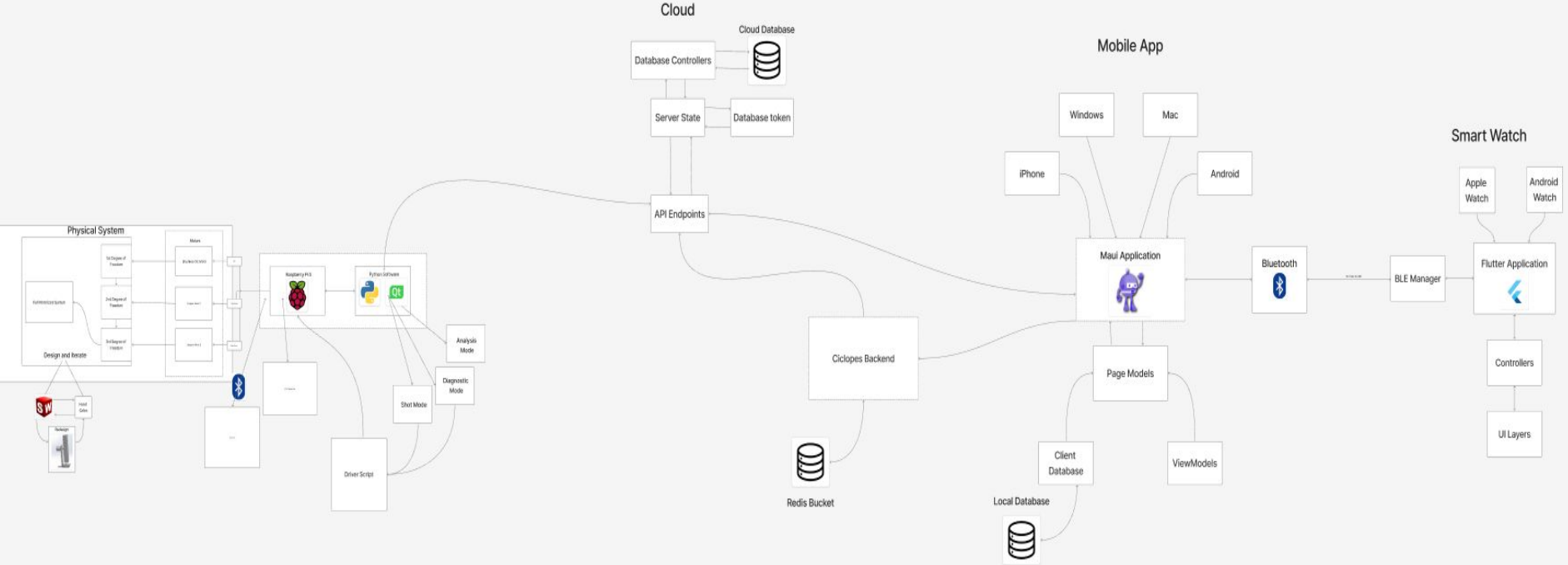
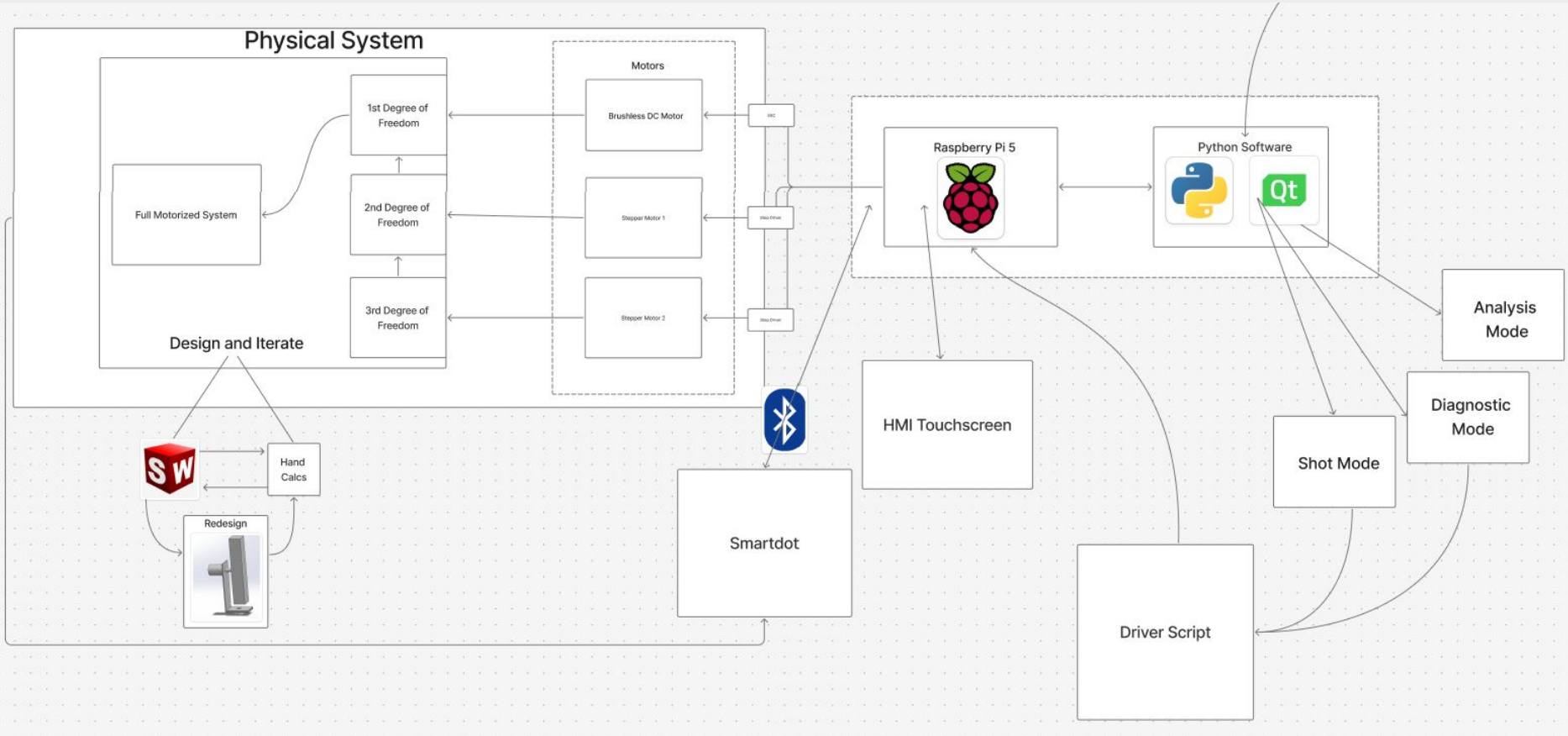
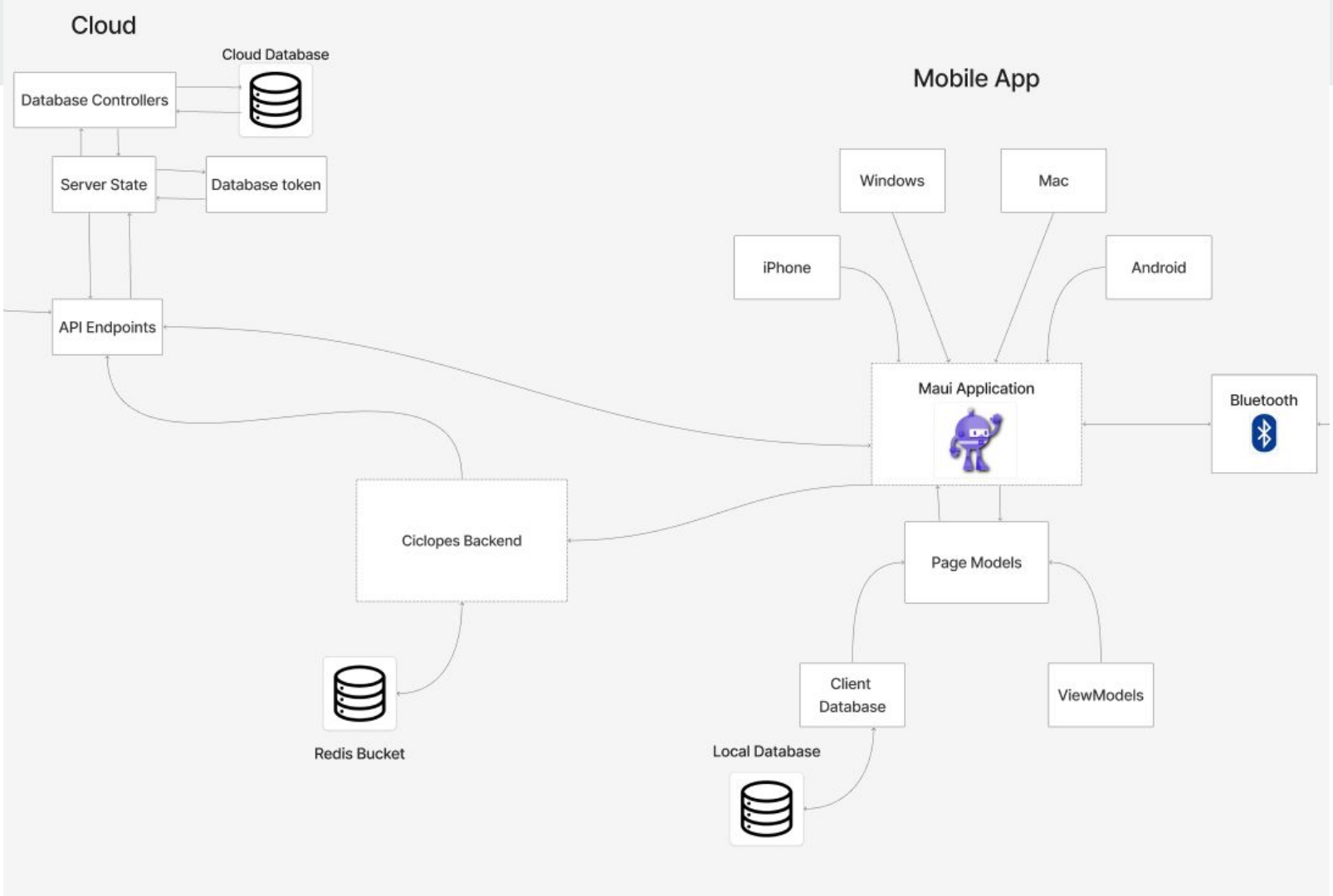
Milestone 1

