
REPORT□



개발자 가이드

1. 시스템 아키텍처

하위 링크 참조

<https://github.com/castelwoah/OSS/blob/main/docs/%5B%EC%98%A4%ED%94%88%EC%86%8C%EC%8A%A4SW%EA%B8%B0%EC%97%AC%5D%20%EC%8B%9C%EC%8A%A4%ED%85%9C%20%EC%84%A4%EA%B3%84.pdf>

2. 개발 프레임워크

2.1. UIKit

UIKit 프레임 워크는 iOS 와 tvOS(애플 티비)에 들어가는 앱을 빌드하는데 필요한 핵심 오브젝트(core objects)를 지원합니다. Xcode 는 UI 기반으로 앱을 빌드하며 UI의 각 요소가 오브젝트(object)로써 기능합니다.

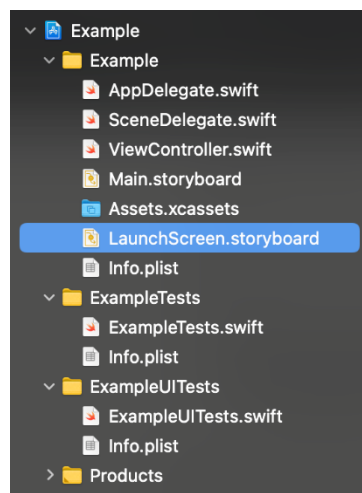
UIKit 를 사용한 애플리케이션은 다음의 두가지 요소를 필수적으로 가져야 합니다.

1) 애플리케이션 아이콘

‘애플리케이션 아이콘’은 말 그대로 앱스토어에서 어플을 다운로드 받을 때 보이는 그림, 보여질 그림입니다. 홈 스크린, 세팅 화면, 알람화면 등에 표시됩니다.

