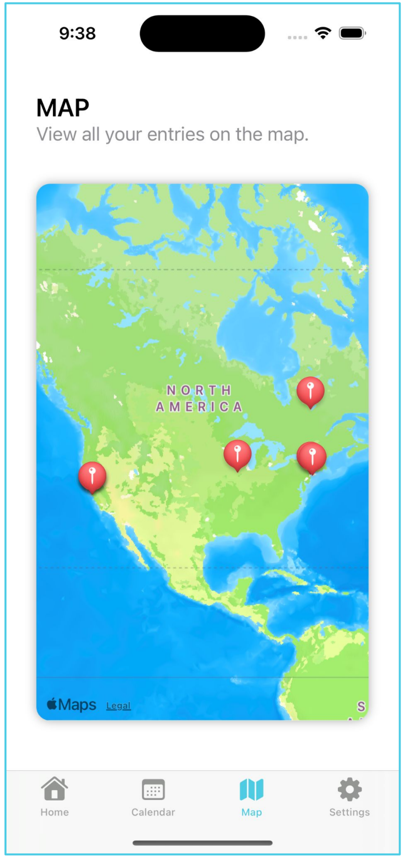
Adjust app color scheme.

[✓ Light] Dark

Drag to change font size.

log out

Home Calendar Map Settings

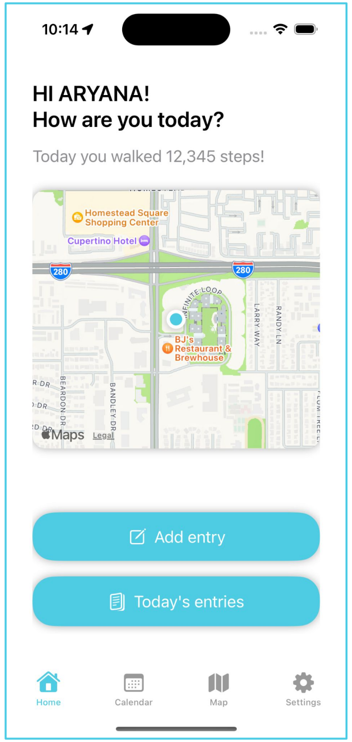

9:38

MAP

View all your entries on the map.

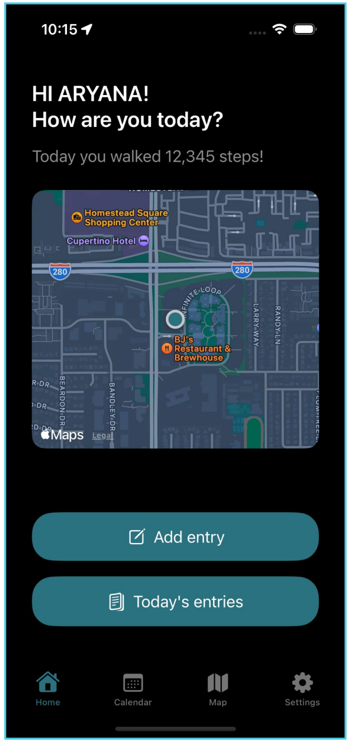
Home Calendar Map Settings

ARYANA


10:14

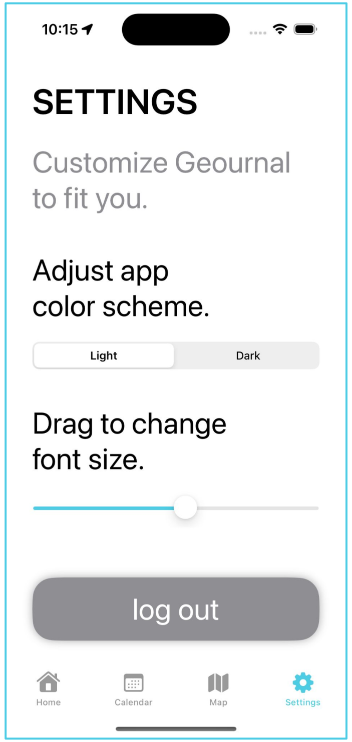
HI ARYANA!
How are you today?

Today you walked 12,345 steps!