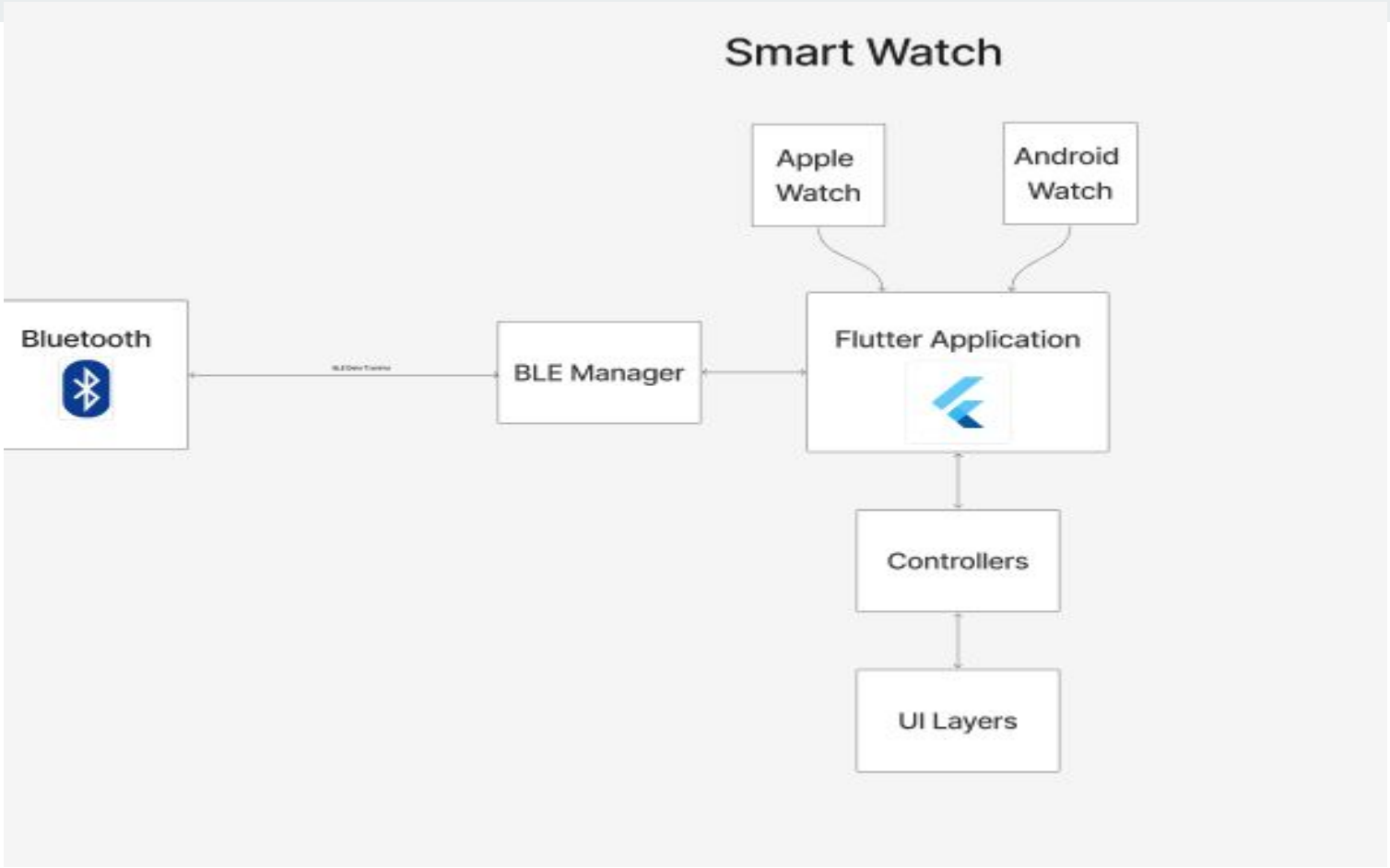
Matt Brown, Josh Byers, Charles Carroll, Zach Cox, Joseph Downey, Gabe Manero, Jakeb Nielsen, Andrew Olvera, Gavin Wentz, Hunter Wolfe

