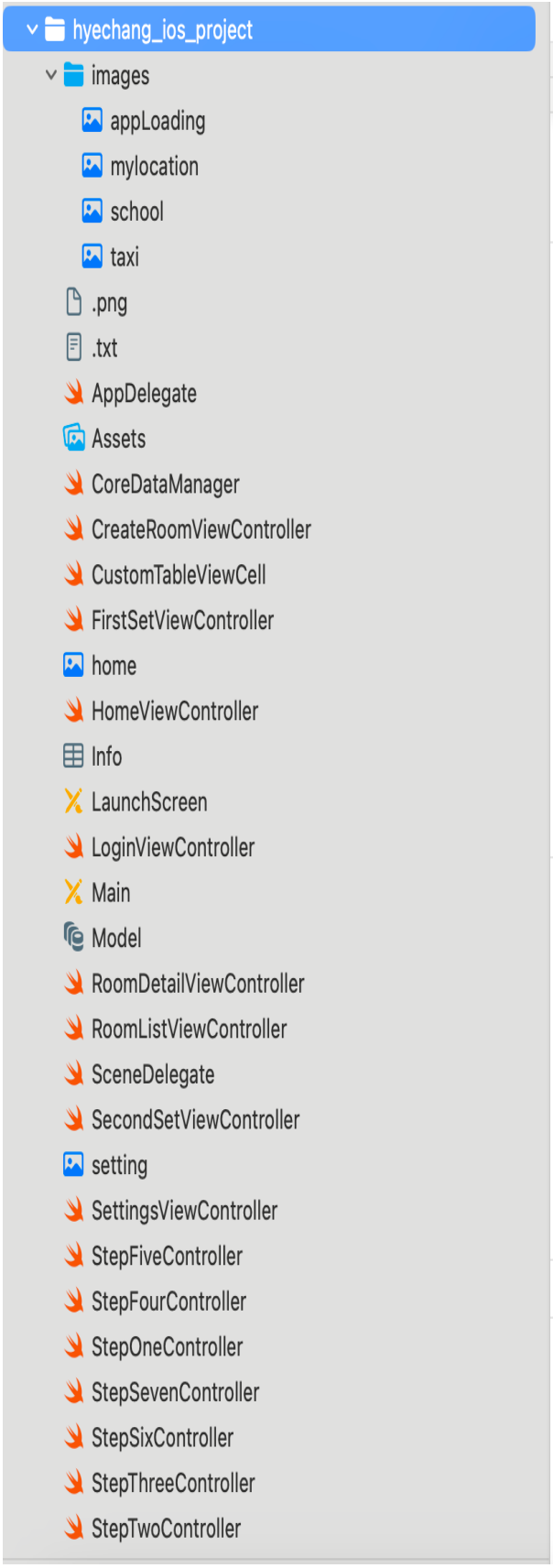
2071148 홍혜창 코드 설명서

프로젝트 파일 구조




```
hyechang_ios_project/
├── images/                        # 이미지 리소스
│   ├── appLoading.png             # 앱 로딩 화면
│   ├── mylocation.png             # 사용자 위치 아이콘
│   ├── school.png                 # 학교 마커 아이콘
│   └── taxi.png                   # 택시 마커 아이콘
├── AppDelegate.swift
├── CoreDataManager.swift          # Core Data 관리자
├── CustomTableViewCell.swift # 커스텀 테이블뷰 셀
├── FirstSetViewController.swift # 1 단계 설정
├── HomeViewController.swift   # 홈 지도 화면
├── LoginViewController.swift # 로그인 화면
├── Model.xcdatamodeld         # Core Data 모델
├── RoomDetailViewController.swift # 방 상세 화면
├── RoomListViewController.swift # 방 목록 화면
├── SecondSetViewController.swift # 2 단계 설정
├── SettingsViewController.swift # 설정 화면
├── CreateRoomViewController.swift # 방 생성 화면
├── StepOneController.swift    # 멤버 대기
├── StepTwoController.swift    # 집합 단계
├── StepThreeController.swift # 택시 검색
├── StepFourController.swift   # 택시 기사 매칭
├── StepFiveController.swift. # 매칭 완료
├── StepSixController.swift.   # 택시 탑승
├── StepSevenController.swift # 요금 정산
├── Main.storyboard            # UI 스토리보드
├── LaunchScreen.storyboard
└── Info.plist
```