10:15

HI ARYANA!
How are you today?

Today you walked 12,345 steps!


10:15

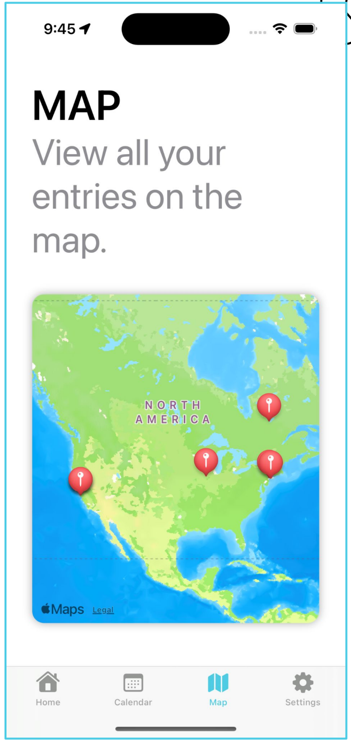
SETTINGS

Customize Geournal to fit you.

Adjust app color scheme.

[✓ Light] Dark

Drag to change font size.


9:45

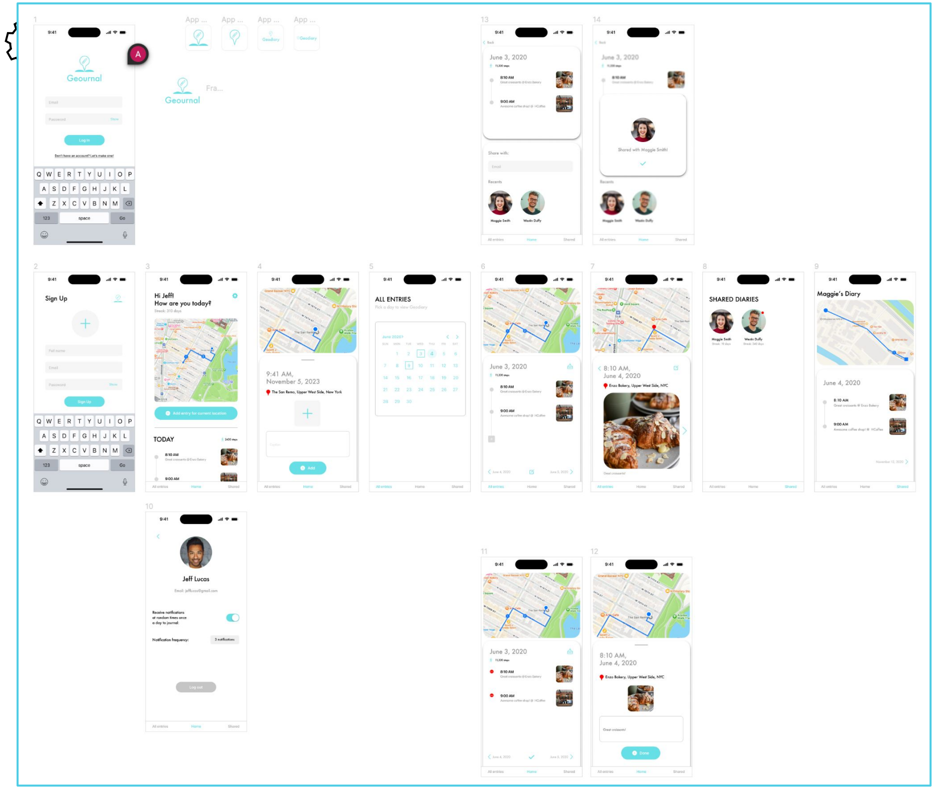
MAP

View all your entries on the map.