HIGH LEVEL OVERVIEW

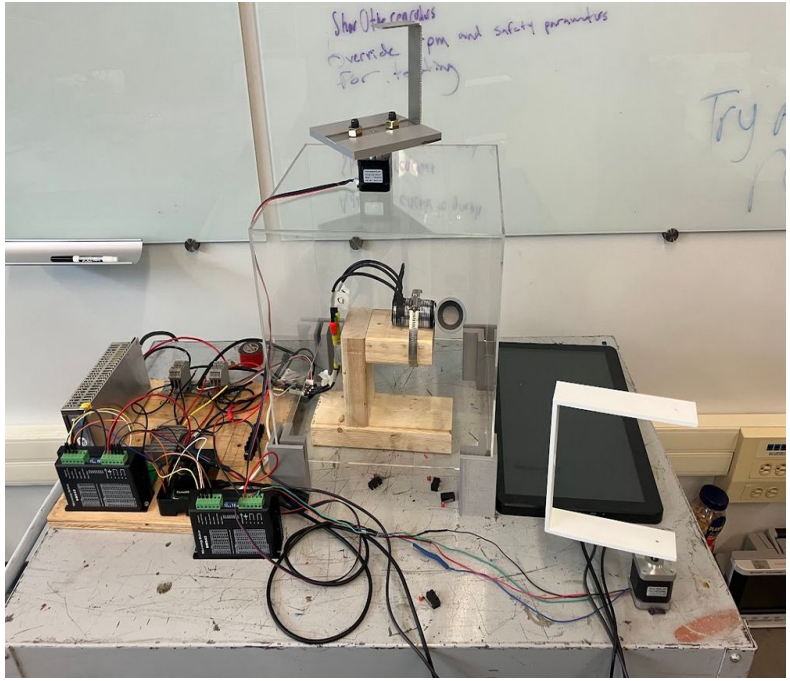



Physical System
Motors
1st Degree of Freedom
2nd Degree of Freedom
3rd Degree of Freedom
Full Motorized System
Design and Iterate
Brushless DC Motor
Stepper Motor 1
Stepper Motor 2
Raspberry Pi 5
Python Software
Qt
Analysis Mode
Diagnostic Mode
Shot Mode
HMI Touchscreen
Smartdot
Driver Script
Hand Calcs
Redesign

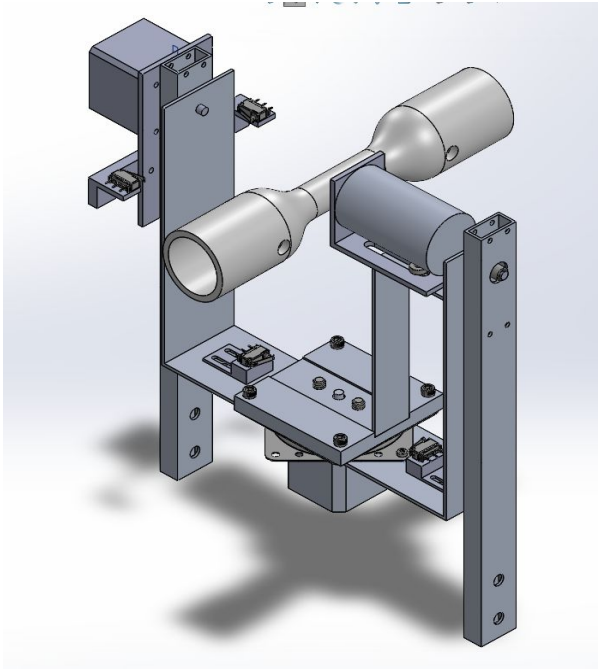

Cloud
Cloud Database
Database Controllers
Server State
Database token
API Endpoints
Mobile App
Windows
Mac
iPhone
Android
Maui Application
Bluetooth
Ciclopes Backend
Page Models
Redis Bucket
Client Database
ViewModels
Local Database


Smart Watch
Apple Watch
Android Watch
Bluetooth
BLE Manager
Flutter Application
Controllers
UI Layers

Physical Design Overview



Prototyped Degrees of Freedom at the End of Fall Milestone 3.



Final Design for Ball Spinner.

Physical Design Team (Goals and Achievements for MS1)

Goals

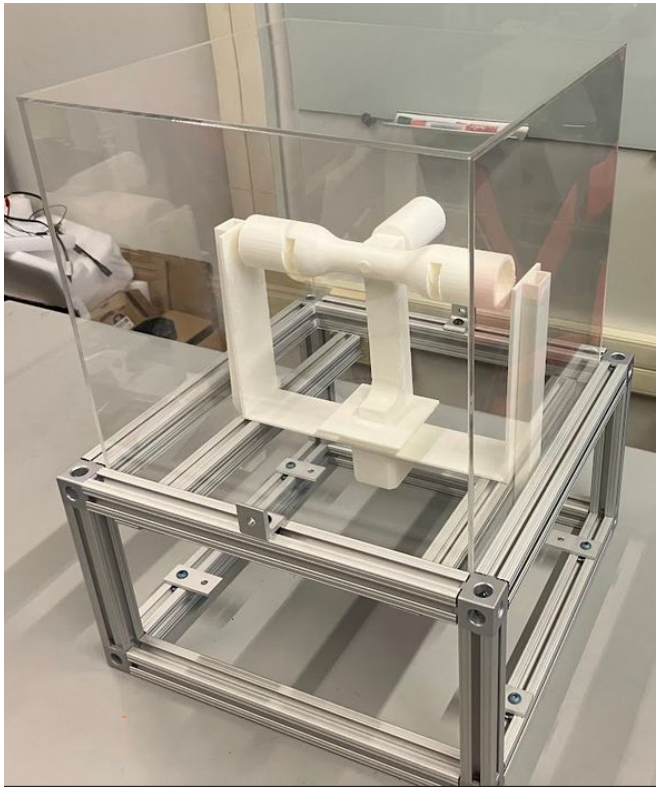
- Fully designed 3D model of ball spinner and electronics enclosure ready for fabrication

Achievements

- Ball Spinner Design Completed
- Enclosure Design Completed and Structural Frame Built
- All material ordered for fabrication
- Plan for fabrication determined

Key Technologies

- SOLIDWORKS - 3D model ball spinner designs as well as enclosure layout
- 3D Print Lab - create rapid prototypes and fabricate non-load bearing components
- Shop - fabricate and adjust metal parts



3D Modeled and Printed Ball Spinner System inside an in progress Electronics Enclosure

Physical Design Team (Future: MS2)

Goals

- Continue work on enclosure (Bottom panel and acrylic box mounting)
- Completed 1st and 2nd degree assembly
- Ensure assembly is ready for motor mounting

Team Pi (Goals for MS1)

Hardware

- Demonstrate temporary BLDC motor operations using software PWM control
- Define safe power supply configuration
- Evaluate motor and esc upgrades

Team Pi (MS1 Achievements)

Hardware

- Successfully integrated and operated BLDC motor
  - Used resourced motor and ESC with the Raspberry Pi via software PWM
- Established safe power delivery
- Compared motor and ESC options for upgrades
  - Evaluated higher torque, sensors
  - Identified trade-offs between torque, weight, cost and feedback capabilities
  - Reviewed programmable ESCs offering braking and finer controls

Team Pi (Future: MS2)

Hardware

- Implement safety
- Develop “shot script” for the primary BLCD through a bowling shot sequence
- Design and evaluate power distribution and source options (battery vs plug in power configs)
- Test motor operations under various conditions
- Integrate sensors and upgrades
- Begin Stepper Motor connections with appropriate drive module