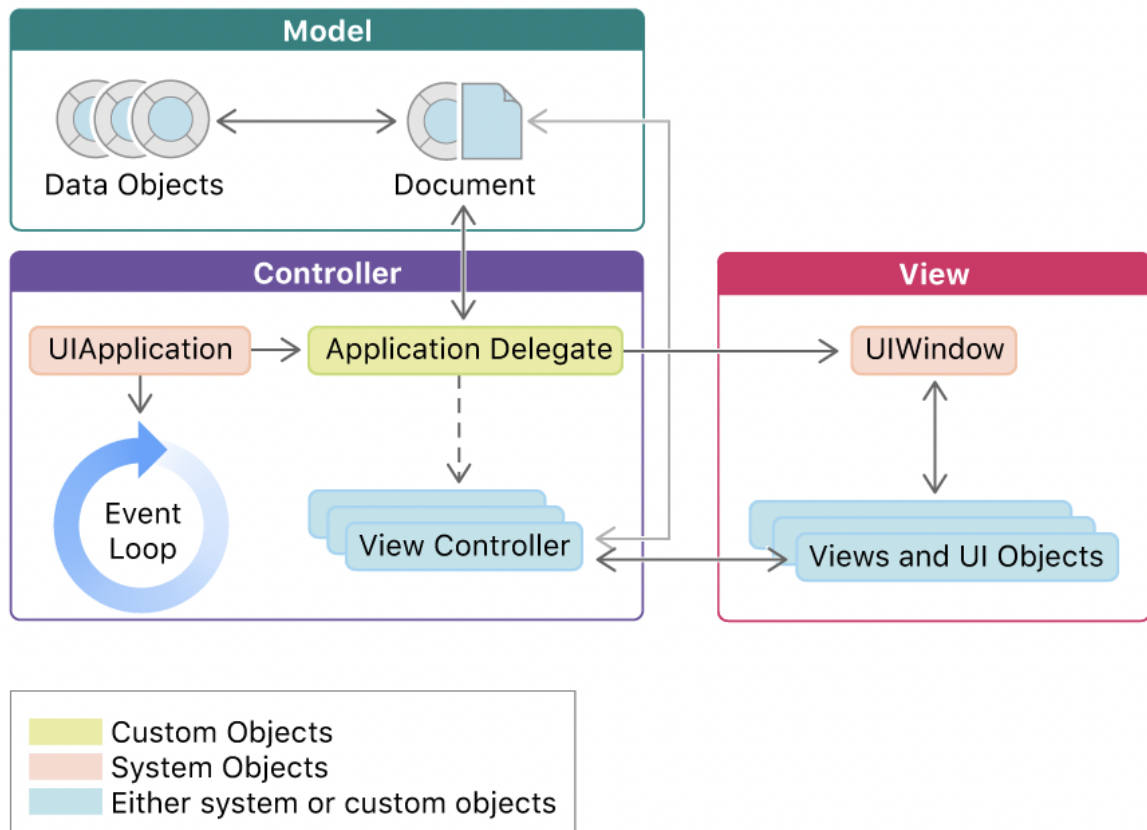
2) Launch screen storyboard

본 파일은 자동으로 프로젝트를 만들면 파일에 생성되어 있습니다. 이름 그대로 LaunchScreen.storyboard 는 애플리케이션이 시작될 때 보여지는 화면을 말합니다.



AppDelegate 는 전반적인 애플리케이션의 라이프 사이클(Life Cycle)을 다루고 있습니다. 이곳에 애플리케이션을 실행하기 전 시간을 뒤야 하므로 sleep() 매서드를 이용해주면 런치 스크린에 작성한 화면이 초단위로 표현됩니다.

2.2. UIKit 애플리케이션의 코드 구조



UIKit 는 앱의 메인 이벤트 루프(Main Event Loop)를 실행하고 화면에 콘텐츠를 표시하며 여러 오브젝트를 제공합니다. 그러기에 앱의 구조가 어떻게 동작하는지 알아야 재사용 가능하고 지속가능한 코드를 짤 수 있습니다.

UIKit 앱의 구조는 MVC(Model - View - Controller)디자인 패턴을 기반으로 동작합니다.

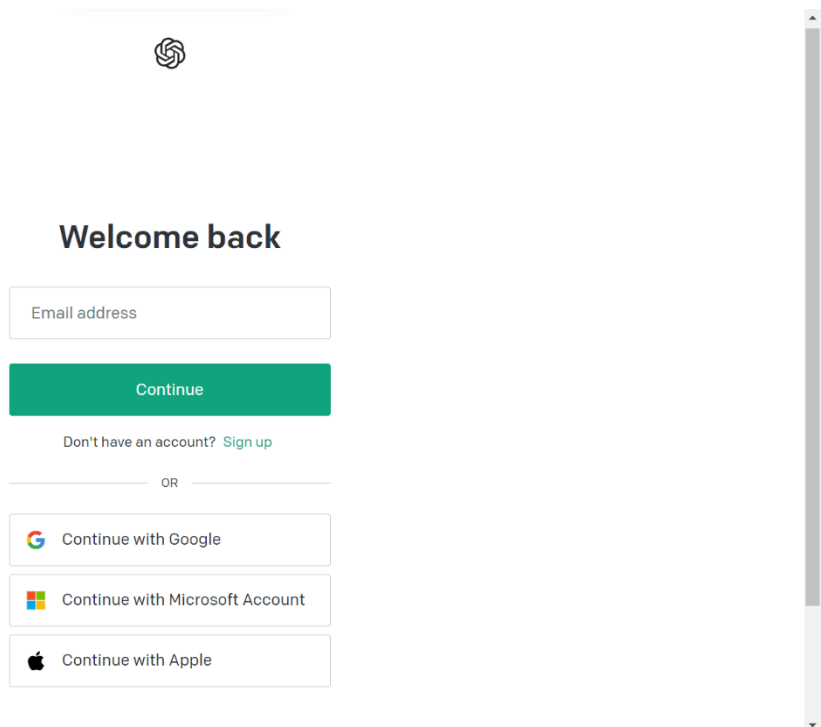
- Model 은 앱의 오브젝트 데이터들을 관리합니다. (동작)
- View 에서는 데이터들의 시각적인 표현을 제공합니다. (표현)
- Controller 에서는 Model 과 View 사이에서 다리 같은 역할을 하면서 적절한 시간에 데이터를 이동시키며 상호작용을 돕습니다.

위 그림에서 Application Delegate 는 오브젝트(객체)들의 동작을 담은 메서드나 함수를 담고 있습니다. 이러한 부분을 뷰와 연결시켜주는 역할을 계속 수행하는 것이 Controller 입니다.