Main.storyboard

사용한 라이브러리

iOS 기본 프레임워크

-UIKit: UI 구성 요소 전체

-MapKit: 지도 및 위치 기반 서비스

-CoreLocation: GPS 위치 추적

-CoreData: 로컬 데이터베이스 관리

외부 라이브러리

-**없음** - 순수 iOS SDK 만 사용

데이터베이스

Room

Attribute	Type
status	String
startLongitude	Double
startLocation	String
startLatitude	Double
roomID	String
ownerID	String
maxMembers	Integer 32
estimatedCost	Double
endLongitude	Double
endLocation	String
endLatitude	Double
currentMembers	Integer 32
costPerPerson	Integer 32

RoomMember

Attribute	Type
userID	String
roomID	String
isReady	Boolean

User

Attribute	Type
userID	String
university	String
password	String
name	String
isLocationActive	Boolean
currentLongitude	Double
currentLatitude	Double
balance	Integer 32

핵심코드

1. CoreDataManager

기능: Core Data 전체 관리 및 비즈니스 로직 처리 **조건**: 싱글톤 패턴으로 전체 데이터 일관성 보장

설명

Core Data 스택 관리, 더미 데이터 생성, 방 생성/입장, 위치 업데이트 등 모든 데이터 처리를 담당하는 핵심 클래스

핵심함수

```
// 방 생성 (CreateRoomViewController 에서 호출)
func createRoom(roomID: String, ownerID: String, startLocation: String,
                startLatitude: Double, startLongitude: Double,
                endLocation: String, endLatitude: Double, endLongitude: Double,
                maxMembers: Int, estimatedCost: Int, costPerPerson: Int) -> Bool {

    let room = Room(context: context)
    room.roomID = roomID
    room.ownerID = ownerID
    room.startLocation = startLocation
    room.startLatitude = startLatitude
    room.startLongitude = startLongitude
    room.endLocation = endLocation
    room.endLatitude = endLatitude
    room.endLongitude = endLongitude
    room.currentMembers = 1
    room.maxMembers = Int32(maxMembers)
    room.estimatedCost = Double(estimatedCost)
    room.costPerPerson = Int32(costPerPerson)
    room.status = "모집중"

    let roomMember = RoomMember(context: context)
    roomMember.roomID = roomID
    roomMember.userID = ownerID
    roomMember.isReady = false

    do {
        try context.save()
        return true
    } catch {
        return false
    }
}


// 실시간 위치 업데이트 (HomeViewController, StepTwoController 에서 호출)
func updateUserLocation(userID: String, latitude: Double, longitude: Double) -> Bool {
    guard let user = getUser(userID: userID) else { return false }

    if user.isLocationActive {
        user.currentLatitude = latitude
        user.currentLongitude = longitude

        do {
            try context.save()
            return true
        } catch {
            return false
        }
    }
    return false
}
```

2. LoginViewController

기능: 로그인 화면 **조건**: 더미 데이터 검증

바인딩한 UI 컴포넌트

```
@IBOutlet weak var userId: UITextField!          // 아이디 입력
@IBOutlet weak var userPwd: UITextField!         // 비밀번호 입력
@IBOutlet weak var errorMessage: UILabel!        // 에러 메시지
```