Team Pi (Goals for MS1)

Software

- Placeholder shot/diagnostic/analysis page
- Basic cloud communication
- SmartDot port and 32 bit destruction
- Placeholder SmartDot graphing
- Testing
- Logging

Team Pi (MS1 Achievements)

Software

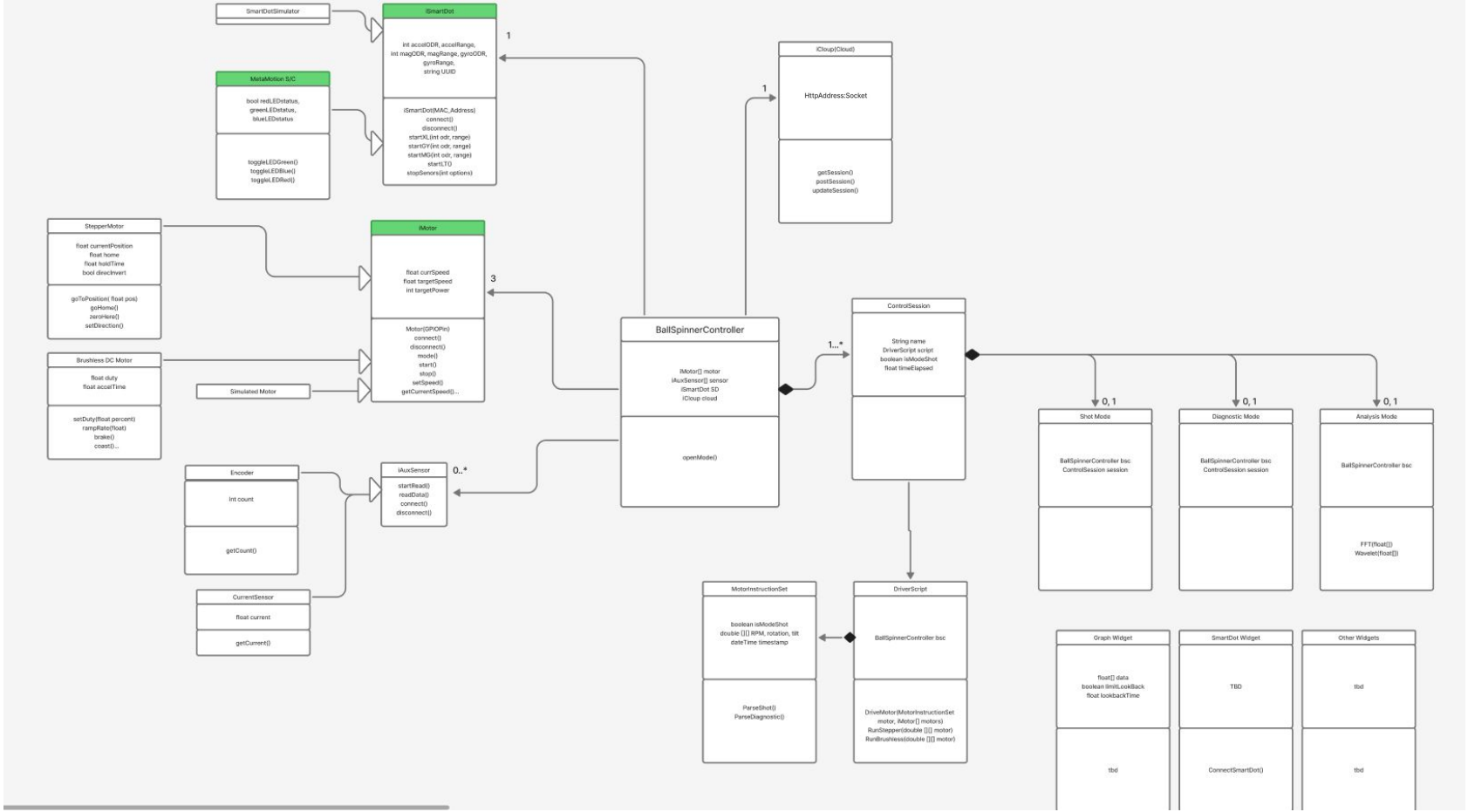
- Main UI with tabbed widgets
- Motor control through the Diagnostic mode (with graphing of motor instructions)
- Compiled and manually linked liblepp, warble, pywarble, metawear-cpp-sdk, metawear-python-sdk for 64 bit
- Graphing real live MetaMotionS data
- Basic logging implementation

Team Pi (Future: MS2)

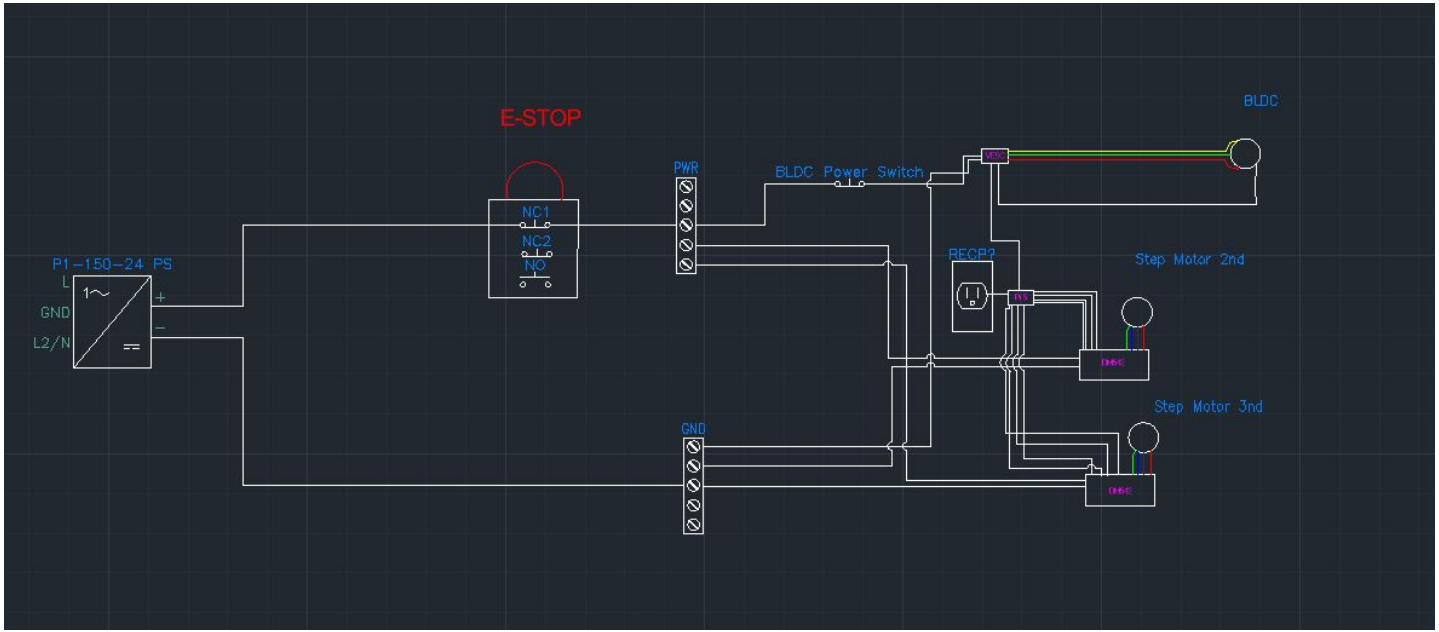
Software

- Organize repository
- E-stop button
- **Unit Test**
- Investigate Frontend Testing utilities
- Add further diagnostic page functionality
- Add logging to all of our important functions
- SmartDot Simulator
- Shot script and database implementation
- Add stored shot/session page