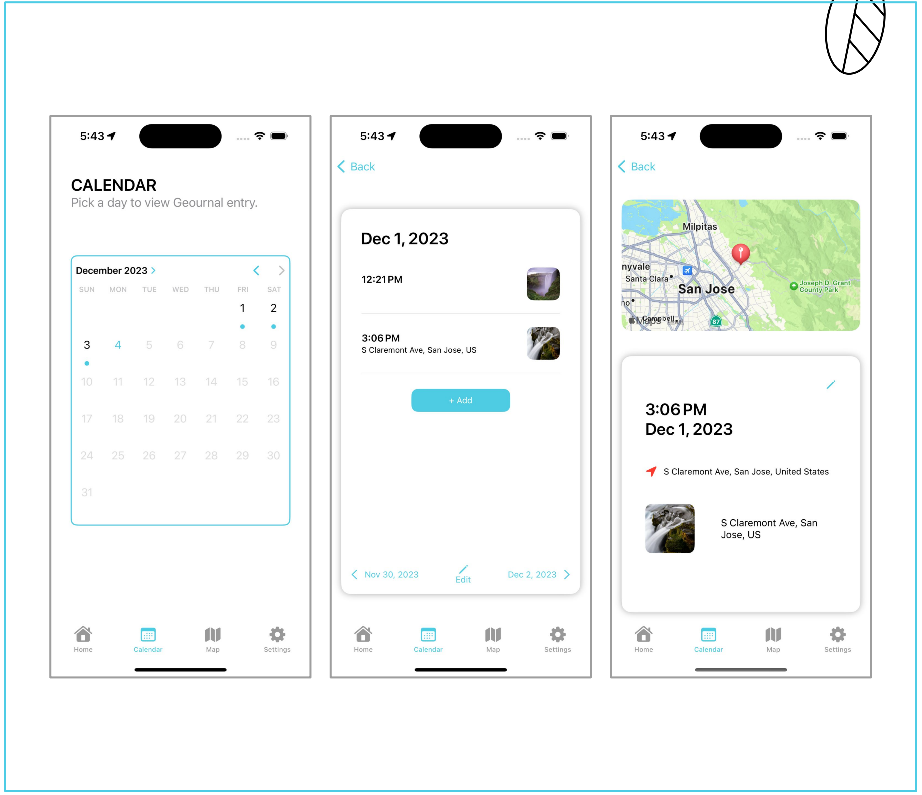
LIGHT MODE + DARK MODE

ADJUSTABLE FONT SIZE

KEVIN

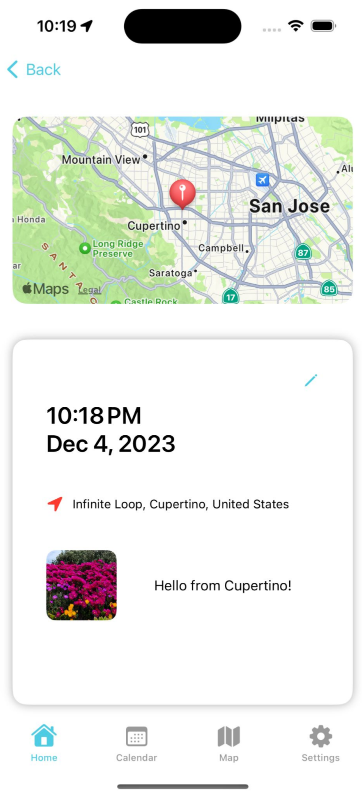


INITIAL WIREFRAMES + LOGO



GEOURNAL ENTRY VIEWS

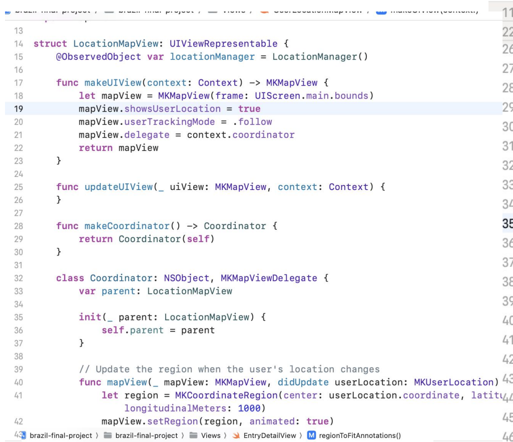
HEZZY



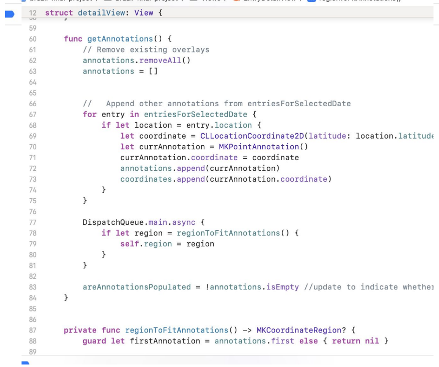
10:18 PM
Dec 4, 2023

Infinite Loop, Cupertino, United States

Hello from Cupertino!

