SWIFTCRAFT
21-24 May 2024

Building Your First Android App:
A Guide for iOS Developers

Vui Nguyen

Building Your First Android App

A Guide for iOS Developers

Vui Nguyen
@sunfishgurl
linktr.ee/sunfishgurl

Our Problem / Challenge

- You’re an iOS developer, tasked with building an Android app 😱
- But where do you start?
- It’s about more than just the little syntactical differences!
- What about feature parity?!

linktr.ee/sunfishgurl

Feature Parity: Ensuring that a feature works the same across platforms. Does not mean the feature is implemented the same across platforms!
- Vui Nguyen (not famous .... yet)

What's One Possible Solution?

- My talk!
- My solution focuses on:
  - architecture
  - process
  - leveraging existing iOS knowledge! You know more than you think you do to get started!
- Are we excited yet? Let's go 🚀

linktr.ee/sunfishgurl

Overview

- Introduce problem
- Introduce myself
- Goals for this talk
- Introducing Task Tracker App: iOS
- How We Built Android and iOS Apps in Parallel

linktr.ee/sunfishgurl

Overview

- Building a CRUD Mobile App on Any Platform
  - Components of Every CRUD Mobile App
  - App Flows:
    - Create - Read: Add Flow
    - Update: Edit Flow
    - Delete: Delete Flow

Overview

- Demo: Task Tracker App for Android
- Putting it All Together: Tips for Building a Standalone Android App
- Resources
- Shoutouts

Hello!👋

I’m Vui Nguyen

- Software engineer with 20+ years experience across multiple tech stacks
- Mobile developer since 2011 (cross-platform, iOS, learning Android)
- Technical Leader
- Former Leadership Fellow for Women Who Code Mobile
- Schnoodle Mom
- Fisherwoman

linktr.ee/sunfishgurl

Non Goals for this Talk

What this talk is NOT about

- Not an analysis on native versus cross platform mobile development
- Not an analysis on how to use generative AI to build our apps for us
  - Spoiler alert: Generative AI cannot do ALL the work for us!
- Not meant to teach you the foundation of mobile development
  - This talk assumes you have intermediate level knowledge of iOS Swift development
- Not about building Android apps using the “best” fill in the blank here: design patterns, architecture, best practices, etc

Goals for this Talk

What this talk is about

- How to **get started** on Android development by **building on your existing iOS knowledge**
  - avoid analysis paralysis
  - progress / improve as you go
- Progress over Perfection
- Demonstrate **process** for building a mobile app on either platform

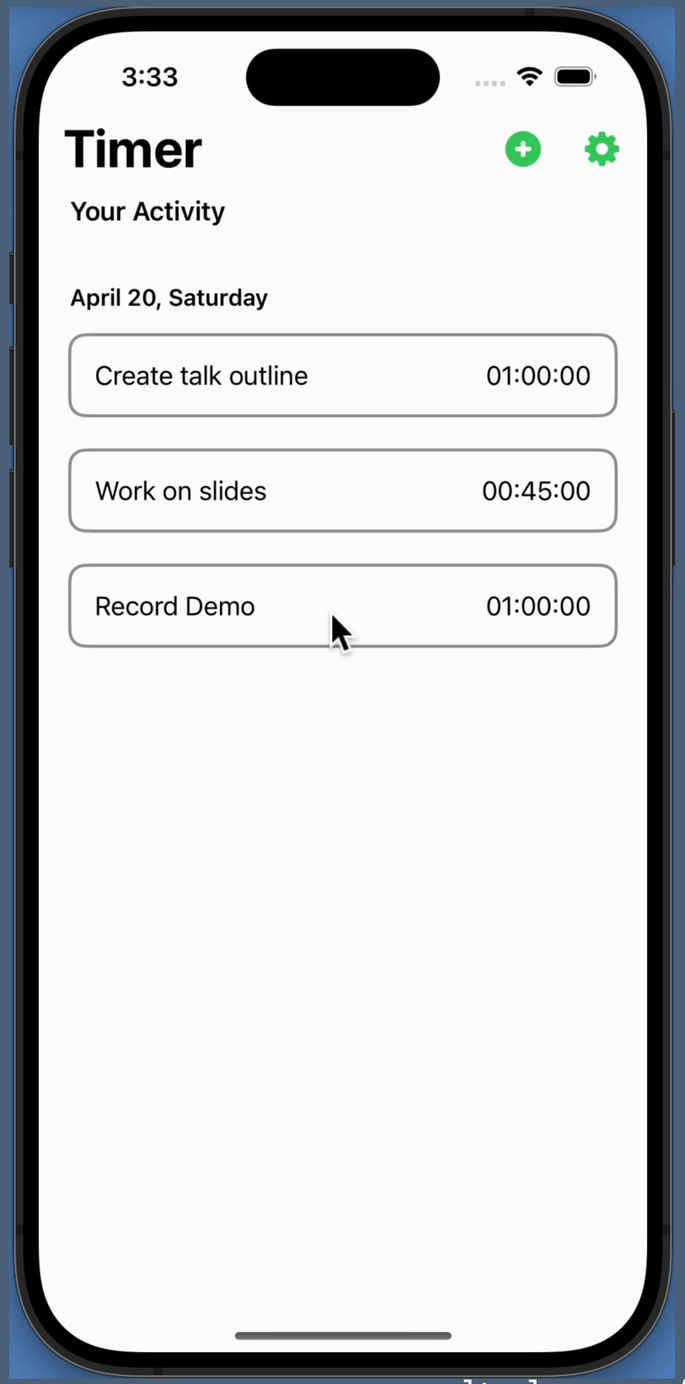
Introducing Task Tracker App

linktr.ee/sunfishgurl

Task Tracker App Demo: iOS

https://bit.ly/TaskTrackerAppiOSDemo

- Open Source Project for Women Who Code Mobile
- Tracks Tasks Being Worked On
- Tracks Time Spent on Tasks
- CRUD Operations



linktr.ee/sunfishgurl

How We Built Android and iOS Apps in Parallel

linktr.ee/sunfishgurl

How We Built Android and iOS Apps in Parallel

- Surprise! 😮 We also built the Task Tracker App for Android at the same time
- How did we do it?
- What lessons did we learn?

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Women Who Code Mobile was an online technical track community, that was part of Women Who Code. (Women Who Code shut down in April 2024 😭)
- Project Maintainer Teams were comprised of volunteers for Women Who Code Mobile
- As the Leadership Fellow of WWCode Mobile, I formed the maintainer teams based on volunteer skill, interest, and availability to commit to the project
- Contributors comprised of regular community members of WWCode Mobile and volunteer team members

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Project Team Breakdown:
  - Each team (Android and iOS) had:
    - 2-3 project maintainers
      - Android: Gauri, Ren, Sepideh
      - iOS: Claudia and Devanshi
    - project manager (Cecelia for both)
    - project lead and architect (Vui for both)

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Preparation Leading Up To Development:
  - October 2023: WWCode Mobile hosted open source contribution workshop
  - November 2023: WWCode Mobile hosted open source maintainers workshop
  - December 2024:
    - finalize maintainer teams
    - start discussion on app idea, app wireframes and MVP project scope

MVP: Minimum Viable Product, is a version of a product with just enough features to be usable
- https://en.wikipedia.org/wiki/Minimum_viable_product

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Project Schedule:
  - January 2024: Project Kick Off (for both Android and iOS)
    - Kick Off Event for each platform over Zoom
    - Introduce app wireframes to community
    - Initial project folders set up (MVVM structure)
    - Introduce open source process and workflow to community
    - Establish Github and official Slack channel for project collaboration / communication

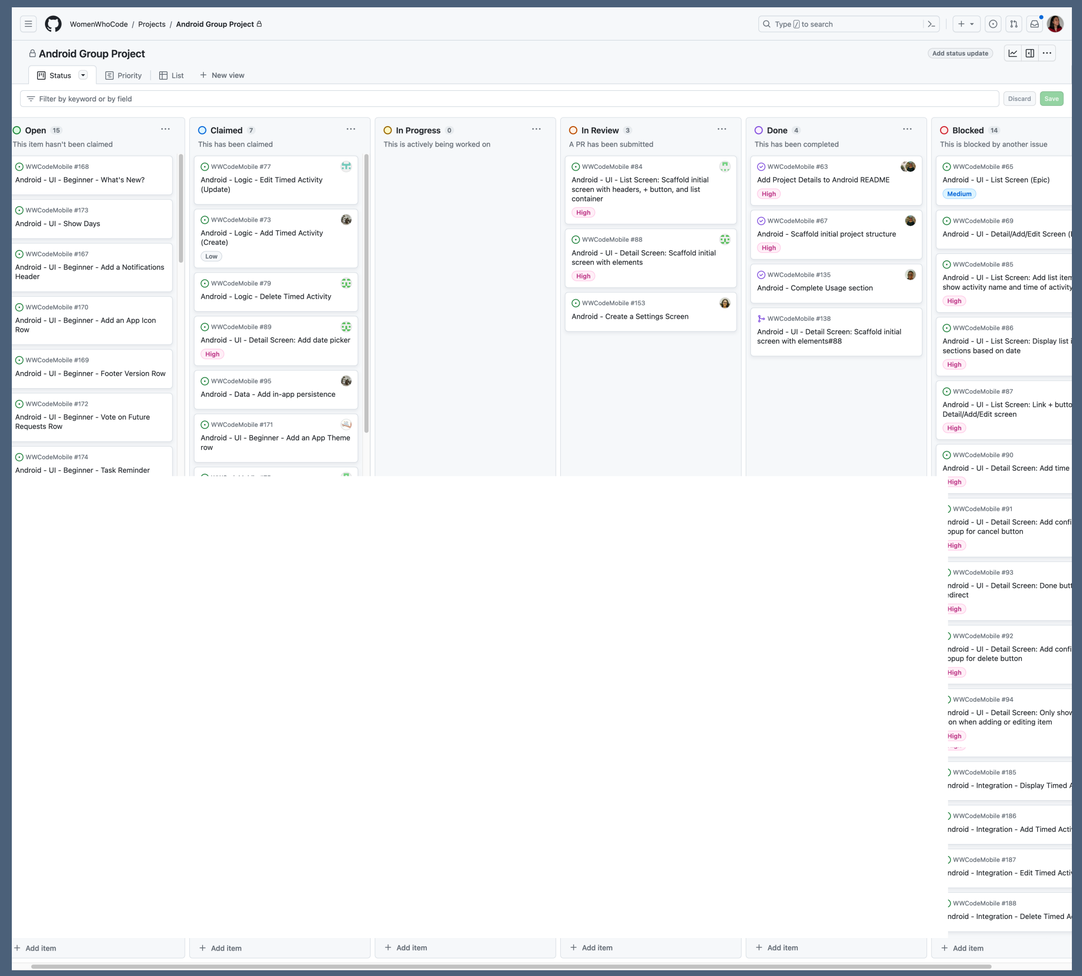
How We Built Android and iOS Apps in Parallel

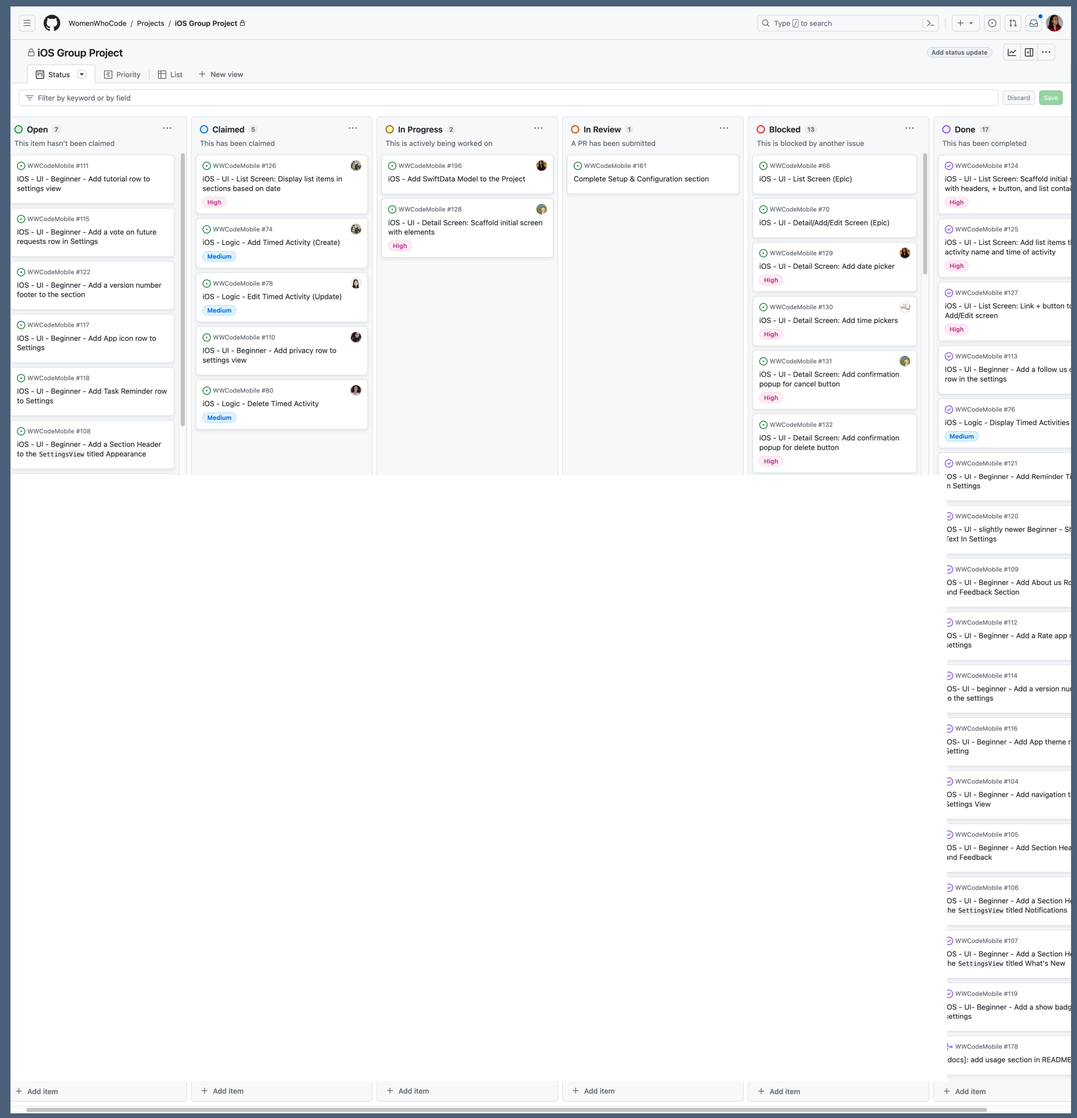
How Women Who Code Mobile Did It

- Project Schedule:
  - January 2024: Project Kick Off (for both Android and iOS)
    - Set up Github project boards
    - Created epics from app wireframes and then broken down further into smaller issues
    - Initial Github issues written up
    - Huge thanks to Cecelia for doing this project management work 🎉

How We Built Android and iOS Apps in Parallel

Android and iOS Github project boards





linktr.ee/sunfishgurl

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Project Schedule:
  - January 2024: Project Kick Off (for both Android and iOS)
    - Thanks for writing up beginner issues:
      - Sepideh (for Android)
      - Claudia (for iOS)
  - Work on projects officially starts! First PRs from contributors start getting merged within the first day or so of kick off events 🚀

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Project Schedule:
  - February 2024: Status Event for Android and iOS
    - Held Status Event for each platform over Zoom
      - iOS delivered MVP demo
      - Android demos progress so far, but struggling to make MVP
      - Celebrate contributors' progress so far
    - Vui promotes active contributor Sepideh to Android project maintainer status
    - Vui creates Github templates for writing issues and PRs - contributors needed guidance beyond a "blank" slate

How We Built Android and iOS Apps in Parallel

How Women Who Code Mobile Did It

- Project Schedule:
  - March 2024: Wrap Up Event for Android and iOS
    - Held Wrap Up Event for each platform over Zoom
      - iOS delivered MVP + bonus features 🎉
        - swipe to delete, task priority
      - Android delivers MVP

How We Built Android and iOS Apps in Parallel

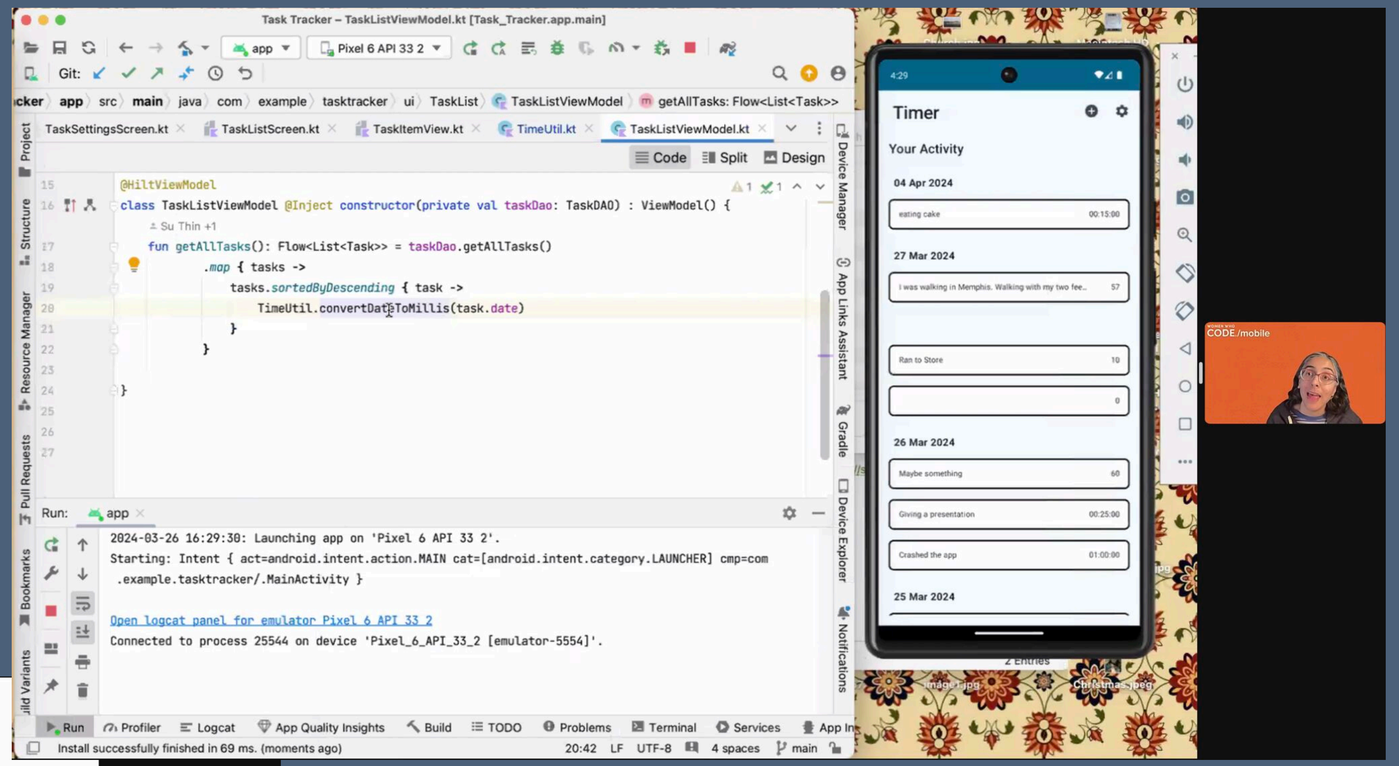
How Women Who Code Mobile Did It

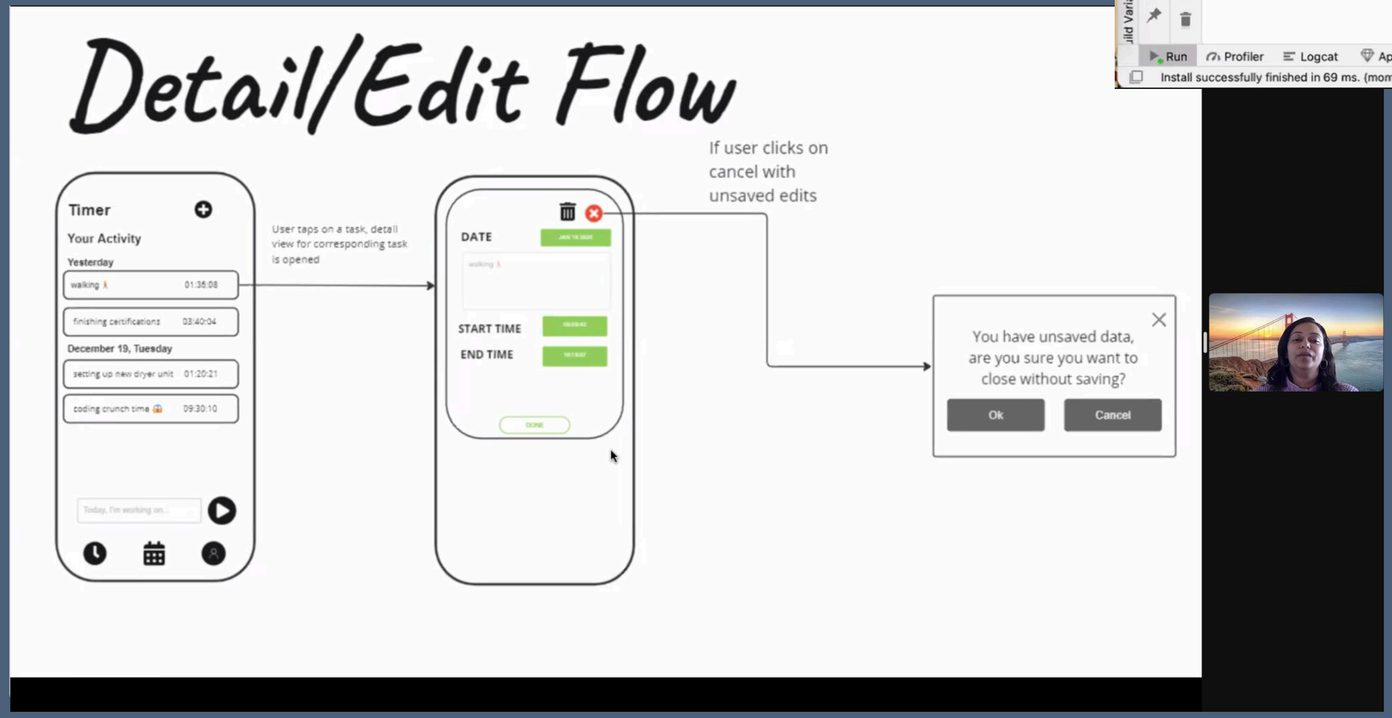
- Project Schedule:
  - March 2024: Wrap Up Event for Android and iOS
    - Projects wrap up and all open issues are closed
    - Final celebration of all contributors 🥳
      - 12+ combined on each platform!
    - Meet and Greet after presentation (not recorded)