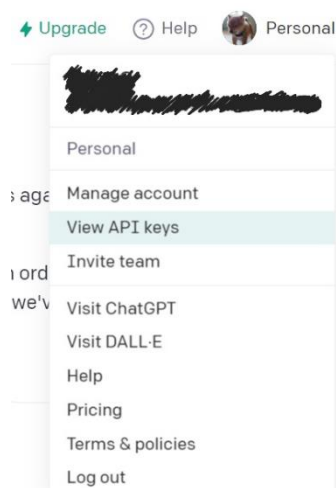
3. 개발환경 설치

3.1. openai api key 발급 받기

<https://platform.openai.com/account/api-keys> 페이지로 접속 후 로그인



우측 상단의 "Personal" 메뉴에서 "View API keys" 클릭



Create new secret key 클릭

The screenshot shows the OpenAI API keys management page. On the left, there's a sidebar with 'ORGANIZATION' and 'USER' sections. Under 'USER', 'API keys' is selected. The main content area is titled 'API keys' and contains instructions about secret API keys. Below the instructions is a table with columns: NAME, KEY, CREATED, and LAST USED. The table has one row for 'test key' with a key starting 'sk-...GzMA'. Below the table, a red box highlights the '+ Create new secret key' button. Further down, there's a 'Default organization' section with a dropdown menu set to 'Personal'.

NAME	KEY	CREATED	LAST USED
test key	sk-...GzMA	2023년 5월 21일	2023년 5월 26일

Secret key 이름을 작성한 후 create 버튼 클릭

This screenshot shows the same OpenAI API keys page, but with a modal dialog open for creating a new secret key. The dialog is titled 'Create new secret key' and has a 'Name' field with the text 'My Test Key' entered. There are 'Cancel' and 'Create secret key' buttons at the bottom of the dialog. The background page is dimmed.

발급 받은 api key 는 절대로 다른 사람과 공유하면 안되며 API 키가 실수로 프로그램 코드를 통해서 노출되지 않도록 각별히 주의가 필요합니다.

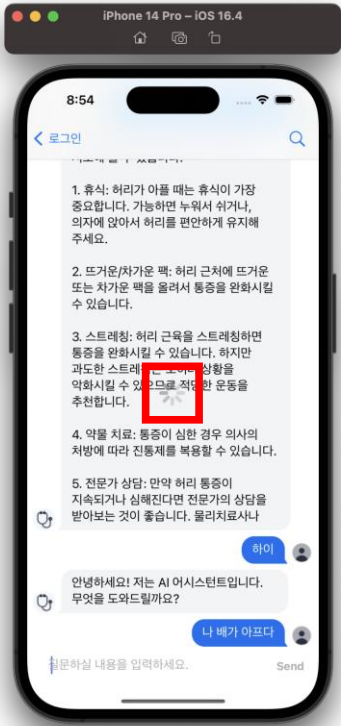
test/View/ChatViewController.swift 코드에 api key 를 입력한다.

```
private var messages: [MessageType] = []  
private let apiKey = "<Your API key>"  
private let botSender = Sender(senderId: "bot_id", displayName: "Bot")  
private let openAI = OpenAISwift(authToken: "<Your API key>")
```

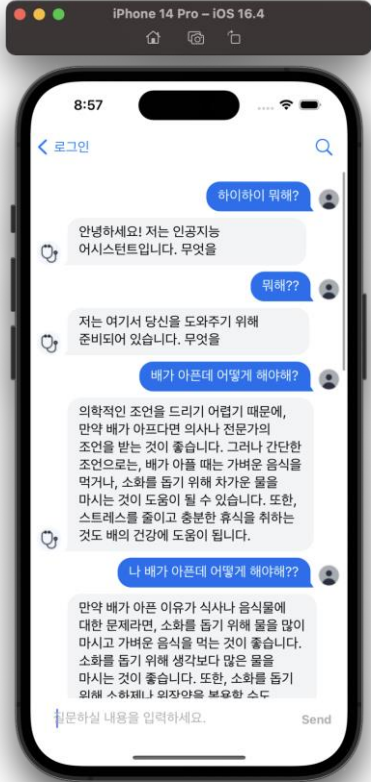
4. 모듈별 기능

기능	써치바
화면	
소스코드	<pre>// MARK: - searchBar private let searchBar: UISearchBar = { let bounds = UIScreen.main.bounds let width = bounds.size.width // 화면 너비 let searchBar = UISearchBar(frame: CGRect(x: 0, y: 0, width: width - 28, height: 0)) searchBar.placeholder = "이전 대화를 검색하세요" searchBar.showsCancelButton = true return searchBar }()</pre>