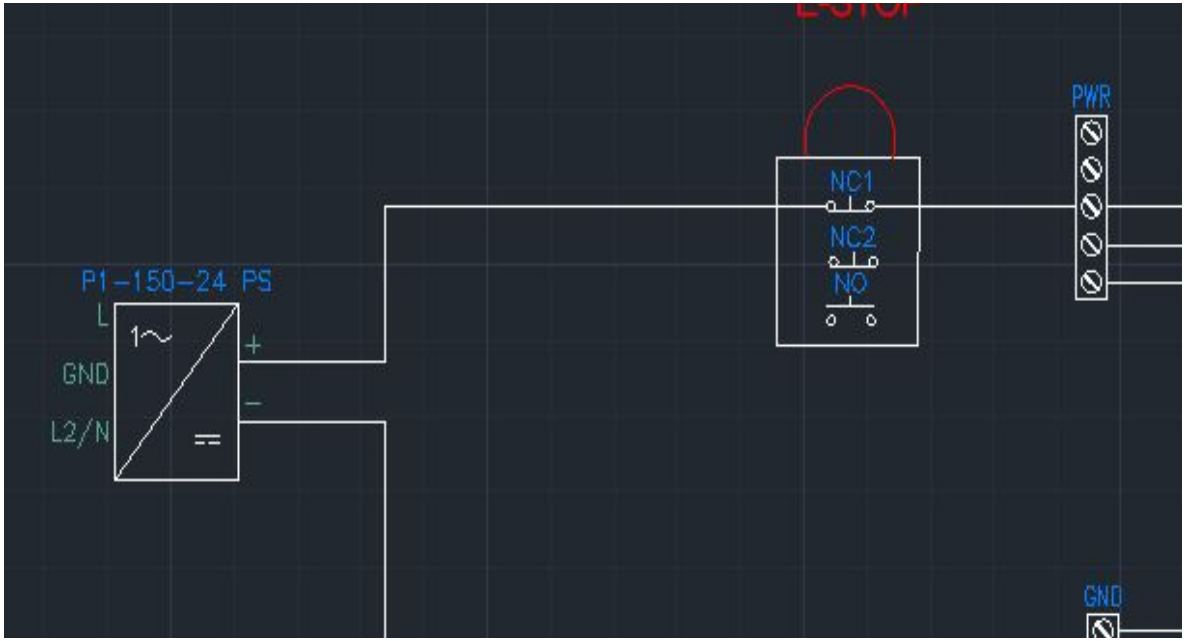
Team Pi UML


SmartDotSimulator
ISmartDot
int accelODR, accelRange,
int magODR, magRange, gyroODR,
gyroRange,
string UUID
ISmartDot(MAC_Address)
connect()
disconnect()
startXL(int odr, range)
startGY(int odr, range)
startIMU(int odr, range)
startLTO
stopSensors(int options)
1
MetaMotion S/C
bool redLEDstatus,
greenLEDstatus,
blueLEDstatus
toggleLEDGreen()
toggleLEDBlue()
toggleLEDRed()
ICloup(Cloud)
HttpAddress:Socket
getSession()
postSession()
updateSession()
1
StepperMotor
float currentPosition
float home
float holdTime
bool direcInvert
goToPosition(float pos)
goHome()
zeroHere()
setDirection()
IMotor
float currSpeed
float targetSpeed
int targetPower
Motor(GPIOPin)
connect()
disconnect()
mode()
start()
stop()
setSpeed()
getCurrentSpeed()...
3
Brushless DC Motor
float duty
float accelTime
setDuty(float percent)
rampRate(float)
brake()
coast()...
Simulated Motor
BallSpinnerController
IMotor[] motor
IAuxSensor[] sensor
ISmartDot SD
ICloup cloud
openMode()
1...*
ControlSession
String name
DriverScript script
boolean isModeShot
float timeElapsed
0, 1
0, 1
0, 1
Shot Mode
BallSpinnerController bsc
ControlSession session
Diagnostic Mode
BallSpinnerController bsc
ControlSession session
Analysis Mode
BallSpinnerController bsc
FFT(float[])
Wavelet(float[])
Encoder
Int count
getCount()
IAuxSensor
startRead()
readData()
connect()
disconnect()
0..*
CurrentSensor
float current
getCurrent()
MotorInstructionSet
boolean isModeShot
double [][] RPM, rotation, tilt
dateTime timestamp
ParseShot()
ParseDiagnostic()
DriverScript
BallSpinnerController bsc
DriveMotor(IMotorInstructionSet
motor, IMotor[] motors)
RunStepper(double [][] motor)
RunBrushless(double [][] motor)
Graph Widget
float[] data
boolean limitLookBack
float lookbackTime
tbd
SmartDot Widget
TBD
ConnectSmartDot()
Other Widgets
tbd
tbd