설명

appLoading.png 로딩 화면, Core Data 더미 데이터 초기화, UITextFieldDelegate

3. HomeViewController

기능: 메인 지도 화면, 방 목록 표시 및 입장 **조건**: 1km 이내 거리 + 예상비용 20% 추가잔액

바인딩한 UI 컴포넌트

```
@IBOutlet weak var mapView: MKMapView!           // 지도
@IBOutlet weak var bottomPopupView: UIView!      // 하단 팝업
@IBOutlet weak var enterRoomButton: UIButton!    // 입장 버튼
@IBOutlet weak var myBalance: UILabel!           // 잔액 표시
@IBOutlet weak var startLabel: UILabel!          // 출발지
@IBOutlet weak var endLabel: UILabel!            // 목적지
@IBOutlet weak var memberInfo: UILabel!          // 인원 정보
@IBOutlet weak var costInfo: UILabel!            // 비용 정보
@IBOutlet weak var errorMessage: UILabel!        // 에러 메시지
```

설명

MapKit 기반 실시간 지도, 커스텀 마커 표시, 방 정보 팝업, 거리 및 잔액 조건 검사를 통한 방 입장 관리

핵심함수

```
// 커스텀 마커 생성 (mylocation.png, taxi.png, school.png 사용)
func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView? {
    if let pointAnnotation = annotation as? MKPointAnnotation,
       pointAnnotation.title == "내 위치" {
        let identifier = "MyLocationAnnotation"
        var annotationView = mapView.dequeueReusableAnnotationView(withIdentifier: identifier)

        if annotationView == nil {
            annotationView = MKAnnotationView(annotation: annotation, reuseIdentifier: identifier)

            if let originalImage = UIImage(named: "mylocation") {
                let newSize = CGSize(width: 20, height: 20)
                let customImage = resizeImage(image: originalImage, targetSize: newSize)

                let circleView = UIView(frame: CGRect(x: 0, y: 0, width: 24, height: 24))
                circleView.backgroundColor = UIColor.systemBlue
                circleView.layer.cornerRadius = 12
                circleView.layer.borderWidth = 2
                circleView.layer.borderColor = UIColor.white.cgColor

                let imageView = UIImageView(image: customImage)
                imageView.frame = CGRect(x: 2, y: 2, width: 20, height: 20)
                imageView.tintColor = .white
                circleView.addSubview(imageView)

                UIGraphicsBeginImageContextWithOptions(circleView.bounds.size, false, 0)
                circleView.layer.render(in: UIGraphicsGetCurrentContext()!)
                let finalImage = UIGraphicsGetImageFromCurrentImageContext()
                UIGraphicsEndImageContext()

                annotationView?.image = finalImage
            }
        }
        return annotationView
    }
    return nil
}
```

4. RoomListViewController

기능: 테이블뷰 기반 방 목록 화면 **조건**: HomeViewController 와 동일 (1km + 20% 추가잔액)

바인딩한 UI 컴포넌트

```
@IBOutlet weak var tableView: UITableView!       // 방 목록 테이블
@IBOutlet weak var myBalance: UILabel!           // 잔액 표시
// CustomTableViewCell 과 연동
```

설명

UITableView + CustomTableViewCell 로 방 목록 표시, Timer 기반 실시간 업데이트, 거리 및 잔액 조건 검사

핵심함수

```
// 테이블뷰 셀 구성 (거리 및 잔액 조건 적용)
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "CustomTableViewCell", for: indexPath) as! CustomTableViewCell

    let room = availableRooms[indexPath.row]

    cell.startLabel?.text = room.startLocation
    cell.endLabel?.text = room.endLocation
    cell.costInfo?.text = "총 \(Int(room.estimatedCost))원 / 인당 \(Int(room.costPerPerson))원"
    cell.memberInfo?.text = "\(Int(room.currentMembers))/\(Int(room.maxMembers))명"

    if let currentLocation = currentLocation {
        let roomLocation = CLLocation(latitude: room.startLatitude, longitude: room.startLongitude)
        let userLocation = CLLocation(latitude: currentLocation.latitude, longitude: currentLocation.longitude)
        let distance = userLocation.distance(from: roomLocation)

        if distance > 1000 {
            cell.enterClick.isEnabled = false
            cell.setInactiveButtonStyle(title: "거리 초과", backgroundColor: .systemGray3)
            cell.errorMessage.text = "거리가 너무 멀어서 입장할 수 없습니다. (현재 거리: \(Int(distance))m)"
            cell.errorMessage.isHidden = false
        } else {
            cell.enterClick.isEnabled = true
            cell.setActiveButtonStyle()
            cell.errorMessage.isHidden = true
        }
    }

    cell.enterClick.tag = indexPath.row
    cell.enterClick.addTarget(self, action: #selector(enterButtonTapped(_:)), for: .touchUpInside)

    return cell
}
```