How We Built Android and iOS Apps in Parallel

Android Wrap Up Event

https://bit.ly/TaskTrackerAppAndroidWrapUpEvent



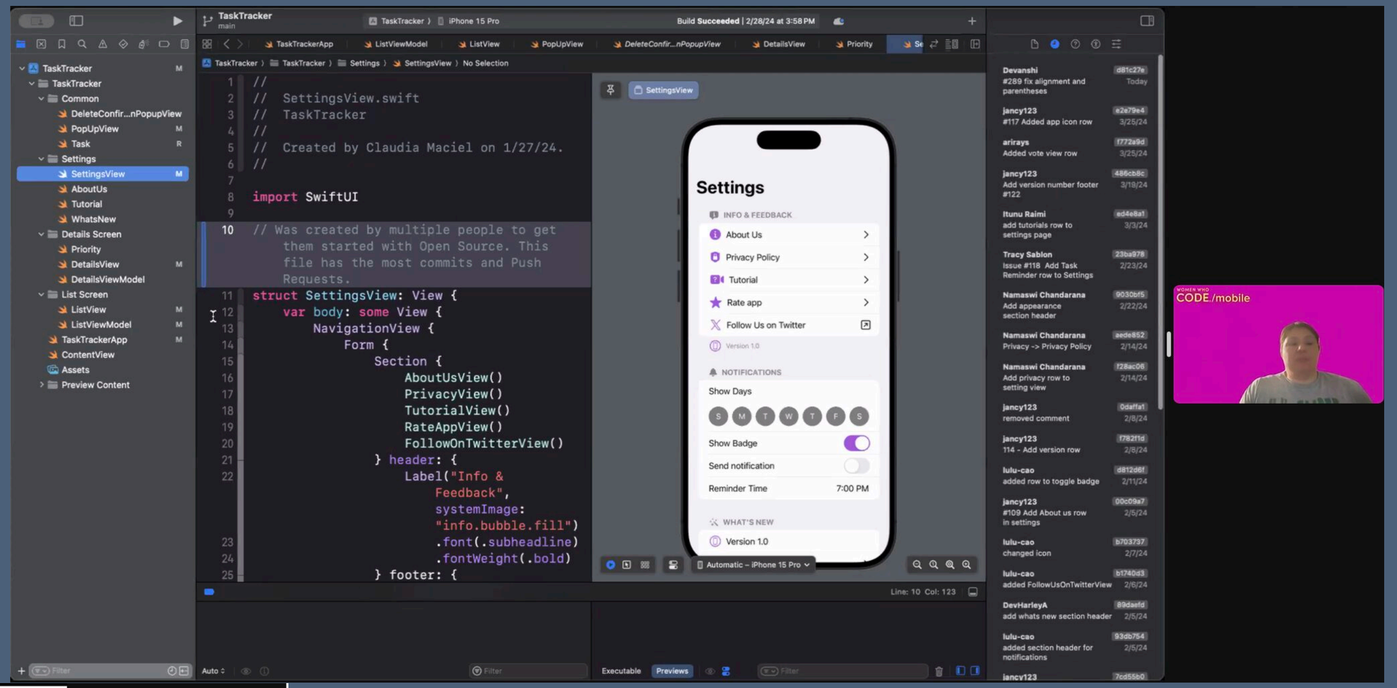


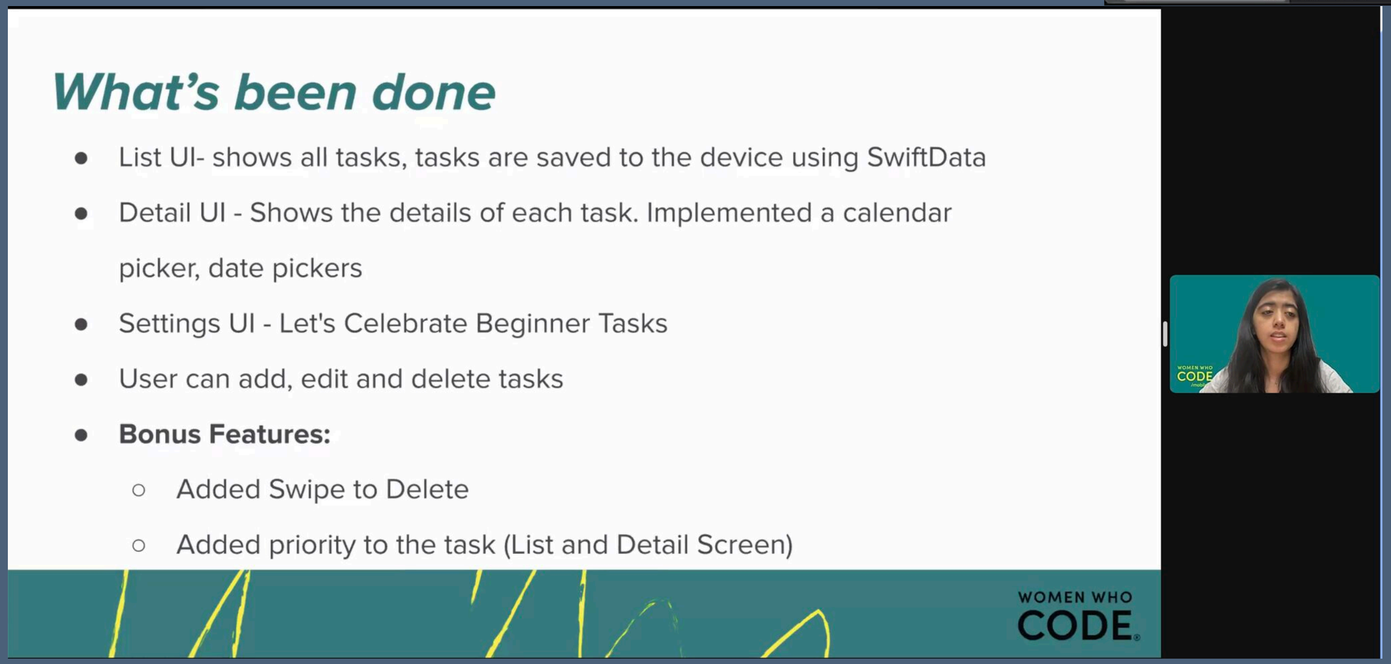
linktr.ee/sunfishgurl

How We Built Android and iOS Apps in Parallel

iOS Wrap Up Event

https://bit.ly/TaskTrackerAppiOSWrapUpEvent





linktr.ee/sunfishgurl

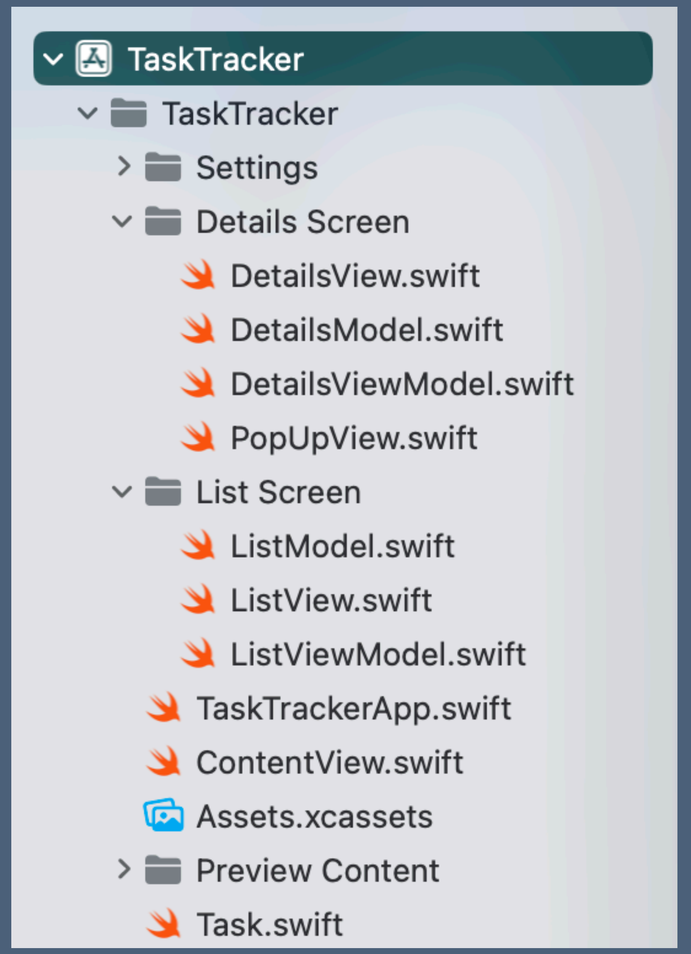
Building a CRUD Mobile App on Any Platform

linktr.ee/sunfishgurl

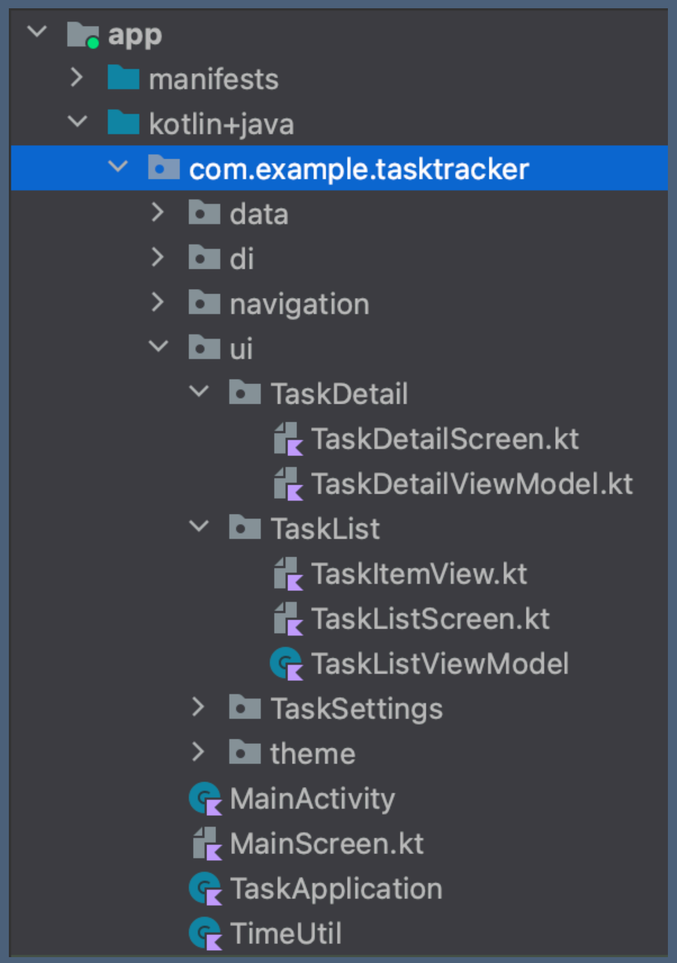
Building a CRUD Mobile App on Any Platform

What Every CRUD Mobile App Needs

- Architecture
  - MVVM: separate UI from business logic



iOS Project Structure



Android Project Structure

linktr.ee/sunfishgurl

Building a CRUD Mobile App on Any Platform

What Every CRUD Mobile App Needs

- UI
  - Android: Jetpack Compose
  - iOS: SwiftUI
- Logic
  - Android: Kotlin, ViewModels
  - iOS: Swift, ViewModels

Building a CRUD Mobile App on Any Platform

What Every CRUD Mobile App Needs

- Navigation
  - Android: Compose support for Navigation (NavController)
  - iOS: NavigationStack, NavigationLink
- Persistence
  - Android:
    - Room database
    - Use coroutines (asynchronous calls) to access database in background thread / scope
  - iOS: SwiftData

linktr.ee/sunfishgurl

Building a CRUD Mobile App on Any Platform

What Every CRUD Mobile App Needs

- Binding data to UI
  - Android:
    - Closures capture updated state from UI
    - StateFlows expose database to the ViewModel
    - ViewModel saves captured state into database through StateFlow
    - ViewModel updates UI from database through StateFlow
  - iOS:
    - SwiftUI property wrapper @Published for Task Model
    - SwiftUI property wrappers @State and @Binding to set / get state from UI

Building a CRUD Mobile App on Any Platform

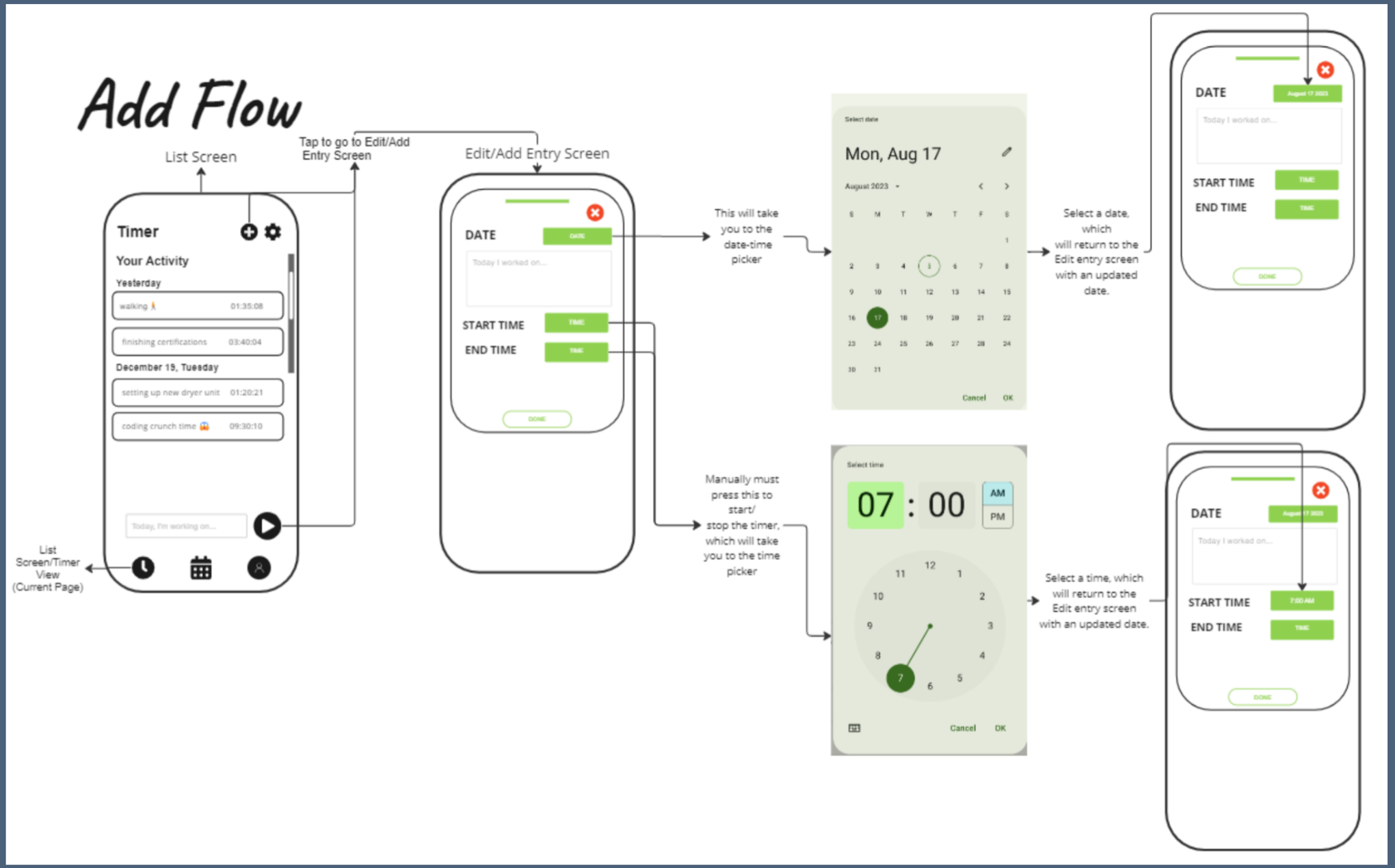
App Flows

- Create - Read: Add Flow
- Update: Edit Flow
- Delete: Delete Flow

Add Flow

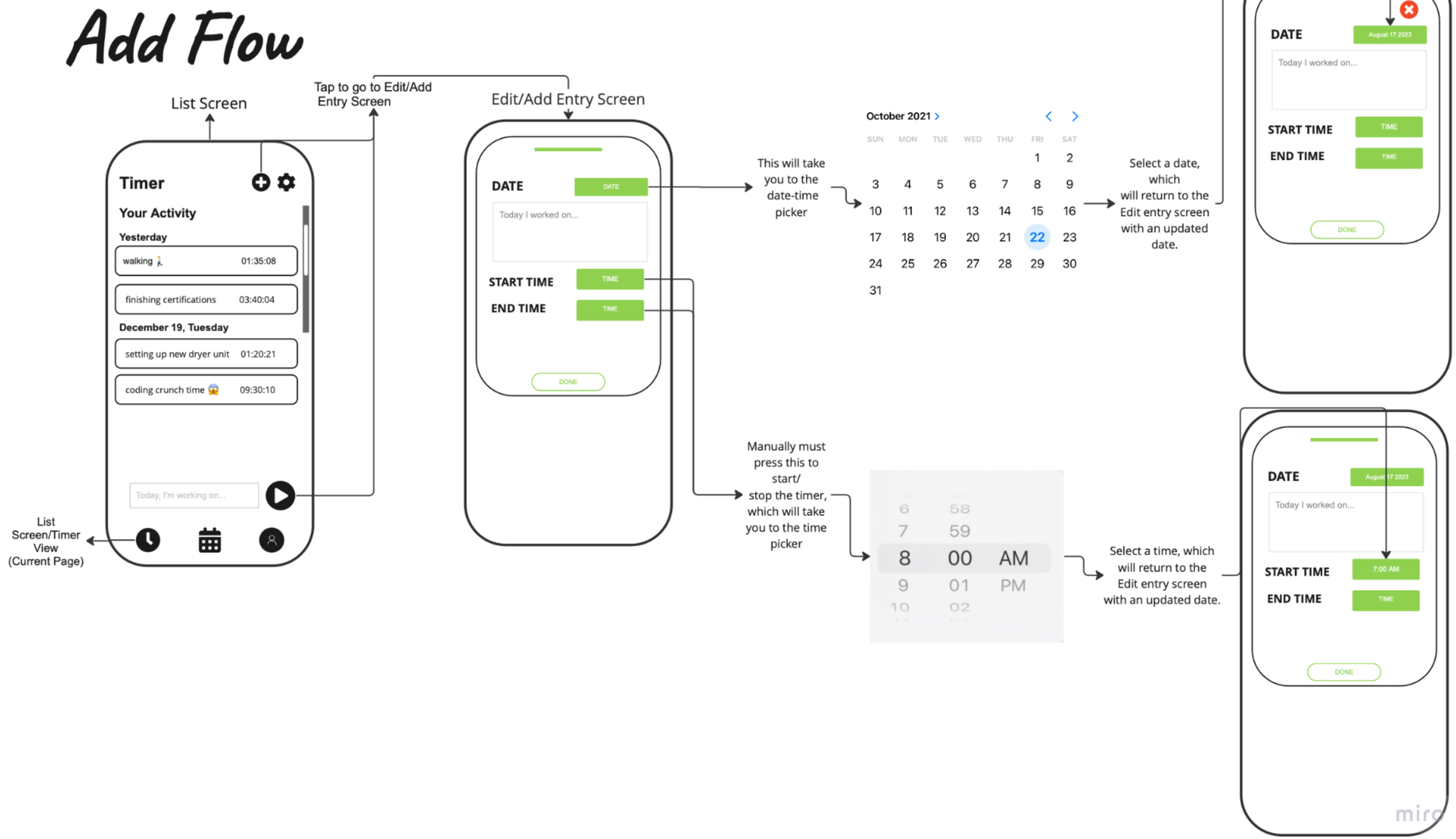
Add Flow - Android

linktr.ee/sunfishgurl



Add Flow - iOS

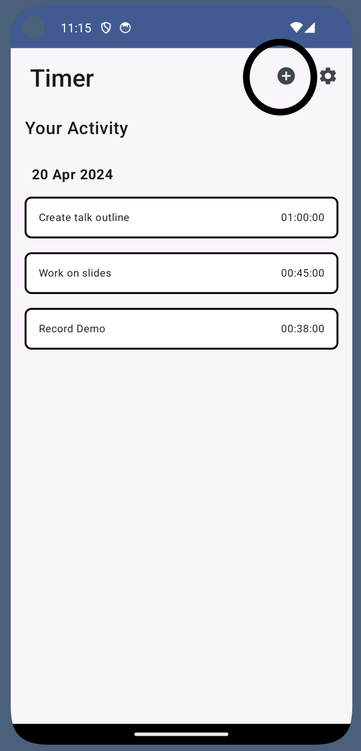
linktr.ee/sunfishgurl



Android Side

List Screen - Navigation

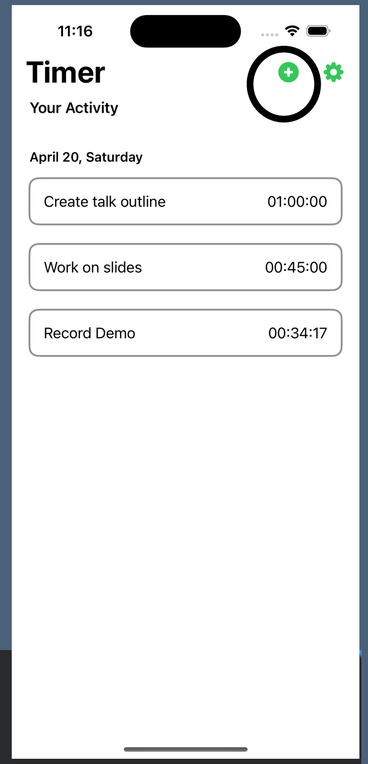
- Display + Button on List Screen




```kotlin
@Composable
fun TaskListScreen(
    onNavigateToSettings: () -> Unit,
    onNavigateToDetail: (id: Int?) -> Unit,
    taskListViewModel: TaskListViewModel
) {
    Scaffold(topBar = {
        ListScreenTopAppBar({ onNavigateToSettings() }, onNavigateToDetail)
    })
    ...
}
```

