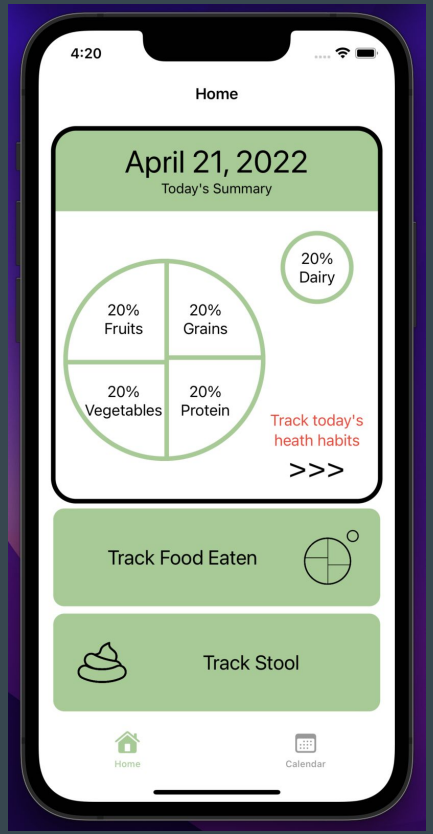


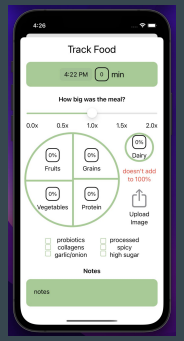
Digestive System App

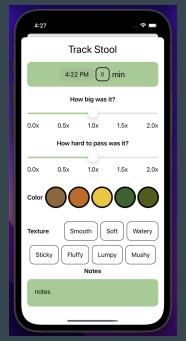
...

Katy Stuparu

Friday, April 22, 2022

SISP Final Presentation






```
ZStack {
    if grains != 0 {
        PieSliceView(pieSliceData: PieSliceData(
            startAngle: Angle(degrees: 0.0),
            endAngle: Angle(degrees: grains / sum * 360.0),
            text1: String(Int(grains)) + "%",
            text2: "Grains"))
    }

    if protein != 0 {
        PieSliceView(pieSliceData: PieSliceData(
            startAngle: Angle(degrees: grains / sum * 360.0),
            endAngle: Angle(degrees: (protein + grains) / sum * 360.0),
            text1: String(Int(protein)) + "%",
            text2: "Protein"))
    }

    if vegetables != 0 {
        PieSliceView(pieSliceData: PieSliceData(
            startAngle: Angle(degrees: (protein + grains) / sum * 360.0),
            endAngle: Angle(degrees: (vegetables + protein + grains) / sum *
                360.0),
            text1: String(Int(vegetables)) + "%",
            text2: "Vegetables"))
    }

    if fruits != 0 {
        PieSliceView(pieSliceData: PieSliceData(
            startAngle: Angle(degrees: (vegetables + protein + grains) / sum *
                360.0),
            endAngle: Angle(degrees: 360.0),
            text1: String(Int(fruits)) + "%",
            text2: "Fruits"))
    }
```

Tracking Digestive System - iOS App

Stage One: Basic Tracking App (SISP)

- Tracks food eaten
- Tracks gastrointestinal symptoms
- Tracks other health habits
- Displays entries

Stage Two: Adding Food/Stool Analysis

- Educates users about digestive system
- Finds patterns in digestion

Stage Three: Optimization

- Code efficiency
- UI design
- Functionality: does app meet goal?

