TRACE

Outcomes and Lessons Learned

Matt Thomas, Ethan Schnaser, Will Dominski, Tony Schulz,
Nathan Frank, Trevor Hamilton, Stone Yang

Overview of Project Objectives

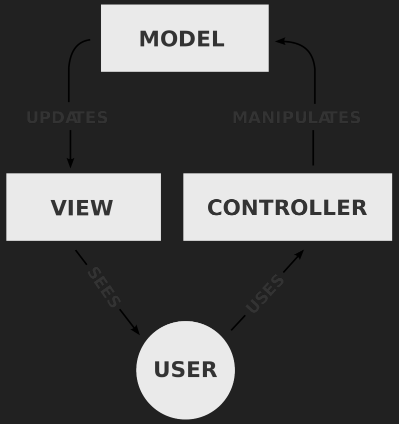
Use Case Aggregation

$$\begin{tikzcd}
\boxed{\text{Use Case Aggregation}} \arrow[r] & \text{Tasks} \\
\arrow[r] & \text{Alerts} \\
\arrow[r] & \text{Cues}
\end{tikzcd}$$

| Minimalism | Time Management |

| Focus | Productivity |

Overview of Architecture & Design



Architectural Framework

→ DB & ContentView as Model

→ EditView as Controller

→ Trace.app as Event source

Tech Stack

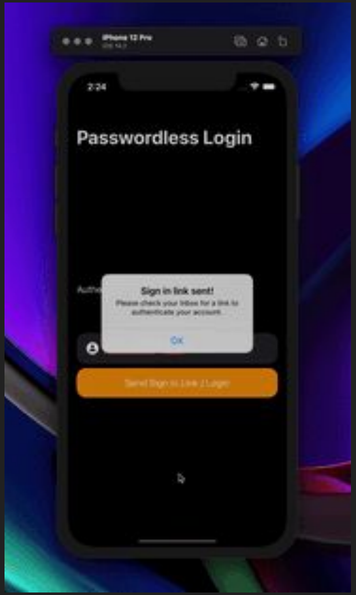
Swift

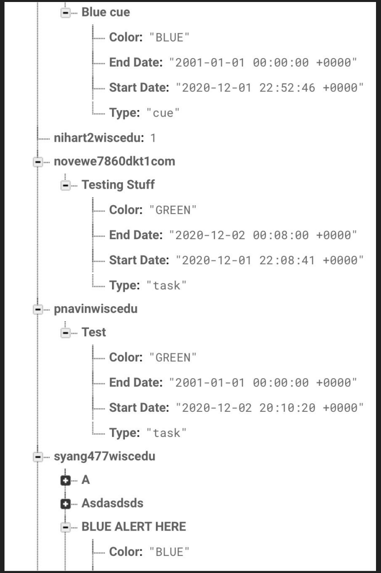
Xcode

iOS

Outcomes: Database

- CocoaPods
  - Modular approach is best
  - Xcode integration
- Firebase
  - Realtime database with Swift/Xcode
  - Setbacks with Swift updates
    - antiquated tutorial guides
- Authentication
  - Linking associated domains/url
  - Firebase ActionCodeSettings
  - Stopping emails & retaining nodes