iOS Side

List Screen - Navigation



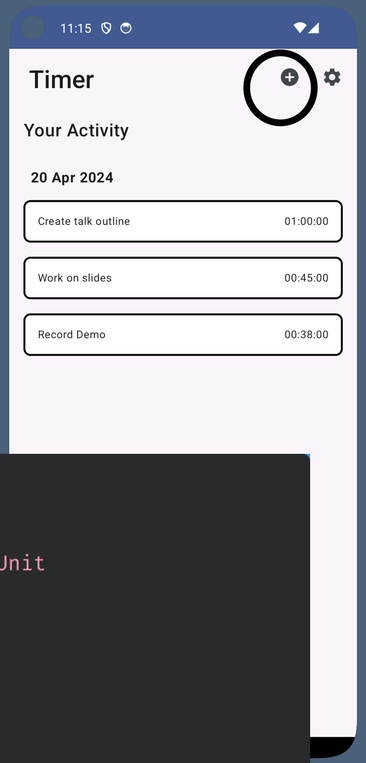

```swift
struct ContentView: View {
    var body: some View {
        NavigationStack {
            ...
            ListView(viewModel: ListViewModel())
                .toolbar {
                    ...
                    ToolbarItemGroup(placement: .topBarTrailing) {
                        AddButtonView()
                        SettingsButtonView()
                    }
                }
        }
    }
}
```

linktr.ee/sunfishgurl

Android Side

List Screen - Navigation

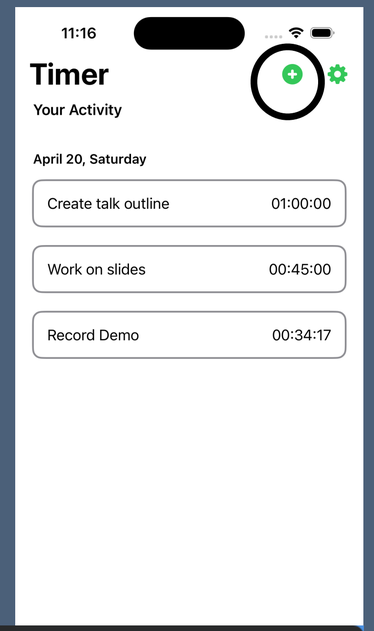
- Press + Button to Navigate to Detail Screen




```
@Composable
fun ListScreenTopAppBar(
    onNavigateToSettings: () → Unit, onNavigateToDetail: (id: Int?) → Unit
) {
    TopAppBar(title = {
        Column(
            modifier = Modifier.padding(
                start = dimensionResource(R.dimen.app_bar_padding),
                top = dimensionResource(R.dimen.app_bar_padding)
            )
        ) {
            Text(
                text = stringResource(R.string.timer),
                fontWeight = FontWeight.Medium,
                fontSize = 28.sp
            )
        }
    }, actions = {
        IconButton(onClick = { onNavigateToDetail(null) }) {
            Icon(
                Icons.Filled.AddCircle, contentDescription = stringResource(R.string.add_button)
            )
        }
        ...
    }

    )
}
```

iOS Side

List Screen - Navigation



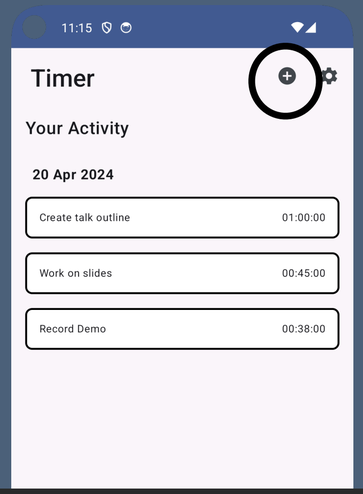

```
private struct AddButtonView: View {
    @State private var showDetailsScreen = false
    var body: some View {
        Button(action: {
            showDetailsScreen.toggle()
        }, label: {
            Image(systemName: "plus.circle.fill")
                .foregroundColor(Color.green)
                .font(Font.body.weight(.black))
        })
        .sheet(isPresented: $showDetailsScreen, content: {
            DetailsScreen(task: nil, isEditingMode: false)
        })
    }
}
```

linktr.ee/sunfishgurl

Android Side

List Screen - Navigation

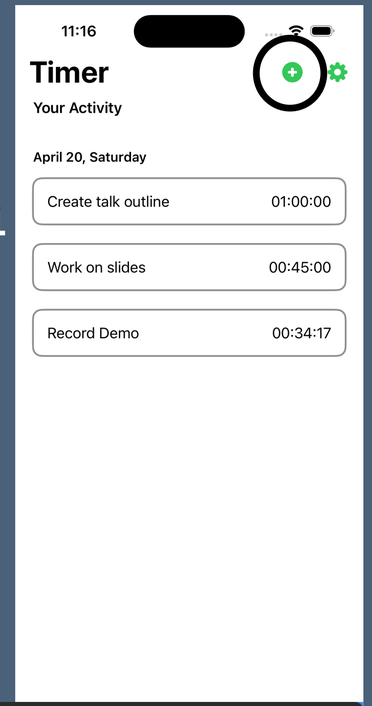
- Navigate to Detail Screen




```
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.rememberNavController

@Composable
fun MainScreen() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = NavScreens.TaskList.route) {
        composable(NavScreens.TaskList.route) {
            TaskListScreen(
                onNavigateToSettings = { navController.navigate(NavScreens.TaskSettings.route) },
                onNavigateToDetail = { taskId ->
                  navController.
                  navigate("${NavScreens.TaskDetail.route}/${getDetailArg(taskId = taskId)}")
                },
            )
        }
    }
}
```

iOS Side

List Screen - Navigation



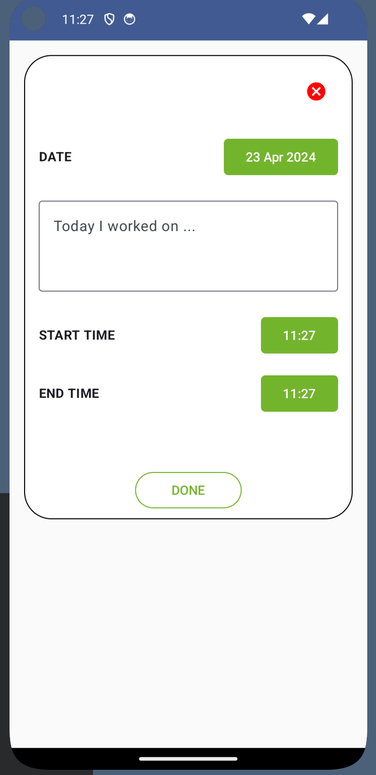

```
private struct AddButtonView: View {
    @State private var showDetailsScreen = false
    var body: some View {
        Button(action: {
            showDetailsScreen.toggle()
        }, label: {
            Image(systemName: "plus.circle.fill")
                .foregroundColor(Color.green)
                .font(Font.body.weight(.black))
        })
        .sheet(isPresented: $showDetailsScreen, content: {
            DetailsScreen(task: nil, isEditingMode: false)
        })
    }
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

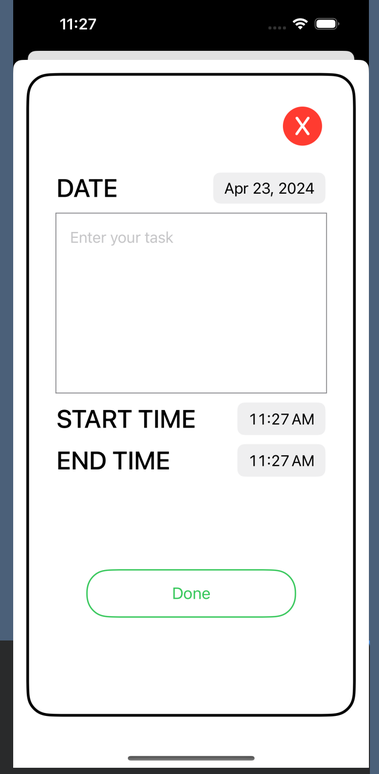
- Display Detail Screen




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {
        ...
        DetailDateButton(
            initialDate = uiState.date,
        )

        OutlinedTextField(
            placeholder = { Text(text = stringResource(id = R.string.textfield_label)) },
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.start_time_label),
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
        )

        OutlinedButton(
            },
        ) {
            Text(text = stringResource(id = R.string.done).uppercase())
        }
    }
}
```

iOS Side

Detail Screen



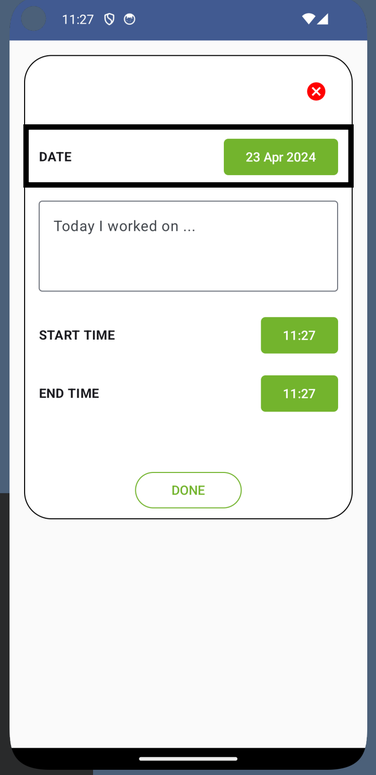

```
struct DetailsScreen: View {
    ...
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate, displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }

                TextField("Enter your task", text: $taskText)

                DatePicker(selection: $startTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }

                DoneButton(shouldDismiss: $shouldDismiss, taskText: $taskText, taskDate: $taskDate,
                           startTime: $startTime, endTime: $endTime, task: task )
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

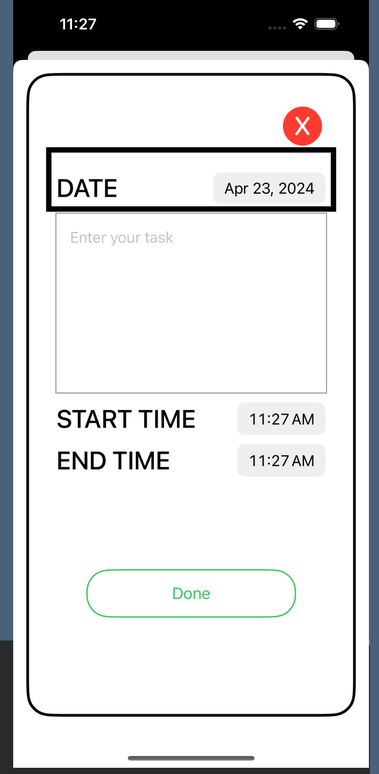
- Display Date Button




```kotlin
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {

        DetailDateButton(
            initialDate = uiState.date,
        )

        OutlinedTextField(
            placeholder = { Text(text = stringResource(id = R.string.textfield_label)) },
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.start_time_label),
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
        )

        OutlinedButton(
            },
        ) {
            Text(text = stringResource(id = R.string.done).uppercase())
        }
    }
}
```

iOS Side

Detail Screen



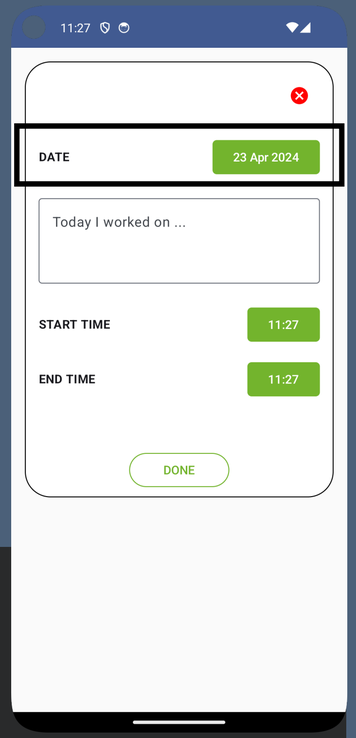

```swift
struct DetailsScreen: View {
    ...
    var body: some View {
        ZStack {
            VStack(spacing: 10) {

                DatePicker(selection: $taskDate, displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }

                TextField("Enter your task", text: $taskText)

                DatePicker(selection: $startTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }

                DoneButton(shouldDismiss: $shouldDismiss, taskText: $taskText, taskDate: $taskDate,
                           startTime: $startTime, endTime: $endTime, task: task )
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

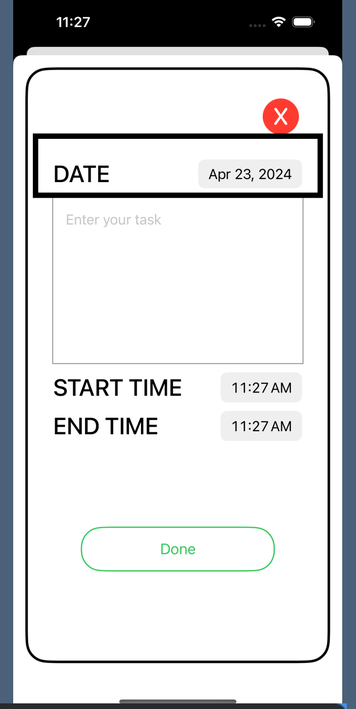
- Date Button




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        ....
    ) {
        ...
        DetailDateButton(
            initialDate = uiState.date,
        ) { selectedDate →
            taskDetailViewModel.updateDate(selectedDate)
        }
        ...
    }
}
```

iOS Side

Detail Screen



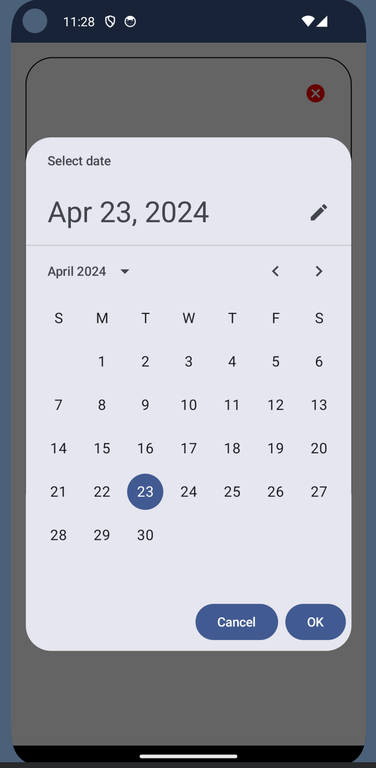

```
struct DetailsScreen: View {
    @State private var taskDate = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate,
                           displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Date Button - display Date Picker




```
@Composable
fun DetailDateButton(initialDate: String, onDateSelected: (String) → Unit) {
    ....
    if (showDatePicker) {
        DetailDatePickerDialog(initialDate, onDateSelected = { newDate →
            date = newDate
            onDateSelected(newDate)
        }, onDismiss = { showDatePicker = false })
    }
}
```

iOS Side

Detail Screen

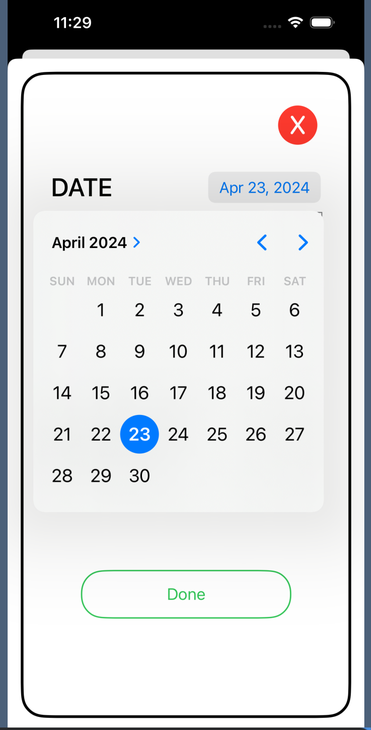
Structure

DatePicker

A control for selecting an absolute date.

iOS 13.0+ | iPadOS 13.0+ | macOS 10.15+ | Mac Catalyst 13.0+ | watchOS 10.0+ | visionOS 1.0+

```
struct DatePicker<Label> where Label : View
```



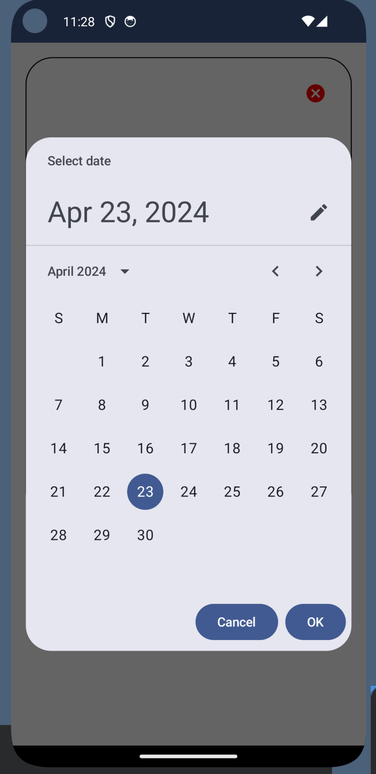

```
struct DetailsScreen: View {
    @State private var taskDate = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate,
                           displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Date Button - display Date Picker




```
import androidx.compose.material3.DatePicker
import androidx.compose.material3.DatePickerDialog

@Composable
fun DetailDatePickerDialog( ... ) {
    DatePickerDialog(onDismissRequest = { onDismiss() }, confirmButton = {
        Button(onClick = {
            onDateSelected(selectedDate)
        }) {
            Text(text = stringResource(id = R.string.ok).uppercase())
        }
    ) {
        DatePicker(state = datePickerState)
    }
}
```

iOS Side

Detail Screen

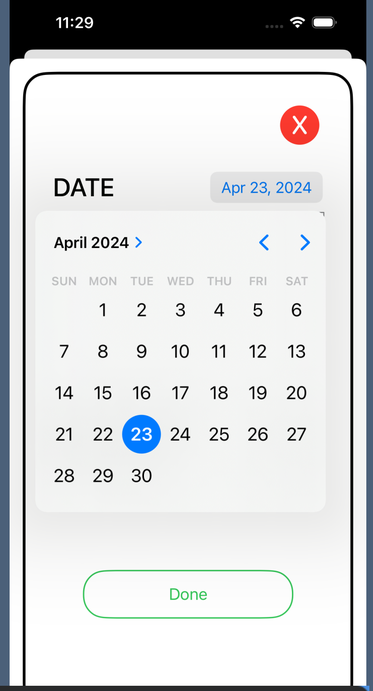
Structure

DatePicker

A control for selecting an absolute date.

iOS 13.0+ | iPadOS 13.0+ | macOS 10.15+ | Mac Catalyst 13.0+ | watchOS 10.0+ | visionOS 1.0+

```
struct DatePicker<Label> where Label : View
```




```
struct DetailsScreen: View {
    @State private var taskDate = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate,
                           displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Date picker - update state

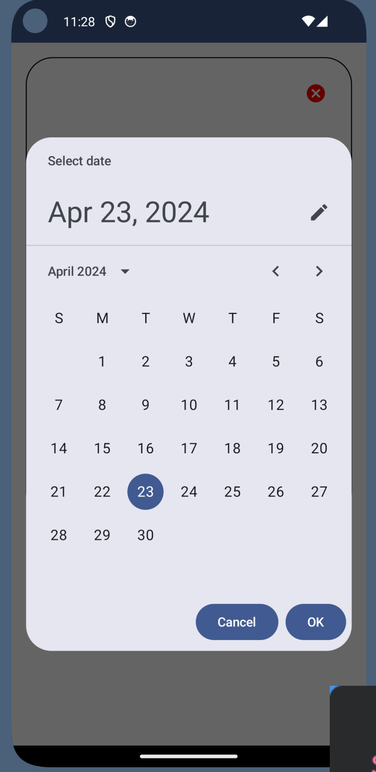
```
if (showDatePicker) {
    DetailDatePickerDialog(initialDate, onDateSelected = { newDate ->
        date = newDate
        onDateSelected(newDate)
    }, onDismiss =
}
```

