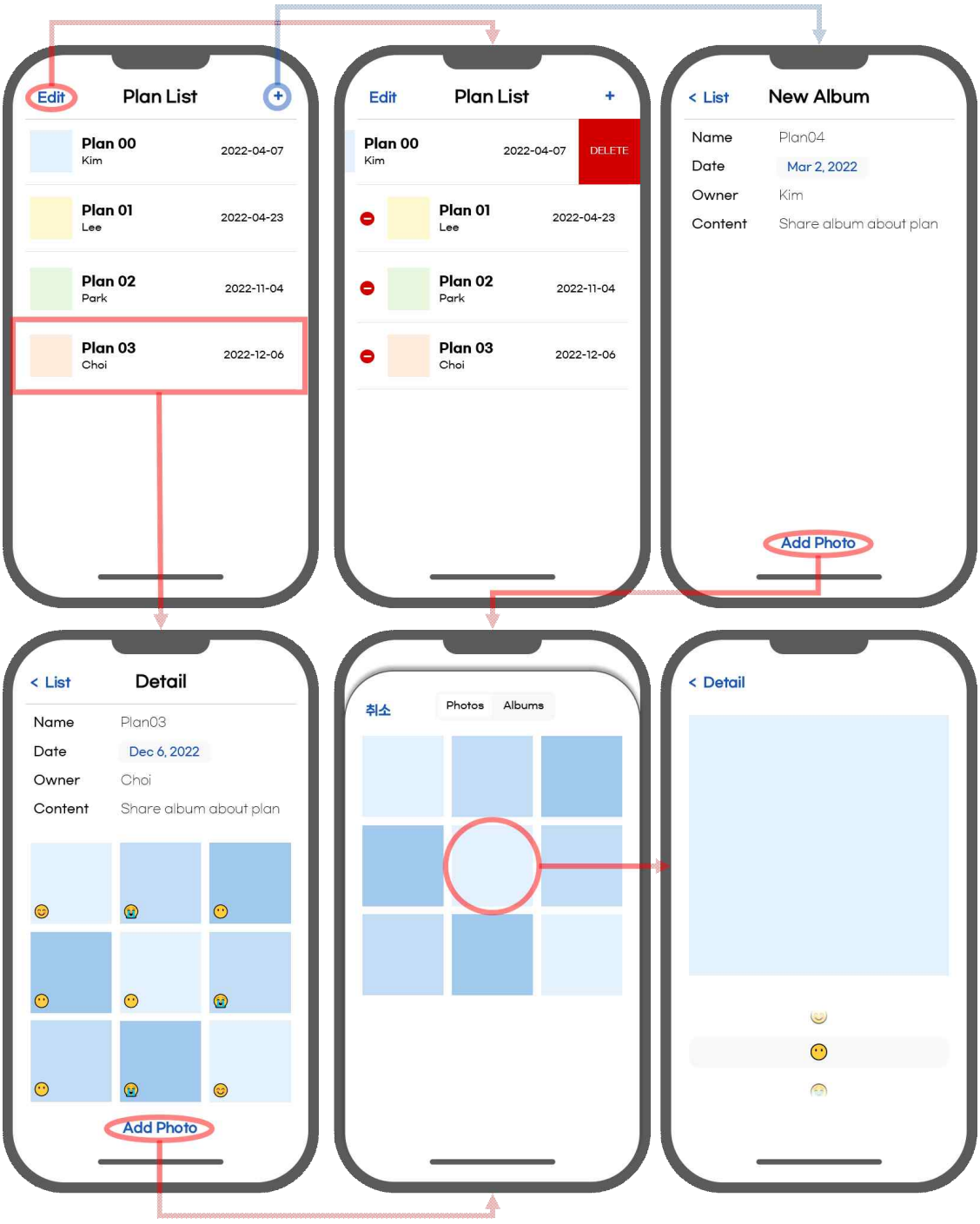
iOS 프로그래밍 Term 프로젝트 제안서

1. **학번 및 이름** : 1971082 김다연
2. **제목** : 일정 공유 앨범
3. **앱 개요 및 구조**

- 개요

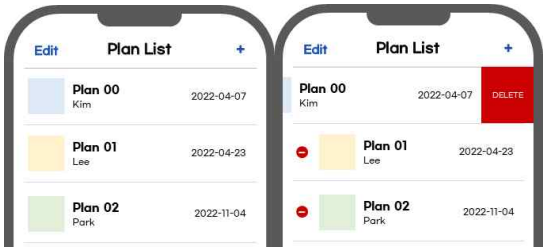
사진을 찍고 SNS에 이를 업로드하는 것이 일상화 되었다. 그러나, 여러 명이 찍은 사진을 공유하고 어떤 사진을 업로드 하는 것이 좋은지 의견을 나누는 과정은 시간이 오래 걸리고, 복잡하기 마련이다. 따라서 본 프로젝트는 일정별로 앨범을 만들고, 이를 다수의 사람이 공유하여 사진에 좋아요, 싫어요, 보통이에요 라는 표현을 할 수 있는 공유 앨범 어플리케이션을 개발하고자 한다.

- 구조



4. **앱 기능**

1) 초기화면



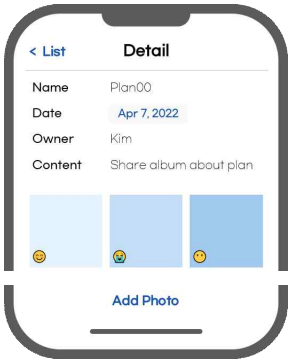
초기화면 Edit 버튼을 눌렀을 때

- 일정별로 생성된 앨범의 목록을 확인 할 수 있다.
- 썸네일 이미지, 앨범 이름, 앨범 생성자, 앨범 생성일을 확인할 수 있다.
- Edit 버튼을 눌러 앨범을 삭제할 수 있다.
- + 버튼을 눌러 새로운 공유 앨범을 생성할 수 있다.