5. CustomTableViewCell

기능: 방 목록용 커스텀 테이블뷰 셀 **조건**: 부모 뷰컨트롤러의 조건에 따라 버튼 상태 변경

바인딩한 UI 컴포넌트

```
@IBOutlet weak var container: UIView!            // 메인 컨테이너
@IBOutlet weak var startLabel: UILabel!          // 출발지
@IBOutlet weak var endLabel: UILabel!            // 목적지
@IBOutlet weak var memberInfo: UILabel!          // 인원 정보
@IBOutlet weak var costInfo: UILabel!            // 비용 정보
@IBOutlet weak var enterClick: UIButton!         // 입장 버튼
@IBOutlet weak var errorMessage: UILabel!        // 에러 메시지
@IBOutlet weak var startAndEndLabel: UILabel!    // 출발지→목적지
```

설명

카드 형태 디자인, 그림자 효과, 동적 버튼 상태 변경, 에러 메시지 표시

6. CreateRoomViewController

기능: 방 생성 2 단계 프로세스 관리자 **조건**: 2 단계에서 잔액 20% 추가 검증

바인딩한 UI 컴포넌트

```
@IBOutlet weak var pageChangeView: UIView!       // 자식 뷰 컨테이너
@IBOutlet weak var pageControl: UIPageControl!   // 페이지 인디케이터
```

설명

UIPageControl + Container View 로 2 단계 프로세스, 자식 뷰컨트롤러 관리, 택시 요금 계산

핵심함수

```
// 방 생성 처리
func handleCreateRoom() {
    let roomID = "room_\(UUID().uuidString.prefix(8))"
    let success = CoreDataManager.shared.createRoom(
        roomID: roomID,
```

```
        ownerID: currentUserID,
        startLocation: startLocationAddress,
        startLatitude: selectedCoordinate!.latitude,
        startLongitude: selectedCoordinate!.longitude,
        endLocation: currentUserUniversity!,
        endLatitude: 37.58616528349631,
        endLongitude: 127.01280516488525,
        maxMembers: selectedMemberCount,
        estimatedCost: estimatedTotalCost,
        costPerPerson: estimatedCostPerPerson
    )

    if success {
        navigateToRoomDetail(roomID: roomID)
    }
}
```

7. FirstSetViewController

기능: 방 생성 1 단계 - 출발지 설정 **조건**: 현재 위치에서 1km 이내 출발지 선택

바인딩한 UI 컴포넌트

```
@IBOutlet weak var mapView: MKMapView!           // 지도
@IBOutlet weak var startLabel: UITextField!      // 출발지 입력
@IBOutlet weak var endLabel: UITextField!        // 목적지 (고정)
@IBOutlet weak var nextButton: UIButton!         // 다음 버튼
@IBOutlet weak var firstSetErrorMessage: UILabel! // 에러 메시지
```