Value captured in a closure

value-parameter onDateSelected: (String) -> Unit

Task_Tracker.app.main

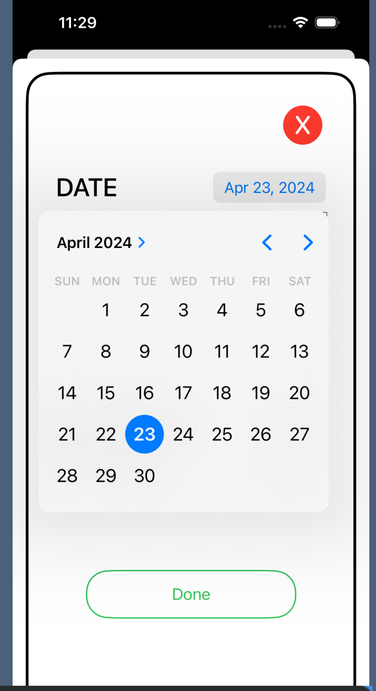
```
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()
    OutlinedCard(
        ....
    ) {
        ...
        DetailDateButton(
            initialDate = uiState.date,
        ) { selectedDate →
            taskDetailViewModel.updateDate(selectedDate)
        }
        ...
    }
}

class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val _detailState = MutableStateFlow(DetailState(false))
    val detailState: StateFlow<DetailState> = _detailState
    ...
    fun updateDate(date: String) {
        _detailState.update { it.copy(date = date) }
    }
}
```



iOS Side

Detail Screen

- state updated in screen and / or view model but NOT in Task Model yet



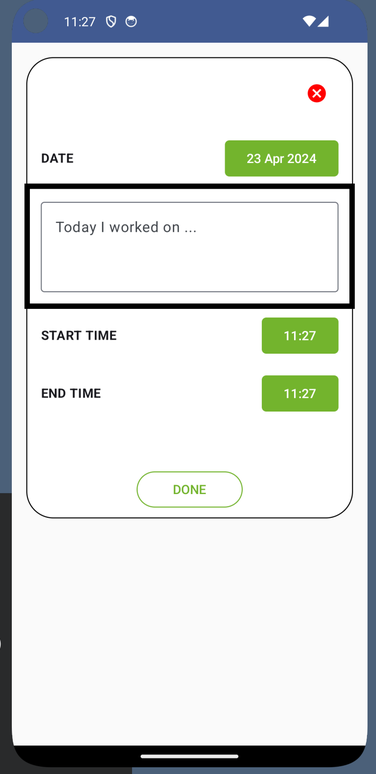

```
struct DetailsScreen: View {
    @State private var taskDate = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate,
                           displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }
            }
        }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

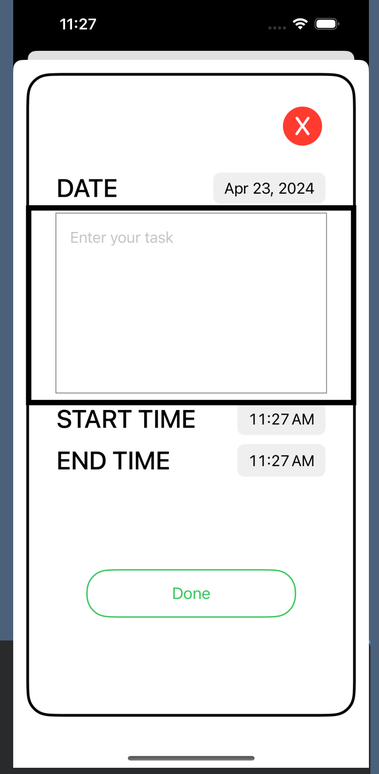
- Display TextField




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {
        ...
        DetailDateButton(
            initialDate = uiState.date,
        )

        OutlinedTextField(
            placeholder = { Text(text = stringResource(id = R.string.textfield_label)) },
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.start_time_label),
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
        )

        OutlinedButton(
            },
        ) {
            Text(text = stringResource(id = R.string.done).uppercase())
        }
    }
}
```

iOS Side

Detail Screen



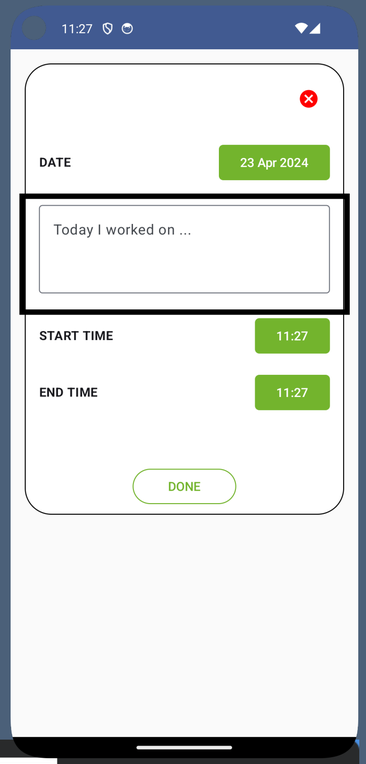

```
struct DetailsScreen: View {
    ...
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate, displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }

                TextField("Enter your task", text: $taskText)

                DatePicker(selection: $startTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }

                DoneButton(shouldDismiss: $shouldDismiss, taskText: $taskText, taskDate: $taskDate,
                           startTime: $startTime, endTime: $endTime, task: task )
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- TextField




```
import androidx.compose.material3.OutlinedTextField

@Composable
fun TaskDetailScreen( ... )
{
    OutlinedTextField(
        value = uiState.activityName,
        onValueChange = { taskDetailViewModel.updateActivity(it) },
        placeholder = { Text(
          text = stringResource(id = R.string.textfield_label)) },
    )
}
```

iOS Side

Detail Screen

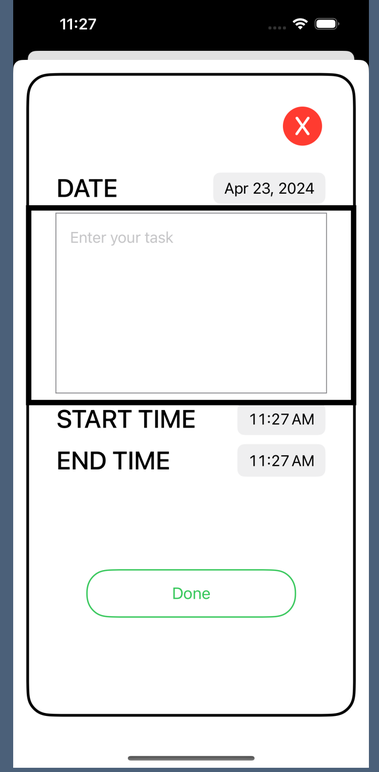
Structure

TextField

A control that displays an editable text interface.

iOS 13.0+ | iPadOS 13.0+ | macOS 10.15+ | Mac Catalyst 13.0+ | tvOS 13.0+ | watchOS 6.0+ | visionOS 1.0+

```
struct TextField<Label> where Label : View
```



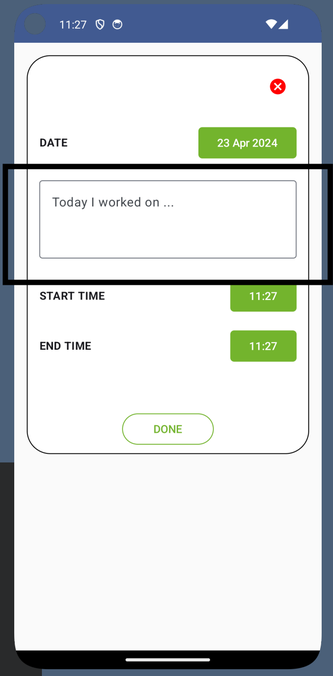

```
struct DetailsScreen: View {
    @State private var taskText: String = ""

    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                TextField("Enter your task", text: $taskText)
                ...
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- TextField - update state

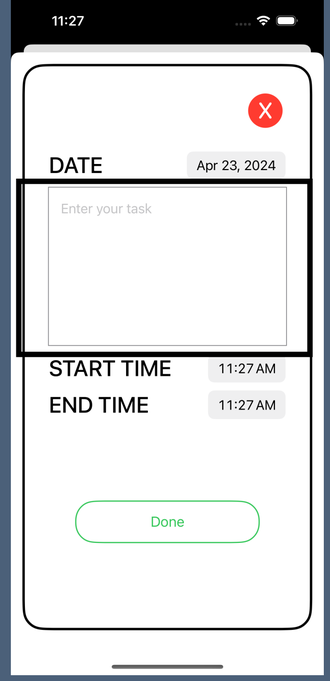



```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()
    OutlinedCard(
    ) {
        OutlinedTextField(
        value = uiState.activityName,
        onValueChange = { taskDetailViewModel.updateActivity(it) },
        placeholder = { Text(
          text = stringResource(id = R.string.textfield_label)) },
        )
    }
}

class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val _detailState = MutableStateFlow(DetailState(false))
    val detailState: StateFlow<DetailState> = _detailState
    ...
    fun updateActivity(activityName: String) {
        _detailState.update { it.copy(activityName = activityName) }
    }
}
```

iOS Side

Detail Screen

- state updated in screen and / or view model but NOT in Task Model yet



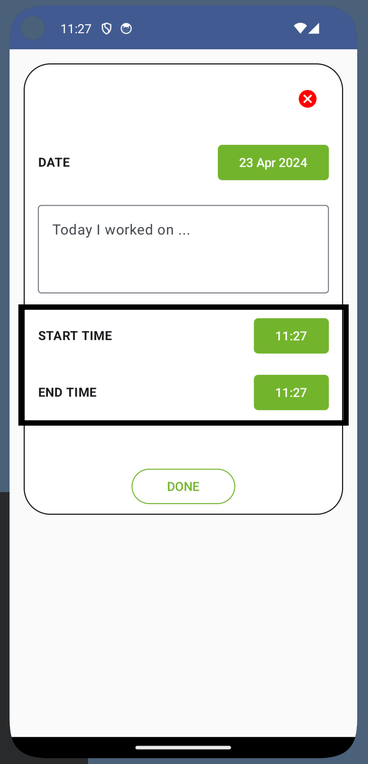

```
struct DetailsScreen: View {
    @State private var taskText: String = ""

    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                TextField("Enter your task", text: $taskText)
                ...
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

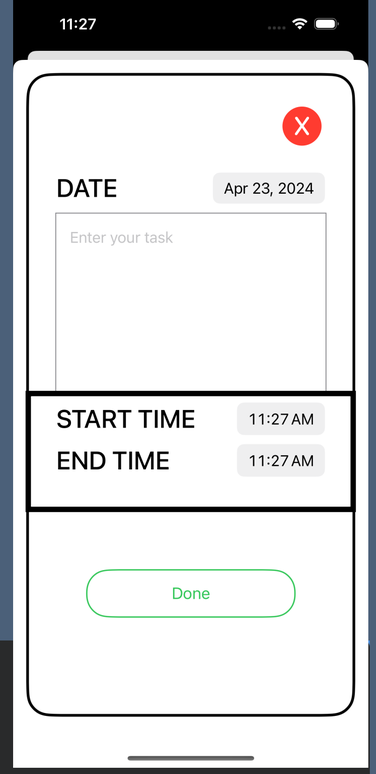
- Display Time Buttons




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {
        ...
        DetailDateButton(
            initialDate = uiState.date,
        )

        OutlinedTextField(
            placeholder = { Text(text = stringResource(id = R.string.textfield_label)) },
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.start_time_label),
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
        )

        OutlinedButton(
            },
        ) {
            Text(text = stringResource(id = R.string.done).uppercase())
        }
    }
}
```

iOS Side

Detail Screen




```
struct DetailsScreen: View {
    ...
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate, displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }

                TextField("Enter your task", text: $taskText)

                DatePicker(selection: $startTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }

                DoneButton(shouldDismiss: $shouldDismiss, taskText: $taskText, taskDate: $taskDate,
                           startTime: $startTime, endTime: $endTime, task: task )
            }
    }
```

linktr.ee/sunfishgurl

Android Side

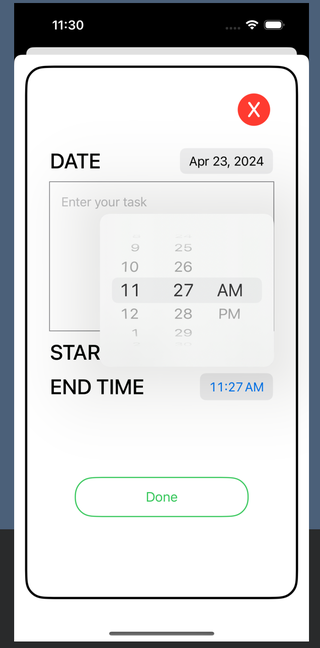
Detail Screen

- Time Buttons - Display Time Pickers


```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        ....
    ) {
        ...
        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.start_time_label),
            initialTime = uiState.startTime,
            onTimeSelected = taskDetailViewModel::updateStartTime
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
            initialTime = uiState.endTime,
            onTimeSelected = taskDetailViewModel::updateEndTime
        )
        ...
    }
}
```

iOS Side

Detail Screen



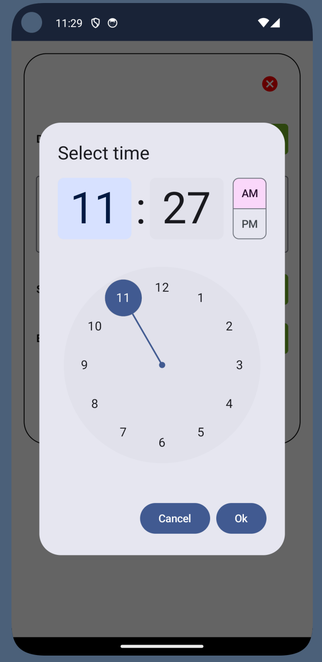

```
struct DetailsScreen: View {
    @State private var startTime = Date.now
    @State private var endTime = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $startTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

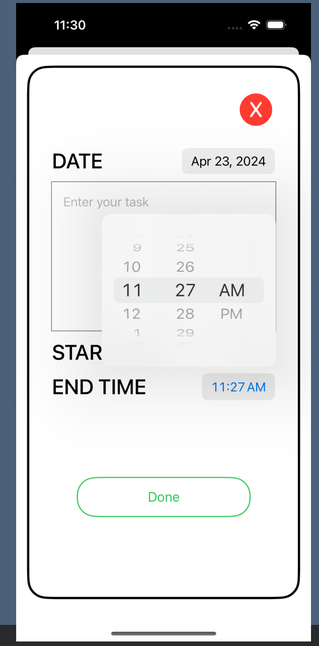
- Time Buttons - Display Time Pickers




```
@Composable
fun TimePickerRow(timeRowLabel: String,
                  initialTime: String,
                  onTimeSelected: (String) -> Unit) {
    ...
    if (showTimePicker) {
        DetailTimePickerDialog(initialTime, onTimeSelected = { selectedTime ->
            onTimeSelected(selectedTime)
        }, onDismiss = { showTimePicker = false })
    }
}
```

iOS Side

Detail Screen



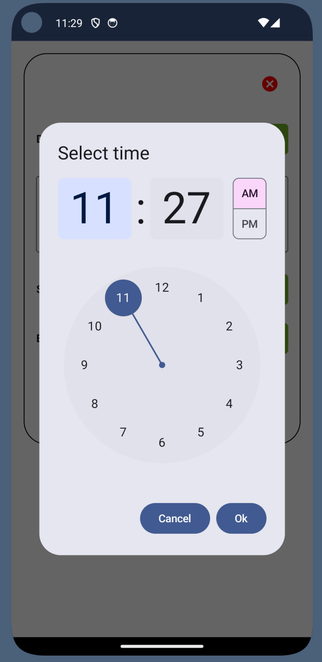

```
struct DetailsScreen: View {
    @State private var startTime = Date.now
    @State private var endTime = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $startTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

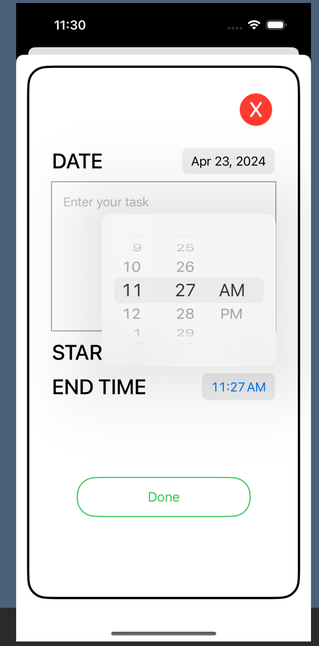
- Time Buttons - Display Time Pickers




```
import androidx.compose.material3.TimePicker

@Composable
fun DetailTimePickerDialog(
    initialTime: String, onTimeSelected: (String) -> Unit,
    onDismiss: () -> Unit) {
    ...
    TimePickerDialog(onDismissRequest = { onDismiss() },
                     onConfirm = {
        onTimeSelected(selectedTime) ...
    }) {
        TimePicker(state = timePickerState)
    }
}
```

iOS Side

Detail Screen




```
struct DetailsScreen: View {
    @State private var startTime = Date.now
    @State private var endTime = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $startTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Time Buttons - Display Time Pickers


```
if (showTimePicker) {
    DetailTimePickerDialog(initialTime, onTimeSelected = { selectedTime ->
        onTimeSelected(selectedTime)
    }, onDi
}
```

Value captured in a closure

value-parameter onTimeSelected: (String) -> Unit

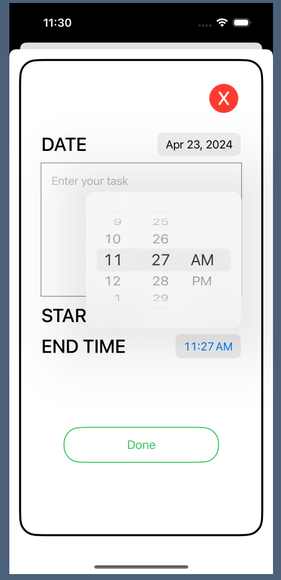
Task_Tracker.app.main

```
import androidx.compose.material3.TimePicker

@Composable
fun DetailTimePickerDialog(
    initialTime: String, onTimeSelected: (String) -> Unit,
    onDismiss: () -> Unit) {
    ...
    TimePickerDialog(onDismissRequest = { onDismiss() },
                     onConfirm = {
        onTimeSelected(selectedTime) ...
    }) {
        TimePicker(state = timePickerState)
    }
}
```

iOS Side

Detail Screen

- Time Pickers - update state




```
struct DetailsScreen: View {
    @State private var startTime = Date.now
    @State private var endTime = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $startTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

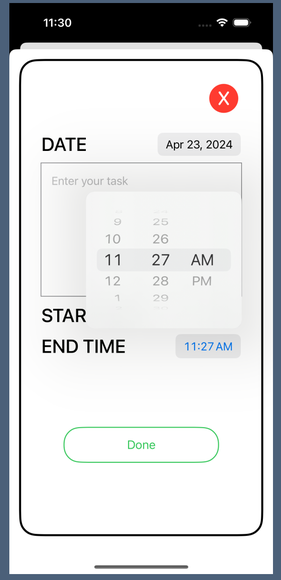
- Time Pickers - update state


```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()
    OutlinedCard(
        ....
    ) {// For the end time picker
        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
            initialTime = uiState.endTime,
            onTimeSelected = taskDetailViewModel::updateEndTime
        )
        ...
    }
}

class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val _detailState = MutableStateFlow(DetailState(false))
    val detailState: StateFlow<DetailState> = _detailState
    ...
    fun updateEndTime(endTime: String) {
        _detailState.update { it.copy(endTime = endTime) }
    }
}
```

iOS Side

Detail Screen

- state updated in screen and / or view model but NOT in Task Model yet



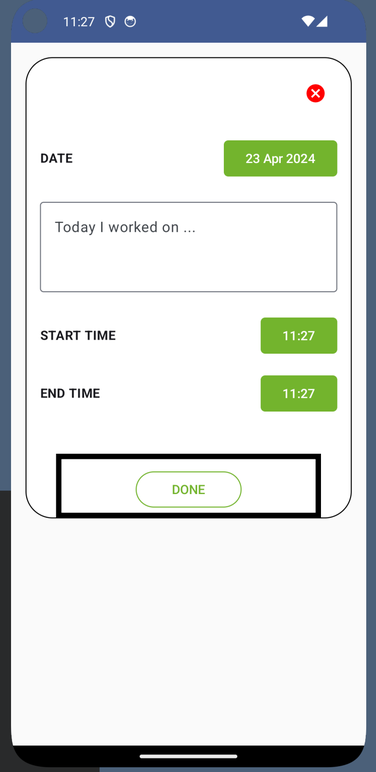

```
struct DetailsScreen: View {
    @State private var startTime = Date.now
    @State private var endTime = Date.now
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $startTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime,
                           displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

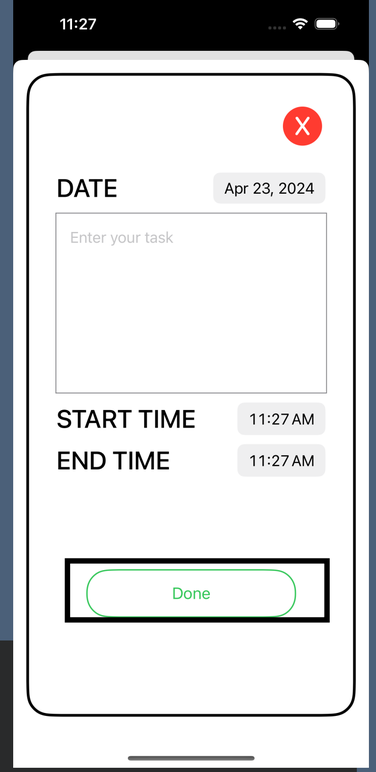
- Display Done Button




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
        shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {
        ...
        DetailDateButton(
            initialDate = uiState.date,
        )

        OutlinedTextField(
            placeholder = { Text(text = stringResource(id = R.string.textfield_label)) },
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.start_time_label),
        )

        TimePickerRow(
            timeRowLabel = stringResource(id = R.string.end_time_label),
        )

        OutlinedButton(
            },
        ) {
            Text(text = stringResource(id = R.string.done).uppercase())
        }
    }
}
```

iOS Side

Detail Screen



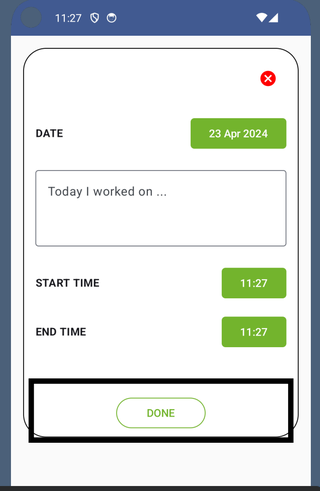

```
struct DetailsScreen: View {
    ...
    var body: some View {
        ZStack {
            VStack(spacing: 10) {
                ...
                DatePicker(selection: $taskDate, displayedComponents: .date) {
                    LeftTitleText(text: "Date")
                }

                TextField("Enter your task", text: $taskText)

                DatePicker(selection: $startTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "Start Time")
                }

                DatePicker(selection: $endTime, displayedComponents: .hourAndMinute) {
                    LeftTitleText(text: "End Time")
                }

                DoneButton(shouldDismiss: $shouldDismiss, taskText: $taskText, taskDate: $taskDate,
                           startTime: $startTime, endTime: $endTime, task: task )
            }
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Done Button

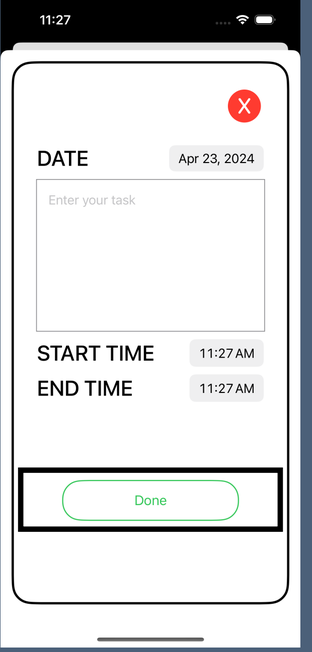



