You can go fast if you go well.
Anup Kumar Mishra Important Notes List:-
Password: ZVR8sMVTiApj77q
LIC Policy Number: - 206392038
StackOverFlow Id user:19680793

If any e-challan Generated On 27-Sep-2023 from 10.30AM to 11.30AM Roshan Bhai Bike for Shantipuram to Civil Lines then it is responsibility to me.

Code- 95407855

Sushil:-2500
// Amitabh Varma :- 3000
// Total = 2500+3000 = 5500
// Sandeep Sir = 4000 Balance

Video:- https://www.youtube.com/watch?v=EwLI3-S5zLo
https://storage.googleapis.com/pdffilesalib/pdf/coverpage/20220807191259_128.png

https://alibrary.page.link/?link=https://alibrary.in/api/student/stack-book-list?id=3887&apn=com.primeitzen.alibrary

Books Earnings API
https://www.alibrary.in/api/publisher/tot-book-sale?month=08&year=2022

Teacher List View Link for web View:

https://www.alibrary.in/student/view-teacher-profile?id=1835&email=amit697598&password=password

https://storage.googleapis.com/pdffilesalib/pdf/protectedpdf/Bountiful_bonsai_20220807192040_128/1.pdf

Using for Searchable functionality:——

ForEach(list.datas.filter {
searchText.isEmpty ? true :
$0.book_detail.title.localizedCaseInsensitiveContains(searchTex

```
t)
}, id: \.id){
item in
Text(item.name).foregroundColor(.gray)
}
```

How to open the non secure url in the custom View in SwiftUI?

API Of Book Request List
https://www.scribd.com/document/562947245/final-new-Hospital-Mangement-System-by-Yash-Bhimanai#

@Anup (Jansatta dal)
https://alibrary.in/api/guest/view-bookrequest?id=&bookSearch="+"&page="+page for other
https://alibrary.in/api/guest/show-linkbookrequest?id=&page= forself