	<pre> @objc private func searchButtonTapped() { searchBar.isHidden = false searchBar.becomeFirstResponder() navigationItem.titleView = searchBar } // MARK: - searchBar // 네비게이션 바 오른쪽에 돋보기 버튼 추가 let searchButton = UIBarButtonItem(image: UIImage(systemName: "magnifyingglass"), style: .plain, target: self, action: #selector(searchButtonTapped)) navigationItem.rightBarButtonItem = searchButton // 써치바 초기화 searchBar.delegate = self searchBar.isHidden = true searchBar.showsCancelButton = true // 써치바를 네비게이션 바의 타이틀 뷰로 설정 navigationItem.titleView = searchBar } func searchBarCancelButtonClicked(_ searchBar: UISearchBar) { searchBar.isHidden = true searchBar.text = "" searchBar.resignFirstResponder() navigationItem.title = "Dr.Bot" // 연립 loadPreviousMessages() } func searchBarSearchButtonClicked(_ searchBar: UISearchBar) { guard let searchText = searchBar.text, !searchText.isEmpty else { return } searchBar.resignFirstResponder() // 이전 메시지 중 검색어를 포함하는 메시지 필터링 let filteredMessages = messages.filter { message in if case let .text(text) = message.kind { return text.localizedCaseInsensitiveContains(searchText) } return false } messages = filteredMessages messagesCollectionView.reloadData() messagesCollectionView.scrollToLastItem(animated: false) } </pre>
<p>설명</p>	<p>네비게이션 바 안에서 가운데쯤 정렬이 되도록 위치를 정해주었다.</p> <p>searchButton 은 돋보기버튼 아이콘으로, 버튼을 누를 시 동작이 일어나도록 selector 를 설정했다. 써치바 구현을 위해 UISearchBar 를 이용해 선언해주었으며, 다음과 같이 초기설정을 해 주었다.</p> <p>네비게이션 바 안에서 가운데쯤 정렬이 되도록 위치를 정해주었다.</p> <p>searchButton 은 돋보기버튼 아이콘으로, 버튼을 누를 시 동작이 일어나도록 selector 를 설정했다.</p> <p>숨겨져 있던 searchBar 를 보이게 된다. 반대로 취소 버튼을 누르면 이전 대화 내용을 다시 불러오면서 네비게이션 바에 보이던 써치바는 사라진다.</p> <p>검색 단어와 비교하는 필터를 거쳐 출력되어 진다.</p>

기능	로딩바 구현
화면	
소스코드	<pre>// MARK: - loadingIndicator private var isLoading = false private let loadingIndicator: UIActivityIndicatorView = { let indicator = UIActivityIndicatorView(style: .large) indicator.color = .gray indicator.translatesAutoresizingMaskIntoConstraints = false return indicator }() // MARK: - loadingIndicator view.addSubview(loadingIndicator) loadingIndicator.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true loadingIndicator.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true</pre>

	<pre> extension ChatViewController: InputBarAccessoryViewDelegate { func inputBar(_ inputBar: InputBarAccessoryView, didPressSendButtonWith text: String) { if isLoading { return // 이미 로딩 중인 경우 중복 요청 방지 } isLoading = true loadingIndicator.startAnimating() let message = Message(text: text, sender: currentSender, messageId: UUID().uuidString, date: Date()) messages.append(message) inputBar.inputTextView.text = "" messagesCollectionView.reloadData() async { do { let result = try await self.openAI.sendChat(with: [ChatMessage(role: .system, content: text)]) if let botResponse = result.choices?.first?.message.content { self.saveMessageToFirestore(message: message, botResponse: botResponse) print("사용자: " + text) print("챗봇: " + botResponse) DispatchQueue.main.async { self.messagesCollectionView.reloadData() self.messagesCollectionView.scrollToLastItem() self.loadingIndicator.stopAnimating() self.isLoading = false } } } catch { print(error.localizedDescription) DispatchQueue.main.async { self.loadingIndicator.stopAnimating() self.isLoading = false } } } } } </pre>
설명	<p>isLoading 을 통해 초기화를 시키고, 로딩중인 상황이 사용자에게 잘 보일 수 있도록 뷰의 중간에 위치하도록 설계했으며 크기도 제일 크게 만들었다.</p> <p>isLoading 은 아래의 코드에서 적용되어 사용된다.</p> <p>InputBar 에 사용자 입력이 받아지면 true 를 반환해 실행되며, 마지막에 메시지가 출력되면 false 로 다시 바뀌는 것을 확인할 수 있다.</p>