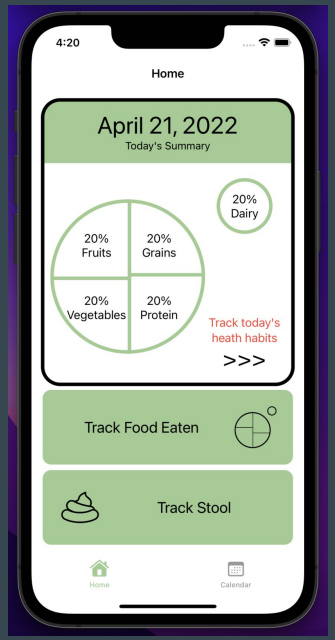
Stage Four: Testing and Publishing App

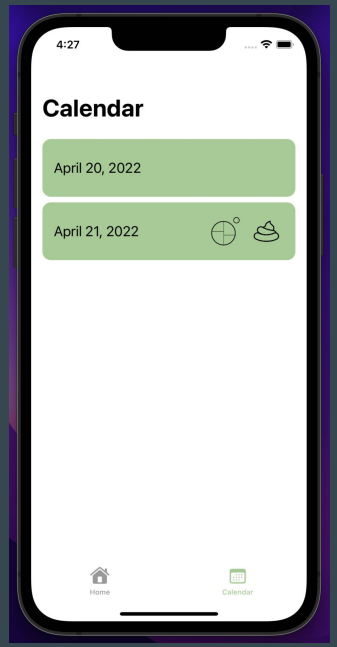
- Testing with simulators and iPhones
- Apple Developer membership
- TestFlight
- Publishing to App Store

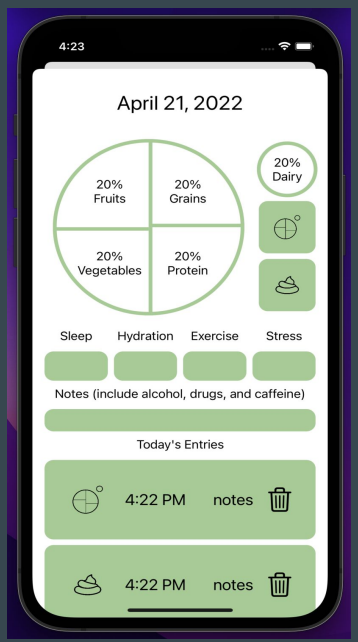
Why I Chose this Project

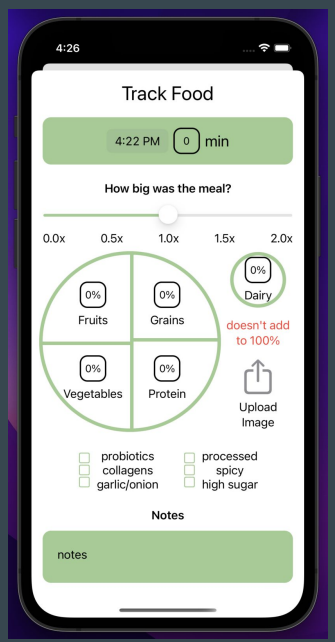
- Personal: trouble explaining my digestive health to my doctor
- Digestive health is a huge factor in mental health and overall ability
- Current apps on the market are not helpful for me
  - Don't track food and stool and other factors (exercise, hydration, sleep, stress)
    - Don't make analysis/connections between food and stool
  - Asking to track too many things (calories, exact amounts of foods eaten, etc.)
  - Are for users with a specific medical condition
- App is targeted towards everyone looking to improve their gut health