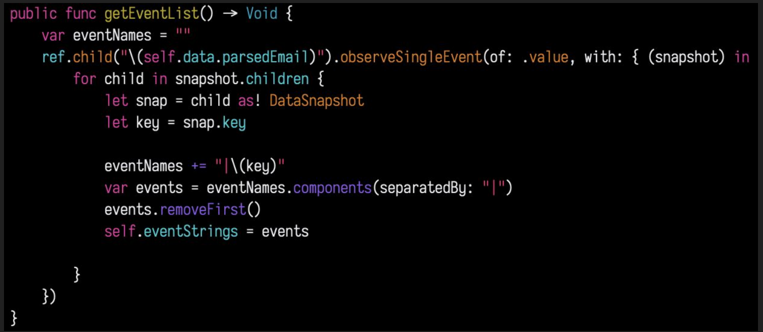




```
Blue cue
    Color: "BLUE"
    End Date: "2001-01-01 00:00:00 +0000"
    Start Date: "2020-12-01 22:52:46 +0000"
    Type: "cue"
nihart2wiscedu: 1
novewe7860dkt1com
  Testing Stuff
    Color: "GREEN"
    End Date: "2020-12-02 00:08:00 +0000"
    Start Date: "2020-12-01 22:08:41 +0000"
    Type: "task"
pnavinwiscedu
  Test
    Color: "GREEN"
    End Date: "2001-01-01 00:00:00 +0000"
    Start Date: "2020-12-02 20:10:20 +0000"
    Type: "task"
syang477wiscedu
  A
  Asdasdsds
  BLUE ALERT HERE
    Color: "BLUE"
```

Outcomes: Backend

- ❖ Linked Local Notifications to TRACE
  - ➢ Reads/writes notifications to/from the database
  - ➢ Tracks dynamic data for scheduling to iOS
- ❖ Offloading Bloated Classes
  - ➢ EditView to shift user control
  - ➢ EventHandler for dynamic UI changes
- ❖ Deployment
  - ➢ Archiving/Versioning
  - ➢ App Store Connect & Test Flight

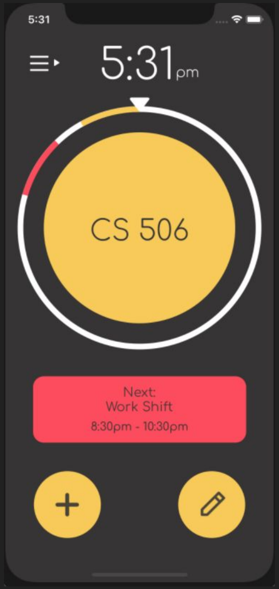


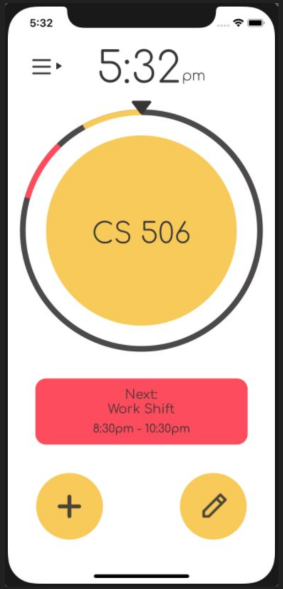
```swift
public func getEventList() → Void {
    var eventNames = ""
    ref.child("\(self.data.parsedEmail)").observeSingleEvent(of: .value, with: { (snapshot) in
        for child in snapshot.children {
            let snap = child as! DataSnapshot
            let key = snap.key

            eventNames += "|\(key)"
            var events = eventNames.components(separatedBy: "|")
            events.removeFirst()
            self.eventStrings = events

        }
    })
}
```

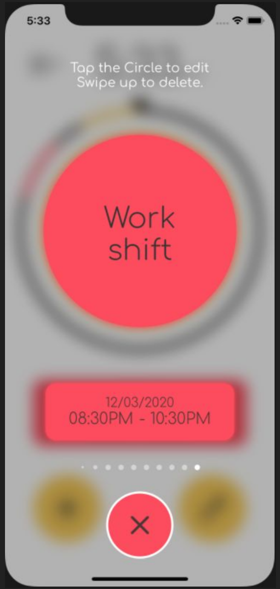
Outcomes: UI

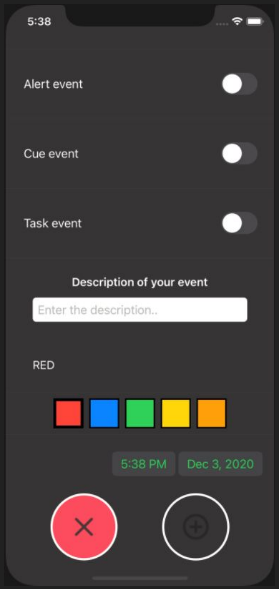
- SwiftUI 2.0
- Implemented the design & functionality