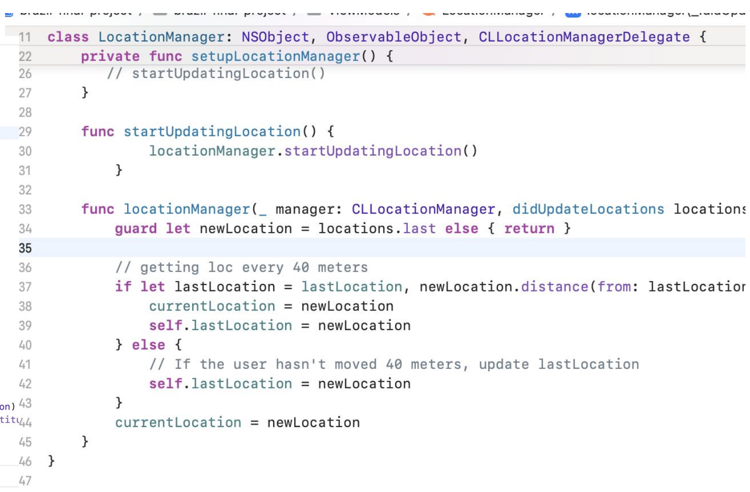


```swift
struct LocationMapView: UIViewRepresentable {
    @ObservedObject var locationManager = LocationManager()

    func makeUIView(context: Context) -> MKMapView {
        let mapView = MKMapView(frame: UIScreen.main.bounds)
        mapView.showsUserLocation = true
        mapView.userTrackingMode = .follow
        mapView.delegate = context.coordinator
        return mapView
    }

    func updateUIView(_ uiView: MKMapView, context: Context) {
    }

    func makeCoordinator() -> Coordinator {
        return Coordinator(self)
    }

    class Coordinator: NSObject, MKMapViewDelegate {
        var parent: LocationMapView

        init(_ parent: LocationMapView) {
            self.parent = parent
        }

        // Update the region when the user's location changes
        func mapView(_ mapView: MKMapView, didUpdate userLocation: MKUserLocation)
            let region = MKCoordinateRegion(center: userLocation.coordinate, latitu
                longitudinalMeters: 1000)
            mapView.setRegion(region, animated: true)
```



brazil-final-project 〉 brazil-final-project 〉 Views 〉 EntryDetailView 〉 regionToFitAnnotations()

```swift
struct detailView: View {

    func getAnnotations() {
        // Remove existing overlays
        annotations.removeAll()
        annotations = []


        //   Append other annotations from entriesForSelectedDate
        for entry in entriesForSelectedDate {
            if let location = entry.location {
                let coordinate = CLLocationCoordinate2D(latitude: location.latitude
                let currAnnotation = MKPointAnnotation()
                currAnnotation.coordinate = coordinate
                annotations.append(currAnnotation)
                coordinates.append(currAnnotation.coordinate)
            }
        }

        DispatchQueue.main.async {
            if let region = regionToFitAnnotations() {
                self.region = region
            }
        }

        areAnnotationsPopulated = !annotations.isEmpty //update to indicate whethe
    }


    private func regionToFitAnnotations() -> MKCoordinateRegion? {
        guard let firstAnnotation = annotations.first else { return nil }