ManuSmriti Book Id:- 296

https://alibrary.in/api/guest/rc-history

Location:
24.212368919064055, 83.03536002659129

Pagination with Filter data:-✺

✖🥰😘

✓

```
struct LatestMagazineView: View {
   @Environment(\.dismiss) var dismiss🐥🐦‍⬛
   let columns = [GridItem(.flexible()),GridItem(.flexible())]
   @StateObject var list = PublicProfilePublisherViewModel()
   @State private var isLoading: Bool = false
   let options = ["All", "Prime", "Free", "Paid"]
   @State var value = ""
   var placeholder = "All"
   @State var selectedIndex = 0
   @State var id: Int
   @FocusState private var isTextFieldFocused: Bool
   @State  var searchText: String = ""
   var body: some View {
      ZStack{
```

```
Image("u")
  .resizable()
  .ignoresSafeArea()
VStack(spacing:0){

  HStack(spacing: 25){
    Button(action: {
      dismiss()
    },
        label: {

      Image(systemName: "arrow.backward")
        .font(.system(size:22, weight:.heavy))
        .foregroundColor(.white)
    })
    Text(list.full_name)
      .font(.system(size: 24, weight: .regular))
      .foregroundColor(.white)
    Spacer()

    AsyncImage(url: URL(string: list.logo)){ img in
      img.resizable()
        .frame(width: 55, height: 55)
        .cornerRadius(28)
    }placeholder: {
      Image("logo_gray").resizable()
        .frame(width: 55, height: 55)
        .cornerRadius(28)
    }


  }.padding(.horizontal)
    .frame(width: UIScreen.main.bounds.width, height: 90)
    .background(Color("orange"))

  ScrollView{
    VStack(spacing: 0){

      AsyncImage(url: URL(string: list.banner)){img in
        img.resizable()
          .frame(height: 225)

      }placeholder: {
        Color.gray
          .frame(height: 225)
      }
      HStack(spacing: 8){
```

```
Image("pdf_white")
   .resizable()
   .frame(width: 25, height: 28)

Text("\(list.total)")
   .foregroundColor(.white)
   .font(.system(size: 24))
Button(action: {

}, label: {
   Text("Follow")
      .foregroundColor(.white)
      .font(.system(size: 24))
      .padding(6)
      .padding(.horizontal,8)
      .overlay(
      RoundedRectangle(cornerRadius: 20)
         .stroke(.white, lineWidth: 1)
   )

})
Image("followers_white")
   .resizable()
   .frame(width: 13, height: 22)
Text("\(list.followers)")
   .foregroundColor(.white)
   .font(.system(size: 24))
Spacer()


Menu {
   ForEach(options.indices, id: \.self) { index in
      Button(options[index]) {
         self.value = options[index]
         self.selectedIndex = index
         list.bookType = ["all", "prime", "free", "paid"][index]
         list.filterData = true
         list.currentPage = 1
         list.getBookData()
      }
   }
} label: {
   Text("\(options[selectedIndex]) \(list.total)")
      .foregroundColor(.white)
      .font(.system(size: 24))
}
```

```
            }.padding(.horizontal).frame(height:
65).background(Color("default_"))
          LazyVGrid(columns: columns, spacing: 10, pinnedViews:
[.sectionHeaders]){
                Section(header:
                    VStack{
                  TextField("", text:
$searchText).font(.title).accentColor(.clear)
                    .padding(4).padding(.trailing,29)
                    .padding(.leading,searchText.isEmpty ? 29 : 4)
                    .foregroundColor(.white)
                    .cornerRadius(8).focused($isTextFieldFocused)
                    .showClearButtonOfSearchField($searchText)
                    .overlay(
                      RoundedRectangle(cornerRadius: 22)
                        .stroke(Color.white, lineWidth: 1)
                    ).overlay(alignment: .leading, content: {
                      if searchText.isEmpty {

Image("magnifying_glass_left" ).resizable().frame(width: 20, height:
20).padding(.leading,8)
                      }
                    })
                    .overlay(alignment: .leading, content: {
                      Text("Search books..").padding(.leading,32)
                        .foregroundColor(.white)
                        .font(.title)
                        .opacity(searchText.isEmpty ? 1 : 0)
                    } ).padding(0)
                }.padding(.horizontal)
                  .frame(height: 65).background(Color("default_")))
                {
                  ForEach(list.datas, id: \.id){item in
                    VStack(alignment: .leading){
                      AsyncImage(url: URL(string: item.book_media.url))
{ img in
                        img .resizable()
                          .frame(height: 255)
                      }placeholder: {
                        Image("logo_gray")
                          .resizable()
                          .frame(height: 255)
                      }

                      Text(item.title).lineLimit(2).font(.system(size:
22,weight: .regular))
```

```
                        Text("Published: \(item.published)")
                           .font(.system(size: 17,weight: .regular))
                        Text("Like \("\(item.tot_likes ?? "0")")")
                           .font(.system(size: 18,weight: .regular))
                        HStack{
                           Spacer()
                           if item.is_free == 1
                           {
                              Text("free").font(.system(size:
22,weight: .bold)).foregroundColor(.red)
                           } else{
                              Text("paid").font(.system(size:
22,weight: .bold)).foregroundColor(.cyan)
                           }

                        }
                     }.padding(8)
                        .background(Color.white)
                        .cornerRadius(4).shadow(radius: 1)

                  }.padding(2)
         if list.currentPage < list.totalPage{

            CircleProgressView()

                        .frame(alignment: .center)

                        .onAppear{

                           list.currentPage = list.currentPage + 1
                           list.getBookData()

 } } } } }}
}.onAppear{
           list.id = self.id
           list.getBookData()
        }
     }
   }
}

struct LatestMagazineView_Previews: PreviewProvider {
   static var previews: some View {
//       LatestMagazineView()
      LatestMagazineView(id: 11)
```

```
    }
}



class PublicProfilePublisherViewModel: ObservableObject{

    @Published var datas = [PublicProfilePublisherDatum]()
    @Published var banner = String()
    @Published var logo = String()
    @Published var followers = Int()
    @Published var total = Int()
    @Published var full_name = String()
    @Published var currentPage = 0
    @Published var totalPage = Int()
    @Published var id = Int()
    var filterData = false
    var bookType = "all"

    func getBookData() {
        let defaults = UserDefaults.standard
        guard let token = defaults.string(forKey: "access_token") else {
            return
        }

        AuthenticationPublicProfilePublisherService().getBookData(token:
token, currentPage: currentPage, type: bookType, id: id) { (result) in
            switch result {
            case .success(let results):
                DispatchQueue.main.async {
                    self.totalPage = results.books.last_page ?? 1
                    self.followers = results.followers ?? 0
                    self.full_name = results.userDetails.full_name
                    self.total = results.books.total ?? 0

                    if self.filterData {
                      self.datas.removeAll()
                    }

                    self.currentPage += 1
                    self.datas.append(contentsOf: results.books.data)

                    self.filterData = false
                }
            case .failure(let error):
                print(error.localizedDescription)
            }
```

```
        }
    }

    func loadMoreContentIfNeeded(currentItem item:
PublicProfilePublisherDatum?) {
        guard let item = item else {
            getBookData()
            return
        }
        let thresholdIndex = datas.index(datas.endIndex, offsetBy: -5)
        if datas.firstIndex(where: { $0.id == item.id }) == thresholdIndex {
            getBookData()
        }
    }

}
```

```
class AuthenticationPublicProfilePublisherService {
    func getBookData(token: String,currentPage:Int,type: String,id: Int,
completion: @escaping (Result<PublicProfilePublisherModel, NetworkError>)
-> Void) {

//
        guard let url = URL(string: "\(APILoginUtility.publicProfileApi)id=\
(id)&bookSearch=&page=\(currentPage)&type=\(type)") else {
            completion(.failure(.invalidURL))
            return
        }

        var request = URLRequest(url: url)
        request.addValue("Bearer \(token)", forHTTPHeaderField: "Authorization")

        URLSession.shared.dataTask(with: request) { (data, response, error) in

            guard let data = data, error == nil else {
                completion(.failure(.noData))
                return
            }

            guard let response = try?
JSONDecoder().decode(PublicProfilePublisherModel.self, from: data) else {
```

```
                completion(.failure(.decodingError))
                return
            }

            completion(.success(response))

        }.resume()

        }

}


struct SearchFieldClearButton: ViewModifier {
    @Binding var fieldText: String

    func body(content: Content) -> some View {
        content
            .overlay{
                if !fieldText.isEmpty {
                    HStack {
                        Spacer()
                        Button {
                            fieldText = ""
                        } label: {
                            Image(systemName: "multiply")
                                .foregroundColor(.white)
                                .font(.title)
                                .padding(.trailing, 4)
                        }
                        .foregroundColor(.white)
                        .padding(.trailing, 4)
                    }
                }
            }
    }
}



import Foundation

// MARK: - PublicProfilePublisher
public struct PublicProfilePublisherModel :Decodable{
    public let userDetails: PublicProfilePublisherDetails
    public let books: PublicProfilePublisherBooks
//    public let stackDetails: [Any?]
```

```
    public let uploadType: [PublicProfilePublisherUploadType]
//    public let teachCount: String
//    public let stuCount: String
    public let followers: Int?  //Followers
//    public let loginUser: PublicProfilePublisherLoginUser
////    public let bookSearch: NSNull
//    public let type: String
//    public let data: Int

}

// MARK: - PublicProfilePublisherBooks
public struct PublicProfilePublisherBooks:Decodable {
    public let current_page: Int
    public let data: [PublicProfilePublisherDatum]
//    public let first_page_url: String
//    public let from: Int
    public let last_page: Int?
//    public let last_page_url: String
//    public let next_page_url: String
//    public let path: String
//    public let per_page: Int
//    public let prev_page_url: String?
//    public let to: Int
    public let total: Int?


}

// MARK: - PublicProfilePublisherDatum
public struct PublicProfilePublisherDatum:Decodable {
    public let id: Int
//    public let upload_type_id: Int
//    public let name: String
//    public let pdf_url: String
//    public let html_url: String
//    public let tot_pages: Int
//
    public let title: String
//    public let long_desc: String
//    public let meta_keyword: String
//    public let author_name: String
//    public let isbn_no: String
    public let is_free: Int?
    public let tot_likes: String?
    public let tot_view: Int?
    public let publish_date: String
```

```
    public let validity_date: String
    public let published: String
    public let book_media: PublicProfilePublisherBookMedia
//    public let book_partner_link: PublicProfilePublisherBookPartnerLink

}

// MARK: - PublicProfilePublisherBookMedia
public struct PublicProfilePublisherBookMedia:Decodable {
    public let id: Int
    public let url: String
    public let created_by: Int
}

// MARK: - PublicProfilePublisherBookPartnerLink
public struct PublicProfilePublisherBookPartnerLink:Decodable {
    public let id: Int
    public let partner_id: Int
    public let book_id: Int

}

// MARK: - PublicProfilePublisherLoginUser
public struct PublicProfilePublisherLoginUser:Decodable {
    public let id: Int
    public let partner_unique_id: String
    public let user_id: Int
    public let first_name: String
    public let last_name: String
    public let full_name: String
    public let dob: String
    public let qr_code_url: String
    public let referal_code: String
    public let partner_role: PublicProfilePublisherPartnerRole

}

// MARK: - PublicProfilePublisherPartnerRole
public struct PublicProfilePublisherPartnerRole:Decodable {
    public let id: Int
    public let name: String
    public let guard_name: String

}

// MARK: - PublicProfilePublisherUploadType
public struct PublicProfilePublisherUploadType:Decodable {
```

```
    public let id: Int
    public let name: String
    public let description: String

}

// MARK: - PublicProfilePublisherDetails
public struct PublicProfilePublisherDetails:Decodable {
    public let id: Int
    public let partner_unique_id: String
    public let full_name: String
    public let qr_code_url: String
    public let partner_media: [PublicProfilePublisherPartnerMedia]
    public let address_link: PublicProfilePublisherAddressLink
    public let partner_role: PublicProfilePublisherPartnerRole
    public let user_data: PublicProfilePublisherUserData
//    public let partnerFollow: NSNull

}

// MARK: - PublicProfilePublisherAddressLink
public struct PublicProfilePublisherAddressLink:Decodable {
    public let id: Int
    public let partner_id: Int
    public let address_id: Int
//    public let createdBy: Int
    public let address: PublicProfilePublisherAddress

}

// MARK: - PublicProfilePublisherAddress
public struct PublicProfilePublisherAddress:Decodable {
    public let id: Int
    public let mobile_1: String
    public let email_id: String

}

// MARK: - PublicProfilePublisherPartnerMedia
public struct PublicProfilePublisherPartnerMedia:Decodable {
    public let id: Int
    public let partner_media_type_id: Int
    public let media_type_id: Int
    public let partner_id: Int
    public let url: String
    public let partnerMediatype: String
    public let partner_media_type: PublicProfilePublisherPartnerMediaType
```

```
}

// MARK: - PublicProfilePublisherPartnerMediaType
public struct PublicProfilePublisherPartnerMediaType:Decodable {
    public let id: Int
    public let type: String
    public let description: String
    public let is_active: Int

}

// MARK: - PublicProfilePublisherUserData
public struct PublicProfilePublisherUserData:Decodable {
    public let id: Int
    public let name: String
    public let username: String
    public let email: String

}
```