2) 새로운 일정 공유 앨범 생성

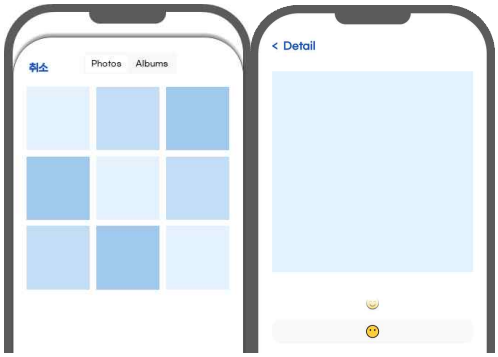
- 앞선 초기화면에서 + 버튼을 눌렀을 때 진입하는 화면이다.
- 일정과 관련된 정보들을 수정한다.
  * Name : 일정의 이름
  * Date : 일정의 일시 (날짜)
  * Owner : 일정을 생성하는 사람 (수정할 수 없다.)
  * Content : 일정에 대한 설명
- Add Photo 버튼을 눌러 앨범에 사진을 추가할 수 있다.
- " < List " 버튼을 눌러 초기화면으로 돌아갈 수 있다.

3) 앨범 상세 정보 확인



- 앞선 초기화면에서 특정 일정 공유 앨범 선택 시의 화면이다.
- 일정과 관련된 정보들을 수정할 수 있다.
  * Name : 일정의 이름
  * Date : 일정의 일시 (날짜)
  * Owner : 일정을 생성하는 사람 (수정할 수 없다.)
  * Content : 일정에 대한 설명
- 앨범내의 사진 목록과 해당 사진에 대한 표현을 확인할 수 있다.
- Add Photo 버튼을 눌러 앨범에 사진을 추가할 수 있다.
- " < List " 버튼을 눌러 초기화면으로 돌아갈 수 있다.

4) 공유 앨범에 사진 추가



사용자의 local 앨범에서 사진 선택 선택한 사진에 대한 표현 선택

- 앨범 생성 화면, 혹은 앨범 상세 정보 확인 화면에서 Add Photo 버튼을 눌렀을 때의 화면이다.
- 사용자의 local 앨범에서 추가할 사진을 선택한다. (취소 버튼을 눌러 사진 추가를 취소할 수 있다.)
- 추가할 사진을 선택하면 좋아요, 보통이에요, 싫어요의 표현을 선택할 수 있다.
- " < List " 버튼을 눌러 초기화면으로 돌아갈 수 있다.

5. 구현 내용

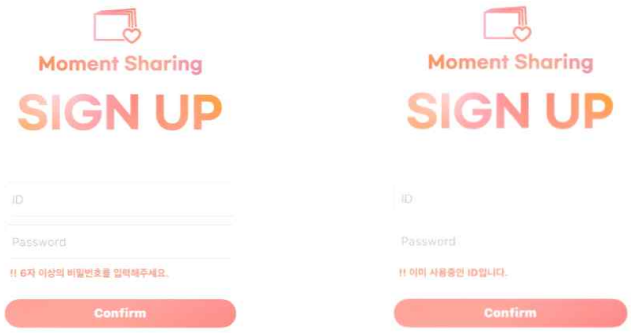
1) 로그인 및 회원 가입

로그인과 회원 가입은 Firebase Auth의 기능을 활용한다. 로그인 화면에는 로그인 버튼과 회원 가입 버튼이 있으며, 회원 가입 버튼 클릭 시 다음 코드를 통해 기존 LoginViewController에서 SignupViewController로 전이한다.

```
@IBAction func signUp(_ sender: UIButton) {
    guard let secondViewController
        = self.storyboard?.instantiateViewController(withIdentifier: "Signup") as? SignupViewController else { return }
    secondViewController.modalTransitionStyle = .crossDissolve      // 화면 전환 애니메이션 설정
    secondViewController.modalPresentationStyle = .fullScreen        // 화면이 보여지는 방법 (fullScreen)
    self.present(secondViewController, animated: true, completion: nil)
}
```

회원 가입시, 사용자가 입력한 ID 뒤에 @test.app을 붙여 이메일 형식으로 ID를 생성한다. 비밀번호가 6자 미만이거나, 이미 사용중인 ID인 경우 에러메세지를 화면에 표시한다. 성공적으로 회원 가입에 성공하면 이전의 LoginViewController로 전이한다.

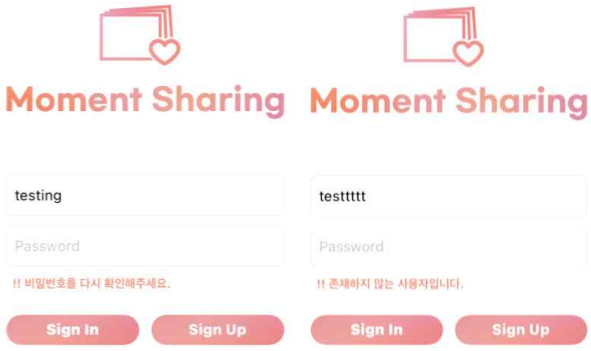
```
@IBAction func confirm(_ sender: UIButton) {
        if let id = IDTextField.text, let passwd = PWTextField.text{
            let email = id + "@test.app"
            Auth.auth().createUser(withEmail: email, password: passwd) { result, error in
                if let error = error as NSError? {
                    let errorCode = AuthErrorCode(_nsError: error)
                    switch errorCode.code {
                    case .invalidEmail:         self.errorLabel.text = "!! ID는 영어와 숫자만 입력할 수 있습니다."
                    case .accountExistsWithDifferentCredential, .credentialAlreadyInUse, .emailAlreadyInUse:
                        self.errorLabel.text = "!! 이미 사용중인 ID입니다."
                    case .weakPassword:       self.errorLabel.text = "!! 6자 이상의 비밀번호를 입력해주세요."
                    default:                  print("Create User Error: ₩(error)")
                    }
                    self.IDTextField.text = ""
                    self.PWTextField.text = ""
                    self.errorLabel.isHidden = false
                    self.btnConstraint.constant = 40
                    return
                }
                print("Reg Success")
                self.presentingViewController?.dismiss(animated: true)
            }
        }
    }
```