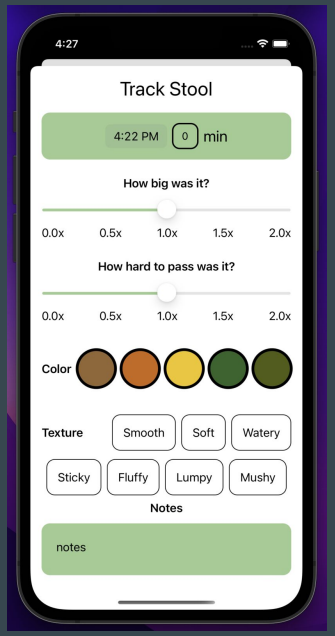
User Interface



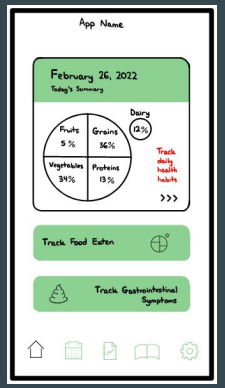


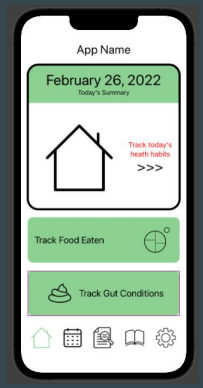


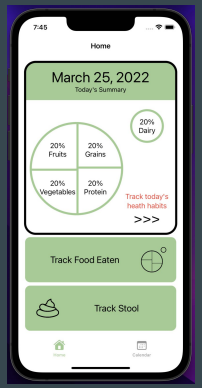


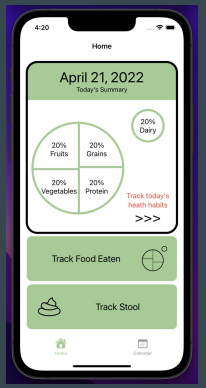


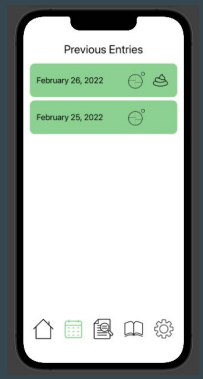
Progression of User Interface

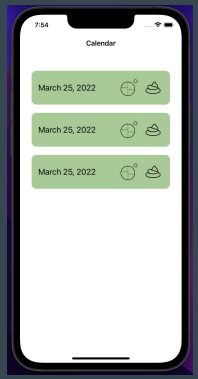


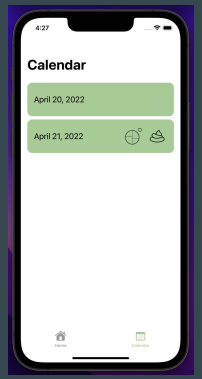


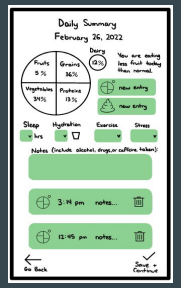


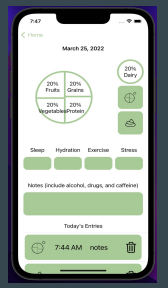


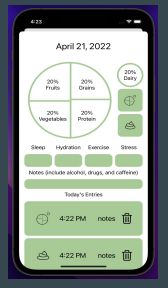


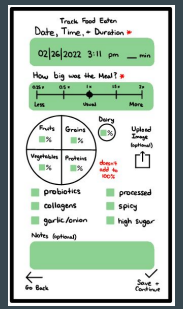


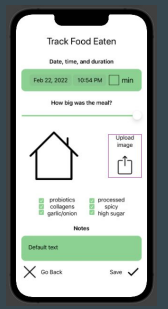


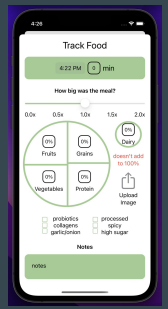


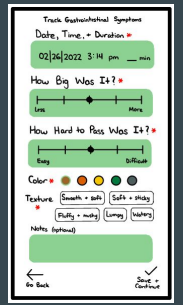


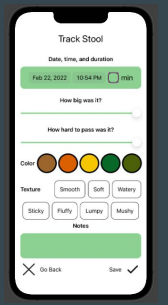


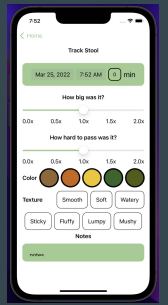


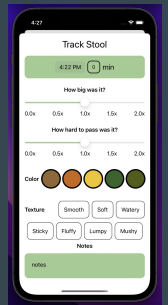






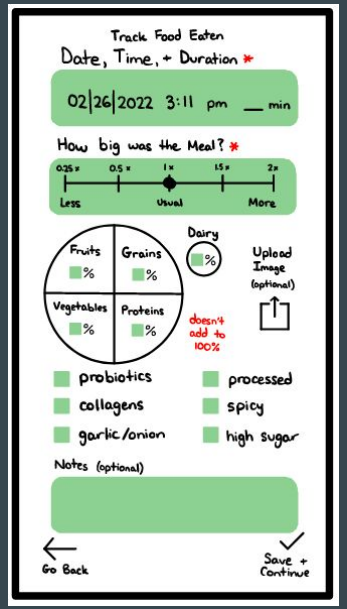


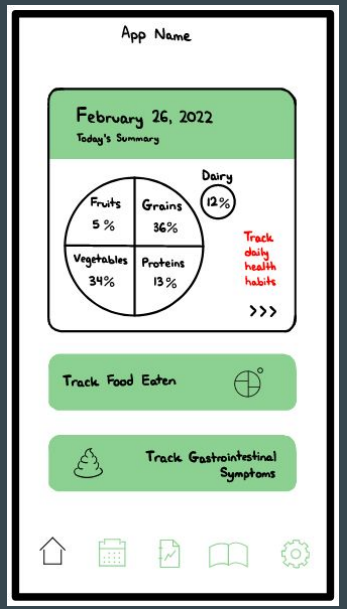




Summary of Progress - 33-Hour Checkpoint

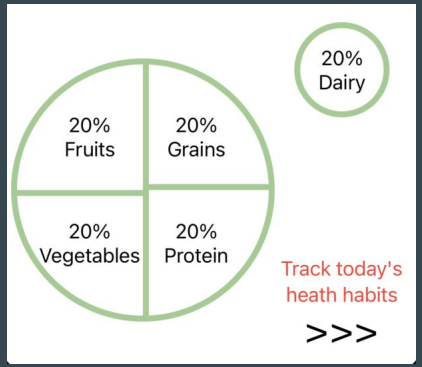
- Watched and worked along with Xcode tutorials
  - Code with Chris: 14-Day Challenge
- Designed first 5 screens
  - Drawn in Notability app on iPad
- Created basic layout for the 5 screens
- Began to add functionality to the screens





Summary of Progress - 66-Hour Checkpoint

- Created data classes
- Created a food plate
  - Dynamic pie chart
  - Static pie chart with pickers
- Created navigation between screens
- Made data persist between screens
- Began creating dynamic lists




```
@Published var grains: Double
@Published var protein: Double
@Published var vegetables: Double
@Published var fruits: Double
@Published var dairy: Double

@Published var sleep: String
@Published var hydration: String
@Published var exercise: String
@Published var stress: String

@Published var notes: String

var entries: [Entry] {

    let unsortedEntries = foodEntries + stoolEntries

    return unsortedEntries.sorted(by: {
        $0.date.compare($1.date) == .orderedAscending
    })
}

func getIndex(uuid: UUID, typeFood: Bool) -> Int {

    if typeFood {
        for i in 0...foodEntries.count - 1 {
            if uuid == self.foodEntries[i].uuid {
                return i
            }
        }
    }
    else {
        for i in 0...stoolEntries.count - 1 {
            if uuid == self.stoolEntries[i].uuid {
                return i
            }
        }
    }
    return -1
}
```

```
struct ContentView: View {

    @StateObject var dayList = DayList()

    var body: some View {
```