//End Pagination and filter data

```
Row = 11;
 1. space = row - 1;
 2. for (j = 1; j<= row; j++)
 3. {
 4. for (i = 1; i<= space; i++)
 5. {
 6. System.out.print(" ");
 7. }
 8. space--;
 9. for (i = 1; i <= 2 * j - 1; i++)
10. {
11. System.out.print("*");
12. }
13. System.out.println(" ");
14. }
15. space = 1;
16. for (j = 1; j<= row - 1; j++)
17. {
```

```
18. for (i = 1; i<= space; i++)
19. {
20. System.out.print(" ");
21. }
22. space++;
23. for (i = 1; i<= 2 * (row - j) - 1; i++)
24. {
25. System.out.print("*");
26. }
27. System.out.println("");
28. }
29. }
```

import SwiftUI	
	struct ContentView: View {
	@State private var index = 0
	var body: some View {
	VStack{
	TabView(selection: $index) {
	ForEach((0..<3), id: \.self) { index in
	CardView()
	}
	}
	.tabViewStyle(PageTabViewStyle(indexDisplayMode: .never))
	HStack(spacing: 2) {
	ForEach((0..<3), id: \.self) { index in
	Circle()
	.fill(index == self.index ? Color.purple : Color.purple.opacity(0.5))
	.frame(width: 20, height: 20)
	}
	}
	.padding()

	HStack(spacing: 2) {
	ForEach((0..<3), id: \.self) { index in
	Rectangle()
	.fill(index == self.index ? Color.purple : Color.purple.opacity(0.5))
	.frame(width: 20, height: 20)
	}
	}
	.padding()
	HStack(spacing: 2) {
	ForEach((0..<3), id: \.self) { index in
	Rectangle()
	.fill(index == self.index ? Color.purple : Color.purple.opacity(0.5))
	.frame(width: 30, height: 5)
	}
	}
	.padding()
	HStack(spacing: 2) {
	ForEach((0..<3), id: \.self) { index in
	Rectangle()
	.fill(index == self.index ? Color.purple : Color.purple.opacity(0.5))
	.frame(height: 5)
	}
	}
	.padding()
	}
	.frame(height: 200)