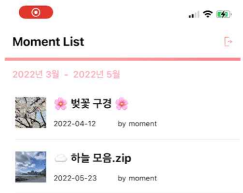
로그인도 회원 가입과 동일하게, 사용자가 입력한 ID 뒤에 @test.app을 붙여 이메일 형식으로 로그인한다. 등록되지 않은 사용자이거나 비밀번호가 맞지 않는 경우 에러메세지를 화면에 표시한다. 로그인에 성공한 경우 PlanGroupViewController로 전이한다.

```
@IBAction func signIn(_ sender: UIButton) {
        if let id = IDTextField.text, let passwd = PWTextField.text{
            let email = id + "@test.app"

            Auth.auth().signIn(withEmail: email, password: passwd) { result, error in
                if let error = error as NSError? {
                    let errorCode = AuthErrorCode(_nsError: error)
                    switch errorCode.code {
                        case .userNotFound, .invalidEmail:
```

```
                        self.errorLabel.text = "!! 존재하지 않는 사용자입니다."
                    case .wrongPassword:
                        self.errorLabel.text = "!! 비밀번호를 다시 확인해주세요."
                        self.IDTextField.text = id
                    default:
                        self.errorLabel.text = "!! 에러가 발생했습니다. 다시 시도해주세요."
                }
                self.PWTextField.text = ""
                self.errorLabel.isHidden = false
                self.btnStackViewConstraint.constant = 40
                return
            }
            print("Login Success")
            let main = UIStoryboard.init(name: "Main", bundle: nil)
            guard let navigationController
             = main.instantiateViewController(withIdentifier: "NavigationController")as? UINavigationController
                else {return}

            navigationController.modalPresentationStyle = .fullScreen
            self.present(navigationController, animated: true, completion: nil)
        }
    }
}
```



2) 일정별 앨범 관리



PlanGroupViewController의 기능은 크게 4가지로 구성된다.

- 로그아웃
- 일정 필터링
- 앨범 정보 표시
- 새로운 일정 정보 생성

우측 상단에 있는 로그아웃 버튼 클릭시, 현재 로그인 한 사용자에서 로그아웃하며 이전의 LoginViewController로 전이한다.

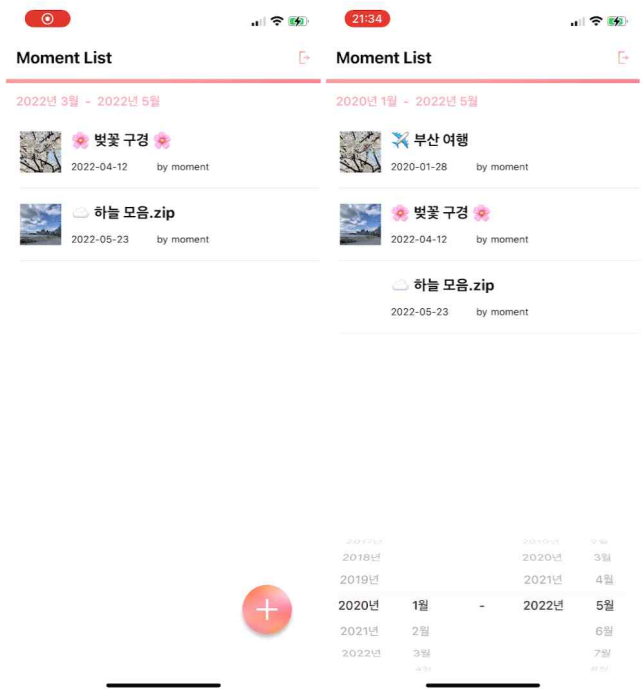
```
@objc func logout(){
    do {
        try Auth.auth().signOut()
        let login = UIStoryboard.init(name: "Login", bundle: nil)
        guard let loginController = login.instantiateViewController(withIdentifier: "LoginController") as?
                                    LoginViewController else {return}
        loginController.modalPresentationStyle = .fullScreen
        self.present(loginController, animated: false, completion: nil)
    }
    catch {
        print(error)
    }
}
```

Navigation View Controller의 title 아래에 위치한 날짜를 클릭하면 연월 선택 화면이 나타난다. 날짜는 UITextField를 이용하였으며, UITextField의 input을 UIPickerView로 설정해