```kotlin
import com.example.tasktracker.TimeUtil.Companion.calculateDuration
import androidx.compose.material3.OutlinedButton

@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()

    OutlinedButton(
        onClick = {
            val duration = calculateDuration(uiState.startTime, uiState.endTime)
            val newTask = Task(
                id = uiState.taskId,
                activityName = uiState.activityName,
                date = uiState.date,
                startTimeInMillis = uiState.startTime,
                endTimeInMillis = uiState.endTime,
                duration = duration
            )
                ...
        },
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}
```

iOS Side

Detail Screen


```swift
struct DetailsScreen: View {
  @State private var shouldDismiss: Bool = false
   ...
  let task: Task?

  var body: some View {
    ZStack {
      VStack(spacing: 10) {
        DoneButton(shouldDismiss: $shouldDismiss,
                   taskText: $taskText,
                   taskDate: $taskDate,
                   startTime: $startTime,
                   endTime: $endTime,
                   task: task)
                ...
}

struct DoneButton: View {
    var body: some View {
      Button("Done") {
                ....
}
```

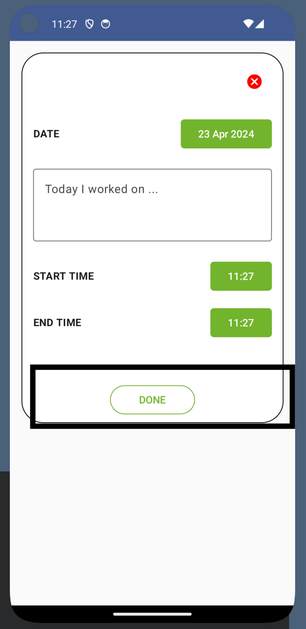


linktr.ee/sunfishgurl

Android Side

Detail Screen

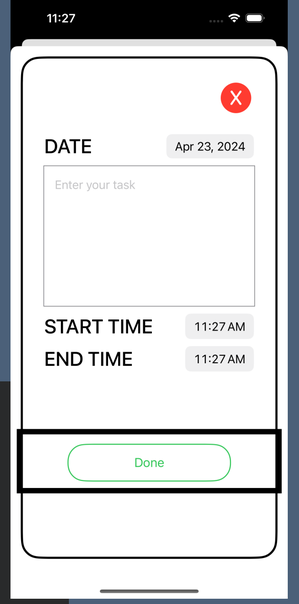
- Tapping on Done Button - Save New Task




```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()

    OutlinedButton(
        onClick = {
            val newTask = Task(
                ...
            )
            if (uiState.isEditMode) {
                    taskDetailViewModel.updateTask(newTask)
                } else {
                    taskDetailViewModel.insertTask(newTask)
                }
                onNavigateToList()
            },
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}
```

iOS Side

Detail Screen



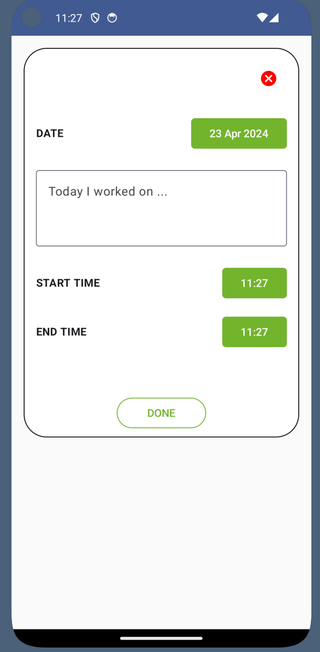

```
struct DetailsScreen: View {
  var body: some View {
        ZStack {
            VStack(spacing: 10) {
                DoneButton( ... )
}
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        Button("Done") {
            if let currentTask = task {
                currentTask.name = taskText
                currentTask.date = taskDate
                currentTask.startTime = startTime
                currentTask.endTime = endTime
            } else {
                var newTask = Task(name: taskText,
                                   date: taskDate,
                                   startTime: startTime,
                                   endTime: endTime )
                modelContext.insert(newTask)
            }
            ....
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Task Model

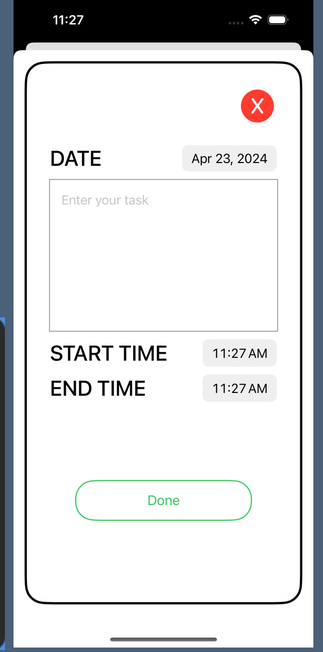



```kotlin
@Entity(tableName = "table_task")
data class Task(
    @PrimaryKey(autoGenerate = true)
    val id: Int = 0,
    val activityName: String,
    val date: String,
    val startTimeInMillis: String,
    val endTimeInMillis: String,
    val duration: String
)
```

iOS Side

Detail Screen

```swift
class DetailsViewModel: ObservableObject {
    @Published var task: Task?
    ....
    }
}
```



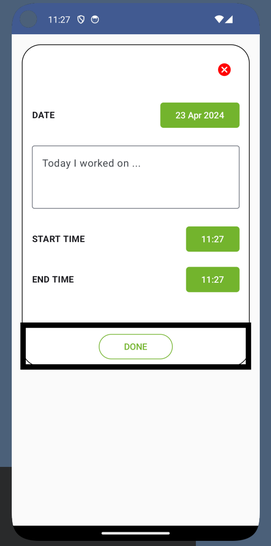

```swift
@Model
class Task {
    var name: String
    var date: Date
    var startTime: Date
    var endTime: Date

    // Calculate the duration between startTime and endTime
    var duration: String {
        let duration = endTime.timeIntervalSince(startTime)
        let hours = Int(duration) / 3600
        let minutes = Int(duration) / 60 % 60
        let seconds = Int(duration) % 60
        return String(format: "%02i:%02i:%02i", hours, minutes, seconds)
    }
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Done Tapped - Save New Task




```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()
    OutlinedCard( ...
    ) {
        OutlinedButton(
        onClick = { ...
            taskDetailViewModel.insertTask(newTask)
        }
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}

import androidx.lifecycle.viewModelScope

class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val repository: TaskRepository, savedStateHandle: SavedStateHandle
    fun insertTask(task: Task) {
        viewModelScope.launch {
            withContext(Dispatchers.IO) {
                repository.insertTask(task)
    ...
}
```

iOS Side

Detail Screen

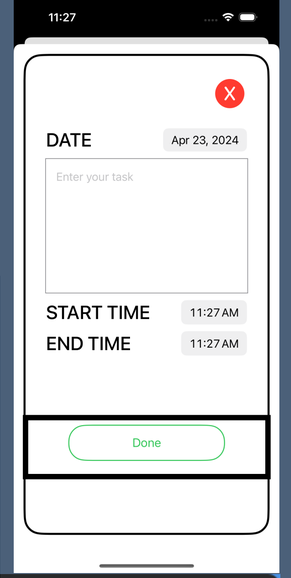
Instance Property

modelContext

The SwiftData model context that will be used for queries and other model operations within this environment.

SwiftData | SwiftUI | iOS 17.0+ | iPadOS 17.0+ | macOS 14.0+ | tvOS 17.0+ | watchOS 10.0+

```
var modelContext: ModelContext { get set }
```



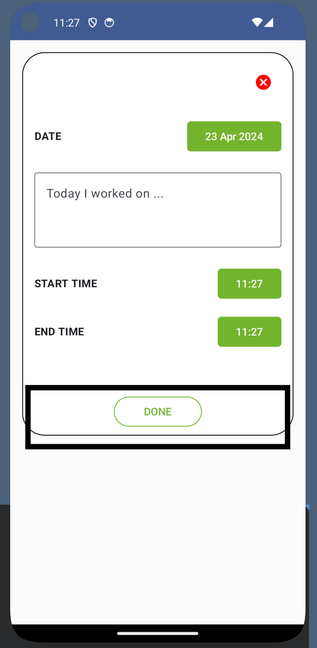

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        Button("Done") {
            if let currentTask = task {
                ...
            } else {
                var newTask = Task(name: taskText,
                                   date: taskDate,
                                   startTime: startTime,
                                   endTime: endTime )
                modelContext.insert(newTask)
            }
            ....
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

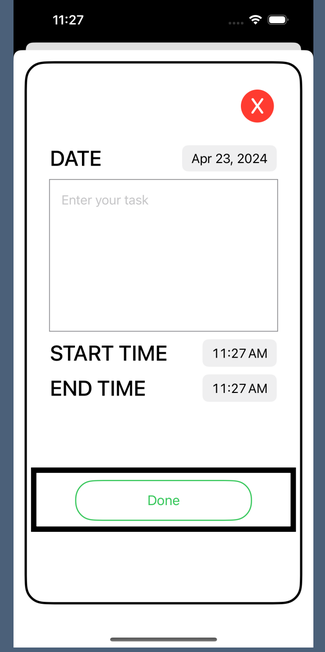
- Done Tapped - Save New Task




```
import androidx.lifecycle.viewModelScope
import androidx.room.Dao
import androidx.room.Insert

class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val repository: TaskRepository, savedStateHandle: SavedStateHandle
    fun insertTask(task: Task) {
        viewModelScope.launch {
            withContext(Dispatchers.IO) {
                repository.insertTask(task)
    ...
}
class TaskRepository @Inject constructor(private val taskDAO: TaskDAO) : ITaskRepository {
    var allTasks: Flow<List<Task>> = taskDAO.getAllTasks()
    override suspend fun insertTask(task: Task) {
        taskDAO.insertTask(task)
    }
}
@Dao
interface TaskDAO {
    @Insert
    fun insertTask(newTask: Task)
}
```

iOS Side

Detail Screen

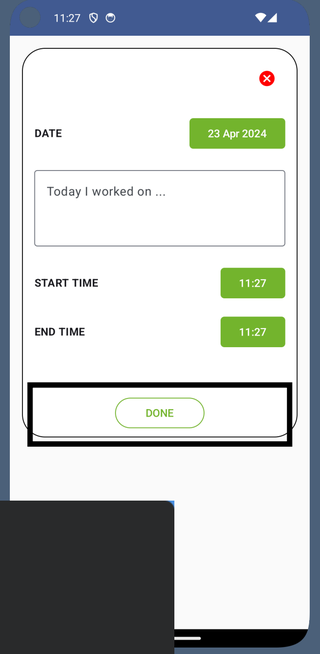


linktr.ee/sunfishgurl

Android Side

Detail Screen

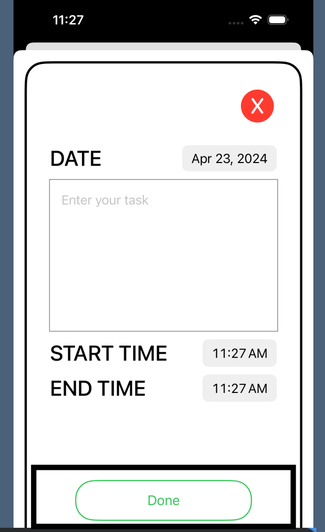
- Done Button - Navigate back to list




```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()

    OutlinedButton(
        onClick = {
            val newTask = Task(
                ...
            )
            if (uiState.isEditMode) {
                taskDetailViewModel.updateTask(newTask)
            } else {
                taskDetailViewModel.insertTask(newTask)
            }
            onNavigateToList()
        },
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}
```

iOS Side

Detail Screen



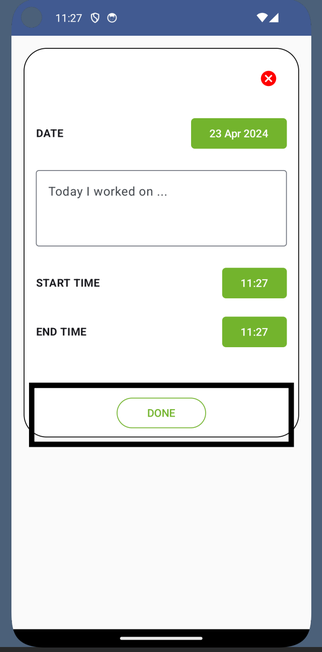

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        Button("Done") {
            if let currentTask = task {
                ...
            } else {
                var newTask = Task(name: taskText,
                                   date: taskDate,
                                   startTime: startTime,
                                   endTime: endTime )
                modelContext.insert(newTask)
            }
            shouldDismiss = true
            ...
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Done Button - Navigate back to list




```
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.rememberNavController

@Composable
fun MainScreen() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = NavScreens.TaskList.route) {
        composable("${NavScreens.TaskDetail.route}/?$TASK_ID_ARG={$TASK_ID_ARG}") {
            val viewModel = hiltViewModel<TaskDetailViewModel>()
            TaskDetailScreen(
                onNavigateToList = { navController.navigate(NavScreens.TaskList.route) },
                taskDetailViewModel = viewModel
            )
        }
    }
}
```

iOS Side

Detail Screen

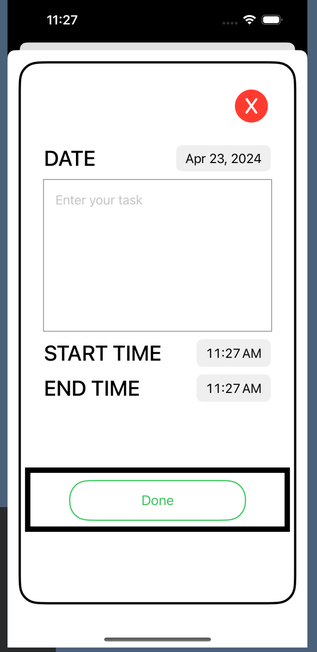
Instance Property

dismiss

An action that dismisses the current presentation.

iOS 15.0+ | iPadOS 15.0+ | macOS 12.0+ | Mac Catalyst 15.0+ | tvOS 15.0+ | watchOS 8.0+ | visionOS 1.0+

```
var dismiss: DismissAction { get }
```



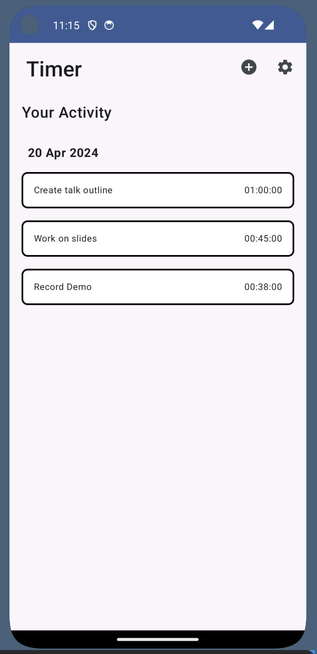

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        Button("Done") {
            ....
            modelContext.insert(newTask)
        }
        shouldDismiss = true
        ...
}
var body: some View {
    ZStack {
        VStack(spacing: 10) {
            DoneButton(shouldDismiss: $shouldDismiss .. )
        }
        .onChange(of: shouldDismiss) {
            if shouldDismiss {
                if viewModel.task != nil {
                    viewModel.updateTask(name: taskText,
                                         date: taskDate,
                                         startTime: startTime,
                                         endTime: endTime)
                }
                dismiss()
            }
}
```

linktr.ee/sunfishgurl

Android Side

List Screen - Return

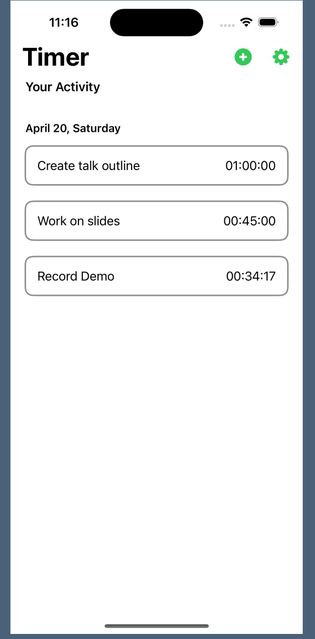
- Display added task




```kotlin
@Composable
fun TaskListScreen(
    onNavigateToSettings: () -> Unit,
    onNavigateToDetail: (id: Int?) -> Unit,
    taskListViewModel: TaskListViewModel
) {
    val allTasks by taskListViewModel.getAllTasks().collectAsState(emptyList())

    Scaffold(topBar = {
        ListScreenTopAppBar({ onNavigateToSettings() }, onNavigateToDetail)
    }) {
        TaskList(
            taskList = allTasks,
            onNavigateToDetail = onNavigateToDetail
        )
    }
}
```

iOS Side

List Screen - Return



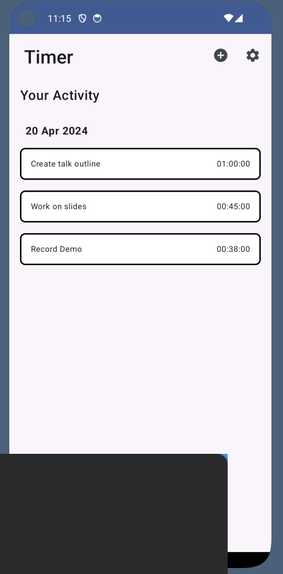

```swift
struct ListView: View {
    @ObservedObject var viewModel: ListViewModel
    @Query var tasks: [Task]

    var body: some View {
      NavigationStack {
        List {
            ForEach( ... ) {
             ...
                let duration = viewModel.formatDuration(start: task.startTime,
                                                        end: task.endTime)
                ActivityItemView(name: task.name, duration: duration)
                NavigationLink(destination: DetailsScreen()  ... )
 ...
```

linktr.ee/sunfishgurl

Android Side

List Screen - Return

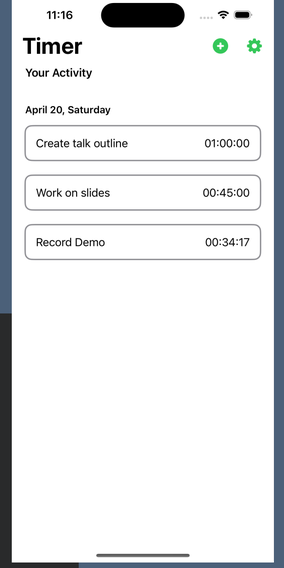
- Display added task




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen - Return

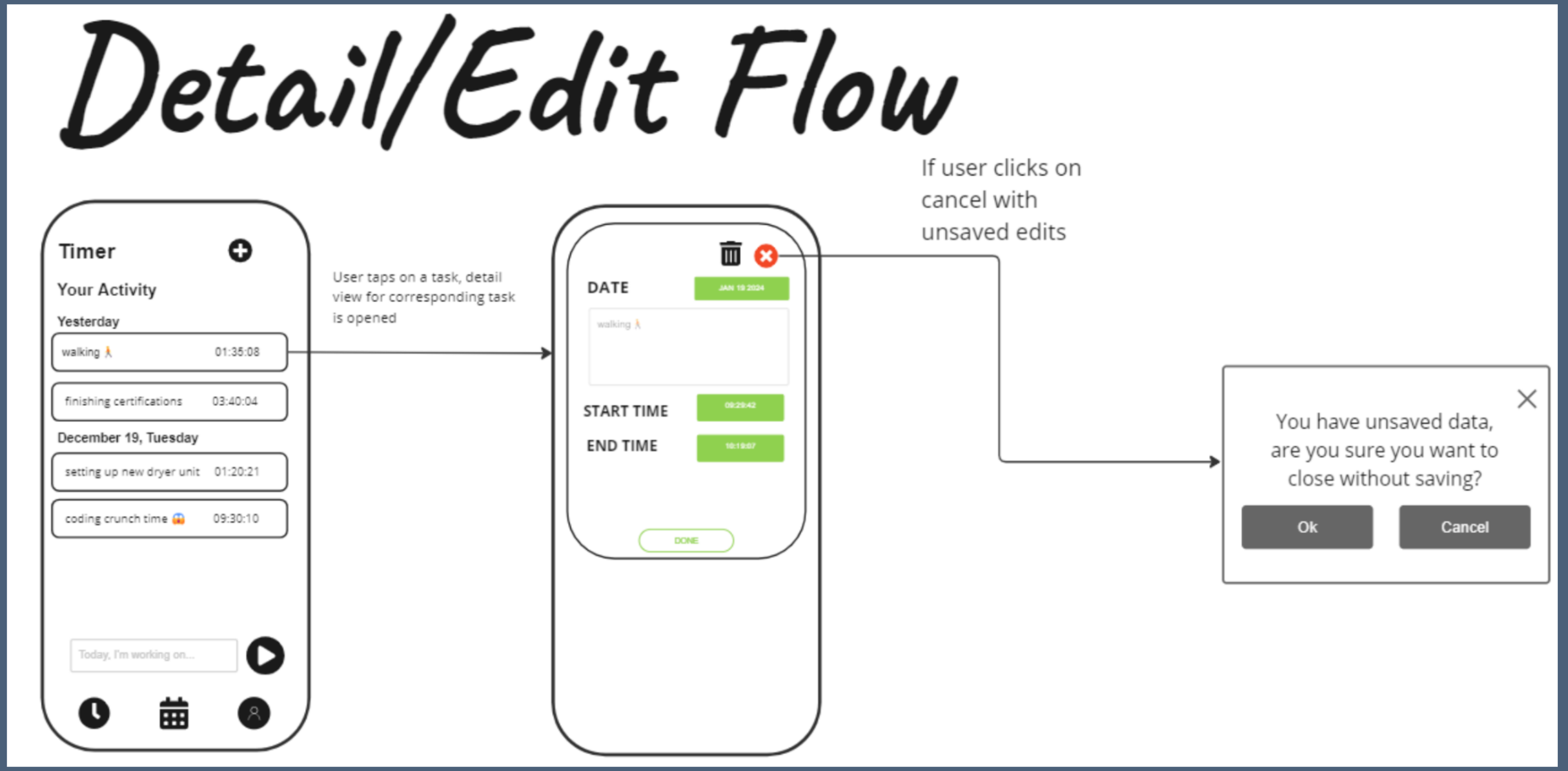



```
struct ListView: View {
     ...
    var body: some View {
       ...
         List {
             ForEach( ... ) {
               ActivityItemView(name: task.name,
                                duration: duration)
                NavigationLink(destination:
                                DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Edit Flow

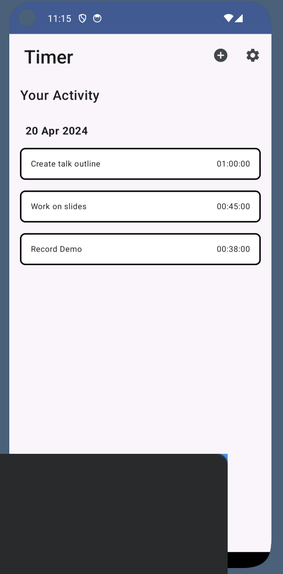
Edit Flow - Android and iOS



Android Side

List Screen

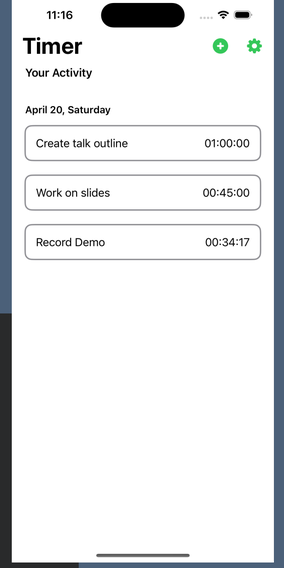
- Display list of tasks




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen



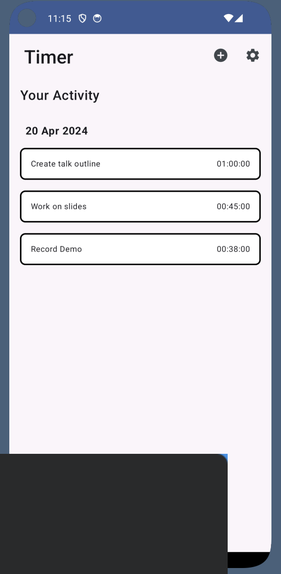