	}
	}
	struct CardView: View{
	var body: some View{
	Rectangle()
	.fill(Color.pink)
	.frame(height: 200)
	.border(Color.black)
	.padding()
	}
	}

```
struct APILoginUtility
{
   static let baseURL: String = "https://www.alibrary.in/api/"

   static let guestBookRequestApi: String = baseURL +   "guest/user-book-
requests?request_type="

  //Upload List
   static let guestUploadListDashboardApi: String? = "https://alibrary.in/api/
guest/ownpdfs?page="
   static let guestUploadListDraftApi: String? = baseURL + "guest/draft-pdf?
page="
   static let guestUploadListDeleteApi: String? = baseURL + "guest/unpublish-
pdf?is_publish=4&page="
   static let guestUnPublishedApi: String = baseURL + "guest/unpublish-pdf?
is_publish="
   static let guestPublishedApi: String = baseURL + "guest/publish-pdf"
   static let guestReportedBookListApi: String = baseURL + "guest/book-
reports"
   static let preViewBooksApi: String = baseURL + "previewbook"  //https://
alibrary.in/api/previewbook
   static let publicProfileApi: String = baseURL + "student/publicProfile?id=17"
   static let studentLibraryApi: String = baseURL + "student/select-library"
   static let guestLibraryApi: String = baseURL + "guest/selectLibrary"
}

enum APIEndpoint {
```

```
    static let baseURL: String = "https://www.alibrary.in/api/"

    enum Guest {

        static let uploadListDashboard: String = baseURL + "guest/ownpdfs?
page="


        static let uploadListDraft: String = baseURL + "guest/draft-pdf?page="

         static let uploadListDelete: String = baseURL + "guest/unpublish-pdf?
is_publish=4&page="

        static let unpublish: String = baseURL + "guest/unpublish-pdf?
is_publish="

        static let publish: String = baseURL + "guest/publish-pdf"

        static let reportedBookList: String = baseURL + "guest/book-reports"

        static let previewBook: String = baseURL + "previewbook"

        static let selectLibrary: String = baseURL + "guest/selectLibrary"
    }

}
```

```
"userslist": [

{
"id": 5,

"totalBooks": 264,
"totalBookViews": "57.472 k",
"totalfollowers": 4,
"full_address": "ANANTAPUR",
"partner_media": [

{
"id": 5,
```

```
"url": "https://alibrary.in/storage/images/partner/logo/
Gyan_logo_181220210443_5.png"
}
]
}
]


// This file was generated from JSON Schema using quicktype, do not modify it
directly.
// To parse the JSON, add this file to your project and do:
//
//   let welcome = try? newJSONDecoder().decode(Welcome.self, from:
jsonData)

import Foundation

// MARK: - Welcome
public struct Welcome {
    public var userslist: [Userslist]

    public init(userslist: [Userslist]) {
        self.userslist = userslist
    }
}

// MARK: - Userslist
public struct Userslist {
    public var id: Int
    public var totalBooks: Int
    public var totalBookViews: String
    public var totalfollowers: Int
    public var fullAddress: String
    public var partnerMedia: [PartnerMedia]
    public var addressLink: AddressLink

    public init(id: Int, totalBooks: Int, totalBookViews: String, totalfollowers: Int,
fullAddress: String, partnerMedia: [PartnerMedia], addressLink: AddressLink) {
        self.id = id
        self.totalBooks = totalBooks
        self.totalBookViews = totalBookViews
        self.totalfollowers = totalfollowers
        self.fullAddress = fullAddress
        self.partnerMedia = partnerMedia
        self.addressLink = addressLink
    }
}
```

```
// MARK: - AddressLink
public struct AddressLink {
    public var id: Int
    public var address: Address

    public init(id: Int, address: Address) {
        self.id = id
        self.address = address
    }
}

// MARK: - Address
public struct Address {
    public var id: Int
    public var countryid: Int
    public var stateid: Int
    public var cityid: Int
    public var postalCode: Int
    public var cityDetails: CityDetails

    public init(id: Int, countryid: Int, stateid: Int, cityid: Int, postalCode: Int,
cityDetails: CityDetails) {
        self.id = id
        self.countryid = countryid
        self.stateid = stateid
        self.cityid = cityid
        self.postalCode = postalCode
        self.cityDetails = cityDetails
    }
}

// MARK: - CityDetails
public struct CityDetails {
    public var id: Int
    public var city: String

    public init(id: Int, city: String) {
        self.id = id
        self.city = city
    }
}

// MARK: - PartnerMedia
public struct PartnerMedia {
    public var id: Int
    public var partnerMediaTypeid: Int
```

```
    public var mediaTypeid: Int
    public var partnerid: Int
    public var url: String

    public init(id: Int, partnerMediaTypeid: Int, mediaTypeid: Int, partnerid: Int,
url: String) {
        self.id = id
        self.partnerMediaTypeid = partnerMediaTypeid
        self.mediaTypeid = mediaTypeid
        self.partnerid = partnerid
        self.url = url
    }
}




struct Menu: Decodable {
    let catagory: String
    let items: [Recipe]
}

struct Recipe: Decodable, Identifiable {
    var id: String { name }
    let name: String
    let totalCalories: Int
    let isFavourite: Bool
    let addedToday: Int
    let components: [RecipeComponent]
}

struct RecipeComponent: Decodable {
    let name: String
    let varietyName: String
    let varietyCalories: Int
    let totalCalories: Int
    let quantity: Int
    let unit: String
}

struct RecipeSumView: View {
    @State private var recipes: [Recipe] = []
    var totalCalories: Int {
        return recipes.reduce(0) { cals, currentItem in
            cals + currentItem.totalCalories
        }
    }
```

```
    var body: some View {
        VStack(spacing: 40) {
            Text("Total Calories = \(totalCalories)")
            VStack(spacing: 20) {
                ForEach(recipes) { recipe in
                    Text(recipe.name).font(.headline)
                    Text(String(recipe.totalCalories))
                }
            }
        }
        .onAppear { loadJSON() }
    }

    func loadJSON() {
        let decoder = JSONDecoder()
        do {
            let menu = try decoder.decode([Menu].self, from: menuJSON)
            recipes = menu[0].items
        } catch {
            print(error)
        }
    }
}



UIApplication
                .shared
                .connectedScenes
                .flatMap { ($0 as? UIWindowScene)?.windows ?? [] }
                .first { $0.isKeyWindow }?.safeAreaInsets.top

//
 var body: some View{

        ZStack {
        VStack {

              VStack {
                 Text(datas).foregroundColor(.white).font(.largeTitle).bold()
                 Text(subdatas).foregroundColor(.white).bold()
                 Text(descBy).foregroundColor(.white)


 // Hstack
```

```
                //End

            }.frame(width: UIScreen.main.bounds.width, height: 162)
                .background(Color.black.opacity(0.2))

        Spacer()

        ScrollView(.horizontal,showsIndicators: false){
            HStack{

                ForEach(imgdatas, id: \.id){ item in
                    AsyncImage(url: URL(string: item.book_media.url)){image in
                        image.resizable().frame(width: 125,height:
160).cornerRadius(7)

                    }placeholder: {
                        Image("logo_gray").resizable().frame(width: 125,height:
160).cornerRadius(7).padding(.horizontal)
                    }
                }



            }
        }
    }

        HStack{
                        Spacer()
            Image(systemName:
"ellipsis").foregroundColor(.white).font(.system(size: 30)).frame(width: 42,
height: 42)
                                .background(Color.gray).cornerRadius(22).opacity(0.
9)
                                .rotationEffect(Angle(degrees: isTapped ? 90 :
0)).padding(.top, -85)
                                .onTapGesture {
                                    isTapped.toggle()
                                        print("Tapped")
                                    }
                                .overlay (

                                    VStack(alignment: .center, spacing: 7) {

                                        if isTapped {
```

```
                                    Spacer(minLength: 1)

                                    Image("whatup_").resizable().frame(width:
40, height: 40).opacity(0.9)
                                    Image("insta_").resizable().frame(width: 40,
height: 40).opacity(0.9)
                                    Image("twiter_").resizable().frame(width: 40,
height: 40).opacity(0.9)
                                    Image("insta_").resizable().frame(width: 40,
height: 40).opacity(0.9)
                                    Image("youtube_").resizable().frame(width:
40, height: 40).opacity(0.9)

                                }

                            }.padding(.top, -44).padding(.trailing),
alignment: .topLeading

                        )
        //
                    }.padding(.trailing)




    }.background(AsyncImage(url: URL(string:  "\(bannerdatas)")){
                    image in
                    image.resizable()
                }placeholder: {
                    Color.orange.opacity(0.5)
            }).frame(height:435)
        .onAppear{
            getData(url: apiurl+"\(id)")
        }
//Pagination In SwiftUI




//    func getDataFromServer(){
//        guard let url = URL(string: "https://www.alibrary.in/api/ros-
explore_categories/1?") else{
//            return
//        }
//
//        self.datas = self.pages.map {
```

```
//          PageDatum(id: $0.categoryBooks.last_page, url:
$0.categoryBooks.next_page_url)
//           }
//
//
//
//
//       let params = "page=\(self.currentPage)"
//       var request = URLRequest(url: url)
//       request.httpMethod = "POST"
//       request.httpBody = params.data(using: .utf8)
//
//       let task = URLSession.shared.dataTask(with: request) { (data, _, error) in
//          if error == nil{
//             if data != nil{
//                do{
//                   print(String(data: data!, encoding: .utf8)!)
//                   let results = try? JSONDecoder().decode(PageBooks.self, from:
data!)
//                   DispatchQueue.main.async {
//                      self.totalPage = results?.categoryBooks.last_page ?? 1
//
//
//
//                      self.datas.append(contentsOf:
(results?.categoryBooks.data)!)
//                   }
//                }
//                catch {
//                   print(error)
//                }
//             }else{
//                print(error?.localizedDescription ?? "N/A")
//             }
//          }
//
//       }
//       task.resume()
//    }

//}
```