설명

MapKit 지도 탭으로 출발지 선택, CLGeocoder 로 주소 변환, 1km 거리 제한

핵심함수

```
// 지도 탭으로 출발지 선택
@objc private func mapTapped(_ gesture: UITapGestureRecognizer) {
    if gesture.state == .ended {
        let touchPoint = gesture.location(in: mapView)
        let coordinate = mapView.convert(touchPoint, toCoordinateFrom: mapView)

        selectedCoordinate = coordinate
        updateMarkerAndUI(coordinate: coordinate)
        view.endEditing(true)
    }
}


// 주소 검색 (포워드 지오코딩)
private func forwardGeocode(address: String) {
    geocoder.geocodeAddressString(address) { [weak self] placemarks, error in
        DispatchQueue.main.async {
            if let placemark = placemarks?.first,
               let location = placemark.location {

                let coordinate = location.coordinate
                self?.selectedCoordinate = coordinate
                self?.removeStartLocationMarker()
                self?.addStartLocationMarker(at: coordinate)

                let region = MKCoordinateRegion(
                    center: coordinate,
                    latitudinalMeters: 1000,
                    longitudinalMeters: 1000
                )
                self?.mapView.setRegion(region, animated: true)
                self?.checkConditions()
            }
        }
    }
}
```

8. SecondSetViewController

기능: 방 생성 2 단계 - 인원 설정 및 잔액 검증 **조건**: 예상 비용의 20% 추가 잔액 필요

바인딩한 UI 컴포넌트

```
@IBOutlet weak var memberSetting: UISegmentedControl! // 2~4 명 선택
@IBOutlet weak var costLabel: UILabel!                // 총 비용
@IBOutlet weak var onePersonCost: UILabel!            // 인당 비용
```

```
@IBOutlet weak var createButton: UIButton!            // 방 만들기 버튼
@IBOutlet weak var secondSetErrorMessage: UILabel!    // 에러 메시지
```

설명

UISegmentedControl 로 인원 선택, 실시간 비용 계산, 잔액 조건 검사

핵심함수

```
// 인원 변경 시 비용 재계산
@IBAction func memberChange(_ sender: UISegmentedControl) {
    currentMemberCount = getCurrentMemberCount()
    parentCreateRoom?.handleMemberCountChange(currentMemberCount)
    updateCostDisplay()
    checkCreateButtonCondition()
}
```

9. RoomDetailViewController

기능: 방 상세 화면 - 7 단계 프로세스 관리 **조건**: 방장 권한, 각 단계별 조건

바인딩한 UI 컴포넌트

```
@IBOutlet weak var pageChangeView: UIView!       // 단계별 뷰 컨테이너
@IBOutlet weak var pageControl: UIPageControl!   // 페이지 인디케이터
@IBOutlet weak var myBalance: UILabel!           // 잔액 표시
```

설명

7 개 Step 컨트롤러 관리, 방장/멤버 권한 구분, 택시 이용 전체 프로세스 관리

핵심함수

```
// Step 간 전환 관리
func showStepTwo() {
    removeCurrentChild()

    let storyboard = UIStoryboard(name: "Main", bundle: nil)
    if let stepTwoVC = storyboard.instantiateViewController(withIdentifier: "StepTwoController") as? StepTwoController {
        stepTwoVC.parentRoomDetail = self
        addChildViewController(stepTwoVC, to: pageChangeView)
        stepTwoController = stepTwoVC

        currentStep = 2
        pageControl.currentPage = 1
        navigationItem.title = "출발지로 모여주세요"
    }
}
```

10. StepOneController

기능: 1 단계 - 멤버 대기 및 Ready 상태 관리 **조건**: 모든 멤버 Ready + 2 명 이상

바인딩한 UI 컴포넌트

```
@IBOutlet weak var dynamicContainer: UIView!     // 멤버 카드 컨테이너
@IBOutlet weak var startButton: UIButton!        // 시작 버튼
@IBOutlet weak var exitButton: UIButton!         // 나가기 버튼
@IBOutlet weak var startLabel: UILabel!          // 출발지
@IBOutlet weak var endLabel: UILabel!            // 목적지
@IBOutlet weak var personCostLabel: UILabel!     // 인당 비용
```