Summary of Progress - 100-Hour Checkpoint

- Added modal navigation
- Added clicking and delete function to lists
- Began implementing local storage
    - Researched SQL and SQLite implementation for Swift
    - Went through SQLite Swift tutorials
    - Started using a wrapper SQLite class for Swift in my project

```
func insertDay(day: Day) throws {

    let insertSql = "INSERT INTO Day (Id, Date, Grains, Protein, Vegetables,
        Fruits, Dairy, Sleep, Hydration, Exercise, Stress, Notes) VALUES (?, ?, ?,
        ?, ?, ?, ?, ?, ?, ?, ?, ?, ?);"

    let insertStatement = try prepareStatement(sql: insertSql)

    defer {
        sqlite3_finalize(insertStatement)
    }

    guard   sqlite3_bind_text(insertStatement, 1,
        nil) == SQLITE_OK &&
            sqlite3_bind_text(insertStatement, 2,
                SQLITE_OK &&
            sqlite3_bind_double(insertStatement, 3
            sqlite3_bind_double(insertStatement, 4
            sqlite3_bind_double(insertStatement, 5
            sqlite3_bind_double(insertStatement, 6
            sqlite3_bind_double(insertStatement, 7
            sqlite3_bind_text(insertStatement, 8,
                &&
            sqlite3_bind_text(insertStatement, 9,
                SQLITE_OK &&
            sqlite3_bind_text(insertStatement, 10,
                SQLITE_OK &&
            sqlite3_bind_text(insertStatement, 11,
                SQLITE_OK &&
            sqlite3_bind_text(insertStatement, 12,
    else {
        throw SQLiteError.Bind(message: errorMessa
    }

    guard sqlite3_step(insertStatement) == SQLITE_
    else {
        throw SQLiteError.Step(message: errorMessa
    }

    print("Successfully inserted row for day: " +
}
```

```
func openTable() {
    try! db.run(days.create(ifNotExists: true) { t in
        t.column(id, primaryKey: true)
        t.column(date)
        t.column(grains)
        t.column(protein)
        t.column(vegetables)
        t.column(fruits)
        t.column(dairy)
        t.column(sleep)
        t.column(hydration)
        t.column(exercise)
        t.column(stress)
        t.column(notes)
    })
}

func addDay(day: Day) {

    try! db.run(days.insert(id <- Int64(day.id), date <- day.dateString, grains <-
        day.grains, protein <- day.protein, vegetables <- day.vegetables, fruits <-
        day.fruits, dairy <- day.dairy, sleep <- day.sleep, hydration <-
        day.hydration, exercise <- day.exercise, stress <- day.stress, notes <-
        day.notes))
}

func fillDayList() {

    for day in try! db.prepare(days) {
        list.append(Day(id: Int(day[id]), date: stringToDate(stringDate:
            day[date]), grains: day[grains], protein: day[protein], vegetables:
            day[vegetables], fruits: day[fruits], dairy: day[dairy], sleep:
            day[sleep], hydration: day[hydration], exercise: day[exercise], stress:
            day[stress], notes: day[notes]))
    }
}
```

What I Learned and Challenges

- Swift
  - Already knew Java, Python, and some C++/C#
  - Swift has a lot of unique syntax that I wasn't familiar with
- Swift UIKit
  - I already had the knowledge for Android app development
  - UIKit doesn't have separate controller and view classes (instead: one class for all)
  - UIKit-specific developer tools and framework
    - Data persistence is unique
- SQL and SQLite
  - Fundamentals of SQL
  - Wrapper class for Swift

```
@Published var date = Date()

var timeStamp: String {
    let dateFormatter = DateFormatter()
    dateFormatter.dateFormat = "h:mm a"
    return dateFormatter.string(from : date)
}
```

```
@EnvironmentObject private var dayList: DayList
```

```
struct CheckBoxView: View {
    @Binding var checked: Bool

    var body: some View {

        Image(systemName: checked ? "checkmark.square.fill" : "square")
            .foregroundColor(Color("AccentColor"))
            .onTapGesture {
                self.checked.toggle()
            }
    }
}
```

What I Learned and Challenges: Time Management

- Started working on the project very late (early February)
- Task-oriented vs. hours-oriented
- Helped to plan out each day in advance
  - Planning out hours and tasks
  - Underestimation of how much rest and sleep I need
- Working under a lot of stress
  - I felt like I didn't have the time I needed
  - Difficulty focusing
  - To-do lists help

What's Next

Stage One: Basic Tracking App (SISP)

- Tracks food eaten
- Tracks gastrointestinal symptoms
- Tracks other health habits
- Displays entries

Stage Two: Adding Food/Stool Analysis

- Educates users about digestive system
- Finds patterns in digestion

Stage Three: Optimization

- Code efficiency
- UI design
- Functionality: does app meet goal?