기능	이전 메시지 불러오는 기능
화면	 <p>The screenshot shows an iPhone 14 Pro screen with the status bar at the top displaying 'iPhone 14 Pro - iOS 16.4'. The time is 8:57. The screen displays a chat interface with a contact named '로그인'. The chat history is visible, showing previous messages and responses. The current message asks for advice on how to deal with a stomach ache, and the assistant provides detailed advice on diet, hydration, and stress management.</p> <p>로그인</p> <p>하이하이 뭐해?</p> <p>안녕하세요! 저는 인공지능 어시스턴트입니다. 무엇을</p> <p>뭐해??</p> <p>저는 여기서 당신을 도와주기 위해 준비되어 있습니다. 무엇을</p> <p>배가 아픈데 어떻게 해야해?</p> <p>의학적인 조언을 드리기 어렵기 때문에, 만약 배가 아프다면 의사나 전문가의 조언을 받는 것이 좋습니다. 그러나 간단한 조언으로는, 배가 아플 때는 가벼운 음식을 먹거나, 소화를 돕기 위해 차가운 물을 마시는 것이 도움이 될 수 있습니다. 또한, 스트레스를 줄이고 충분한 휴식을 취하는 것도 배의 건강에 도움이 됩니다.</p> <p>나 배가 아픈데 어떻게 해야해??</p> <p>만약 배가 아픈 이유가 식사나 음식물에 대한 문제라면, 소화를 돕기 위해 물을 많이 마시고 가벼운 음식을 먹는 것이 좋습니다. 소화를 돕기 위해 생각보다 많은 물을 마시는 것이 좋습니다. 또한, 소화를 돕기 위해 수화제나 위장약을 복용할 수도</p> <p>문하실 내용을 입력하세요. Send</p>

소스코드

```
func loadPreviousMessages() {
    guard let currentUser = Auth.auth().currentUser else {
        return
    }

    let database = Firestore.firestore()
    let query = database.collection("chats")
        .whereField("senderId", isEqualTo: currentUser.uid)
        .orderBy(by: "sentDate")

    query.getDocuments { [weak self] (snapshot, error) in
        guard let self = self, let snapshot = snapshot else {
            if let error = error {
                print("Error messages: \(error)")
            }
            return
        }

        self.messages.removeAll() // 기존 메시지 삭제

        for document in snapshot.documents {
            let data = document.data()

            guard let senderId = data["senderId"] as? String,
                  let senderName = data["senderName"] as? String,
                  let messageId = data["messageId"] as? String,
                  let sentDateTimestamp = data["sentDate"] as? TimeInterval,
                  let text = data["text"] as? String,
                  let botResponse = data["botResponse"] as? String else {
                continue
            }

            let sender: Sender
            if senderId == currentUser.uid {
                sender = Sender(senderId: senderId, displayName: "User")
            } else {
                sender = Sender(senderId: senderId, displayName: "Bot")
            }

            let sentDate = Date(timeIntervalSince1970: sentDateTimestamp)
            let message = Message(text: text, sender: sender, messageId: messageId, date: sentDate)
            let botMessage = Message(text: botResponse, sender: self.botSender, messageId: UUID().uuidString)

            self.messages.append(message)
            self.messages.append(botMessage)
        }

        DispatchQueue.main.async {
            self.messagesCollectionView.reloadData()
            self.messagesCollectionView.scrollToLastItem()
        }
    }
}
```

설명

사용자가 회원가입을 하게되면 각 고유의 UID 를 갖게된다.

<input type="text" value="이메일 주소, 전화번호 또는 사용자 UID로 검색"/> 사용자 추가 🔄 ⋮				
식별자	제공업체	생성한 날짜 ↓	로그인한 날짜	사용자 UID
test@gmail.com		2023. 6. 14.	2023. 6. 14.	POAaLVJ6YLQI1f60RdA6M4SYQO...
<div>페이지당 행 수: 50 ▼ 1 - 1 of 1 < ></div>				

해당 UID 를 senderID 에 저장하게하여, 쿼리문을 이용해 데이터베이스에서 채팅내용을 불러오도록 설정했다.

복합

단일 필드

색인 추가

컬렉션 ID	색인이 생성된 필드 ②	쿼리 범위	상태
chats	senderId 오름차순 sentDate 오름차순 __name__ 오름차순	컬렉션	사용 설정됨

파이어베이스에서 쿼리문을 실행되도록 하기 위해 사진과 같이
파이어베이스의 복합필드를 설정해 주어 정상적으로 작동할 수 있었다.

