Pitch Snitch Future Development

SDMAY25-11

Quick Start

Video QuickStart Instructions:
https://sdmay25-11.sd.ece.iastate.edu/QuickstartFlutterApplication.mp4

Android Development

1. Clone the GitLab repository
    a. `git clone https://git.ece.iastate.edu/sd/sdmay25-11.git`
    b. `cd softball-tracker`

2. Replace the link with your actual repository URL.Download Flutter from the official site: Flutter SDK
3. Extract the Flutter SDK to a preferred location (e.g., `C:\flutter` on Windows)
4. Add Flutter to the system path:
    a. Add `C:\flutter\bin` to `Environment Variables`
4. Restart your terminal
5. Run the `flutter doctor` command in your terminal to ensure the installation is complete.
6. Download Android Studio from the official site: Android Studio Download
7. Install the Android Studio Dart and Flutter plugins from Settings --> Plugins
8. For Emulating:
    a. Set up a device in the Device Manager tab
    b. Edit the device and set `Graphics = Software`
    c. Set up the camera environment by going into `Advanced Settings` and set `Front Camera = Emulated` and `Back Camera = Emulated`
9. For Physical Device:
    a. Enable **Developer Mode** on your Android phone:
    b. Go to **Settings > About phone**
    c. Tap **Build number 7 times** to enable Developer Mode
    d. In **Developer Options**, enable **USB Debugging**
    e. Connect your Android device via USB
    f. Run: `flutter devices`. Your device should appear in the list.

Common Windows Issues:

- If your device becomes unresponsive with errors about `*.lock files` delete all files ending in `.lock` in your Android Studio installation location within the `avd` directory
- When running the app, if the installation fails due to storage limitations, select to edit your emulated device and select the `Wipe Data` option to clear the device's app storage.

- If Android Studio is unable to run Dart, end all tasks in Task Manager using `dart`. Some programs can be left dangling.

iOS Development

1. Install **Xcode** from the App Store
2. Open **Xcode > Preferences > Accounts**
3. Sign in with your Apple ID (free account works, but requires manual app signing)
4. Connect your iPhone via USB
5. Trust the device when prompted
6. Install **Flutter** using Homebrew:
   brew install flutter
7. Ensure **Xcode** is installed via the App Store
8. Install CocoaPods for iOS dependencies: `sudo gem install cocoapods`
9. Run: `flutter devices`. Your iPhone should appear in the list.
10. Open **ios/Runner.xcworkspace** in Xcode
11. Select your **development team** under **Signing & Capabilities**
12. Run the app in Xcode or via: `flutter run`

Common iOS Issues

- **iOS Signing Issues?**
  - Ensure Xcode is signed with a valid development team in **Signing & Capabilities**.

Running the Application

- First, verify your flutter installation with `flutter doctor`. If there are any missing dependencies, follow the instructions provided by `flutter doctor`.
- Next, clean the flutter cache and get the dependencies from the flutter project.
  ```
  flutter clean
  flutter pub get
  ```
- If `flutter pub get` fails, run it again:
- Run the Flutter app with `flutter run`

- If running on iOS and encountering permission errors, try:
  ```
  cd ios
  pod install
  cd ..
  flutter run
  ```

Common Errors

App not detecting device?

- Run `flutter devices` to confirm the device is recognized.
- Ensure USB debugging (Android) or proper provisioning (iOS) is set up.

Errors during `flutter pub get`?

- Run `flutter clean` and try `flutter pub get` twice.

Dependency Errors

- Run `flutter clean` and try `flutter pub get` twice.

Quick Start Video

Git Repository Overview

Repo Link: https://git.ece.iastate.edu/sd/sdmay25-11

README

The **README.md** file features the poster outlining the project's main ideas as well as the Quickstart Instructions. This file, viewable on the main repository page, will be the main navigational details of the project, showing an overview of the project's design and mission as well as developmental guides for running the project.

Design Document

The document **sdmay25-11_Design_Document.pdf** is an extensive documentation document of our application containing the following topics:

- Project Introduction
- Requirements, Constraints, and Standards
- Project Plan
- Design
- Testing
- Implementation
- Professional Responsibility
- Closing Material
- Team Information
- Appendices
    - Instruction Manual
    - Design Considerations
    - Alternate Designs
    - Code Analysis

Final Presentation

The **492_Final_Presentation.pdf** presentation summarizes this project's application, motives, and development practices. The presentation document is a reduced version of the Design Document. The contents of the presentation are listed below.