```

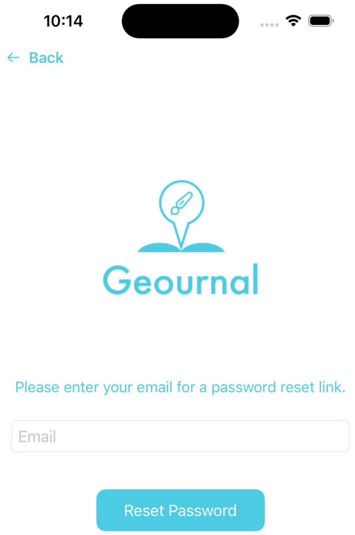


```swift
class LocationManager: NSObject, ObservableObject, CLLocationManagerDelegate {
    private func setupLocationManager() {
       // startUpdatingLocation()
    }

    func startUpdatingLocation() {
            locationManager.startUpdatingLocation()
        }

    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations
        guard let newLocation = locations.last else { return }

        // getting loc every 40 meters
        if let lastLocation = lastLocation, newLocation.distance(from: lastLocatio
            currentLocation = newLocation
            self.lastLocation = newLocation
        } else {
            // If the user hasn't moved 40 meters, update lastLocation
            self.lastLocation = newLocation
        }
        currentLocation = newLocation
    }
}
```

LOCATION UPDATES TRACK CURRENT LOCATION—CAN ALSO CONSTANTLY UPDATE USER LOCATION IN FUTURE

MAP RENDERING WITH ANNOTATION

ARI


10:14

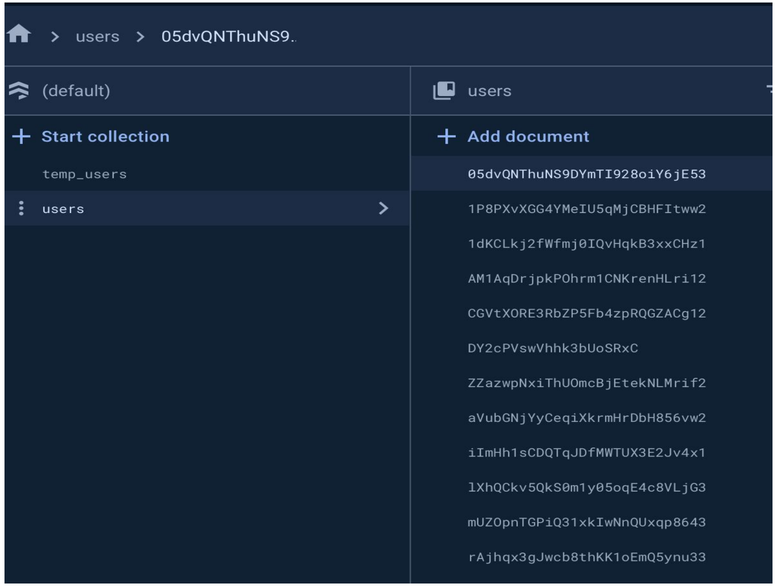
← Back

Geournal

Please enter your email for a password reset link.

Email

Reset Password


users > 05dvQNThuNS9..

(default)

Start collection

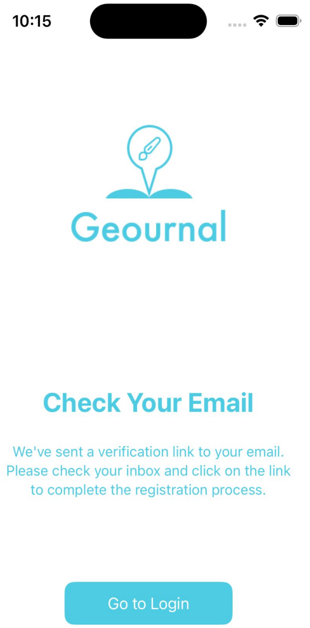
temp_users

users

users

Add document

```
05dvQNThuNS9DYmTI928oiY6jE53
1P8PXvXGG4YMeIU5qMjCBHFItww2
1dKCLkj2fWfmj0IQvHqkB3xxCHz1
AM1AqDrjpkPOhrm1CNKrenHLri12
CGVtXORE3RbZP5Fb4zpRQGZACg12
DY2cPVswVhhk3bUoSRxC
ZZazwpNxiThUOmcBjEtekNLMrif2
aVubGNjYyCeqiXkrmHrDbH856vw2
iImHh1sCDQTqJDfMWTUX3E2Jv4x1
lXhQCkv5QkS0m1y05oqE4c8VLjG3
mUZOpnTGPiQ31xkIwNnQUxqp8643
rAjhqx3gJwcb8thKK1oEmQ5ynu33
```