```
struct ListView: View {
     ...
    var body: some View {
       ...
        List {
            ForEach( ... ) {
                ActivityItemView(name: task.name,
                                 duration: duration)
                NavigationLink(destination:
                                 DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Android Side

List Screen

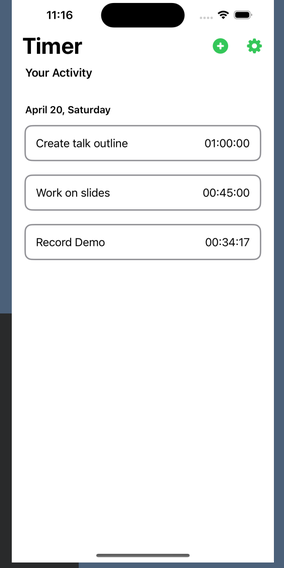
- Tap on Row - Navigate to Detail Screen




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen



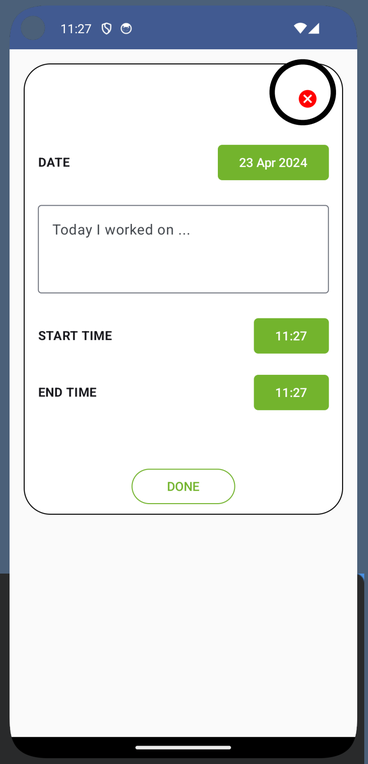

```
struct ListView: View {
     ...
    var body: some View {
       ...
        List {
            ForEach( ... ) {
              ActivityItemView(name: task.name,
                               duration: duration)
              NavigationLink(destination:
                             DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

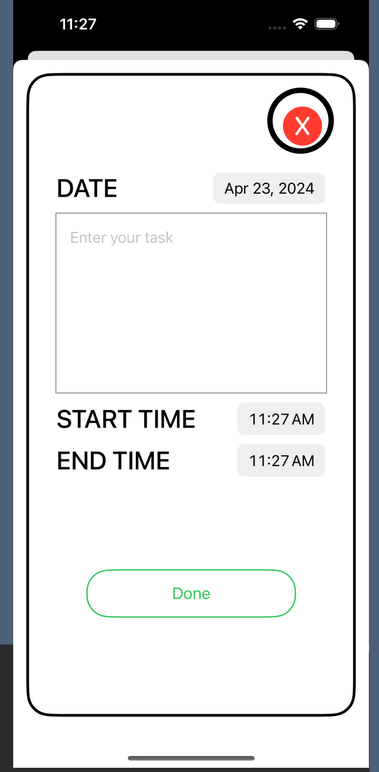
- Display Cancel Button




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
      shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {
        Row( ... )
        {
            if (uiState.isEditMode) {
                DeleteButton (onClick = {setShowDeleteConfirmationPopup(true)})
            }
            CancelButton(onClick = { setShowCancelConfirmationPopup(true) })
        }
        DetailDateButton(
            initialDate = uiState.date,
        )
        ...
    }
}
```

iOS Side

Detail Screen



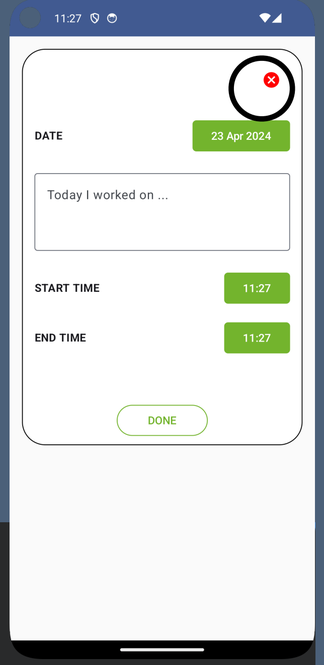

```
struct DetailsScreen: View {
  var body: some View {
        ZStack {
            VStack(spacing: 10) {
              TopBarView( ... )

struct TopBarView: View {
    var body: some View {
        HStack(alignment: .lastTextBaseline, spacing: 16) {
            if isEditingMode {
                Button(action: {
                    showDeleteConfirmationPopup = true
                }
              Button(action: {
                  showCancelConfirmationPopup = true
              }
            }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

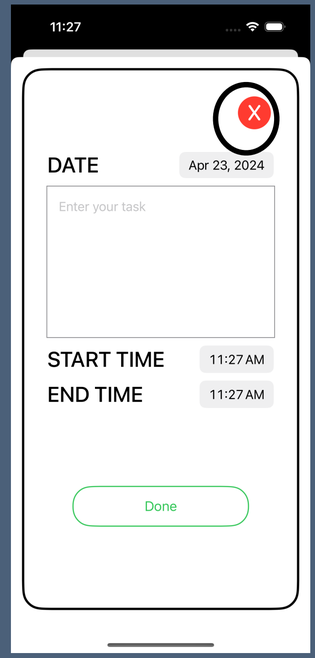
- Display Cancel Button




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard()
    {
        CancelButton(onClick = { setShowCancelConfirmationPopup(true) })
    }
}

import androidx.compose.material3.IconButton
@Composable
fun CancelButton(onClick: () → Unit) {
    IconButton(onClick = onClick) {
        Icon(
            painter = painterResource(id = R.drawable.baseline_cancel_24),
            contentDescription = stringResource(id = R.string.cancel),
            tint = Color.Red
        )
    }
}
```

iOS Side

Detail Screen

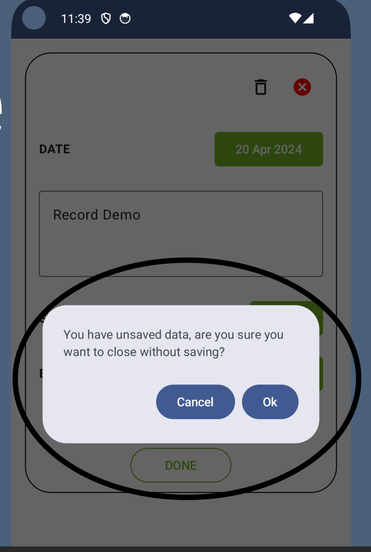


linktr.ee/sunfishgurl

Android Side

Detail Screen

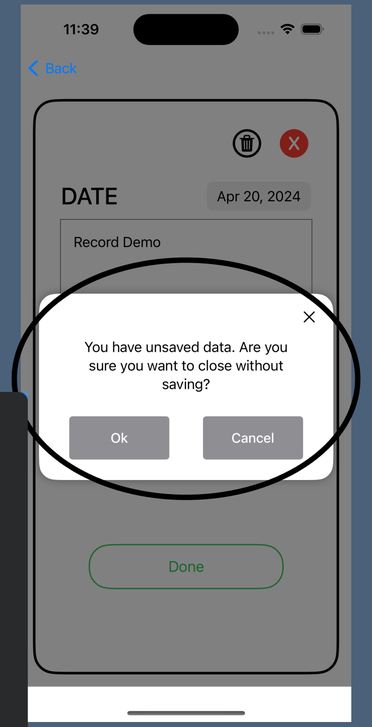
- Display Cancel Confirmation Dialog




```
if (showCancelConfirmationPopup) {
    ConfirmationDialog(message = stringResource(id = R.string.cancel_popup_message),
    onConfirm = onCancelConfirmed,
    onCancel = { setShowCancelConfirmationPopup(false) })
}

import androidx.compose.material3.AlertDialog
@Composable
fun ConfirmationDialog(
    message: String, onConfirm: () -> Unit, onCancel: () -> Unit
) {
    AlertDialog(onDismissRequest = onCancel, text = { Text(text = message) },
                confirmButton = {
        Button(onClick = { onConfirm() }) {
            Text(text = stringResource(id = R.string.ok))
        }
    }, dismissButton = {
        Button(onClick = { onCancel() }) {
            Text(text = stringResource(id = R.string.cancel))
        }
    })
}
```

iOS Side

Detail Screen



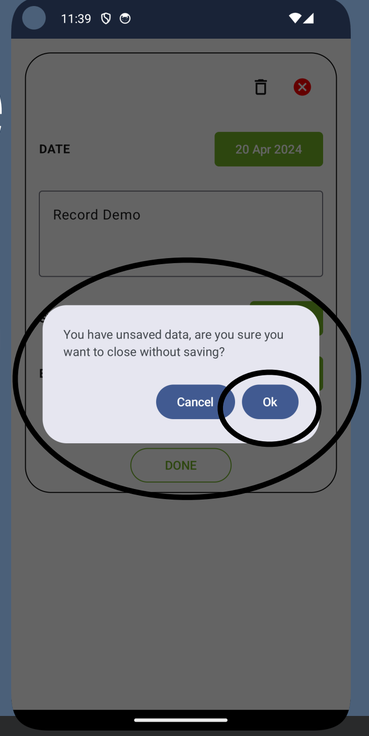

```
struct DetailsScreen: View {
    .....
    if $showCancelConfirmationPopup.wrappedValue {
        PopUpView(
            text: "You have unsaved data.  ... "
            firstButtonText: "Ok",
            secondButtonText: "Cancel",
            showButton: $showCancelConfirmationPopup,
            shouldDismiss: $shouldDismiss)
    }
struct PopUpView: View {
    init(text: String, firstButtonText: String,
         secondButtonText: String?,
         showButton: Binding<Bool>,
         shouldDismiss: Binding<Bool>) {
        .....
    }

    var body: some View {
            Text(text)
            Button {}
            Button {}
    }
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Display Cancel Confirmation Dialog: Tap Okay button
- Result: Dismiss Confirmation Dialog and return to List screen

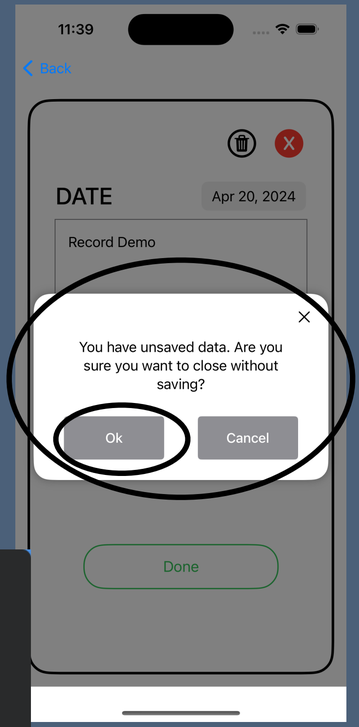



```
if (showCancelConfirmationPopup) {
    ConfirmationDialog(message =
        stringResource(id = R.string.cancel_popup_message),
    onConfirm = onCancelConfirmed,
    onCancel = { setShowCancelConfirmationPopup(false) })
}

val onCancelConfirmed = {
    setShowCancelConfirmationPopup(false)
    onNavigateToList()
}
```

iOS Side

Detail Screen

- Disclaimer: carbon.now.sh autocorrects != to ≠ 😥



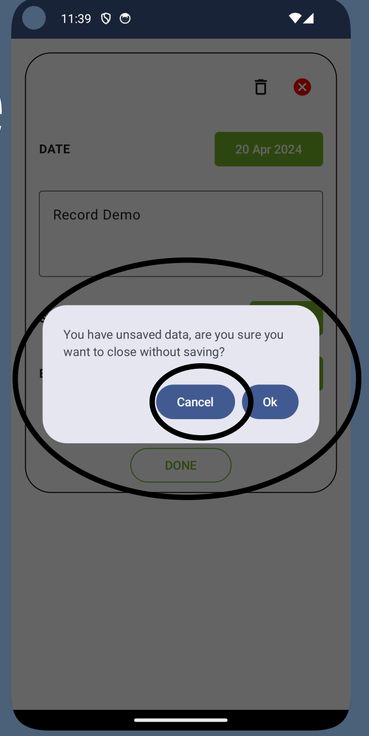

```
struct DetailsScreen: View {
    .onChange(of: shouldDismiss) {
        if shouldDismiss {
            if viewModel.task != nil {
                viewModel.updateTask(name: taskText,
                                     date: taskDate,
                                     startTime: startTime,
                                     endTime: endTime)
            }
            dismiss()
        }
    }
    if $showCancelConfirmationPopup.wrappedValue {
        PopUpView(
            text: "You have unsaved data. ..."
            firstButtonText: "Ok",
            secondButtonText: "Cancel",
            showButton: $showCancelConfirmationPopup,
            shouldDismiss: $shouldDismiss)
    }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Display Cancel Confirmation Dialog: Tap Cancel button
- Result: Dismiss Confirmation Dialog and stay on Detail screen



Value captured in a closure

Move lambda argument out of parentheses ⌥⇧⏎ More actions... ⌥⏎

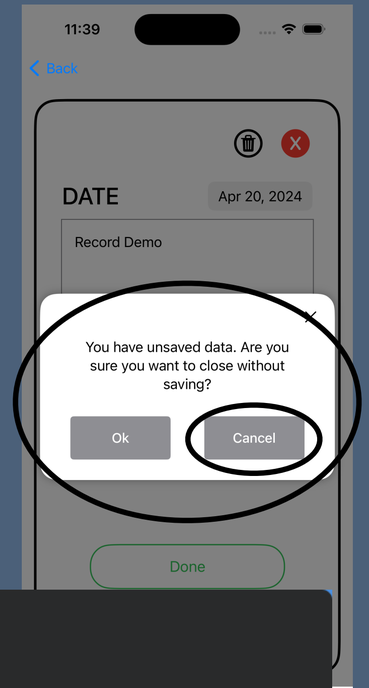
```
val setShowCancelConfirmationPopup: (Boolean) -> Unit
```

Task_Tracker.app.main

```
if (showCancelConfirmationPopup) {
    ConfirmationDialog(message =
        stringResource(id = R.string.cancel_popup_message),
    onConfirm = onCancelConfirmed,
    onCancel = { setShowCancelConfirmationPopup(false) })
}
```

iOS Side

Detail Screen



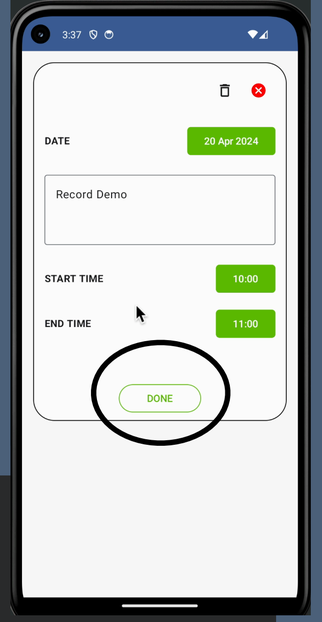

```
struct DetailsScreen: View {

  if showDeleteConfirmationPopup || showCancelConfirmationPopup {
    .onTapGesture {
        showDeleteConfirmationPopup = false
        showCancelConfirmationPopup = false
    }
  }
}
if $showCancelConfirmationPopup.wrappedValue {
    PopUpView(
        text: "You have unsaved data.  ... "
        firstButtonText: "Ok",
        secondButtonText: "Cancel",
        showButton: $showCancelConfirmationPopup,
        shouldDismiss: $shouldDismiss)
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

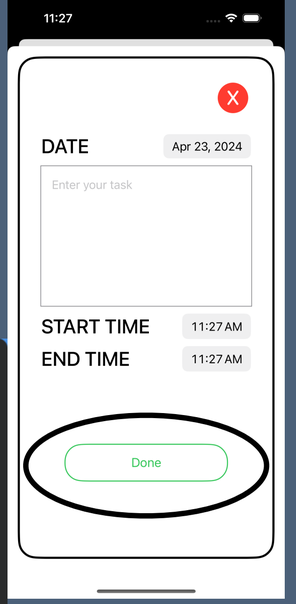
- Tapping on Done Button: Save Edited Task




```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()

    OutlinedButton(
        onClick = {
            val newTask = Task(
                ...
            )
            if (uiState.isEditMode) {
                taskDetailViewModel.updateTask(newTask)
            } else {
                taskDetailViewModel.insertTask(newTask)
            }
            onNavigateToList()
        },
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}
```

iOS Side

Detail Screen



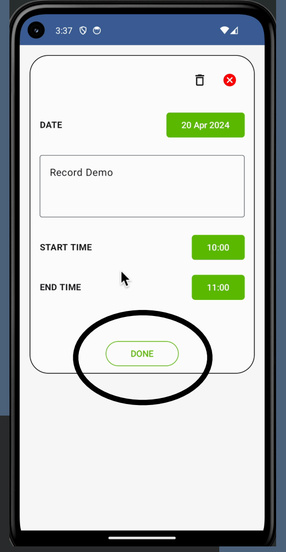

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        if let currentTask = task {
            currentTask.name = taskText
            currentTask.date = taskDate
            currentTask.startTime = startTime
            currentTask.endTime = endTime
        }
        shouldDismiss = true
        ...
}
var body: some View {
    ZStack {
        VStack(spacing: 10) {
            DoneButton(shouldDismiss: $shouldDismiss .. )
        }
        .onChange(of: shouldDismiss) {
            if shouldDismiss {
                if viewModel.task != nil {
                    viewModel.updateTask( ... )
                }
                dismiss()
            }
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

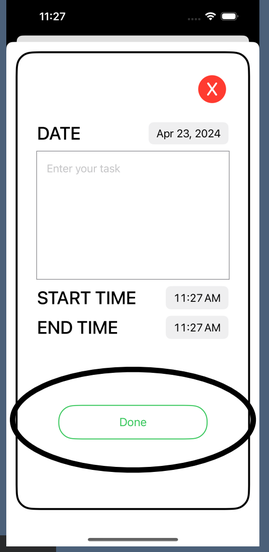
- Done Tapped - Save Edited Task




```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()
    OutlinedCard(  ...
    ) {
        OutlinedButton(
        onClick = {  ...
            taskDetailViewModel.updateTask(newTask)
        },
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}

import androidx.lifecycle.viewModelScope

class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val repository: TaskRepository, savedStateHandle: SavedStateHandle
    fun updateTask(task: Task) {
        viewModelScope.launch {
            withContext(Dispatchers.IO) {
                repository.updateTask(task)
    ...
}
```

iOS Side

Detail Screen



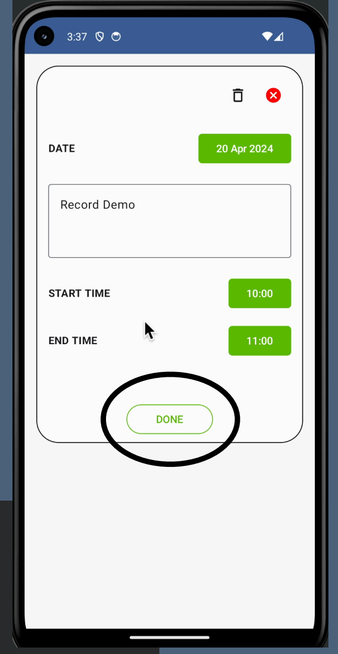

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
      DoneButton(shouldDismiss:
                 $shouldDismiss  ... )
}
var body: some View {
    ZStack {
        VStack(spacing: 10) {
            DoneButton(shouldDismiss ... )
        }
        .onChange(of: shouldDismiss) {
            if shouldDismiss {
              if viewModel.task ≠ nil {
                viewModel.updateTask( ... )
            }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

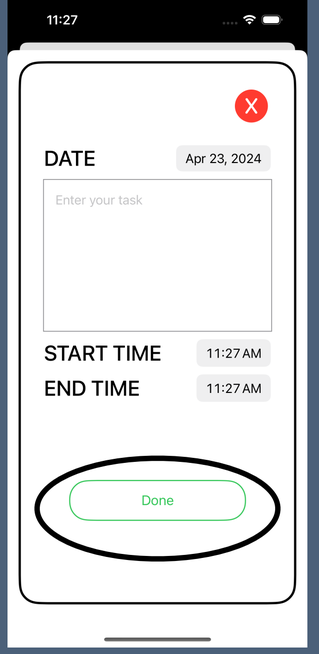
- Done Button - Navigate back to list




```
@Composable
fun TaskDetailScreen( ... )
{
    val uiState by taskDetailViewModel.detailState.collectAsState()