설명

동적 멤버 카드 생성, Timer 기반 시뮬레이션, Ready 상태 관리

핵심함수

```
// 동적 멤버 카드 생성 (UIStackView 기반)
```

```
private func createMemberCards(in container: UIView, maxMembers: Int) {
    let columns = 2
    let cardWidth: CGFloat = 70
    let cardHeight: CGFloat = 90
    let spacing: CGFloat = 6

    let mainStackView = UIStackView()
    mainStackView.axis = .vertical
    mainStackView.distribution = .fillEqually
    mainStackView.spacing = spacing
    mainStackView.translatesAutoresizingMaskIntoConstraints = false
    container.addSubview(mainStackView)

    let rows = (maxMembers + 1) / 2
    for row in 0..<rows {
        let rowStackView = UIStackView()
        rowStackView.axis = .horizontal
        rowStackView.distribution = .fillEqually
        rowStackView.spacing = spacing

        let startIndex = row * columns
        let endIndex = min(startIndex + columns, maxMembers)

        for i in startIndex..<endIndex {
            let memberCard = createSingleMemberCard(index: i)
            memberCards.append(memberCard)
            rowStackView.addArrangedSubview(memberCard)
        }

        mainStackView.addArrangedSubview(rowStackView)
    }
}
```

11. StepTwoController

기능: 2 단계 - 집합 단계 (가장 복잡)　　**조건**: 모든 멤버 100m 이내 + 방장만 택시 호출 가능

바인딩한 UI 컴포넌트

```
@IBOutlet weak var mapView: MKMapView!           // 실시간 위치 지도
@IBOutlet weak var realTimeCheckView: UIView!    // 멤버 상태 컨테이너
@IBOutlet weak var markerInfoView: UIView!       // 범례 뷰
@IBOutlet weak var callButton: UIButton!         // 택시 호출 버튼
@IBOutlet weak var notifyLabel: UILabel!         // 알림 라벨
```

설명

실시간 위치 추적, 100m 원형 오버레이, 멤버별 색상 구분 마커, 위치 시뮬레이션

핵심함수

```
// 100m 원형 오버레이 렌더링
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if let circle = overlay as? MKCircle {
        let renderer = MKCircleRenderer(circle: circle)
        renderer.strokeColor = UIColor.systemRed.withAlphaComponent(0.8)
        renderer.fillColor = UIColor.systemRed.withAlphaComponent(0.2)
        renderer.lineWidth = 2
        return renderer
    }
    return MKOverlayRenderer()
}

// 멤버별 색상 구분 마커 생성
private func createCustomMarkerImage(color: UIColor) -> UIImage {
    let size = CGSize(width: 20, height: 20)
    let renderer = UIGraphicsImageRenderer(size: size)

    return renderer.image { context in
        color.setFill()
        let rect = CGRect(origin: .zero, size: size)
        let path = UIBezierPath(ovalIn: rect)
        path.fill()

        UIColor.white.setStroke()
        path.lineWidth = 2
        path.stroke()
    }
}
```

12. StepThreeController

기능: 3 단계 - 택시 검색　　**조건**: 자동 진행

바인딩한 UI 컴포넌트

```
@IBOutlet weak var indicator: UIActivityIndicatorView! // 로딩 인디케이터
@IBOutlet weak var lookingView: UIView!                // 검색 뷰
@IBOutlet weak var startAndEndLabel: UILabel!          // 출발지→목적지
```

설명

택시 검색 애니메이션, 맥박 효과, 자동 Step4 이동

13. StepFourController

기능: 4 단계 - 택시 기사 매칭 **조건**: 자동 진행

바인딩한 UI 컴포넌트

```
@IBOutlet weak var indicator: UIActivityIndicatorView! // 로딩 인디케이터
@IBOutlet weak var matchingView: UIView!               // 매칭 뷰
@IBOutlet weak var startAndEndLabel: UILabel!          // 출발지→목적지
```

설명

기사 매칭 애니메이션, 회전 효과, 자동 Step5 이동

14. StepFiveController

기능: 5 단계 - 매칭 완료 **조건**: 수동 진행 (탑승 시작 버튼)