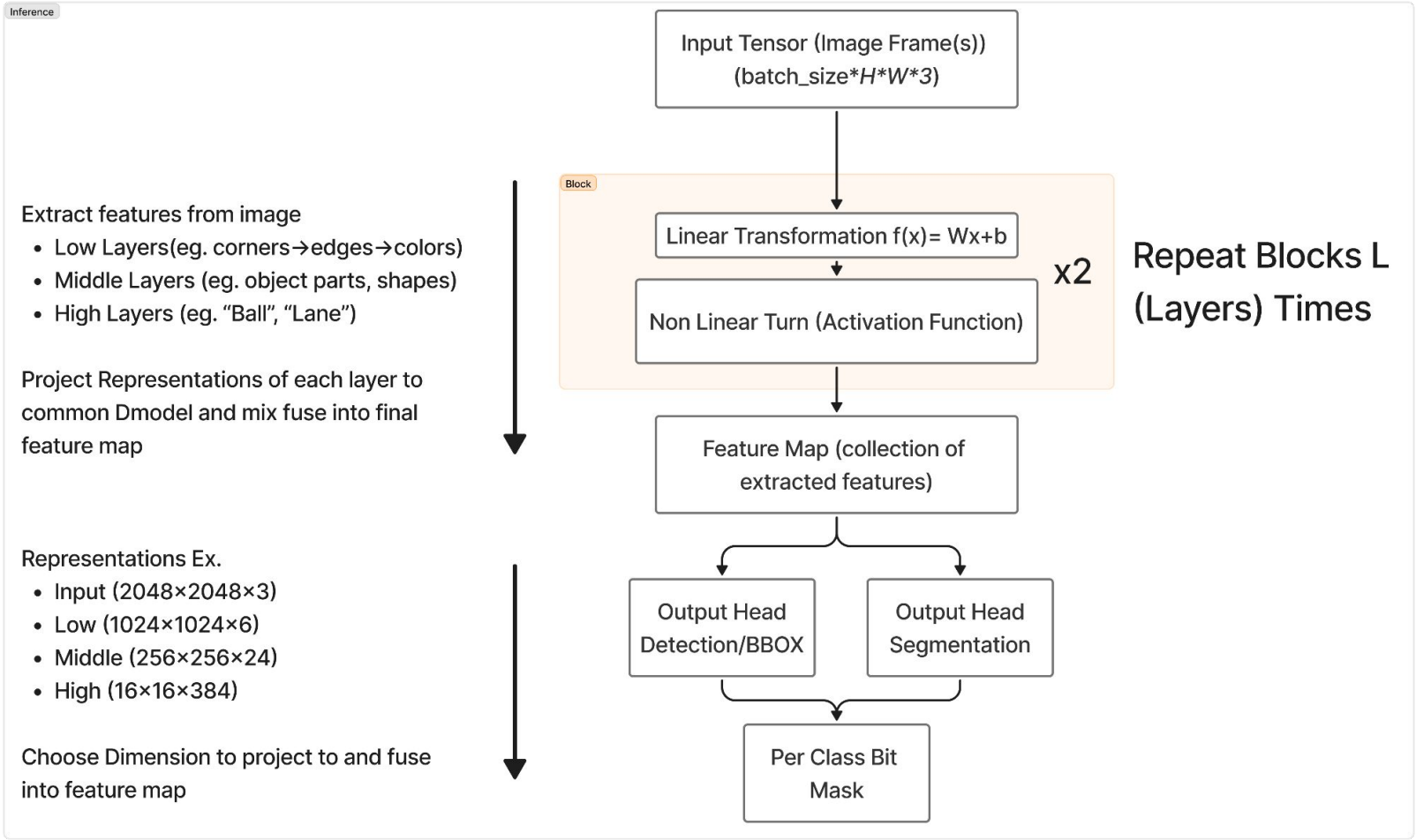
Team Pi Current Electrical System



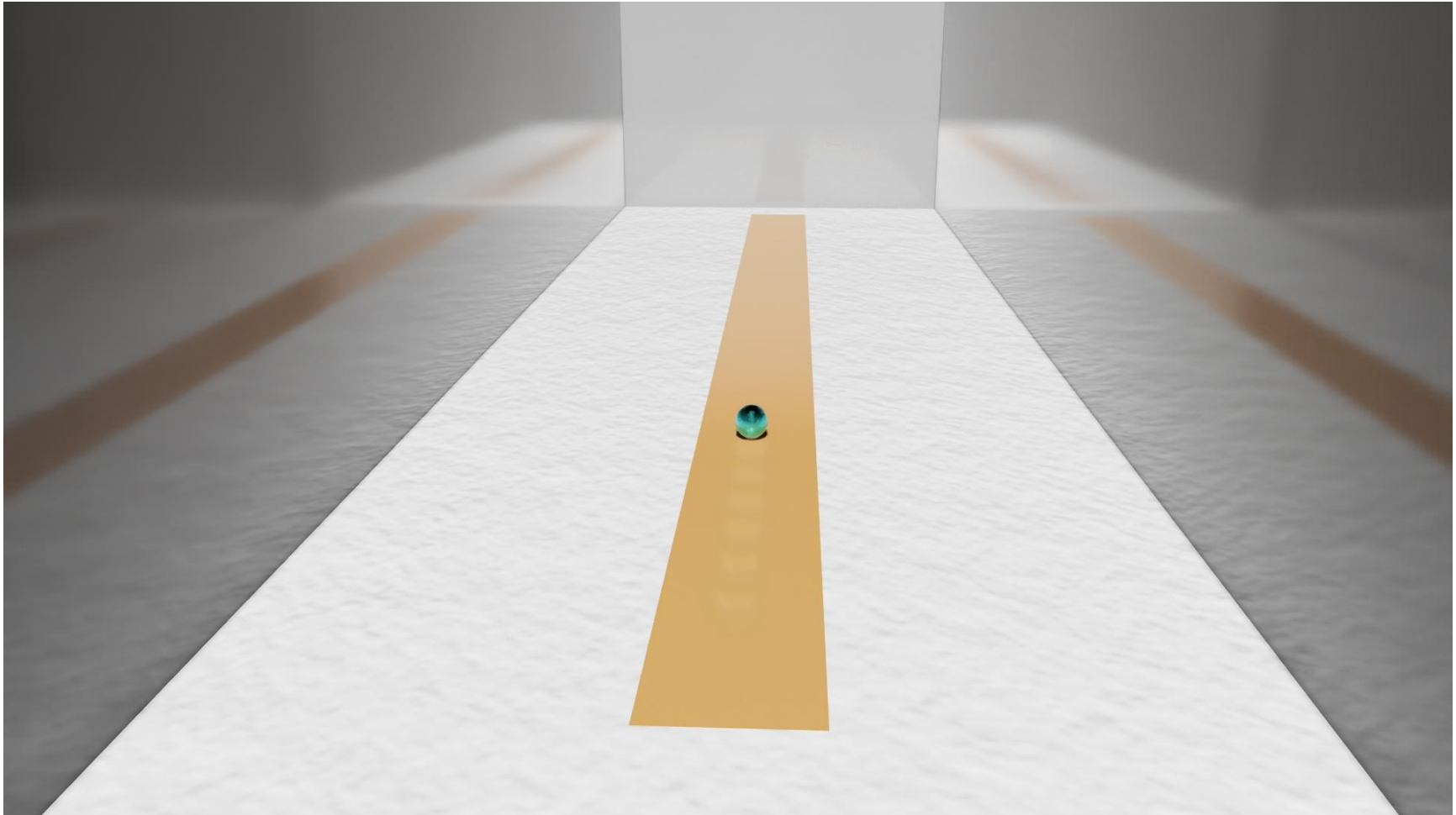
E-STOP



Ciclopes - Overview



Example over Image



Ciclopes - Goals for MS1

- Basic scene in Isaac Sim
- Generate synthetic dataset in Isaac Sim
- Setup training/evaluation engine
- Run first training and evaluate

Ciclopes - Accomplishments for MS1

- Basic scene in Isaac Sim
    - Low rendering quality, lighting, etc.
- Generate synthetic dataset in Isaac Sim
    - 5100 frames (png, polygon labels) - 4590 train, 510 val
- Setup training/evaluation engine
- Run first training and evaluate