    OutlinedButton(
        onClick = {
            val newTask = Task(
                ...
            )
            if (uiState.isEditMode) {
                taskDetailViewModel.updateTask(newTask)
            } else {
                taskDetailViewModel.insertTask(newTask)
            }
            onNavigateToList()
        },
    ) {
        Text(text = stringResource(id = R.string.done).uppercase())
    }
}
```

iOS Side

Detail Screen



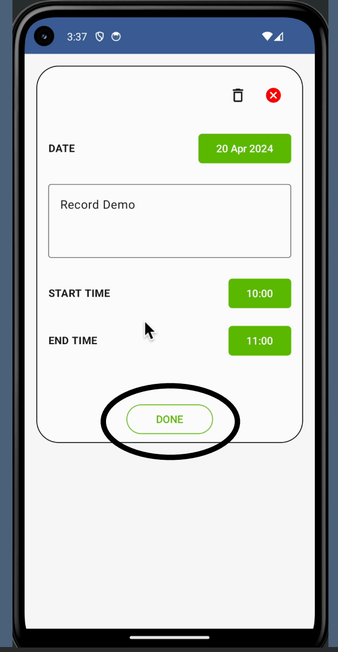

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        if let currentTask = task {
            currentTask.name = taskText
            ...
        }
        shouldDismiss = true
        ...
}
var body: some View {
    ZStack {
        VStack(spacing: 10) {
            DoneButton(shouldDismiss ... )
        }
        .onChange(of: shouldDismiss) {
            if shouldDismiss {
              if viewModel.task ≠ nil {
                viewModel.updateTask( ... )
            }
            dismiss()
        }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Done Button - Navigate back to list




```
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.rememberNavController

@Composable
fun MainScreen() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination = NavScreens.TaskList.route) {
        composable("${NavScreens.TaskDetail.route}/?$TASK_ID_ARG={$TASK_ID_ARG}") {
            val viewModel = hiltViewModel<TaskDetailViewModel>()
            TaskDetailScreen(
                onNavigateToList = { navController.navigate(NavScreens.TaskList.route) },
                taskDetailViewModel = viewModel
            )
        }
    }
}
```

iOS Side

Detail Screen

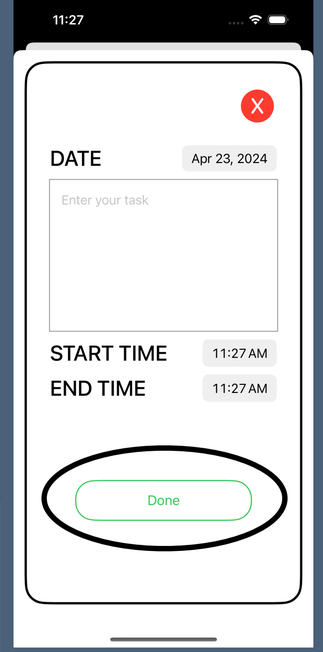
Instance Property

dismiss

An action that dismisses the current presentation.

iOS 15.0+ | iPadOS 15.0+ | macOS 12.0+ | Mac Catalyst 15.0+ | tvOS 15.0+ | watchOS 8.0+ | visionOS 1.0+

```
var dismiss: DismissAction { get }
```



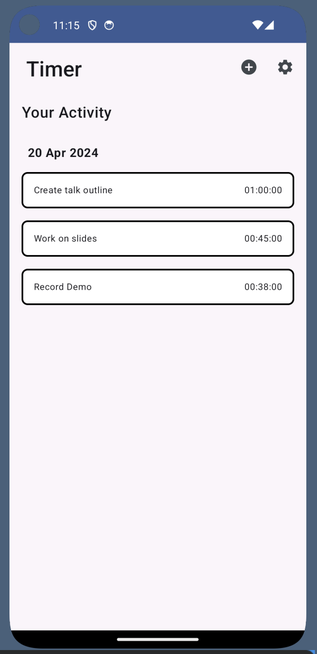

```
struct DoneButton: View {
    @Environment(\.modelContext) var modelContext
    var body: some View {
        if let currentTask = task {
            currentTask.name = taskText
            ...
        }
        shouldDismiss = true
        ...
}
var body: some View {
    ZStack {
        VStack(spacing: 10) {
            DoneButton(shouldDismiss ... )
        }
        .onChange(of: shouldDismiss) {
            if shouldDismiss {
              if viewModel.task ≠ nil {
                viewModel.updateTask( ... )
            }
            dismiss()
        }
```

linktr.ee/sunfishgurl

Android Side

List Screen - Return

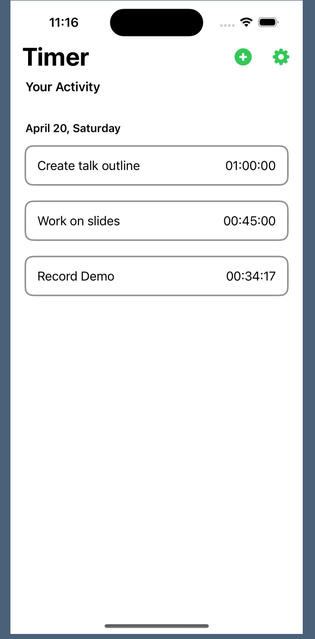
- Display updated task




```kotlin
@Composable
fun TaskListScreen(
    onNavigateToSettings: () -> Unit,
    onNavigateToDetail: (id: Int?) -> Unit,
    taskListViewModel: TaskListViewModel
) {
    val allTasks by taskListViewModel.getAllTasks().collectAsState(emptyList())

    Scaffold(topBar = {
        ListScreenTopAppBar({ onNavigateToSettings() }, onNavigateToDetail)
    }) {
        TaskList(
            taskList = allTasks,
            onNavigateToDetail = onNavigateToDetail
        )
    }
}
```

iOS Side

List Screen - Return



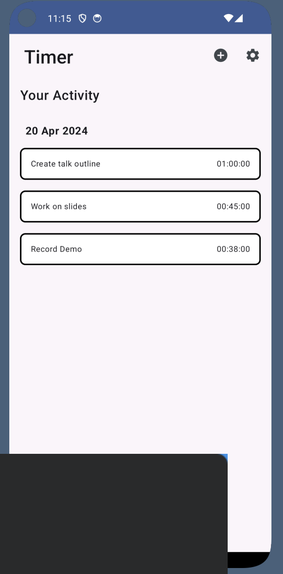

```swift
struct ListView: View {
    @ObservedObject var viewModel: ListViewModel
    @Query var tasks: [Task]

    var body: some View {
      NavigationStack {
        List {
            ForEach( ... ) {
             ...
                let duration = viewModel.formatDuration(start: task.startTime,
                                                        end: task.endTime)
                ActivityItemView(name: task.name, duration: duration)
                NavigationLink(destination: DetailsScreen()  ... )
 ...
```

linktr.ee/sunfishgurl

Android Side

List Screen - Return

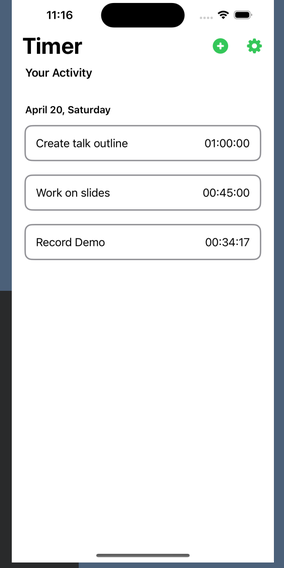
- Display updated task




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen - Return

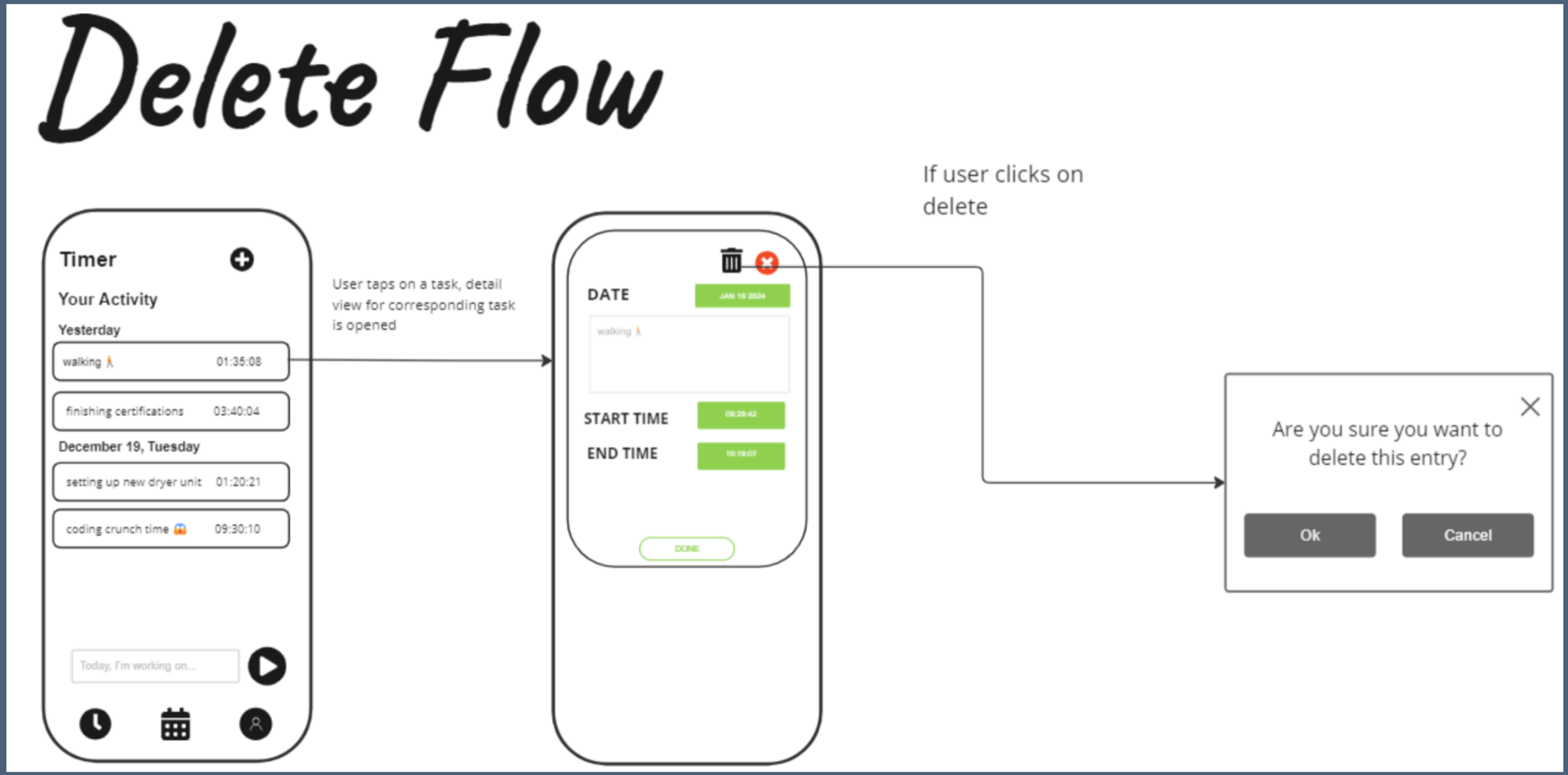



```
struct ListView: View {
     ...
    var body: some View {
       ...
          List {
              ForEach( ... ) {
                ActivityItemView(name: task.name,
                                 duration: duration)
                 NavigationLink(destination:
                                DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Delete Flow

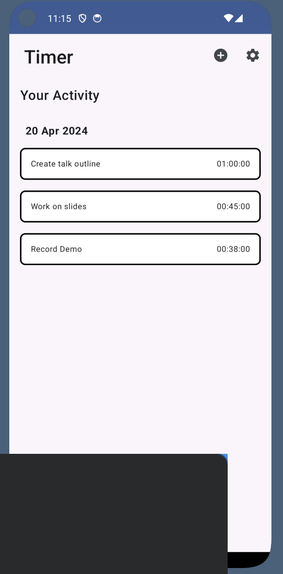
Delete - Android and iOS



Android Side

List Screen

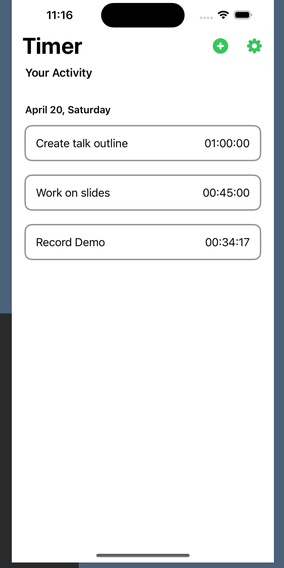
- Display list of tasks




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen



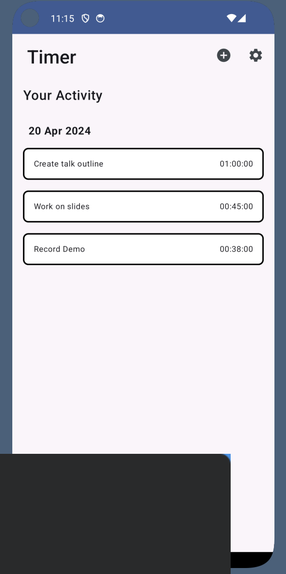

```
struct ListView: View {
     ...
    var body: some View {
       ...
        List {
            ForEach( ... ) {
                ActivityItemView(name: task.name,
                                 duration: duration)
                NavigationLink(destination:
                                 DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Android Side

List Screen

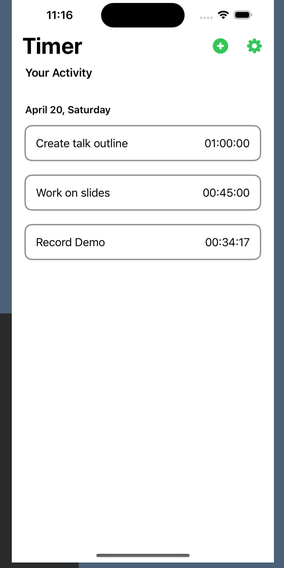
- Tap on Row - Navigate to Detail Screen




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen



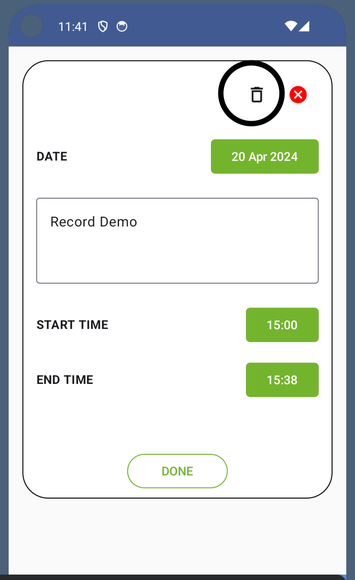

```
struct ListView: View {
     ...
    var body: some View {
       ...
        List {
            ForEach( ... ) {
              ActivityItemView(name: task.name,
                               duration: duration)
            NavigationLink(destination:
                           DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Display Delete Button




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard(
      shape = RoundedCornerShape(dimensionResource(R.dimen.detail_card_shape))
    ) {
        Row( ... )
        {
            if (uiState.isEditMode) {
                DeleteButton (onClick = {setShowDeleteConfirmationPopup(true)})
            }
            CancelButton(onClick = { setShowCancelConfirmationPopup(true) })
        }
        DetailDateButton(
            initialDate = uiState.date,
        )
        ...
    }
}
```

iOS Side

Detail Screen

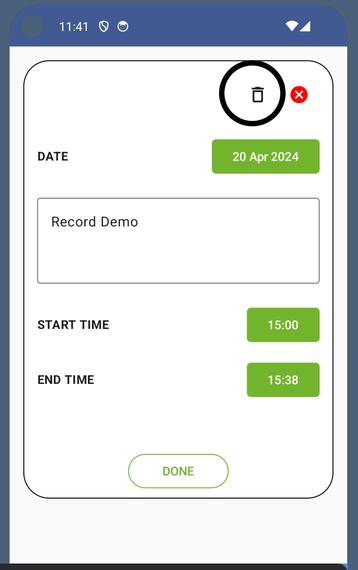

```
struct DetailsScreen: View {
  var body: some View {
        ZStack {
            VStack(spacing: 10) {
                TopBarView( ... )

struct TopBarView: View {
    var body: some View {
        HStack(alignment: .lastTextBaseline, spacing: 16) {
            if isEditingMode {
                Button(action: {
                    showDeleteConfirmationPopup = true
                }
                Button(action: {
                    showCancelConfirmationPopup = true
                }
            }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

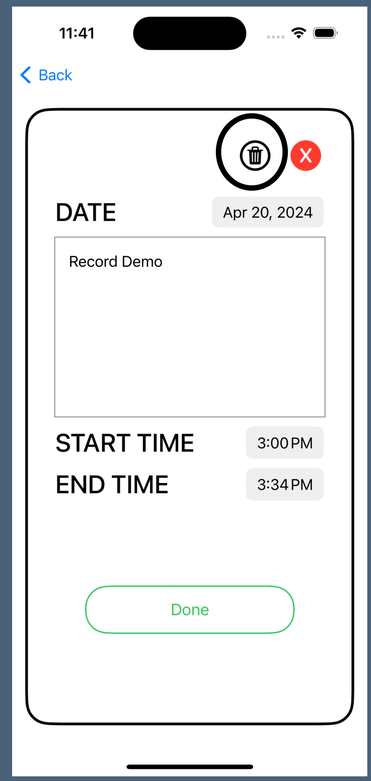
- Display Delete Button




```
@Composable
fun TaskDetailScreen( ... )
{
    OutlinedCard()
    {
        DeleteButton (onClick = {setShowDeleteConfirmationPopup(true)})
    }
}

@Composable
fun DeleteButton(onClick: () → Unit) {
    IconButton(onClick = onClick) {
        Icon(
            painter = painterResource(id = R.drawable.outline_delete_24),
            contentDescription = stringResource(id = R.string.delete),
        )
    }
}
```

iOS Side

Detail Screen

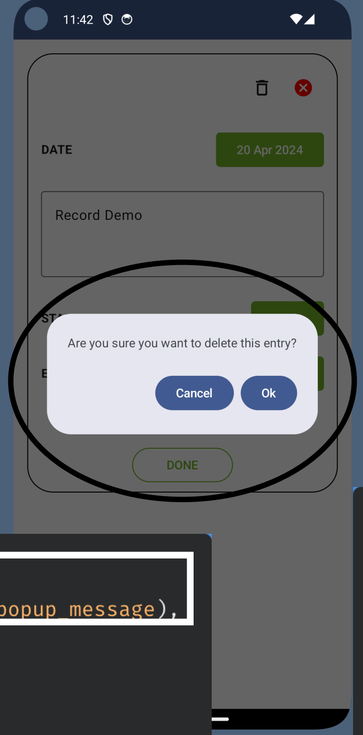


linktr.ee/sunfishgurl

Android Side

Detail Screen

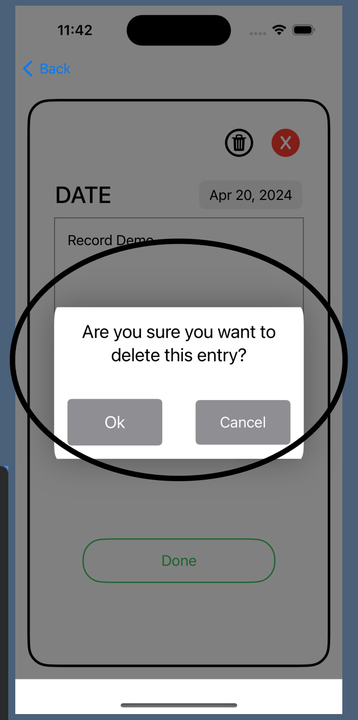
- Display Delete Confirmation Dialog




```
if (showDeleteConfirmationPopup) {
    ConfirmationDialog(message = stringResource(id = R.string.delete_popup_message),
    onConfirm = {
        setShowDeleteConfirmationPopup(false)
        taskDetailViewModel.deleteTask(task)
        onNavigateToList()
    },
    onCancel = {setShowDeleteConfirmationPopup(false)})
}

import androidx.compose.material3.AlertDialog
@Composable
fun ConfirmationDialog(
    message: String, onConfirm: () -> Unit, onCancel: () -> Unit
) {
    AlertDialog(onDismissRequest = onCancel, text = { Text(text = message) },
                confirmButton = {
        Button(onClick = { onConfirm() }) {
            Text(text = stringResource(id = R.string.ok))
        }
    }, dismissButton = {
        Button(onClick = { onCancel() }) {
            Text(text = stringResource(id = R.string.cancel))
        }
    })
}
```

iOS Side

Detail Screen



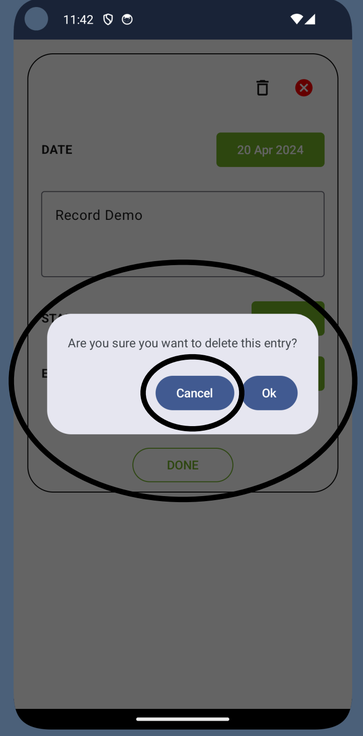

```
struct DetailsScreen: View {

    if $showDeleteConfirmationPopup.wrappedValue {
        if let unwrappedTask = task {
            DeleteConfirmationPopupView(
              showDeleteConfirmationPopup: ...
              task: ... )
            }
    }
}
private struct DeleteConfirmationPopupView: View {
    @Environment(\.modelContext) var modelContext
    @Environment(\.presentationMode) var presentationMode
    @Binding var showDeleteConfirmationPopup: Bool
    let task: Task

    var body: some View {
     ...
        Text("Are you sure you want to delete this entry?")
        Button("Ok") {
            modelContext.delete(task)
            self.showDeleteConfirmationPopup = false
            presentationMode.wrappedValue.dismiss()
        }
        Button("Cancel") {
            self.showDeleteConfirmationPopup = false
        }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Display Delete Confirmation Dialog: Tap Cancel button
- Result: Dismiss Confirmation Dialog and stay on Detail screen



Value captured in a closure

Move lambda argument out of parentheses ⌥⇧⏎ More actions... ⌥⏎

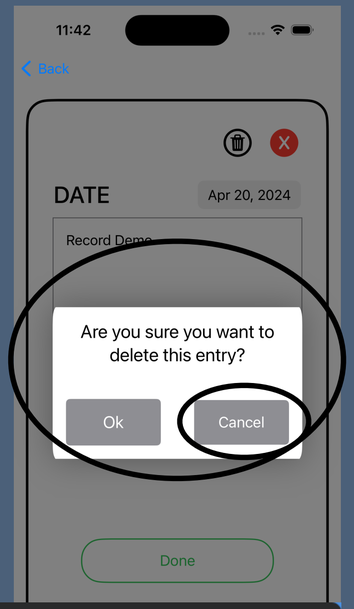
```
val setShowDeleteConfirmationPopup: (Boolean) -> Unit
```

Task_Tracker.app.main