UITextField 클릭 시 UIPickerView가 나타나도록 구현하였다. UIDatePickerView에는 연월만 선택하는 기능이 없기 때문에, UIPickerView의 component를 5로 설정하여 각각 시작 년도, 시작 월, 구분선(-), 끝나는 년도, 끝나는 월을 나타내도록 설정하였다. UIPickerView의 내용이 변경될 때 마다 queryPlan()을 호출해 해당 날짜에 해당하는 일정 앨범들만 화면에 출력한다.

```
// for datePicker(pickerView) - textField
extension PlanGroupViewController: UIPickerViewDelegate, UIPickerViewDataSource {
    func configureDatePicker(){
        datePicker.delegate = self
        datePicker.dataSource = self
        datePicker.backgroundColor = .white
        dateTextField.inputView = datePicker
        dateTextField.tintColor = .clear
        setAvailableDate()
    }
    func numberOfComponents(in pickerView: UIPickerView) -> Int { return 5 }
    func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
        switch component {
        case 0:     return availableYear.count
        case 1:     return allMonth.count
        case 2:     return 1
        case 3:     return availableYear.count
        case 4:     return allMonth.count
        default:    return 0
        }
    }
    func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
        switch component {
        case 0:     return String(availableYear[row]) + "년"
        case 1:     return String(allMonth[row]) + "월"
        case 2:     return "-"
        case 3:     return String(availableYear[row]) + "년"
        case 4:     return String(allMonth[row]) + "월"
        default:    return ""
        }
    }
    func pickerView(_ pickerView: UIPickerView, didSelectRow row: Int, inComponent component: Int) {
        switch component {
        case 0:     selectedStartYear = availableYear[row]
        case 1:     selectedStartMonth = allMonth[row]
        case 3:     selectedEndYear = availableYear[row]
        case 4:     selectedEndMonth = allMonth[row]
        default:    break
        }
        // 미래 연월 방지
        if (Int(todayYear) == selectedStartYear && Int(todayMonth)! < selectedStartMonth) {
            pickerView.selectRow(Int(todayMonth)!-1, inComponent: 1, animated: true)
            selectedStartMonth = Int(todayMonth)!
        }
        if (Int(todayYear) == selectedEndYear && Int(todayMonth)! < selectedEndMonth) {
            pickerView.selectRow(Int(todayMonth)!-1, inComponent: 4, animated: true)
            selectedEndMonth = Int(todayMonth)!
        }
        // 미래 < 과거 방지 : 년
        if (selectedStartYear > selectedEndYear) {
            pickerView.selectRow(selectedEndYear - 2000, inComponent: 0, animated: true)
            selectedStartYear = selectedEndYear
            // 월 다시 확인
            if (self.datePicker.selectedRow(inComponent: 1) > self.datePicker.selectedRow(inComponent: 4)) {
                pickerView.selectRow(selectedEndMonth - 1, inComponent: 1, animated: true)
                selectedStartMonth = selectedEndMonth
            }
        }
        // 미래 < 과거 방지 : 월
        if (selectedStartYear == selectedEndYear && selectedStartMonth > selectedEndMonth) {
            pickerView.selectRow(selectedEndMonth - 1, inComponent: 1, animated: true)
            selectedStartMonth = selectedEndMonth
        }
         dateTextField.text =
                "₩(selectedStartYear)년 ₩(selectedStartMonth)월  -  ₩(selectedEndYear)년 ₩(selectedEndMonth)월"
        let startdate = stringToDate(string: "₩(selectedStartYear)년 ₩(selectedStartMonth)월 1일")
        let enddate = stringToDate(string:"₩(selectedEndYear)년 ₩(selectedEndMonth)월 1일")
        planGroup.queryPlan(from: startdate!, to:enddate!)
    }

  func pickerView(_ pickerView: UIPickerView, viewForRow row: Int, forComponent component: Int, reusing view: UIView
```

```
?) -> UIView {
        let view = UIView(frame: CGRect(x:0, y:0, width: 100, height: 60))
        let subLabel = UILabel(frame: CGRect(x: 0, y: 0, width: 100, height: 60))
        subLabel.textAlignment = .center
        subLabel.font = UIFont.systemFont(ofSize: 15, weight: .medium)
        switch component{
        case 0:      subLabel.text = String(availableYear[row]) + "년"
        case 1:      subLabel.text = String(allMonth[row]) + "월"
        case 2:      subLabel.text = "-"
        case 3:      subLabel.text = String(availableYear[row]) + "년"
        case 4:      subLabel.text = String(allMonth[row]) + "월"
        default:     subLabel.text = ""
        }
        view.addSubview(subLabel)
        return view
    }
    func setAvailableDate() {
        let formatterYear = DateFormatter()
        formatterYear.dateFormat = "yyyy"
        todayYear = formatterYear.string(from: Date())
        formatterYear.dateFormat="MM"
        todayMonth = formatterYear.string(from: Date())
        for i in 2000...Int(todayYear)! { availableYear.append(i) }
        selectedStartYear = Int(todayYear)!
        selectedStartMonth = Int(todayMonth)! - 2 > 0  ? Int(todayMonth)! - 2 : 1
        selectedEndYear = Int(todayYear)!
        selectedEndMonth = Int(todayMonth)!
        dateTextField.text =
                "₩(selectedStartYear)년 ₩(selectedStartMonth)월  -  ₩(selectedEndYear)년 ₩(selectedEndMonth)월"
        // pickerView 미리 변경
        datePicker.selectRow(selectedStartYear - 2000, inComponent: 0, animated: false)
        datePicker.selectRow(selectedStartMonth - 1, inComponent: 1, animated: false)
        datePicker.selectRow(selectedEndYear - 2000, inComponent: 3, animated: false)
        datePicker.selectRow(selectedEndMonth - 1, inComponent: 4, animated: false)
    }
    func dateToString(date: Date) -> String! {
        let formmater = DateFormatter()
        formmater.dateFormat = "yyyy년 MM월 dd일"
        formmater.locale = Locale(identifier: "ko_KR")
        return formmater.string(from: date)
    }
    func stringToDate(string: String) -> Date! {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy년 MM월 dd일"
        dateFormatter.locale = Locale(identifier: "ko_KR")
        return dateFormatter.date(from: string)
    }
}
```



앨범 리스트를 표시하는 부분은 UITableView를 이용하였고, 상세 정보를 표시하는 부분은 UITableViewCell을 custom하여 구현하였다. 앨범의 내용으로는 plan.name에 해당하는 앨범 이름, plan.date에 해당하는 일시, plan.owner에 해당하는 앨범 소유자, plan.album의 첫 요소에 해당하는 썸네일이 있다. 특히 썸네일의 경우, [사진이 저장된 경로 String : 감정표시 Index]의 형태로 저장되는 plan.album의 첫 요소를 가져와 이 사진이 저장된 경로를 바탕으로 Firebase Storage에서 이미지를 가져온다. 각 요소를 클릭하면 해당 앨범에 해당하는 정보와 함께 PlanDetailViewController로 전이한다.

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        //let cell = UITableViewCell(style: .value1, reuseIdentifier: "") // TableViewCell을 생성한다
        let cell = tableView.dequeueReusableCell(withIdentifier: "PlainTableViewCell")

        // planGroup는 대략 1개월의 플랜을 가지고 있다.
        let plan = planGroup.getPlans()[indexPath.row] // Date를 주지않으면 전체 plan을 가지고 온다