10:15

Geournal

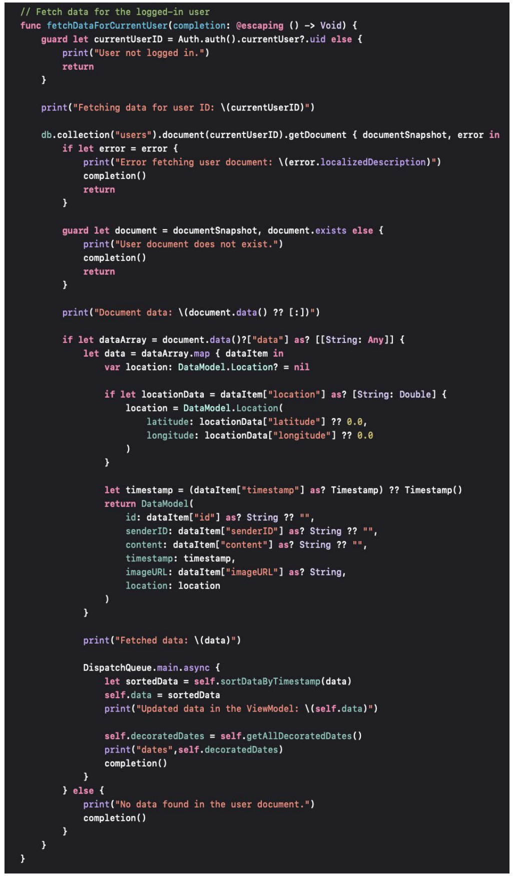
Check Your Email

We've sent a verification link to your email. Please check your inbox and click on the link to complete the registration process.

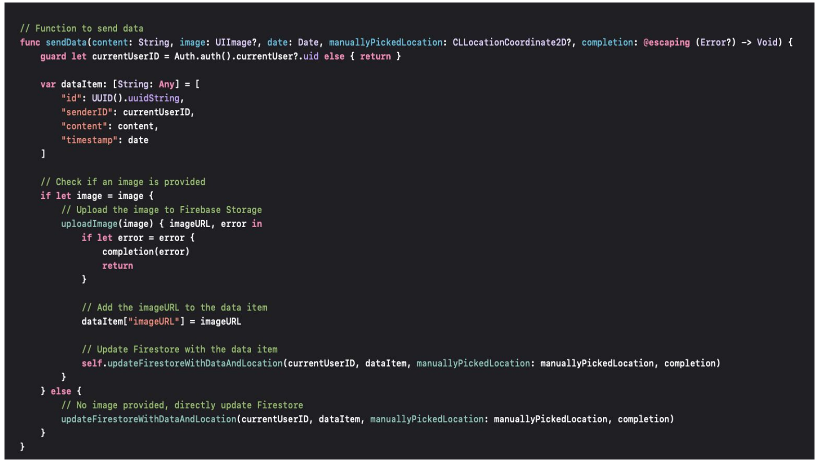
Go to Login

LOGIN REQUIRES USER VERIFICATION. USERS WHO SIGN UP ARE GRANTED A TEMPORARY DATA FOLDER ON FIREBASE UNTIL THEY VERIFY THEIR EMAIL ADDRESS. INCLUDED PASSWORD RESET AS WELL.