```
if (showDeleteConfirmationPopup) {
    ConfirmationDialog(message = stringResource(id = R.string.delete_popup_message),
    onConfirm = {
        setShowDeleteConfirmationPopup(false)
        taskDetailViewModel.deleteTask(task)
        onNavigateToList()
    },
    onCancel = {setShowDeleteConfirmationPopup(false)})
}
```

iOS Side

Detail Screen



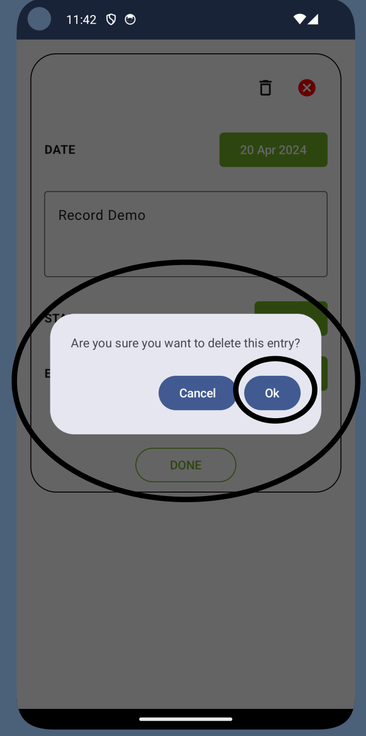

```
private struct DeleteConfirmationPopupView: View {
    @Environment(\.modelContext) var modelContext
    @Environment(\.presentationMode) var presentationMode
    @Binding var showDeleteConfirmationPopup: Bool
    let task: Task

    var body: some View {
     ...
        Text("Are you sure you want to delete this entry?")
        Button("Ok") {
            modelContext.delete(task)
            self.showDeleteConfirmationPopup = false
            presentationMode.wrappedValue.dismiss()
        }
        Button("Cancel") {
            self.showDeleteConfirmationPopup = false
        }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

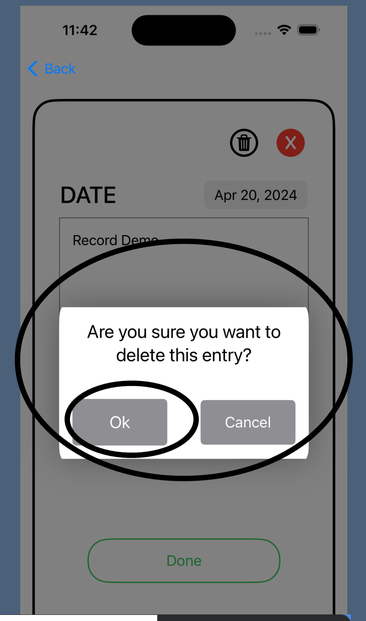
- Display Delete Confirmation Dialog: Tap Okay button
- Result:
  - ✅ Dismiss Confirmation Dialog
  - Delete Task
  - Return to List screen




```
if (showDeleteConfirmationPopup) {
    ConfirmationDialog(
      message = stringResource(id = R.string.delete_popup_message),
    onConfirm = {
        setShowDeleteConfirmationPopup(false)
        taskDetailViewModel.deleteTask(task)
        onNavigateToList()
    },
    onCancel = {setShowDeleteConfirmationPopup(false)})
}
```

iOS Side

Detail Screen



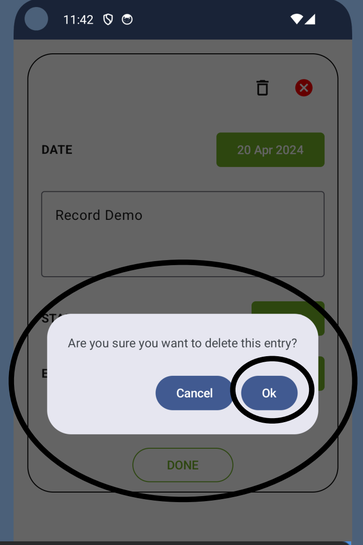

```
private struct DeleteConfirmationPopupView: View {
    @Environment(\.modelContext) var modelContext
    @Environment(\.presentationMode) var presentationMode
    @Binding var showDeleteConfirmationPopup: Bool
    let task: Task

    var body: some View {
    ...
        Text("Are you sure you want to delete this entry?")
        Button("Ok") {
            modelContext.delete(task)
            self.showDeleteConfirmationPopup = false
            presentationMode.wrappedValue.dismiss()
        }
        Button("Cancel") {
            self.showDeleteConfirmationPopup = false
        }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Display Delete Confirmation Dialog: Tap Okay button

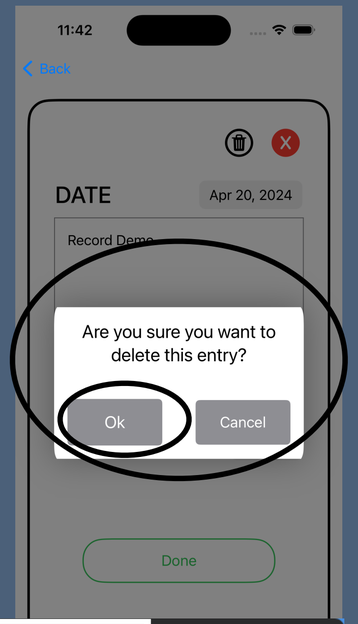



```
if (showDeleteConfirmationPopup) {
    ConfirmationDialog(
      message = stringResource(id = R.string.delete_popup_message),
    onConfirm = {
        setShowDeleteConfirmationPopup(false)
        taskDetailViewModel.deleteTask(task)
        onNavigateToList()
    },
    onCancel = {setShowDeleteConfirmationPopup(false)})
}

import androidx.lifecycle.viewModelScope
class TaskDetailViewModel @Inject constructor( ... ) : ViewModel() {
    private val repository: TaskRepository, savedStateHandle: SavedStateHandle
    fun deleteTask(task: Task) {
        viewModelScope.launch {
            withContext(Dispatchers.IO) {
                repository.deleteTask(task)
    ...
}
```

iOS Side

Detail Screen

- Result:
  - ✅ Dismiss Confirmation Dialog
  - ✅ Delete Task
  - Return to List screen



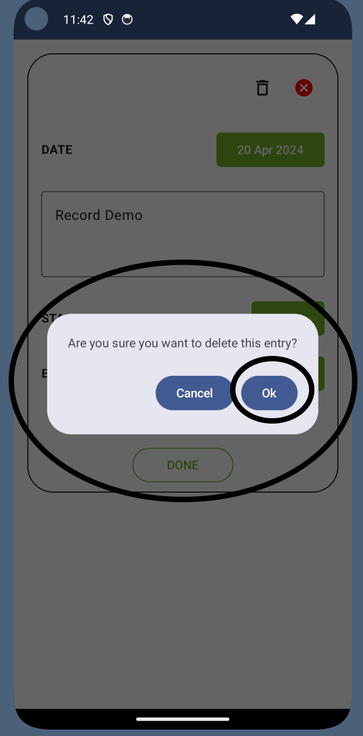

```
private struct DeleteConfirmationPopupView: View {
    @Environment(\.modelContext) var modelContext
    @Environment(\.presentationMode) var presentationMode
    @Binding var showDeleteConfirmationPopup: Bool
    let task: Task

    var body: some View {
    ...
        Text("Are you sure you want to delete this entry?")
        Button("Ok") {
            modelContext.delete(task)
            self.showDeleteConfirmationPopup = false
            presentationMode.wrappedValue.dismiss()
        }
        Button("Cancel") {
            self.showDeleteConfirmationPopup = false
        }
```

linktr.ee/sunfishgurl

Android Side

Detail Screen

- Display Delete Confirmation Dialog: Tap Okay button

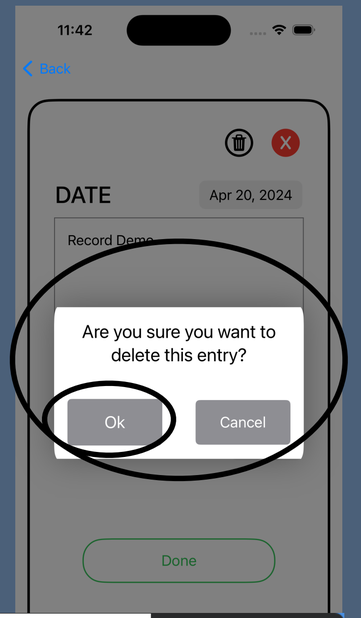



```
if (showDeleteConfirmationPopup) {
    ConfirmationDialog(
      message = stringResource(id = R.string.delete_popup_message),
    onConfirm = {
        setShowDeleteConfirmationPopup(false)
        taskDetailViewModel.deleteTask(task)
        onNavigateToList()
    },
    onCancel = {setShowDeleteConfirmationPopup(false)})
}
```

iOS Side

Detail Screen

- Result:
  - ✅ Dismiss Confirmation Dialog
  - ✅ Delete Task
  - ✅ Return to List screen



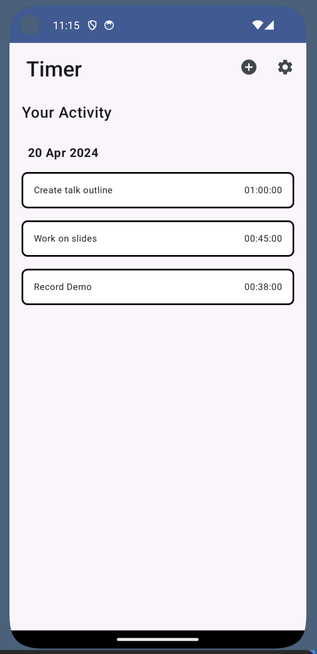

```
private struct DeleteConfirmationPopupView: View {
    @Environment(\.modelContext) var modelContext
    @Environment(\.presentationMode) var presentationMode
    @Binding var showDeleteConfirmationPopup: Bool
    let task: Task

    var body: some View {
    ...
        Text("Are you sure you want to delete this entry?")
        Button("Ok") {
            modelContext.delete(task)
            self.showDeleteConfirmationPopup = false
            presentationMode.wrappedValue.dismiss()
        }
        Button("Cancel") {
            self.showDeleteConfirmationPopup = false
        }
```

linktr.ee/sunfishgurl

Android Side

List Screen - Return

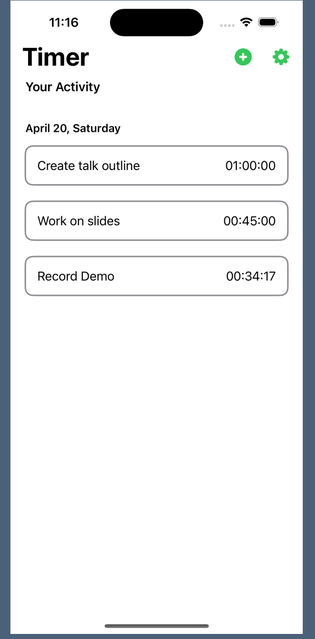
- Display updated tasks, deleted task gone




```kotlin
@Composable
fun TaskListScreen(
    onNavigateToSettings: () -> Unit,
    onNavigateToDetail: (id: Int?) -> Unit,
    taskListViewModel: TaskListViewModel
) {
    val allTasks by taskListViewModel.getAllTasks().collectAsState(emptyList())

    Scaffold(topBar = {
        ListScreenTopAppBar({ onNavigateToSettings() }, onNavigateToDetail)
    }) {
        TaskList(
            taskList = allTasks,
            onNavigateToDetail = onNavigateToDetail
        )
    }
}
```

iOS Side

List Screen - Return



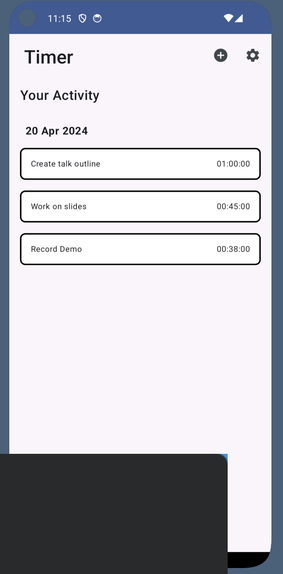

```swift
struct ListView: View {
    @ObservedObject var viewModel: ListViewModel
    @Query var tasks: [Task]

    var body: some View {
      NavigationStack {
        List {
            ForEach( ... ) {
             ...
                let duration = viewModel.formatDuration(start: task.startTime,
                                                        end: task.endTime)
                ActivityItemView(name: task.name, duration: duration)
                NavigationLink(destination: DetailsScreen()  ... )
 ...
```

linktr.ee/sunfishgurl

Android Side

List Screen - Return

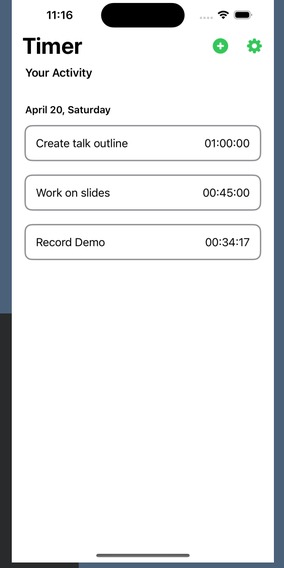
- Display updated tasks, deleted task gone




```
@Composable
fun TaskList(
    taskList: List<Task>, ... onNavigateToDetail: (id: Int?) → Unit
) {
    LazyColumn( ...
    ) { ...
        val grouped = taskList.groupBy { it.date }

        grouped.forEach { (date, tasks) →
            item {
                Header(date)
            }

            items(tasks) { task →
                TaskCard(task = task
                    ), onClick = { onNavigateToDetail(task.id) })
            }
        }
    }
}
```

iOS Side

List Screen - Return




```
struct ListView: View {
     ...
    var body: some View {
       ...
        List {
            ForEach( ... ) {
                ActivityItemView(name: task.name,
                                 duration: duration)
                NavigationLink(destination:
                                 DetailsScreen() ... )
}
struct ActivityItemView: View {
    let name: String
    let duration: String

    var body: some View {
        HStack {
            Text(name)
            Text(duration)
        }
    }
}
```

linktr.ee/sunfishgurl

Task Tracker App Demo: Android

https://bit.ly/TaskTrackerAppAndroidDemo

- Open Source Project for Women Who Code Mobile
- Tracks Tasks Being Worked On
- Tracks Time Spent on Tasks
- CRUD Operations

Putting it All Together

Tips for Building a Standalone Android App

- Use kickstarter tutorials for foundational knowledge:
  - Setting up development environment, Run a HelloWorld app in simulator, etc
- Use sample applications as a starting point for what you're building:
  - for architecture, infrastructure, app flow, etc.
- Focus on learning the why: architecture, theory, process for mobile development

linktr.ee/sunfishgurl

Putting it All Together

Tips for Building a Standalone Android App

- For syntax help and deeper dives in certain areas, use the following:
  - APIs: Android Studio, Google documentation
  - Generative AI tools : Gemini AI in Android Studio, Github copilot
- Bonus, Optional:
  - To practice in real life, contribute to open source and start with beginner issues

linktr.ee/sunfishgurl

Resources

Open Source Projects - Task Tracker links

- WWCodeMobile Github repo has been taken down 😭
- Fork: https://github.com/vuinguyen/WWCodeMobile
  - coding-projects/android, branch: android-mvp-app
  - coding-projects/ios, branch: ios-mvp-app
- Projects have ended, no longer taking contributions (but can use as a reference)

Resources

Android

- Kotlin: https://developer.android.com/kotlin/samples
- Compose: https://developer.android.com/courses/android-basics-compose/course
- Architecture: https://github.com/android/architecture-samples
- Architecture and Compose: https://github.com/skydoves/DisneyCompose/

linktr.ee/sunfishgurl

Resources

Android

- Navigation: https://developer.android.com/develop/ui/compose/navigation
- StateFlow: https://developer.android.com/kotlin/flow/stateflow-and-sharedflow
- Coroutines: https://developer.android.com/kotlin/coroutines
- Room database: https://developer.android.com/courses/pathways/android-basics-compose-unit-6-pathway-2
- Room database (with coroutines & concurrency) code lab:
  - https://developer.android.com/codelabs/basic-android-kotlin-compose-persisting-data-room

linktr.ee/sunfishgurl

Shoutouts 😍

linktr.ee/sunfishgurl

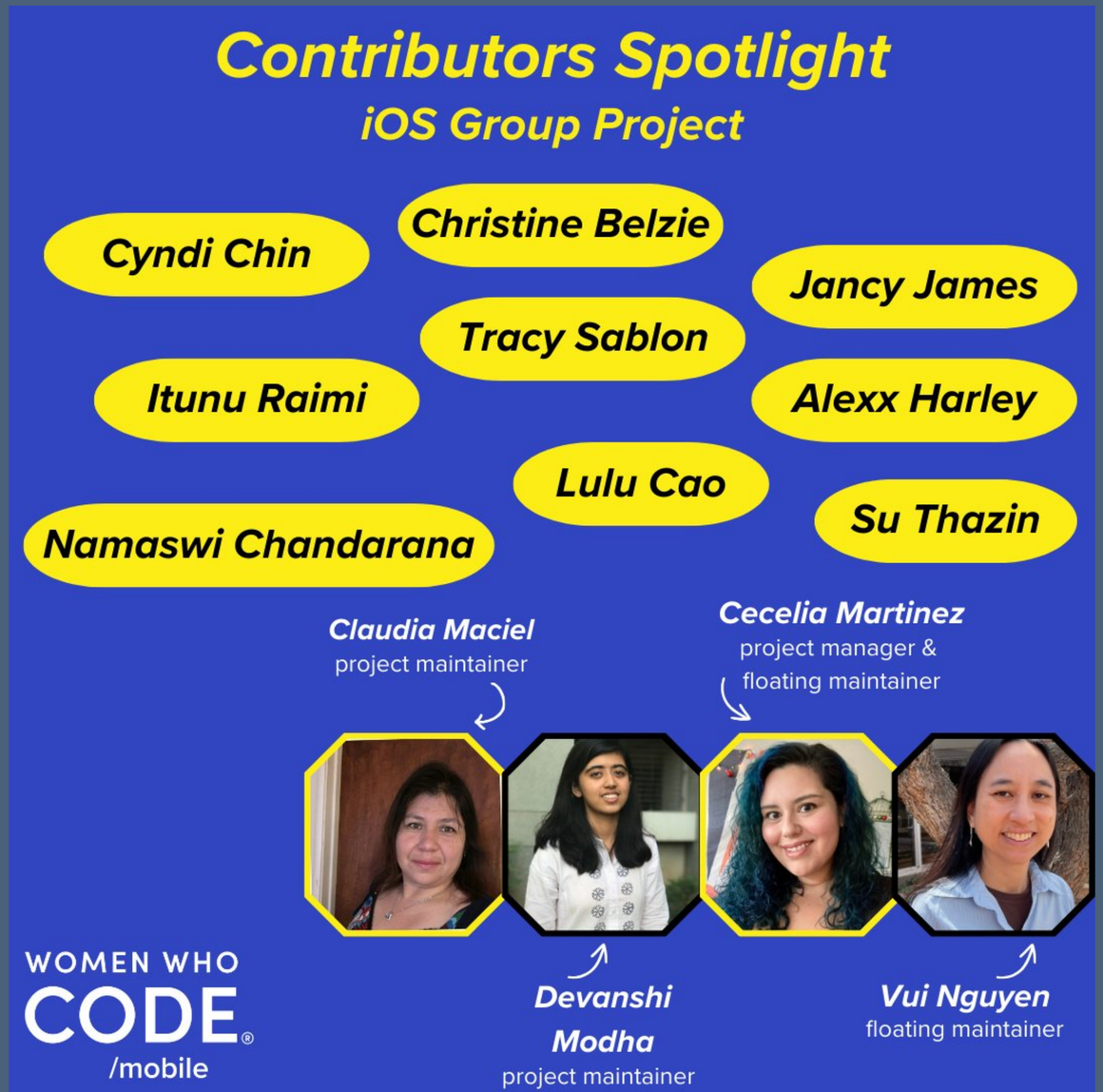
Android Group Project

Maintainers and Contributors

linktr.ee/sunfishgurl

iOS Group Project

Maintainers and Contributors



linktr.ee/sunfishgurl

My Talk Reviewers

linktr.ee/sunfishgurl

Promotion Award
Android Group Project

Sepideh Miller

Contributor to Maintainer

WOMEN WHO CODE /mobile

Spotlight Award
iOS Group Project

Itunu Raimi

Top Contributor

WOMEN WHO CODE /mobile

What We Covered Today

- Introduce problem
- Introduce myself
- Goals for this talk
- Introducing Task Tracker App: iOS
- How We Built Android and iOS Apps in Parallel

linktr.ee/sunfishgurl

What We Covered Today

- Building a CRUD Mobile App on Any Platform
  - Components of Every CRUD Mobile App
  - App Flows:
    - Create - Read: Add Flow
    - Update: Edit Flow
    - Delete: Delete Flow

linktr.ee/sunfishgurl

What We Covered Today

- Demo: Task Tracker App for Android
- Putting it All Together: Tips for Building a Standalone Android App
- Resources
- Shoutouts

linktr.ee/sunfishgurl

Thanks! Questions?

- I’m Vui Nguyen
- @sunfishgurl
- linktr.ee/sunfishgurl