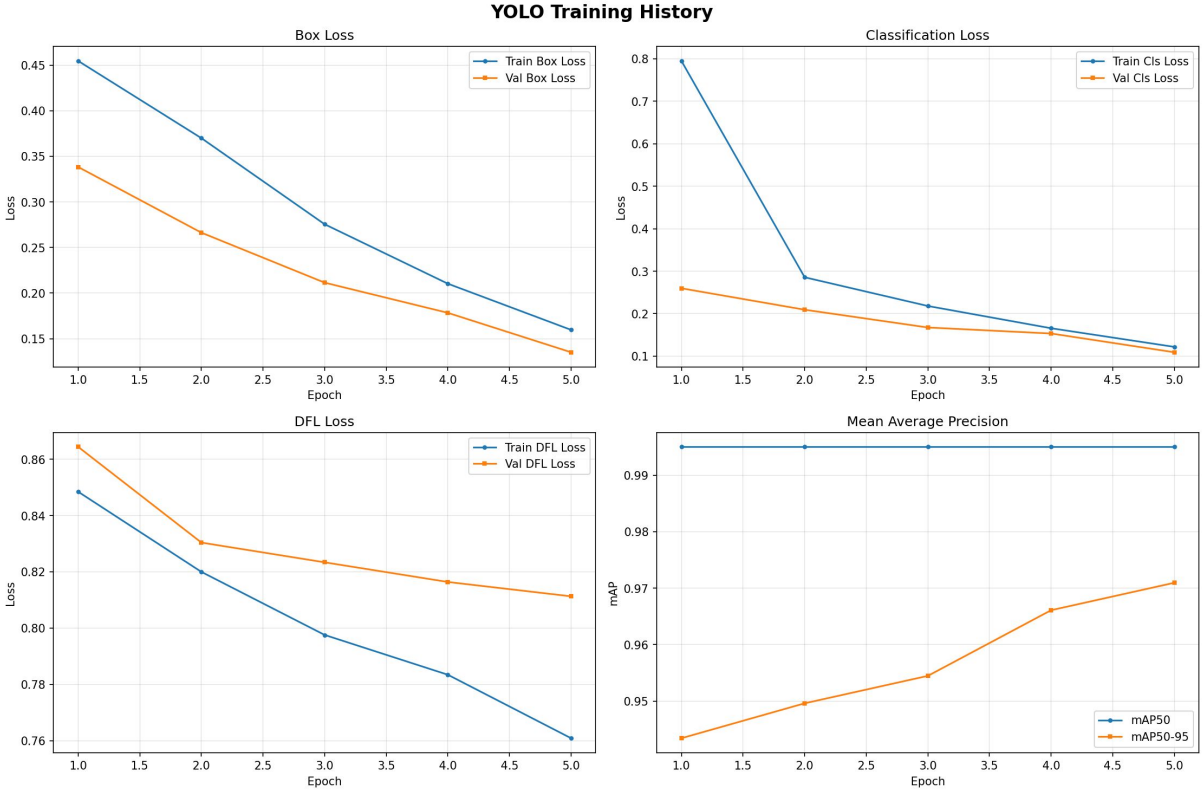
Train Results

- Image Size = 1240x1240 (downsample from 1920x1080 and pad)
- Batch Size = 8

Metrics:

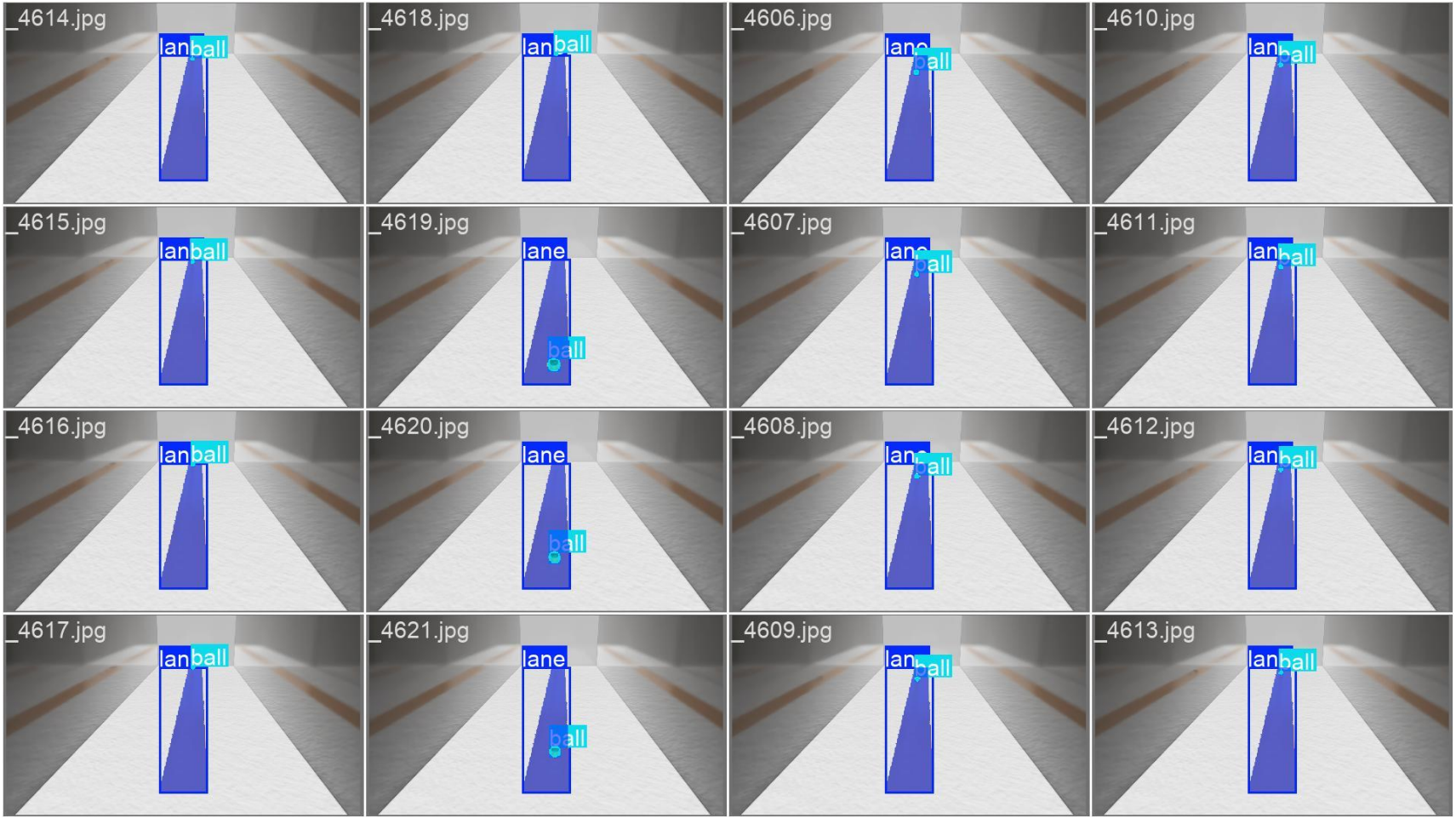
- mAP 50: mean precision across classes - 50% overlapping = correct

- mAP 50-95: mean precision across classes and various overlapping percentages (50-95) - incrementing by 5