바인딩한 UI 컴포넌트

```
@IBOutlet weak var taxiDriverView: UIView!       // 택시 기사 뷰
@IBOutlet weak var startAndEndLabel: UILabel!    // 출발지→목적지
@IBOutlet weak var startButton: UIButton!        // 탑승 시작 버튼
```

설명

택시 기사 정보 표시, 탑승 시작 버튼으로 수동 진행

15. StepSixController

기능: 6 단계 - 택시 탑승 중 **조건**: 자동 진행

바인딩한 UI 컴포넌트

```
@IBOutlet weak var movingView: UIView!           // 이동 뷰
@IBOutlet weak var indicator: UIActivityIndicatorView! // 로딩 인디케이터
@IBOutlet weak var startAndEndLabel: UILabel!    // 출발지→목적지
```

설명

택시 이동 애니메이션, 좌우 움직임 효과, 자동 Step7 이동

16. StepSevenController

기능: 7 단계 - 요금 정산 **조건**: 실제 잔액 차감 (예상비용 ±10% 랜덤)

바인딩한 UI 컴포넌트

```
@IBOutlet weak var myBalanceChange: UILabel!     // 잔액 변화
@IBOutlet weak var onePersonCost: UILabel!       // 인당 비용
```

```
@IBOutlet weak var totalCost: UILabel!           // 총 비용
```

설명

실제 비용 계산, 잔액 차감, NSAttributedString 으로 변화 시각화

핵심함수

```
// 실제 잔액 차감
private func updateBalance() {
    guard let currentUserID = parentRoomDetail?.currentUserID else { return }
    guard let user = CoreDataManager.shared.getUser(userID: currentUserID) else { return }

    user.balance = Int32(newBalance)

    do {
        try CoreDataManager.shared.context.save()
    } catch {
        print("❌ 잔액 업데이트 실패: \(error)")
    }
}
```

17. SettingsViewController

기능: 설정 화면 - 가상 계좌 관리　　**조건**: 충전/출금 금액 검증

바인딩한 UI 컴포넌트

```
@IBOutlet weak var currentBalance: UILabel!      // 현재 잔액 (큰 글씨)
@IBOutlet weak var myBalance: UILabel!           // 상단 잔액
@IBOutlet weak var userName: UILabel!            // 사용자 이름
@IBOutlet weak var userSchool: UILabel!          // 학교명
@IBOutlet weak var firstView: UIView!            // 충전 컨테이너
@IBOutlet weak var secondView: UIView!           // 출금 컨테이너
@IBOutlet weak var thirdView: UIView!            // 잔액 표시 컨테이너
```

설명

3 개 컨테이너 뷰 카드 디자인, UIActionSheet 로 금액 선택, 실시간 잔액 업데이트

핵심함수

```
// 충전 처리
private func processDeposit(amount: Int) {
    guard let user = currentUser else { return }

    let currentBalance = Int(user.balance)
    let newBalance = currentBalance + amount

    user.balance = Int32(newBalance)

    do {
        try CoreDataManager.shared.context.save()
        updateBalanceLabels(balance: newBalance)

        showSuccessAlert(
            title: "💰 충전 완료",
            message: "충전 금액: \(NumberFormatter.localizedString(from: NSNumber(value: amount), number: .decimal))원\n현재 잔액: \(NumberFormatter.localizedString(from: NSNumber(value: newBalance), number: .decimal))원"
        )
    } catch {
        showAlert(title: "충전 실패", message: "데이터 저장 중 오류가 발생했습니다.")
    }
}

// 뷰 컨테이너 스타일링
private func setupViewContainers() {
    firstView.layer.cornerRadius = 12
    firstView.backgroundColor = UIColor.systemBackground
    firstView.layer.shadowColor = UIColor.black.cgColor
    firstView.layer.shadowOffset = CGSize(width: 0, height: 2)
    firstView.layer.shadowOpacity = 0.05
    firstView.layer.shadowRadius = 4

    thirdView.layer.cornerRadius = 12
    thirdView.layer.borderWidth = 1
    thirdView.layer.borderColor = UIColor.systemBlue.cgColor
}
```