आदतें आत्म सुधार के यौगिक हित हैं। जिस तरह से चक्रवृद्धि ब्याज से पैसा बढ़ता है, उसी तरह आपकी आदतों का प्रभाव उन्हें दोहराते ही कई गुना बढ़ जाता है। ऐसा लगता है कि वे किसी भी दिन बहुत कम अंतर लाते हैं और फिर भी वे महीनों और वर्षों में जो प्रभाव डालते हैं वह बहुत बड़ा हो सकता है। दो, पांच या

शायद दस साल बाद जब पीछे मुड़कर देखा जाए तो अच्छी आदतों की कीमत और बुरी आदतों की कीमत स्पष्ट रूप से स्पष्ट हो जाती है।

```
struct HeaderViewEx: View {
   @State var datas = String()
   @State var descBy = String()
   @State var subdatas = String()
   @State var bannerdatas = String()
   @State var allbtn = String()
   @State var imgdatas = [ExploreBookDetail]()
   @State var btns = [Subcategorycol]()
   @State var id = 9
   let apiurl = "https://alibrary.in/api/explore_categories/"
   @State var isTapped: Bool = false
   var body: some View{

      ZStack {
      VStack {

            VStack {
               Text(datas).foregroundColor(.white).font(.largeTitle).bold()
               Text(subdatas).foregroundColor(.white).bold()
               Text(descBy).foregroundColor(.white)


 // Hstack

               //End

            }.frame(width: UIScreen.main.bounds.width, height: 162)
               .background(Color.black.opacity(0.2))

         Spacer()

         ScrollView(.horizontal,showsIndicators: false){
            HStack{

               ForEach(imgdatas, id: \.id){ item in
                  AsyncImage(url: URL(string: item.book_media.url)){image in
                     image.resizable().frame(width: 125,height:
160).cornerRadius(7)

                  }placeholder: {
```

```
                    Image("logo_gray").resizable().frame(width: 125,height:
160).cornerRadius(7).padding(.horizontal)
                }
            }




        }
      }
    }

      HStack{
                    Spacer()
        Image(systemName:
"ellipsis").foregroundColor(.white).font(.system(size: 30)).frame(width: 42,
height: 42)
                        .background(Color.gray).cornerRadius(22).opacity(0.
9)
                        .rotationEffect(Angle(degrees: isTapped ? 90 :
0)).padding(.top, -85)
                        .onTapGesture {
                          isTapped.toggle()
                              print("Tapped")
                          }
                        .overlay (

                          VStack(alignment: .center, spacing: 7) {

                            if isTapped {

                              Spacer(minLength: 1)
                              Link("\(Image("insta_"))", destination:
URL(string: "fb://facebook.com/FacebookPageName")!)
                              Image("whatup_").resizable().frame(width:
40, height: 40).opacity(0.9)
                              Image("insta_").resizable().frame(width: 40,
height: 40).opacity(0.9)
                              Image("twiter_").resizable().frame(width: 40,
height: 40).opacity(0.9)
                              Image("insta_").resizable().frame(width: 40,
height: 40).opacity(0.9)
                              Image("youtube_").resizable().frame(width:
40, height: 40).opacity(0.9)

                            }

                          }.padding(.top, -44).padding(.trailing),
```

```
alignment: .topLeading

                                    )
                                }.padding(.trailing)


        }.background(AsyncImage(url: URL(string:  "\(bannerdatas)")){
                            image in
                            image.resizable()
                        }placeholder: {
                            Color.orange.opacity(0.5)
                    }).frame(height:435)
            .onAppear{
                getData(url: apiurl+"\(id)")
            }

    }
    func getData(url: String) {
        guard let url = URL(string: url) else { return }
        URLSession.shared.dataTask(with: url) { (data, _, _) in
            if let data = data {
                do {
                    let results = try JSONDecoder().decode(ExploreModel.self, from:
data)
                    DispatchQueue.main.async {
                        self.datas = results.bookcategory.category_name
                        self.subdatas = results.bookcategory.description
                        self.descBy = results.bookcategory.desc_by
                        self.bannerdatas = results.bookcategory.banner
                        self.imgdatas = results.bookDetails
                        self.btns = results.subcategorycol
                        self.allbtn = results.subCategoryname
                    }
                }
                catch {
                    print(error)
                }
            }
        }.resume()
    }



    }
```