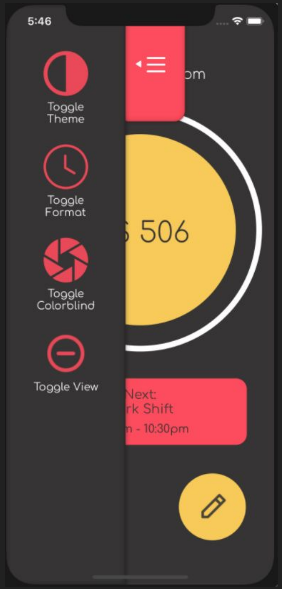
- New update, less documentation
- Big learning curve (View Updating)





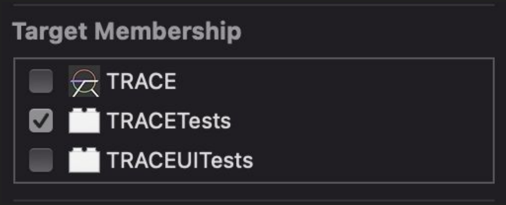




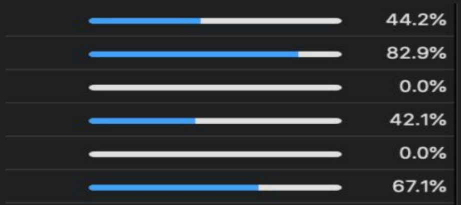


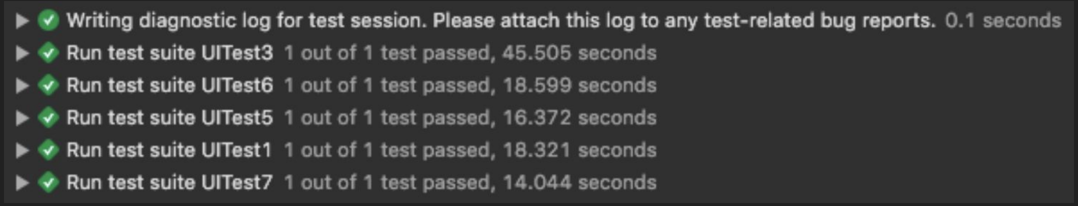
Outcomes: Testing

- Used the XCTest framework
  - Unit Tests
  - UI Tests
- Code Coverage
  - Measuring progress
- Dependency Conflicts
  - Compile targets


Target Membership

☐	TRACE
✓	TRACETests
☐	TRACEUITests


44.2%
82.9%
0.0%
42.1%
0.0%
67.1%


```
▶ ✓ Writing diagnostic log for test session. Please attach this log to any test-related bug reports. 0.1 seconds
▶ ✓ Run test suite UITest3  1 out of 1 test passed, 45.505 seconds
▶ ✓ Run test suite UITest6  1 out of 1 test passed, 18.599 seconds
▶ ✓ Run test suite UITest5  1 out of 1 test passed, 16.372 seconds
▶ ✓ Run test suite UITest1  1 out of 1 test passed, 18.321 seconds
▶ ✓ Run test suite UITest7  1 out of 1 test passed, 14.044 seconds
```

Lessons Learned

- Swift fundamentals, XCode, XCTest knowledge (iOS App Development)
- Integrating a realtime database (Firebase) with an in-dev product
- Difficulties and limitations of developing for iOS exclusively on macOS
- Need for more enforcement of communication (stand-ups)
- Value of separate branches/merging for each user to ensure main branch is stable
- Should have tackled testing first for a better grasp of group member strengths/needs for role rotations
- Value of keeping a tasks/goals queue to ensure there is always a clear view of what can be worked on