Stage Four: Testing and Publishing App

- Testing with simulators and iPhones
- Apple Developer membership
- TestFlight
- Publishing to App Store

Reflection and Summary

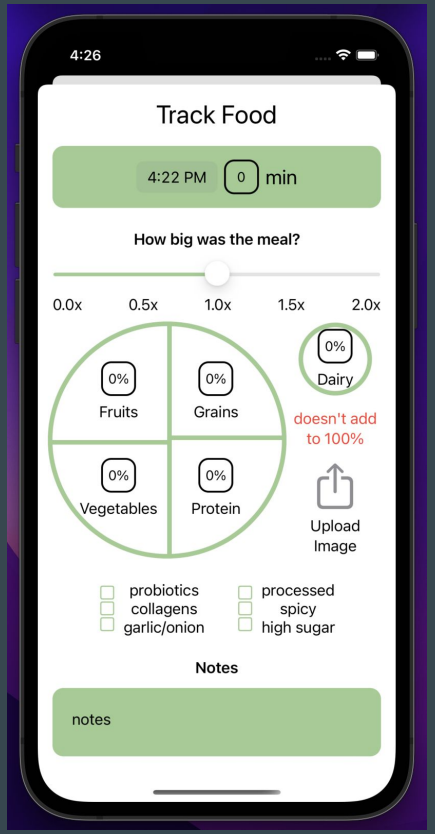
- What I learned:
  - Technical skills
    - Swift and UIKit
    - App development
    - SQL
    - Project and time management
  - Problem-solving skills for CS
  - A lot about myself and my working habits
    - Planning is helpful
    - How to handle hours-oriented tasks
- Recommendations and advice for SISP

```swift
class FoodEntry: Entry {

    @Published var duration = 0
    @Published var size = 1.0

    @Published var probiotics = false
    @Published var collagens = false
    @Published var garliconion = false
    @Published var processed = false
    @Published var spicy = false
    @Published var highsugar = false

    @Published var grains = "0%"
    @Published var protein = "0%"
    @Published var vegetables = "0%"
    @Published var fruits = "0%"
    @Published var dairy = "0%"

    func foodToInt(food: String) -> Int {
        return Int(food.dropLast()) ?? -1
    }

    func foodSum() -> Int {
        return foodToInt(food: grains) +
        foodToInt(food: protein) +
        foodToInt(food: vegetables) +
        foodToInt(food: fruits) +
        foodToInt(food: dairy)
    }

}
```

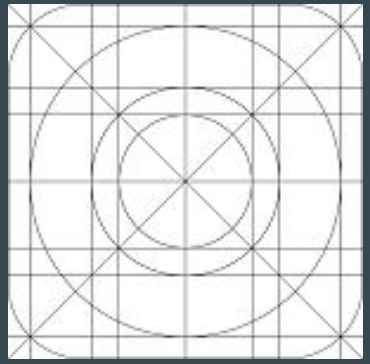
Thank You

...

Questions? Suggestions?



Installing App on iPhone



Instructions:

1. Plug in your iPhone to my laptop
2. Click “Trust” and enter your passcode
3. After I initiate the installation from my laptop, you can unplug your iPhone
4. In Settings, navigate to General -> VPN & Device Management
5. Click on “Apple Development: katystuparu@icloud.com” and then click “Trust”
6. Open the new app on your Home Screen titled “digestivesystem”
    a. Feel free to let me know of any bugs you find or any suggestions you have!
    b. SQLite local storage is not working yet, so data you enter will not be saved when you close the app
    c. I have not done much testing yet, so the app won’t look as intended in Dark Mode and on some iPhones