When the Button click then change the foreground text colour.

```
struct ContentView: View {
   @State private var selectedButtonIndex = 0

   var body: some View {
      HStack {
         Button(action: {
            self.selectedButtonIndex = 0
         }, label: {
            Text("All").padding(.horizontal)
               .padding(.vertical, 4)
               .background(Color("default_"))
               .cornerRadius(18)
               .foregroundColor(selectedButtonIndex == 0 ? Color.orange :
Color.white)
               .font(.system(size: 22, weight: .regular))
         })
         ScrollView(.horizontal) {
            HStack {
               ForEach(list.rcSources.indices, id: \.self) { index in
                  Button(action: {
                     self.selectedButtonIndex = index + 1
                  }, label: {
                     Text(list.rcSources[index].description)
                        .padding(.horizontal)
                        .padding(.vertical, 4)
                        .background(Color("default_"))
                        .cornerRadius(18)
                        .foregroundColor(selectedButtonIndex == index + 1 ?
Color.orange : Color.white)
                        .font(.system(size: 22, weight: .regular))
                  })
               }
            }
         }
      }
   }
}
```

```
struct ExCatBtn: View{
   let apiurl = "https://alibrary.in/api/explore_categories/"
   @State var id = 6
   @State var allbtn = String()
   @State var btns = [Subcategorycol]()
   var body: some View{
```

```
        HStack(spacing: 10){
            ScrollView(.horizontal, showsIndicators: false){

                HStack {
                    Text(allbtn).font(.system(size:
15)).foregroundColor(.white).padding().frame(height:
35).background(Color("default")).cornerRadius(22)
                    ForEach(btns,id: \.self){item in
                        Text(item.category_name).font(.system(size:
15)).foregroundColor(.white).padding().frame(height:
35).background(Color("default")).cornerRadius(22)
                    }
                }.frame(height: 65).background(Color.white)

            }
        }.onAppear{
            getData(url: apiurl+"\(id)")
//            getData1(url: apiurl+"\(id)")
        }

    }
 func getData(url: String) {
        guard let url = URL(string: url) else { return }
        URLSession.shared.dataTask(with: url) { (data, _, _) in
            if let data = data {
                do {
                    let results = try JSONDecoder().decode(ExploreModel.self, from:
data)
                    DispatchQueue.main.async {
                        self.btns = results.subcategorycol
                        self.allbtn = results.subCategoryname
                    }
                }
                catch {
                    print(error)
                }
            }
        }.resume()
    }



    }

    //Example of Add list after one page infinite scroll
class pagegetData: ObservableObject{
    @Published var data = [Doc]()
```

```
    @Published var count = 1

    init(){
        updateData()

    }
    func updateData(){
        let url = "https://api.plos.org/search?q=title:%22Food%22&start=\
(count)&rows=50"
        let session = URLSession(configuration: .default)
        session.dataTask(with: URL(string: url)!){(data, _, err) in
            if err != nil{
                print((err?.localizedDescription)!)
                return
            }
            do{
                let json = try JSONDecoder().decode(Detail.self, from: data!)
                let oldData = self.data
                DispatchQueue.main.async {
                    self.data = oldData + json.response.docs
                    self.count += 1
                }
            }catch{
                print(error.localizedDescription)
            }

        }.resume()
    }
}

struct Detail:Decodable{
    var response: Response
}
struct Response:Decodable{
    var docs:[Doc]
}
struct Doc:Decodable{
    var id: String
    var publication_date:String
    var article_type: String
    var eissn:String
}


struct cellView: View{
    var data: Doc
    var isLast: Bool
```

```
    @StateObject var listData:pagegetData
    var body: some View{
        VStack(alignment: .leading, spacing: 12){
            Text(data.id).fontWeight(.bold)
            Text(data.eissn)
            Text(data.article_type)
            if isLast{
                Text(data.publication_date).font(.caption)
                    .onAppear{

                        DispatchQueue.main.asyncAfter(deadline: .now() + 0.5){
//                          if self.listData.data.count != 50{
                              self.listData.updateData()
//                          print("Loading..... ")
                          }
//                      }
                    }
            }
            else{
                Text(data.publication_date).font(.caption)
            }
        }
        .frame(width: 235,height: 325).background(Color.red).cornerRadius(12)

        .padding(.top, 10)
    }
}

// View Change On click on sideMenu Item
import Foundation
import SwiftUI
import Combine




//struct ContentView: View {
//
//    @State private var showMenu = false
//    @State private var selected: SelectedScreen = .home
//
//    var body: some View {
//
//
//
//        NavigationView {
//
```

```
//
//
//             ZStack {
//
//
//                VStack{
//                   HStack{
//                      Button {
//                         withAnimation {
//                            self.showMenu.toggle()
//                         }
//                      } label: {
//                         Image(systemName: "line.horizontal.3")
//                            .imageScale(.large).foregroundColor(.gray)
//                      }
//                      Spacer()
//
//                   }.padding()
//                   Spacer(minLength: 45)
//                }
//             // show selected screen
//             switch selected {
//             case .home:
//                MainView()
//                   .disabled(self.showMenu ? true : false)
//             case .screen1:
//                OtherView(screen: 1)
//
//             case .screen2:
//                OtherView(screen: 2)
//             }
//
//             // put menu over it
//             if self.showMenu {
//                MenuViewQ(showMenu: $showMenu, selected: $selected)
//                   .transition(.move(edge: .leading))
//             }
//          }
//       }
//    }
//}
//
//
//enum SelectedScreen {
//    case home
//    case screen1
//    case screen2
```

```
//}
//
//
//struct MenuViewQ: View {
//
//    @Binding var showMenu: Bool
//    @Binding var selected: SelectedScreen
//
//    var body: some View {
//        HStack {
//        VStack(alignment: .leading, spacing: 24) {
//            Button {
//                selected = .home
//                showMenu = false
//            } label: {
//                Label("Home", systemImage:
"rectangle.righthalf.inset.fill.arrow.right")
//            }
//
//            Button {
//                selected = .screen1
//                showMenu = false
//           } label: {
//                Label("Screen 1", systemImage: "1.circle")
//            }
//
//            Button {
//                selected = .screen2
//                showMenu = false
//            } label: {
//                Label("Screen 2", systemImage: "2.circle")
//            }
//        }
//        .padding()
//        .frame(maxHeight: .infinity)
//        .background(Color(red: 32/255, green: 32/255, blue: 32/255))
//
//            Spacer()
//    }
//    }
//}
//
//struct MainView: View {
//    var body: some View {
//        Text("This is Main View")
//            .font(.largeTitle)
//    }
```

```
//}
//
//struct OtherView: View {
//    let screen: Int
//    var body: some View {
//        Text("Other View: Screen \(screen)")
//            .font(.largeTitle)
//    }
//}
//
//struct ContentView_Previews: PreviewProvider {
//    static var previews: some View {
//        ContentView()
//
//
//    }
//}
```

The package manifest at \Package.swift cannot be accessed (/package.swift doesn’t exist in file system)

Dear User,please note that if you have fully organized your stack and all book content placed in it matches your title according to you, You may allow it to be publicly placed in the public media. Please note that if the content you put in this type does not match your original title,then the library may block it and unnecessarily cancle your stack as aviolation.\n \n Note: Books placed in the stack will not appear in the public stack if the number is less than 3.


```
{
   "studentStacks": {
      "current_page": 1,
      "data": [
         {
            "id": 772,
            "partner_id": 622,

            "stack_book_link_count": 7,
            "stack_book_link": [
               {
                  "id": 2176,

                  "book_url": "https://storage.googleapis.com/pdffilesalib/pdf/
thumbnail/The_Universe_in_a_Nutshell_20221014203906_128/1.png",
                  "stack_book": {
                     "id": 3161,
```

```
                    "upload_type_id": 1,
                    "name": "The Universe in a Nutshell",

                }
            },
            {
                "id": 2150,

                "book_url": "https://storage.googleapis.com/pdffilesalib/pdf/
thumbnail/Comprehensive_Gynecology_7th_Ed_20220707194143_128/1.png",
                "stack_book": {
                    "id": 2647,
                    "upload_type_id": 1,
                    "name": "Comprehensive Gynecology 7th Ed"
                }
            },
            {
                "id": 2149,
                "book_url": "https://storage.googleapis.com/pdffilesalib/pdf/
thumbnail/Williams_Gynecology_3rd_Ed_11/1.png",
                "stack_book": {
                    "id": 65,
                    "upload_type_id": 1,
                    "name": "Williams Gynecology 3rd Ed"
                }
            }
        ],
        "bookcount": 7,

        "copy_stack_name": "Gynecology",
        "copy_partner_id": 28,
        "copy_partner_name": "Guest User",
        "stackBookCount": 7,
        "stack_detail": {
            "id": 790,
            "name": "Gynecology"
        }
    }
  ],

  "total": 1
 },
 "data": 1
}
```

https://alibrary.in/api/guest/ownpdfs?page=1 for dashboard data (done)

https://alibrary.in/api/guest/publish-pdf?page=1 **for publish pdf**

https://alibrary.in/api/guest/unpublish-pdf **for unpublish pdf**
1 =
4= deleted
2= disapproved

https://alibrary.in/api/guest/draft-pdf?page=1 **for draft pdf (done)**

https://alibrary.in/api/guest/unpublish-pdf?is_publish=4&page=1 **for delete (done)**

https://github.com/NuPlay/RichText.git

An online library


```
{
   "userbookrequest": {
      "id": 112,
      "category_name": "Pets",
      "subcategory_name": "Birds"
   },
   "requestBookLinks": {
      "current_page": 1,
      "data": [
      ]

   }
```

```
}
```

and same API for Other user is

```
{
   "userbookrequest": {
      "id": 112,
      "category_name": "Pets",
      "subcategory_name": "Birds"
   },
   "requestBookLinks": {
      "current_page": 1,
      "data": [

 {
            "id": 220,
            "user_request_id": 112,
            "title": "Dogs (DK Handbooks)",
            "category_name": "Pets",
            "subcategory_name": "Dogs & Cats"
         }
      ]

   }
}
```