기능	로그인 뷰
화면	
소스코드	<pre> // MARK: - 이메일 입력하는 텍스트 뷰 private lazy var emailTextFieldView: UIView = { let view = UIView() view.backgroundColor = #808080 view.layer.cornerRadius = 5 view.clipsToBounds = true view.addSubview(emailTextField) view.addSubview(emailInfoLabel) return view }() // "이메일 또는 전화번호" 안내문구 private var emailInfoLabel: UILabel = { let label = UILabel() label.text = "이메일주소" label.font = UIFont.systemFont(ofSize: 18) label.textColor = #808080 return label }() private lazy var emailTextField: UITextField = { var tf = UITextField() tf.frame.size.height = 48 tf.backgroundColor = .clear tf.textColor = .white tf.tintColor = .white tf.autocapitalizationType = .none tf.spellCheckingType = .no tf.autocorrectionType = .no tf.keyboardType = .emailAddress tf.addTarget(self, action: #selector(textFieldEditingChanged(_:)), for: .editingChanged) return tf }() </pre>

```

// MARK: - 비밀번호 입력하는 텍스트 뷰
private lazy var passwordTextFieldView: UIView = {
    let view = UIView()
    view.frame.size.height = 48
    view.backgroundColor = #808080
    view.layer.cornerRadius = 5
    view.clipsToBounds = true
    view.addSubview(passwordTextField)
    view.addSubview(passwordInfoLabel)
    view.addSubview(passwordSecureButton)
    return view
}()

// 패스워드 텍스트필드의 안내문구
private var passwordInfoLabel: UILabel = {
    let label = UILabel()
    label.text = "비밀번호"
    label.font = UIFont.systemFont(ofSize: 18)
    label.textColor = #808080
    return label
}()

// 로그인 - 비밀번호 입력 필드
private let passwordTextField: UITextField = {
    let tf = UITextField()
    tf.backgroundColor = #808080
    tf.frame.size.height = 48
    tf.backgroundColor = .clear
    tf.textColor = .white
    tf.tintColor = .white
    tf.autocapitalizationType = .none
    tf.autocorrectionType = .no
    tf.spellCheckingType = .no
    tf.isSecureTextEntry = true
    tf.clearsOnBeginEditing = false
    tf.addTarget(self, action: #selector(textFieldEditingChanged(_:)), for: .editingChanged)
    return tf
}()

// 패스워드에 "표시" 버튼 비밀번호 가리기 기능
private let passwordSecureButton: UIButton = {
    let button = UIButton(type: .custom)
    button.setTitle("표시", for: .normal)
    button.setTitleColor(#colorLiteral(red: 0.8374180198, green: 0.8374378085, blue: 0.8374271393, alpha: 1), for: .normal)
    button.titleLabel?.font = UIFont.systemFont(ofSize: 14, weight: .light)
    button.addTarget(self, action: #selector(passwordSecureModeSetting), for: .touchUpInside)
    return button
}()

// MARK: - 회원가입 버튼

private let registerButton: UIButton = {
    let button = UIButton(type: .custom)
    button.backgroundColor = .clear
    button.layer.cornerRadius = 5
    button.clipsToBounds = true
    button.layer.borderWidth = 1
    button.layer.borderColor = #808080
    button.setTitle("회원가입", for: .normal)
    button.setTitleColor(.black, for: .normal)
    button.titleLabel?.font = UIFont.boldSystemFont(ofSize: 16)
    button.isEnabled = true
    button.addTarget(self, action: #selector(registerButtonTapped), for: .touchUpInside)
    return button
}()

```

	<pre> // MARK: - 로그인 버튼 private let loginButton: UIButton = { let button = UIButton(type: .custom) button.backgroundColor = .clear button.layer.cornerRadius = 5 button.clipsToBounds = true button.layer.borderWidth = 1 button.layer.borderColor = #000000 button.setTitle("로그인", for: .normal) button.setTitleColor(.black, for: .normal) button.titleLabel?.font = UIFont.boldSystemFont(ofSize: 16) button.isEnabled = true button.addTarget(self, action: #selector(loginButtonTapped), for: .touchUpInside) return button }() lazy var stackView: UIStackView = { let st = UIStackView(arrangedSubviews: [emailTextFieldView, passwordTextFieldView, loginButton, registerButton]) st.spacing = 18 st.axis = .vertical st.distribution = .fillEqually st.alignment = .fill print("stack") return st }() </pre>
설명	<p>로그인 화면에서 보여지는 각 버튼을 구현하는 코드들이다.</p> <p>만약 파이어베이스에 등록된 이메일, 비밀번호와 일치하지 않을 시 다음과 같은 알림창을 팝업하여 사용자에게 알려준다.</p> <p>비밀번호를 틀리게 쓸 경우도 있기 때문에 사용자에게 편의를 주기 위해 표시버튼을 통해 비밀번호를 보여주며, 텍스트필드의 부분을 클릭하게 되면 좌측상단으로 작게 보이도록 구현했다.</p>