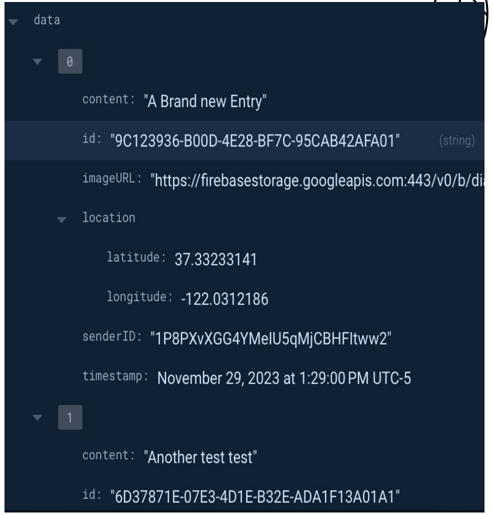
ARI



```swift
// Fetch data for the logged-in user
func fetchDataForCurrentUser(completion: @escaping () -> Void) {
    guard let currentUserID = Auth.auth().currentUser?.uid else {
        print("User not logged in.")
        return
    }

    print("Fetching data for user ID: \(currentUserID)")

    db.collection("users").document(currentUserID).getDocument { documentSnapshot, error in
        if let error = error {
            print("Error fetching user document: \(error.localizedDescription)")
            completion()
            return
        }

        guard let document = documentSnapshot, document.exists else {
            print("User document does not exist.")
            completion()
            return
        }

        print("Document data: \(document.data() ?? [:])")

        if let dataArray = document.data()?["data"] as? [[String: Any]] {
            let data = dataArray.map { dataItem in
                var location: DataModel.Location? = nil

                if let locationData = dataItem["location"] as? [String: Double] {
                    location = DataModel.Location(
                        latitude: locationData["latitude"] ?? 0.0,
                        longitude: locationData["longitude"] ?? 0.0
                    )
                }

                let timestamp = (dataItem["timestamp"] as? Timestamp) ?? Timestamp()
                return DataModel(
                    id: dataItem["id"] as? String ?? "",
                    senderID: dataItem["senderID"] as? String ?? "",
                    content: dataItem["content"] as? String ?? "",
                    timestamp: timestamp,
                    imageURL: dataItem["imageURL"] as? String,
                    location: location
                )
            }

            print("Fetched data: \(data)")

            DispatchQueue.main.async {
                let sortedData = self.sortDataByTimestamp(data)
                self.data = sortedData
                print("Updated data in the ViewModel: \(self.data)")

                self.decoratedDates = self.getAllDecoratedDates()
                print("dates",self.decoratedDates)
                completion()
            }
        } else {
            print("No data found in the user document.")
            completion()
        }
    }
}
```



```swift
// Function to send data
func sendData(content: String, image: UIImage?, date: Date, manuallyPickedLocation: CLLocationCoordinate2D?, completion: @escaping (Error?) -> Void) {
    guard let currentUserID = Auth.auth().currentUser?.uid else { return }

    var dataItem: [String: Any] = [
        "id": UUID().uuidString,
        "senderID": currentUserID,
        "content": content,
        "timestamp": date
    ]

    // Check if an image is provided
    if let image = image {
        // Upload the image to Firebase Storage
        uploadImage(image) { imageURL, error in
            if let error = error {
                completion(error)
                return
            }

            // Add the imageURL to the data item
            dataItem["imageURL"] = imageURL

            // Update Firestore with the data item
            self.updateFirestoreWithDataAndLocation(currentUserID, dataItem, manuallyPickedLocation: manuallyPickedLocation, completion)
        }
    } else {
        // No image provided, directly update Firestore
        updateFirestoreWithDataAndLocation(currentUserID, dataItem, manuallyPickedLocation: manuallyPickedLocation, completion)
    }
}
```



```
data
    0
        content: "A Brand new Entry"
        id: "9C123936-B00D-4E28-BF7C-95CAB42AFA01"    (string)
        imageURL: "https://firebasestorage.googleapis.com:443/v0/b/di
        location
            latitude: 37.33233141
            longitude: -122.0312186
        senderID: "1P8PXvXGG4YMeIU5qMjCBHFItww2"
        timestamp: November 29, 2023 at 1:29:00 PM UTC-5
    1
        content: "Another test test"
        id: "6D37871E-07E3-4D1E-B32E-ADA1F13A01A1"
```

FUNCTIONS LIKE THE SCREENSHOTS SHOWN WERE CREATED FOR BACKEND USE TO ALLOW DATA, INCLUDING IMAGES TO BE SENT, FETCHED, SORTED BY USER ENTRY DATE, EDITED, DELETED, AND MORE.

DATA IS STORED ON FIREBASE AS AN ARRAY INSIDE OF A USER'S INDIVIDUAL FOLDER WHICH CONTAINS THEIR ENTRY STRING, AS WELL AS ANY IMAGES, LOCATION INFORMATION AND THE TIME THE ENTRY WAS CREATED.