How to find the data[] and set to the view if data have value exist otherwise it prints A Message using MVVM in swiftui?

Sure, I can help you create a basic resume for a Junior iOS developer with one year of experience. Here is a sample resume that you can use as a starting point:

Anup Kumar Mishra

Village and Post:- Kambli veer Bhan Pur
Kunda Pratapgarh 230129
Uttar Pradesh
+91-8299544315
email:- anoopm678@gmail.com

Objective:

To secure a position as a Junior iOS Developer where I can utilize my

skills and knowledge to contribute to the development of innovative mobile applications.
Education:
Master of Arts (MA)
[CSJMA University Kanpur], [2017]

Skills:

- **Proficient in Swift and Objective-C programming languages**
- **Experience in iOS app development using Xcode and Interface Builder**
- **Strong understanding of iOS application architecture patterns (MVC, MVVM)**
- **Familiarity with RESTful API integration and JSON parsing**
- **Knowledge of version control systems such as Git and SVN**
- **Familiarity with agile software development methodologies**

Professional Experience:

Junior iOS Developer
PrimeItZen Software Solution Pvt. Ltd., 21-Mar-2023 to Today

Responsibilities:

- **Worked on developing and maintaining iOS applications using Swift and Objective-C**
- **Collaborated with the development team to design and implement new app features**
- **Conducted code reviews to ensure code quality and adherence to best practices**
- **Implemented unit and UI tests to ensure app stability and reliability**
- **Troubleshot and fixed bugs reported by users**
- **Participated in daily stand-up meetings and sprint retrospectives**

Projects:

[aLibrary.in], **An online library**

- **Developed a aLibrary using Swift and Xcode**
- **Integrated RESTful APIs to fetch data and displayed it in a clean user interface**
- **Implemented various features including [feature 1], [feature 2], and [feature 3]**
- **Conducted unit and UI tests to ensure app stability and reliability**
- **[Name of Project], [Description], [Link to Github Repo]**
- **Worked on a team to develop a [description of project] using Swift**

and Xcode
- **Collaborated with designers to create a visually appealing user interface**
- **Integrated [API] to fetch data and displayed it in the app**
- **Implemented various features including [feature 1], [feature 2], and [feature 3]**

Certifications:

A Level, NIELIT, 2021,O Level, NIELIT, 2019

References:

Tab view For Header Section tab items

```
//extension View{
//    func pageLable() -> some View {
//        self
//            .frame(maxWidth: .infinity, alignment: .center)
//    }
//    func pageView(ignoreSafeArea: Bool = false, edges: Edge.Set = []) -> some
View {
//        self.frame(width: getScreenBounds().width, alignment: .center)
//            .ignoresSafeArea(ignoreSafeArea ? .container : .init(), edges: edges)
//    }
//    func getScreenBounds() -> CGRect{
//
//        return UIScreen.main.bounds
//    }
//}
//
//
//struct PagerTabView<Content: View, Lable: View>: View {
//    var content: Content
//    var lable: Lable
//    //
//    var tint: Color
//    //selection
//    @Binding var selection: Int
//    init( tint: Color, selection: Binding<Int>, @ViewBuilder lables: @escaping ()
-> Lable , @ViewBuilder content: @escaping () -> Content) {
//        self.content = content()
//        self.lable = lables()
//        self.tint = tint
```

```
//        self._selection = selection
//    }
//
//    //offset for pagescroll
//    @State var offset : CGFloat = 0
//    @State var maxTabs : CGFloat = 0
//    @State var tabOffset : CGFloat = 0
//
//    var body: some View{
//        VStack(spacing: 0){
//            HStack(spacing: 0){
//                lable
//            }
//            .overlay(
//                HStack{
//                    ForEach(0..<Int(maxTabs), id:\.self){ index in
//                        Rectangle().fill(Color.blue.opacity(0.01)).onTapGesture {
//                            let newOffset = CGFloat(index) * getScreenBounds().width
//                            self.offset = newOffset
//                        }
//
//                    }
//
//            })
//
//
//            .foregroundColor(tint)
//
//        //Indicators
//
//            Capsule().fill(tint).frame(width: maxTabs == 0 ? 0 :
(getScreenBounds().width/maxTabs), height: 5).padding(.top, 10)
//                .frame(maxWidth: .infinity, alignment: .leading).offset(x: tabOffset)
//
//            OffsetPageTabView(selection: $selection,offset: $offset){
//                HStack(spacing: 0){
//                    content
//                }.overlay(
//                    GeometryReader{ proxy in
//                        Color.clear
//                            .preference(key: TabPreferenceKey.self, value:
proxy.frame(in: .global))
//                    }
//
//
//                )
//                .onPreferenceChange(TabPreferenceKey.self){
```

```
//                proxy in
//                let minX =  -proxy.minX
//                let maxWidth = proxy.width
//                let screenWidth = getScreenBounds().width
//                let maxtabs = (maxWidth/screenWidth).rounded()
//                let progress = minX/screenWidth
//                let tabOffset = progress * (screenWidth/maxTabs)
//                self.tabOffset = tabOffset
//                self.maxTabs = maxtabs
//            }
//        }
//
//    }
//
// }
//}
//
//
//struct TabPreferenceKey: PreferenceKey{
//    static var defaultValue: CGRect = .init()
//    static func reduce(value: inout CGRect, nextValue: () -> CGRect) {
//        value = nextValue()
//    }
//}
//
//
//
//struct OffsetPageTabView<Content: View>: UIViewRepresentable{
//
//
//    var content : Content
//    @Binding var offset: CGFloat
//    @Binding var selection: Int
//    func makeCoordinator() -> Coordinator {
////        return OffsetPageTabView.Coordinator(parent: self)
//        return OffsetPageTabView.Coordinator(parent: self)
//    }
//    init(selection: Binding<Int>,offset: Binding<CGFloat>, @ViewBuilder
content: @escaping () -> Content) {
//        self.content = content()
//        self._offset = offset
//        self._selection = selection
//    }
//
//    func makeUIView(context: Context) -> UIScrollView{
//
//
```

```
//          let scrollview = UIScrollView()
//          let hostview = UIHostingController(rootView: content)
//          hostview.view.translatesAutoresizingMaskIntoConstraints = false
//
//      let constraints = [hostview.view.topAnchor.constraint(equalTo:
scrollview.topAnchor), hostview.view?.leadingAnchor.constraint(equalTo:
scrollview.leadingAnchor),
//                         hostview.view.trailingAnchor.constraint(equalTo:
scrollview.trailingAnchor),
//                         hostview.view.bottomAnchor.constraint(equalTo:
scrollview.bottomAnchor),
//                         hostview.view.heightAnchor.constraint(equalTo:
scrollview.heightAnchor)
//      ].compactMap { $0 }
//
//      scrollview.addSubview(hostview.view)
//          NSLayoutConstraint.activate(constraints)
//      //Enabling Paging
//      scrollview.showsVerticalScrollIndicator = true
//      scrollview.showsHorizontalScrollIndicator = true
//      return scrollview
//  }
//
//  func updateUIView(_ uiView: UIScrollView, context: Context) {
//      let currentOffset = uiView.contentOffset.x
//      if currentOffset != offset {
//          uiView.setContentOffset(CGPoint(x: offset, y: 0), animated: true)
//      }
//  }
//  class Coordinator: NSObject, UIScrollViewDelegate{
//      var parent: OffsetPageTabView
//      init(parent: OffsetPageTabView) {
//          self.parent = parent
//      }
//
//
//      func scrollViewDidScroll(_ scrollView: UIScrollView) {
//          let offset = scrollView.contentOffset.x
//
//          let maxSize = scrollView.contentSize.width
//          let currentSelection = (offset/maxSize).rounded()
//          parent.selection = Int(currentSelection)
//
//          parent.offset = offset
//      }
//  }
//}
```