기능	회원가입 뷰
화면	 <p>The image displays four sequential screenshots of a mobile application's sign-up process on an iPhone 14 Pro (iOS 16.4). Each screen shows the '회원가입' (Sign Up) page with a back arrow and a '로그인' (Login) link.</p> <ul style="list-style-type: none"> Top Left (9:04): The initial sign-up screen. It features two input fields: '이메일주소' (Email Address) and '비밀번호' (Password). A '가입하기' (Sign Up) button is at the bottom. Top Right (9:05): A confirmation screen for the email 'test@gmail.com'. It displays a message: '해당 이메일은 이미 사용 중입니다.' (This email is already in use). A '확인' (Confirm) button is at the bottom. Bottom Left (9:05): A confirmation screen for the email 'test2@gmail.com'. It displays a message: '비밀번호가 너무 짧습니다. 6글자 이상으로 해주세요.' (Password is too short. Please use 6 or more characters). A '확인' (Confirm) button is at the bottom. Bottom Right (9:06): A confirmation screen for the email 'test2@gmail.com'. It displays a message: '유효하지 않은 이메일 주소입니다.' (Invalid email address). A '확인' (Confirm) button is at the bottom.

소스코드	<pre> let firebaseErrorMessages: [AuthErrorCode.Code: String] = [.emailAlreadyInUse: "해당 이메일은 이미 사용 중입니다.", .invalidEmail: "유효하지 않은 이메일 주소입니다.", .weakPassword: "비밀번호가 너무 짧습니다. 6글자 이상으로 해주세요.",] @objc func signUpButtonTapped() { guard let email = emailTextField.text, let password = passwordTextField.text else { return } Auth.auth().createUser(withEmail: email, password: password) { (authResult, error) in if let error = error { // 회원가입 실패 self.showSignUpFailureAlert() print("Error creating user: \(error.localizedDescription)") } else { // 회원가입 성공 guard let uid = authResult?.user.uid else { // uid를 가져올 수 없는 경우 if let error = error { print("Error getting UID: \(error)") } return } _ = Sender(senderId: uid, displayName: "") let alert = UIAlertController(title: "", message: "회원가입이 완료되었습니다.", preferredStyle: .alert) let checkAction = UIAlertAction(title: "확인", style: .default) { _ in // 확인 버튼을 눌렀을 때의 동작을 처리 if let navigationController = self.navigationController { navigationController.popToRootViewController(animated: true) } } alert.addAction(checkAction) self.present(alert, animated: true, completion: nil) } } } </pre>
설명	<p>로그인화면과 유사한 뷰들로 구현이 되어서, 회원가입 버튼을 눌렀을 때 실패하면 회원가입이 실패했을 경우, 파이어베이스의 각 에러 코드별로 각기 다른 에러를 출력하도록 설계했다.</p> <p>이후 모든 조건에 부합하면, 회원가입이 완료되었다는 알림창을 띄우고, 로그인화면으로 돌아간다.</p>

5. 테스트 방법

하위 링크 참조

<https://github.com/castelwoah/OSS/blob/85fb2db313ee37cecb34a148f48e5033753c04ac/docs/%ED%85%8C%EC%8A%A4%ED%8A%B8%20%EA%B3%84%ED%9A%8D%EC%84%9C.pdf>

5.1. 테스트 단계별 수행 내용

테스트 단계	주요 수행내용
단위 테스트	단위 테스트는 코딩 직후 소프트웨어 설계의 최소 단위인 ‘모듈’이나 ‘컴포넌트’에 초점을 맞춰 테스트 사용자의 요구사항을 기반으로 기능성 테스트를 최우선으로 수행한다.
통합 테스트	통합 테스트는 단위 테스트가 완료된 모듈들을 결합하여 하나의 시스템으로 완성시키는 과정의 테스트 통합 테스트는 모듈 간, 통합된 컴포넌트 간의 상호 작용 오류를 검사한다.
시스템 테스트	개발된 소프트웨어가 해당 컴퓨터 시스템에서 완벽하게 수행하는지를 점검하는 테스트 환경적인 장애 리스크를 최소화하기 위해 실제 사용 환경과 유사한 테스트 환경에서 테스트 해야한다.
인수 테스트	인수 테스트는 개발한 소프트웨어가 사용자의 요구사항을 충족하는지에 중점을 두고 수행하는 테스트 인수 테스트는 개발한 소프트웨어를 사용자가 직접 테스트한다.

