# Project Proposal: Staggered Break Schedule App

## Team Name:

I need a break

## Team Member's Name:

Emma Capirchio

## Business Opportunity / Problem

In high-traffic retail and warehouse environments, managing employee break schedules manually is inefficient, error-prone, and leads to overlapping breaks that can disrupt operations and violate labor regulations. Most existing scheduling systems are too rigid, lack real-time updates, or don't support complex team structures.

This app addresses the need for a dynamic, role-based break scheduling system that allows for staggered, conflict-free breaks across departments while ensuring compliance with company policies.

## Users

- Store Directors: Oversee the entire scheduling process and have the highest authority.
- Executive Team Leads (ETLs): Manage department-specific teams and adjust break times as needed.
- Team Leads (TLs): View and monitor schedules for their respective teams.
- Team Members (TMs): View their own break schedule.

## High-Level User Requirements

1. Authentication:
    - Sign in with Apple and username/password support
    - Role-based access control (Director > ETL > TL > TM)
2. Break Scheduling:
    - Create, view, and adjust staggered break schedules
    - Prevent overlapping breaks within departments
3. Employee Management:
    - Add, edit, or remove employees
    - Assign roles and departments
4. Department-Specific Goals/Notes:
    - Include department notes and goals in schedules
5. Cloud Sync and Offline Support:
    - Sync with CloudKit for real-time data access
    - Allow viewing schedules offline
6. In-App Messaging:
    - Send and receive messages within the app
    - View message history
    - Filter by sender or department
    - Optional push notifications for new messages

## Technology That Will Be Used

- Frontend: SwiftUI (Xcode)

- Backend/Cloud: CloudKit (Apple's iCloud database)
- Authentication: Sign in with Apple and custom username/password
- Storage: iCloud containers with private and shared records
- Other Features:
    - PDF generation for print-ready schedules
    - Dark mode and accessibility support

### High-Level Database Architecture

| Record Type | Fields |
|---|---|
| User | username, passwordHash, role, department, appleIdentifier |
| Employee | name, role, department, availability |
| Schedule | employeeReference, shiftStart, shiftEnd, breaks[], goals, notes |
| Break | startTime, endTime, employeeReference |
| Message | senderReference, recipientReference, timestamp, content, isRead |

## Proposed Solution

### Technical Approach:

- Role-based app architecture to display features based on user role
- Modular ViewModels and Managers (AuthManager, CloudKitManager) for clean separation of concerns
- Real-time syncing with CloudKit for seamless multi-device support
- UserDefaults for local caching and offline access
- Error handling and data validation across the app for reliability
- Integrate a modular MessageManager using CloudKit to handle role-based, real-time in-app messaging between users.

### Non-Technical Approach:

- User training sessions for managers and team leads to adopt the system
- Internal documentation and guides to help with onboarding
- Feedback loop via surveys to improve usability after rollout
- Regular check-ins with teams to refine scheduling policies and app usage
- Encourage teams to use in-app messaging for all schedule-related communication and provide onboarding guidance during rollout.