- Project Overview
    - What is Slow-Pitch softball?
    - Why is our application needed?

- User Needs
    - Who are our users?
    - What do they expect and need from our application?
- Project Requirements
    - What are the fundamental standards our project must meet?
- Ethical Requirements
    - What does our team believe our ethical standards must be during development?
- Project Planning
    - What management style does our team use?
    - Task Decomposition graph to allocate tasks to complete our main objective.
    - Gantt Chart showing our timeline of goals throughout the project.
- Design
    - Technical Needs
        - Flutter Development Framework
        - Ultralytics YOLO
    - Our Yolo Model
        - The collection, annotation and augmentation of our data used in the model.
        - The training specifications used for our YOLO model.
        - The integration of our YOLO model into our application.
    - Height Determination Design
        - How do we calculate the height of a ball using identified points?
    - Armed System Design
        - How can we identify and track **only** pitched softballs?
    - Module Design
- Testing
    - Model Testing
        - How do we verify that our model can accurately and efficiently detect a ball throughout a pitch?
    - User Testing
        - User feedback and testing of the tracking, UI, and setup process.
- Future Development Recommendations

Demo Video

Demo_Video.mp4 shows a quick 3-minute demo of our application's functionality, going through each screen. The application goes through the instructions, initialization of game parameters, calibration of the field, and tracking of the ball.

Flutter Application

The Flutter application can be viewed in the **softball_tracker** directory. The Flutter application directory can be split up into several directories:

- **`android/`**: Contains Android-specific files, including Gradle build scripts and native code.
- **`ios/`**: Contains iOS-specific files, including Xcode project files and native code.
- **`lib/`**: The main directory for Flutter/Dart code.
    - **`views/`**: Contains UI screens such as `yolo_screen.dart` and `setup_camera_capture_screen.dart`.
    - **`utils/`**: Utility functions and constants used across the app.
    - **`widgets/`**: Reusable Flutter widgets.
- **`assets/`**: Contains images, videos, and other static resources.
- **`ultralytics_yolo/`**: A custom Flutter plugin for YOLO-based object detection.
    - **`ultralytics_yolo/lib/`**: Contains the Dart interface for the plugin.
        - **`ultralytics_yolo_platform_interface.dart`**: Defines the platform interface for the plugin.
        - **`ultralytics_yolo_platform_channel.dart`**: Implements the platform interface using method channels.
        - **`camera_preview/`**: Contains classes for managing the camera preview and bounding box overlays.
    - **`ultralytics_yolo/android/`**: Contains Android-specific implementation.
        - **`MethodCallHandler.java`**: Handles method calls from Dart and communicates with the Android CameraX API and TensorFlow Lite.
- **`ultralytics_yolo/ios/`**: Contains iOS-specific implementation.
    - **`SwiftUltralyticsYoloPlugin.swift`**: Handles method calls from Dart and communicates with the iOS AVFoundation and CoreML frameworks.
- **`test/`**: Directory to contain unit and widget tests for the app, run by `flutter test`.

Future Development Recommendations

Training a New YOLO Model

The current model implemented within the application is a YOLOv11 model trained on a singular field, in broad daylight lighting, and from one sideline angle. Collecting data from different fields and lighting with a noisy background can allow for a more versatile model, increasing the reliability of our system. A new YOLO model will help our application function at night games, on multiple fields, with different camera angles, and not be susceptible to background noise.

Height Testing

During our prototype and application testing, we were able to validate the height being calculated near the ground where the ball can be held and directly measured (e.g., 1 ft - 8 ft). More testing of the height calculation's consistency can help validate our system's accuracy.

Stress Testing

Just like any application preparing for deployment, extensive stress testing must be completed. Stress testing the UI, such as screen orientation, input validation, and navigating through all state paths, can reveal hidden bugs that users may experience. Additionally, stress testing the model in all environments can help pinpoint faults where the model may crash. Reducing the crashing of the model can improve the user experience and reliability of the application.

Review Tool

Our development team believes that the added feature of a review tool can improve the overall gameplay experience. This tool should be able to save the data of a pitched softball, recording the YOLO coordinates and associated heights. Using these data points, a visual can be made for each pitch, indicating if the pitch was detected as illegal. Maximum and minimum heights can be displayed on the visual, and the detected maximum height should be clearly labeled, visually showing the legality of the pitch.