        // 0, 1, 2순서가 맞아야 한다. 안맞으면 스트로보드에서 다시 맞도록 위치를 바꾸어야 한다
        // Thumbnail img는 앨범에 사진이 있으면 앨범의 첫번째 사진으로,
        // 앨범에 사진이 없거나 불러올 수 없는 상태이면 defaultImg로 설정된다.
        let ref = Storage.storage().reference().child(plan.key);
        ref.listAll { (result, error) in
            if let error = error {
                print(error)
                (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(named: "logo.png")
            }
            else {
                let item = result!.items[safe: 0]
                if item != nil {
                    item!.getData(maxSize: 1*1024*1024) { [self] data, error in
                        if let error = error {
                            print(error)
                            (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(named: "logo.png")
                        }
                        else {
                            (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(data: data!)!
                        }
                    }
                }
                else {
                    (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(named: "logo.png")
                }
            }
        }
        (cell?.contentView.subviews[1] as! UILabel).text = plan.name
        (cell?.contentView.subviews[2] as! UILabel).text = plan.owner
        (cell?.contentView.subviews[4] as! UILabel).text = plan.date.toStringDate()

        return cell!
    }
```

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {

        //let cell = UITableViewCell(style: .value1, reuseIdentifier: "") // TableViewCell을 생성한다
        let cell = tableView.dequeueReusableCell(withIdentifier: "PlainTableViewCell")

        // planGroup는 대략 1개월의 플랜을 가지고 있다.
        let plan = planGroup.getPlans()[indexPath.row] // Date를 주지않으면 전체 plan을 가지고 온다

        // 0, 1, 2순서가 맞아야 한다. 안맞으면 스트로보드에서 다시 맞도록 위치를 바꾸어야 한다
        // Thumbnail img는 앨범에 사진이 있으면 앨범의 첫번째 사진으로,
        // 앨범에 사진이 없거나 불러올 수 없는 상태이면 defaultImg로 설정된다.
        let ref = Storage.storage().reference().child(plan.key);
        ref.listAll { (result, error) in
            if let error = error {
                print(error)
                (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(named: "logo.png")
            }
            else {
                let item = result!.items[safe: 0]
                if item != nil {
                    item!.getData(maxSize: 1*1024*1024) { [self] data, error in
                        if let error = error {
                            print(error)
                            (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(named: "logo.png")
```

```
                        }
                        else {
                            (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(data: data!)!
                        }
                    }
                }
                else {
                    (cell?.contentView.subviews[0] as! UIImageView).image = UIImage(named: "logo.png")
                }
            }
        }


        (cell?.contentView.subviews[1] as! UILabel).text = plan.name
        (cell?.contentView.subviews[2] as! UILabel).text = plan.owner
        (cell?.contentView.subviews[4] as! UILabel).text = plan.date.toStringDate()

        return cell!
    }
```

```
func tableView(_ tableView: UITableView, moveRowAt sourceIndexPath: IndexPath, to destinationIndexPath: IndexPath) {
        // 이것은 데이터베이스에 까지 영향을 미치지 않는다. 그래서 planGroup에서만 위치 변경
        let from = planGroup.getPlans()[sourceIndexPath.row]
        let to = planGroup.getPlans()[destinationIndexPath.row]
        planGroup.changePlan(from: from, to: to)
        tableView.moveRow(at: sourceIndexPath, to: destinationIndexPath)
    }
```

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        print("prepare in pgc")
        if segue.identifier == "ShowPlan"{
            let planDetailViewController = segue.destination as! PlanDetailViewController
            // plan이 수정되면 이 saveChangeDelegate를 호출한다
            planDetailViewController.saveChangeDelegate = saveChange
            planDetailViewController.plan =
                        planGroup.getPlans()[planGroupTableView.indexPathForSelectedRow!.row].clone()
        }
        // 이하 생략
    }
```

우측 하단의 원형 + 버튼을 클릭하면 새로운 일정 앨범을 생성할 수 있다. 새로운 Plan 객체 newPlan을 하나 생성하여 newPlan의 정보와 함께 PlanDetailViewController로 전이한다.

```
@IBAction func addingPlans(_ sender: UIButton) { performSegue(withIdentifier: "AddPlan", sender: self) }
```

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // 생략
        if segue.identifier == "AddPlan"{
            print("AddPlan")
            let planDetailViewController = segue.destination as! PlanDetailViewController
            planDetailViewController.saveChangeDelegate = saveChange

            // 빈 plan을 생성하여 전달한다
            let newPlan =
                        Plan(date:Date(), owner: Auth.auth().currentUser!.email!.components(separatedBy: "@")[0],
                        withData: false)
            planDetailViewController.plan = newPlan
            planGroupTableView.selectRow(at: nil, animated: true, scrollPosition: .none)
        }
    }
```

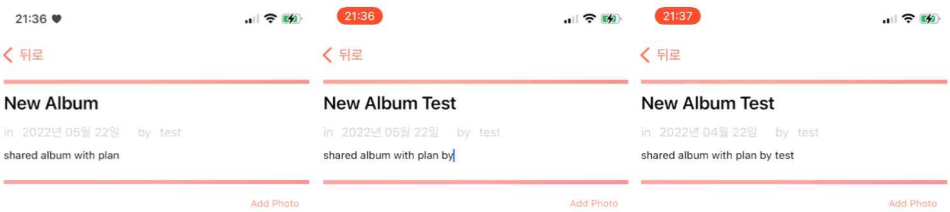
3) 새로운 일정 생성 및 일정 관리

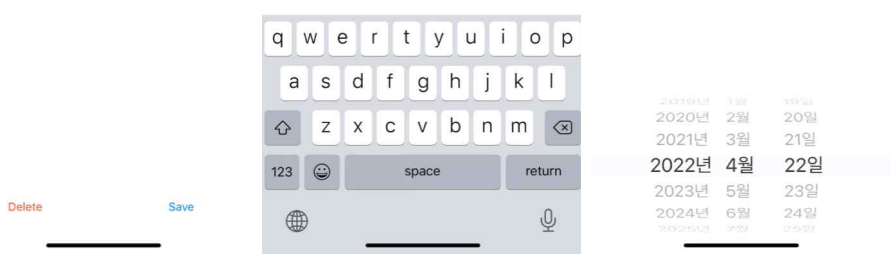
PlanDetailViewController에서는 앨범의 정보를 수정하거나 사진을 추가할 수 있으며, 앨범을

삭제할 수 있다. 그러나 앨범의 정보를 수정하거나 삭제하는 기능은 앨범의 소유자에게만 활성화된다. 앨범 소유자는 앨범의 이름 (plan.name), 앨범의 일시 (plan.date), 앨범에 대한 짧은 설명 (plan.content)를 수정할 수 있다. 또한 하단의 Save 버튼을 눌러 변경된 앨범의 정보를 Firebase DB에 저장할 수 있으며, Delete 버튼을 눌러 삭제할 수 있다.