5.2. 단위 테스트 시나리오

			테스트 결과 범례: pass - 성공, fail - 실패, N/A - 테스트 불가, 기타				
업무		테스트 시나리오			테스트 수행결과		
분류	테스트 ID	케이스 설명	테스트 데이터	예상결과	테스트결과	테스트일자	성명
backend	BE-001	이름 / 생년월일을 입력하면 DB 에 저장되는가?		이름 / 생년월일을 입력하면 DB 에 저장됨			
	BE-002	DB 에서 각 메시지에 고유키값을 부여하는가?		DB 에 저장할때 각 메시지에 고유키값이 부여됨			
	BE-003	user 와 assistant, 전체 message 로 분류 하여 저장되는가?		user 와 assistant, 전체 message 로 분류 하여 저장됨			
	BE-004	메시지 기록을 잘 불러오는가?		메시지 기록을 잘 불러옴			
	BE-005	API 에 설정된 persona 값이 적용이 잘 되었는가?	"자기소개해 주세요"	지정된 이름과 역할을 출력한다.			
	BE-006	만족도 조사 선택항목이 DB 에		만족도 조사 선택항목이			

		저장되었는가?		DB 에 저장됨			
frontend	FE-001	메시지 전송 버튼 클릭 시 이벤트가 잘 발생 되는가?		텍스트필드 에 담긴 메시지가 뷰레이아웃 에 잘 전송되어 나타난다.			
	FE-002	검색 버튼 클릭시 이벤트가 잘 발생 되는가?		이전 메시지 기록 검색을 할 수 있는 상태로 바뀐다.			
	FE-003						
	FE-004						
	FE-005						

5.3. 통합 테스트 시나리오

				테스트 결과 범례: pass - 성공, fail - 실패, N/A - 테스트 불가, 기타				
업무			테스트 시나리오			테스트 수행결과		
분류	테스트 ID	테스트명	케이스 설명	테스트 데이터	예상결과	테스트결과	테스트일자	성명
실행		실행	Dr.Bot 앱을 실행한다	앱을 터치한다.	의사 아이콘이 화면에 출력되며 함께 앱이 실행된다.			
					환영 인사와 함께 입력란과 검색 아이콘이 화면에 출력된다.			
사용		질문	사용자가 본인의 증상에 대해 포괄적으로 질문한다.	"머리가 아파요"	증상 발현 주기, 통증정도, 정확한 통증위치 등을 되묻는다.			
		사용자 요구사항	사용자가 통증 완화에 도움이 되는	"손목 통증이 오래 지속되는데 통증	통증 완화에 도움이 되는 대처법			

			대처법 3 가지를 질문한다.	완화에 도움이 되는 방법 3 가지 알려줘"	3 가지를 제공한다.			
					증상이 지속되거 나 통증이 악화될시 병원방문 진료를 권고한다 는 답변을 출력한다.			
		사용자 요구사 항	사용자가 증상 완화에 도움이 되는 약 3 가지를 질문한다.	"두통이 지속되서 불편한데 증상 완화에 도움이 되는 약 3 가지 알려줘"	증상 완화에 도움이 되는 약 3 가지와 복용시 주의 사항을 출력한다.			
					약물 사용에는 특히 주의가 필요함을 명시한다.			
		이외 정보입 력	의료정보 와 관련없는 내용을 질문한다.	"자동차 엔진 3 가지 알려줘"	답변하되 의료정보 제공이 본역할임 을 제시한다.			

		검색	키워드로 대화 기록을 검색한다.	검색(돋보기) 아이콘을 클릭하여 키워드를 입력한다.	키워드에 해당하는 글씨가 하이라이트 처리되어 출력되며 해당 위치로 화면이 이동한다.			
		만족도 조사	사용자에게 만족도 조사를 버튼식으로 제공한다.		만족도 조사 말풍선을 화면에 출력한다.			