//
//

RESUME

Name: Anup Kumar Mishra
Contact Information:

- **Address: Pratapgarh, Uttar Pradesh**
- **Phone: +91-8299544315**
- **Email: anoopm678@gmail.com**

Objective:
To secure a iOS Developer position where I can leverage my skills and knowledge to contribute to the development of innovative mobile applications.
Education:
Master of Arts (MA) in Hindi Literature, [CSJMA University Kanpur], [2017]
Skills:

- **Proficient in Java , Swift and Objective-C programming languages**
- **Experienced in iOS app development using Xcode and Interface Builder**
- **Strong understanding of iOS application architecture patterns (MVC, MVVM)**
- **Familiarity with RESTful API integration and JSON parsing**
- **Knowledge of version control systems such as Git.**
- **Familiarity with agile software development methodologies**

Professional Experience:

Junior iOS Developer, PrimeItZen Software Solution Pvt. Ltd., 21 Mar 2022 to Today

- **Developed and maintained iOS applications using Swift and Xcode.**
- **Collaborated with the development team to design and implement new app features.**
- **Conducted code reviews to ensure code quality and adherence to best practices.**
- **Implemented unit and UI tests to ensure app stability and reliability.**
- **Troubleshot and fixed bugs reported by users.**
- **Participated in daily stand-up meetings and sprint retrospectives.**

Projects:

- aLibrary.in, e-book business.
- Developed aLibrary using Swift language and SwiftUI framework and Xcode IDE.
- Integrated RESTful APIs to fetch data and displayed it in a clean user interface.
- Implemented various features including User authentication, Multimedia support,In-app. purchases,Search functionality, Location-based services and Accessibility features.
- Conducted unit and UI tests to ensure app stability and reliability.
- aLibrary, e-book business.
- Worked on a team to develop aLibrary e-book business iOS apps using Swift and Xcode.
- Collaborated with designers to create a visually appealing user interface.
- Integrated RestAPI to fetch data and displayed it in the app.

Certifications:

- A Level, NIELIT, 2021
- O Level, NIELIT, 2019

```
class RCTransactionViewModel: ObservableObject {

   @Published var datas = [RCTransDatum]()
   @Published var rcSources = [RCTransSource]()
   @Published var totalRC = Int()
   @Published var totalPage = Int()
   @Published var currentPage = 1


   func getRCTransactionData(source: String, page: Int = 1) {
      let defaults = UserDefaults.standard
      guard let token = defaults.string(forKey: "access_token")
else {
         return
      }
```

```
      RCTransactionService().getRCTransactionData(token: token,
source: source, page: page){ (result) in
        switch result {
          case .success(let results):
            DispatchQueue.main.async {
              if page == 1 {
                self.datas = results.rcTransHistories.data
                self.rcSources = results.rcSources
              } else {
                self.datas.append(contentsOf:
results.rcTransHistories.data)
              }
              self.totalRC = results.totalRc.total_rc
              self.totalPage = results.rcTransHistories.total
            }
          case .failure(let error):
            print(error.localizedDescription)
        }
      }
    }

    func loadNextPage(source: String) {
      guard currentPage < totalPage else { return }
      currentPage += 1
      getRCTransactionData(source: source, page: currentPage)
    }
}
```

STUDENT LOGIN

1. **SWAT997060.**
2. **password**

https://www.alibrary.in/api/getbookmedias?bookmediatype=video&bookid=4152

Teacher Login:-
UserId:- rohansingh@gmail.com
password:- password

UserId:- rosh273206
password:- password

DOB 21-11-1961 Rajkumar Baijnath & Mamta Srivastava

Attendance Chart:-

Present	*Absent*
05/03/2023	*11/05/2023*
//18/04/2023 Half Day	*15/05/2023*
14/05/2023	*30/05/2023*
28/05/2023	*27/06/2023*
04/06/2023	*12/07/2023*
25/06/2023	*27/06/2023*
29/06/2023	*26/10/2023*
24/09/2023	*06/11/2023*
08/10/2023	*23/11/2023*
22/10/2023	*11/12/2023*
// 09/11/2023 Half day	*12/12/2023*
//18/12/2023 Half day	

Jalata tel aur Baati, Prakaash ke lie lekin log Kahate hain diya jal raha hai.

जलता तेल और बाती, प्रकाश के लिए लेकिन लोग कहते हैं दिया जल रहा है|

Burning oil and wick, for light but people say the lamp is burning.

Alibrary | Online Library | Online Digital Library | Digital Library
alibrary.in

API For Exam and test.
This API Use :-

https://alibrary.in/api/student/getsubjects
https://alibrary.in/api/student/showSubjectCourses?subject_id=
https://alibrary.in/api/guest/guest-course
https://alibrary.in/api/student/showResult?test_id=

I want to make my and my family's economic life easy.
Password bonus21

Sandeep Sir Balance Total = 15000
I Returned 5000 Cash + 4000 GPay on 09 August + 2000 online
Now Remaining Balance 4000