앨범의 정보를 수정하는 부분은 UITextField와 UIDatePickerView를 이용하여 구현했다. 앨범의 이름과 설명은 UITextField로 수정하며, 앨범의 일시를 수정하는 부분 역시 UITextField로 구현되었으나 이전 PlanGroupViewController에서의 일정 필터링 기능과 동일한 방법으로 UITextField의 input을 UIDatePickerView로 주어 UITextField 클릭 시 UIDatePickerView가 나타나도록 설정했다. UIDatePickerView의 내용이 바뀌게 되면 UITextField의 내용 역시 즉시 변경된다.

```swift
// for datePicker - textField
extension PlanDetailViewController {
    func configureDatePicker(){
        self.datePicker.datePickerMode = .date
        self.datePicker.preferredDatePickerStyle = .wheels
        self.datePicker.backgroundColor = .white
        self.datePicker.setDate(plan!.date, animated: false)
        self.datePicker.addTarget(self, action: #selector(datePickerValueDidChanged(_:)), for: .valueChanged)
        self.dateTextField.inputView = self.datePicker
        self.dateTextField.tintColor = .clear
    }
    @objc func datePickerValueDidChanged(_ datePicker: UIDatePicker){
        let formmater = DateFormatter()
        formmater.dateFormat = "yyyy년 MM월 dd일"
        formmater.locale = Locale(identifier: "ko_KR")
        self.planDate = datePicker.date
        self.datePicker.locale = Locale(identifier: "ko-KR")
        self.dateTextField.text = formmater.string(from: datePicker.date)
        planDate = datePicker.date
    }
    func dateToString(date: Date) -> String! {
        let formmater = DateFormatter()
        formmater.dateFormat = "yyyy년 MM월 dd일"
        formmater.locale = Locale(identifier: "ko_KR")
        return formmater.string(from: date)
    }
    func stringToDate(string: String) -> Date! {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "yyyy년 MM월 dd일"
        dateFormatter.locale = Locale(identifier: "ko_KR")
        return dateFormatter.date(from: string)
    }
}
```

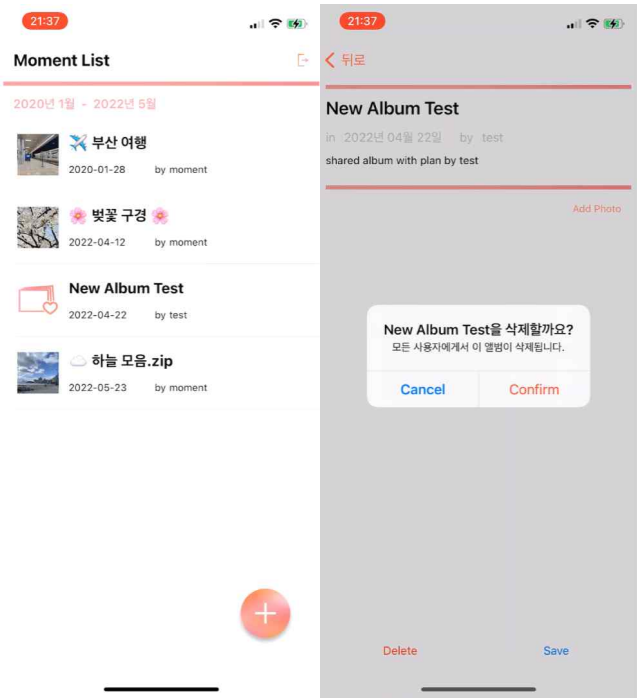




하단의 Save 버튼을 클릭하면 saveChangeDelegate 함수를 이용해 변경된 내용을 저장한다. 이 때, plan.album의 내용은 저장하지 않는다. Delete 버튼을 클릭한 경우, 사용자에게 이 앨범을 정말 삭제할 것인지 Alert 창을 통해 재확인한다. 이 Alert에서 사용자가 Confirm 버튼을 클릭하면 Firebase DB에서 해당 앨범이 삭제된다. Save와 Delete 버튼 모두 작업이 완료되면 popUpViewController() 함수를 통해 이전 PlanGroupViewController로 전이한다.

```
@IBAction func deletePlan(_ sender: Any) {
    let title = "₩(planName.text!)을 삭제할까요?"
    let message = "모든 사용자에게서 이 앨범이 삭제됩니다."
    let alertController = UIAlertController(title: title, message: message, preferredStyle: .alert)
    let cancelAction = UIAlertAction(title: "Cancel", style: .cancel, handler: nil)
    let deleteAction = UIAlertAction(title: "Confirm", style: .destructive, handler: {
        [self] (action:UIAlertAction) -> Void in
            let pgvc = self.navigationController?.children[0] as! PlanGroupViewController
            pgvc.planGroup.saveChange(plan: plan!, action: .Delete)
            self.navigationController?.popViewController(animated: true)
    })
    alertController.addAction(cancelAction)
    alertController.addAction(deleteAction)
    return present(alertController, animated: true)
}
```

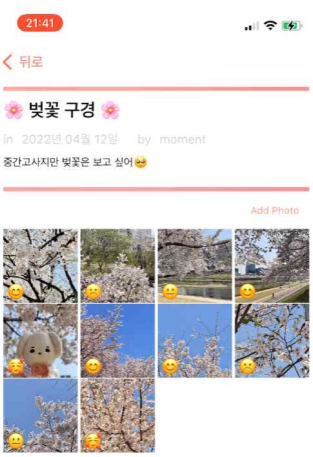
```
// saveBtn
@IBAction func gotoBack(_ sender: UIButton) {
    if let saveChangeDelegate = saveChangeDelegate{
        plan!.name = planName.text
        plan!.date = planDate!
        plan!.owner = ownerLabel.text
        plan!.content = contentTextField.text
        saveChangeDelegate(plan!)
    }
    navigationController?.popViewController(animated: true)
}
```