NEXT STEPS

ROUTES

Implement background location tracking to generate daily user routes

CUSTOMIZATION

Add additional customization features (e.g., font, accent color, etc.)

EFFICIENCY

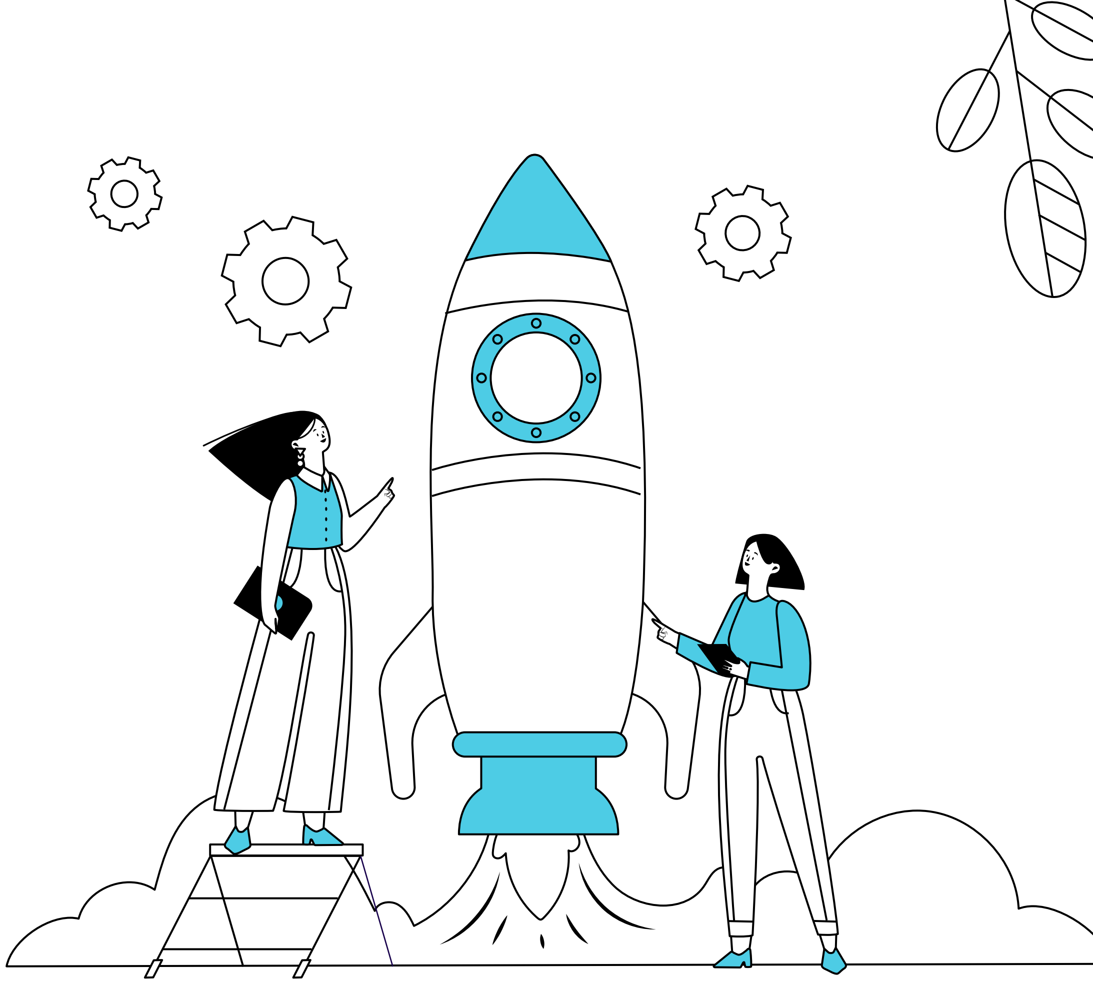
Increase efficiency of data fetching algorithms

ANNOTATIONS

Add additional interactions with map annotations on map views

THANKS!

Do you have any questions?



Ari Wilford
Aryana Mohammadi
Hezzy Segal
Kevin Li