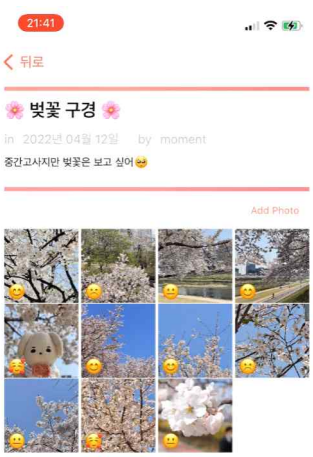


중간의 Add Photo 버튼을 클릭하면 UIImagePickerController로 전이한다. 사용자는 자신의 Local 앨범에서 사진을 선택하여 본 프로젝트의 일정 앨범에 저장할 수 있다. 사용자가 Local 앨범에서 사진을 선택하면, Firebase Storage의 plan.key로 된 폴더에 해당 이미지를 저장한다. 이후 plan.album에 이미지의 reference 경로와 함께 감정 Index의 default 값인 2를 저장한다. 변경된 plan 객체의 내용을 saveDelegate() 함수를 통해 Firebase DB에 저장한다.

```
extension PlanDetailViewController: UIImagePickerControllerDelegate, UINavigationControllerDelegate{
    // local album에서 사진 가져와 저장
    @IBAction func addPhoto(_ sender: UIButton) {
```

```
        let pickerViewController = UIImagePickerController()
        pickerViewController.sourceType = .photoLibrary
        pickerViewController.delegate = self
        present(pickerViewController, animated: true, completion: nil)
    }
    func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
        picker.dismiss(animated: true, completion: nil)
    }
    func imagePickerController
    (_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {
        // 이미지를 가져온다
        let image = info[UIImagePickerController.InfoKey.originalImage] as! UIImage
        picker.dismiss(animated: true, completion: nil)
        var data = Data()        // 앨범에 저장을 요청하면서 저장 완료에 대한 handler를 제공한다
        data = image.jpegData(compressionQuality: 0.8)!
        let metaData = StorageMetadata()
        metaData.contentType = "image/png"
        if let filePath = plan?.key {
            let fileName = "/"+random(15)+".png"
            storage.reference().child(filePath+fileName).putData(data, metadata: metaData){
                (metaData, error) in if let error = error {
                    print(error)
                    return
                }else{
                    let albumKey = self.fireStoreID + filePath + fileName
                    let storageRef = self.storage.reference(forURL: albumKey)
                    self.plan!.album[albumKey] = 2
                    self.saveChangeDelegate?(self.plan!)
                    self.afterSaveImage(image)
                }
            }
        }
    }
    func afterSaveImage(_ image: UIImage) {
        // plan의 key를 이용해 해당 폴더 내의 이미지들을 가져와 imgList에 저장한다.
        refList = []
        let ref = storage.reference().child(plan!.key);
        ref.listAll { (result, error) in
            if let error = error {
                print(error)
            }
            else {
                for item in result!.items {
                    self.refList.append(item)
                    self.collectionView.reloadData()
                }
            }
        }
    }
}
```





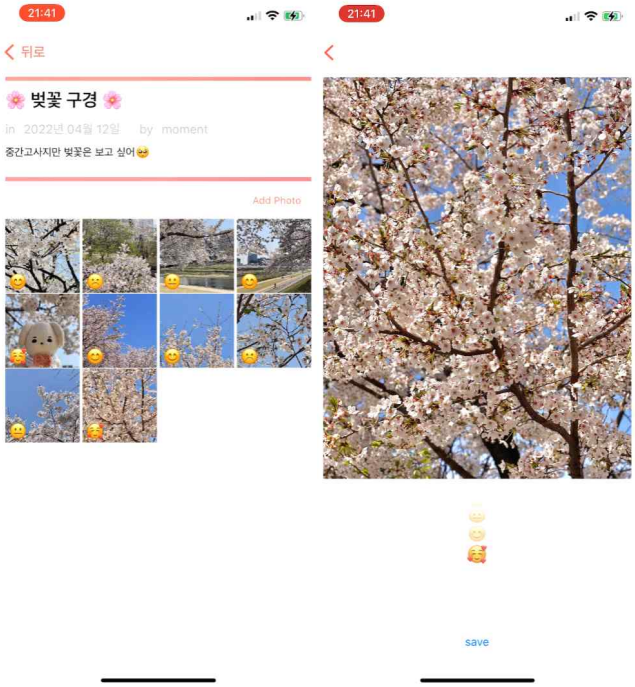
4) 앨범 정보 확인 및 사진에 감정 표시

일정 앨범에서 사진을 확인하는 부분은 UICollectionView로 구현했다. 각 CollectionViewCell은 이미지와 감정을 나타내는 UILabel로 구성되어 있다. CollectionView의 요소를 클릭하면 해당 이미지와 감정 Index번호와 함께 AlbumDetailViewController로 전이한다.

```
// for album
extension PlanDetailViewController: UICollectionViewDelegate, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout{
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        // 사진의 갯수를 리턴한다.
        return refList.count
    }

                                func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

                    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "ImageCollectionViewCell", for: indexPath) as! ImageCollectionViewCell
        // refList를 바탕으로 이미지를 가져온다.
        refList[indexPath.row].getData(maxSize: 1*1024*1024) { [self] data, error in
            if let error = error {
                print(error)
                cell.imageView.image = nil
            }
            else {
                cell.imageView.image = UIImage(data: data!)!
            }
        }

        let url = refList[indexPath.row].downloadURL(){ url, error in
            if let error = error {
                print(error)
            }
            else {
                let albumKey = self.fireStoreID +
                url!.absoluteString.components(separatedBy: ".com/o/")[1]
                .components(separatedBy: "?alt=")[0].replacingOccurrences(of: "%2F", with: "/")

                let index = try? self.plan?.album[albumKey]
                cell.emotion.text = self.emotionLists[index ?? 2]
                print("&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&")
                print(index, cell.emotion.text)

            }
        }
        return cell
    }

  func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {
            // CollectionView에 하나의 이미지의 크기를 리턴한다.
            // indexPath에 따라 크리를 조정하는 것도 가능하다.
        return CGSize(width: 90, height: 90)
    }
    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
        // 이 이미지를 클릭하면 자세히 보기로 전이한다. Send가 self가 아니고 클릭된 Cell의 indexPath이다.
        print("is clicked.")
        performSegue(withIdentifier: "album", sender: indexPath)
    }

    func setFlowLayout(){
        let flowLayout: UICollectionViewFlowLayout = UICollectionViewFlowLayout()
        flowLayout.minimumLineSpacing = 1
        flowLayout.minimumInteritemSpacing = 1
        let myWidth: CGFloat = self.collectionView.frame.width / 3
        flowLayout.itemSize = CGSize(width: myWidth, height: myWidth)
        self.collectionView.collectionViewLayout = flowLayout
    }
}
```

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "album" {
            let albumDetailViewController = segue.destination as! AlbumDetailViewController
```

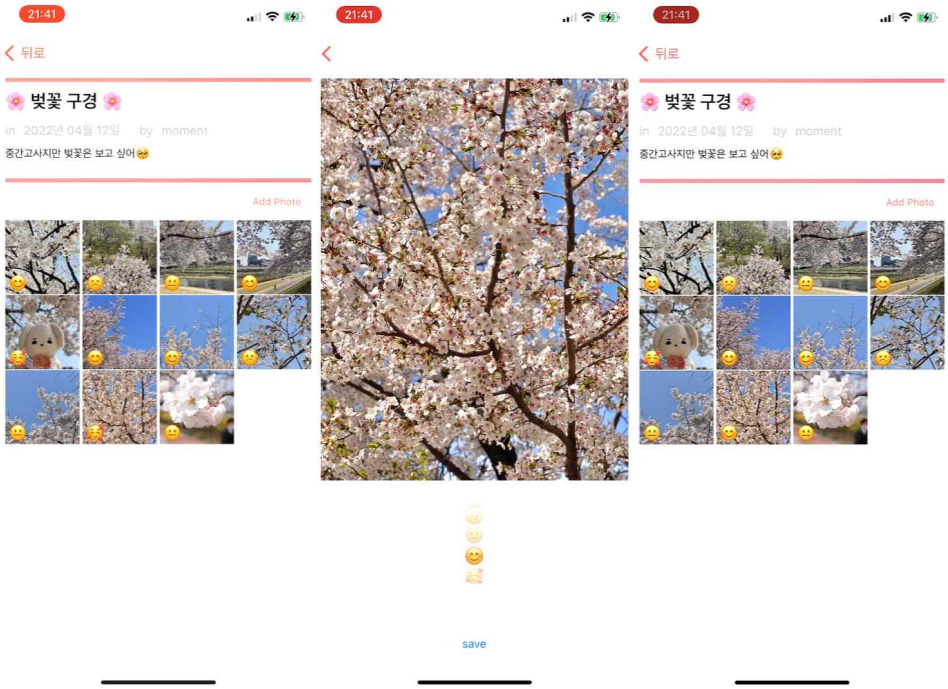
```
            // 이미지에 대한 정보를 가져온다
            let indexPath = sender as! IndexPath     // sender이 indexPath이다.
            var image: UIImage!
            var emotionIndex: Int!

            refList[indexPath.row].getData(maxSize: 1*1024*1024) { [self] data, error in
                if let error = error {
                    print(error)
                    image = nil
                }
                else {
                    image = UIImage(data: data!)!

                    let url = refList[indexPath.row].downloadURL(){ url, error in
                        if let error = error {
                            print(error)
                        }
                        else {
                            let albumKey = self.fireStoreID +
                            url!.absoluteString.components(separatedBy: ".com/o/")[1]
                            .components(separatedBy: "?alt=")[0].replacingOccurrences(of: "%2F", with: "/")
                            emotionIndex = try? self.plan?.album[albumKey]
                            print(emotionIndex)
                            albumDetailViewController.key = albumKey            // for return
                            albumDetailViewController.image = image             // albumDetailViewController이미지 변경
                            albumDetailViewController.emotionIndex = emotionIndex ?? 2
                            // albumDetailViewController의 emotion 변경
                        }
                    }
                }
            }
        }
    }
```



AlbumDetailViewController에서는 사진을 크게 확인할 수 있으며, UIPickerView를 이용해 5개의 감정 이모티콘 중 하나를 선택할 수 있다. 하단의 Save 버튼을 클릭하면 현재의 Controller를 종료 (Exit)하고, PlanDetailViewController의 unwind 함수를 실행한다. 이는 현재의 AlbumDetailViewController에서 이미지의 reference String과 감정 Index의 번호를 받아 plan.album의 정보를 업데이트하고, saveChangeDelegate() 함수를 이용해 Firebase DB에 변경 내용을 저장한다.

```
// AlbumDetailViewController로부터 emotionIndex 얻기
    @IBAction func unwind(sender: UIStoryboardSegue) {
        if let albumDetailViewController = sender.source as? AlbumDetailViewController {
            let key = albumDetailViewController.key
            let index = albumDetailViewController.emotionPickerView.selectedRow(inComponent: 0)
```

```
            plan?.album[key!] = index
            self.saveChangeDelegate?(self.plan!)
        }
    }
```



6. 결론

안드로이드나 웹 기반 서비스에 비해 Swift에 대한 자료는 많이 부족하다는 것을 알게 되었다. 자료가 부족하다보니 Firebase와 Swift의 공식 문서를 더욱 많이 참고하게 되었다. 특히 Firebase storage에서 이미지를 다운받아 가져오는 부분을 구현할 때, 그동안 잘 알지 못했던 후행 클로저에 대해 제대로 공부할 수 있게 되었다. Firebase storage에서 데이터를 가져올 때, Firestorage reference로 가져오기 때문에 이를 다시 data와 UIImage로 변환하는 과정이 필요하다. 데이터를 가져오는 과정에서 후행 클로저는 필수적이며, 이 내용은 언제 실행될지 정확하게 알 수 없기 때문에 에러처리를 확실하게 해야한다는 점 또한 알게 되었다.

처음 계획했던 기능 이상으로 구현해냈으나, 모든 사용자가 앨범에 접근하여 사진을 추가하고 감정을 표현할 수 있는 점은 보안상으로 문제가 있다고 생각한다. 앨범 소유자가 앨범에 비밀번호를 설정하거나, 다른 User들을 자신의 앨범에 초대하여 앨범에 대한 접근권한이 있는 User들만 사진을 추가하고 감정을 표현할 수 있는 기능을 넣으면 더 좋은 프로젝트 결과물이 될 것이라고 예상한다.