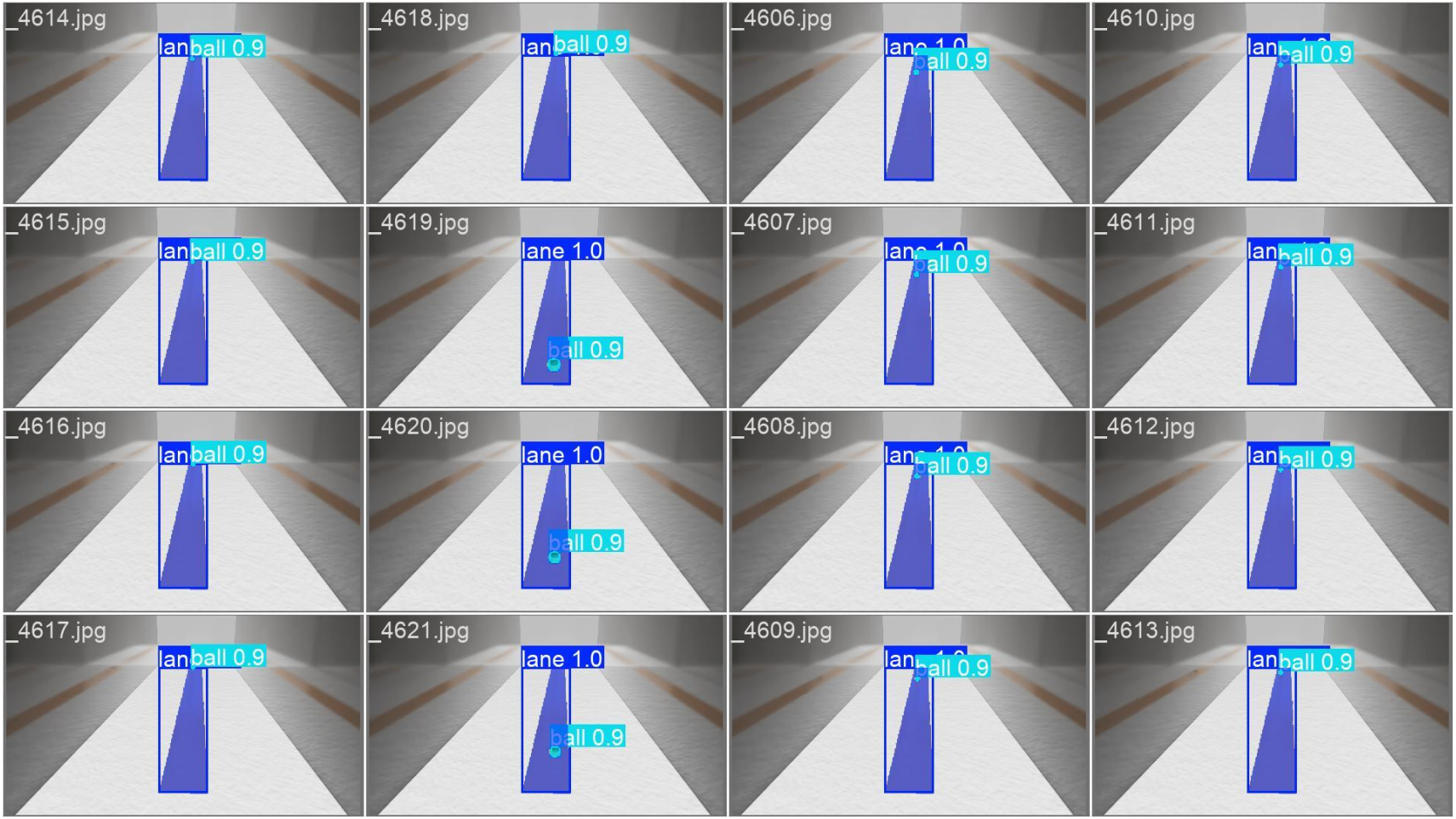


Final mAP 50-95: 0.97 – Almost Perfect, suspiciously high

Train Results - Labels



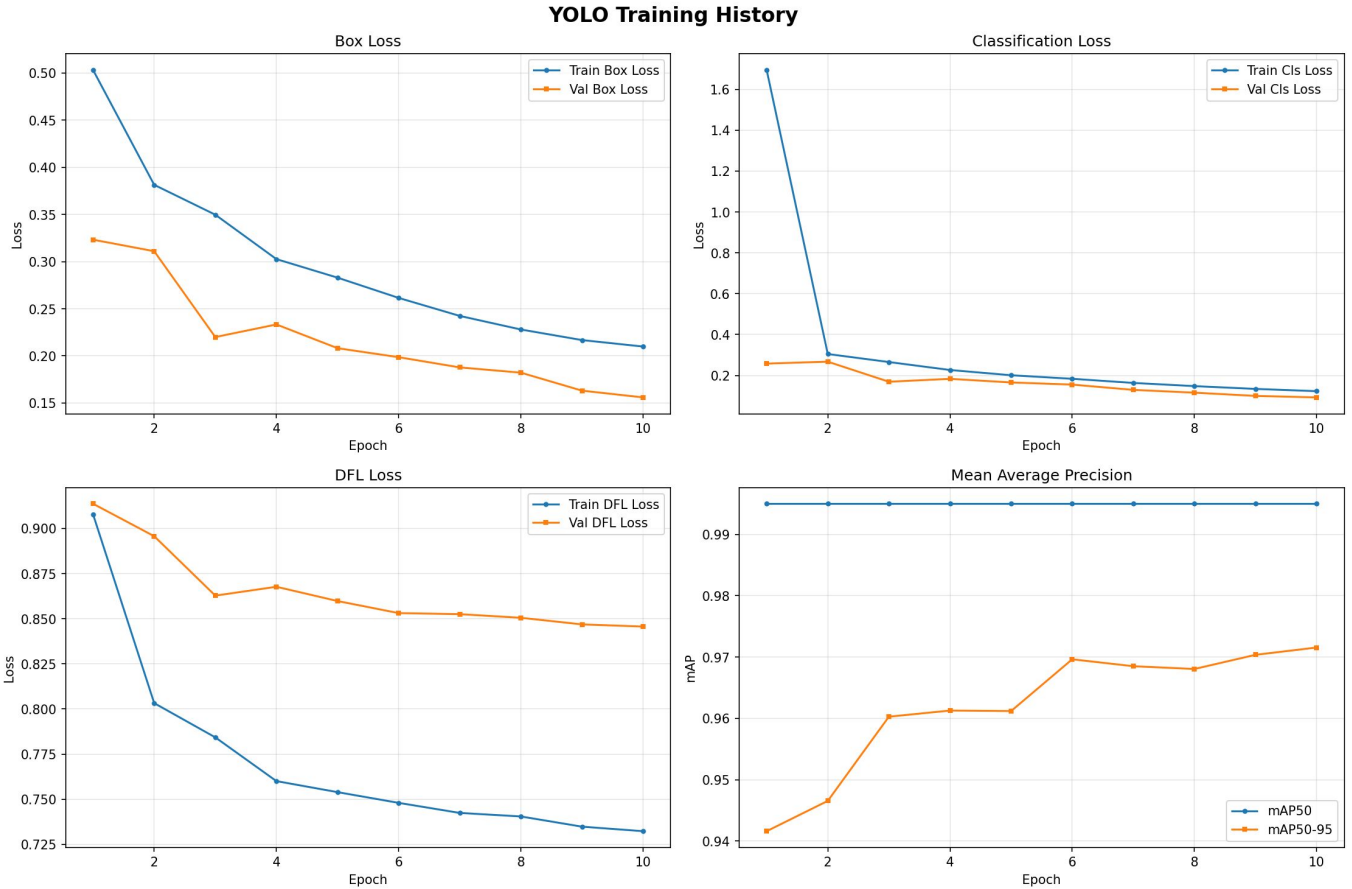
Train Results - Predictions (Inference)



Train Results - “Scaled” Run (Overfitting)

Metrics:

- Image Size = 2048x2048 (Actual 1920x1080 padded inside)
- Smaller Batch Size = 4
- Requires more augmentation to train batches



Ciclopes - Goals for MS2

- Generate episodes of more realistic trajectories for testing
    - Export with per frame BEV positions and kinematics labels
- Develop homography warping functionality
    - Test with labeled BEV positions as ground truth
- Develop BEV episode buffer -> kinematics / features functionality
    - Visualizations? What is actionable data?
    - Post processing tricks (eg. frame interpolation, detection hacks)
- Run end to end segmentation -> warp -> features testing on labeled episodes

Mobile App (MS1 Goals)

- Improve event and session creation/navigation
- Update ball arsenal for ball colors

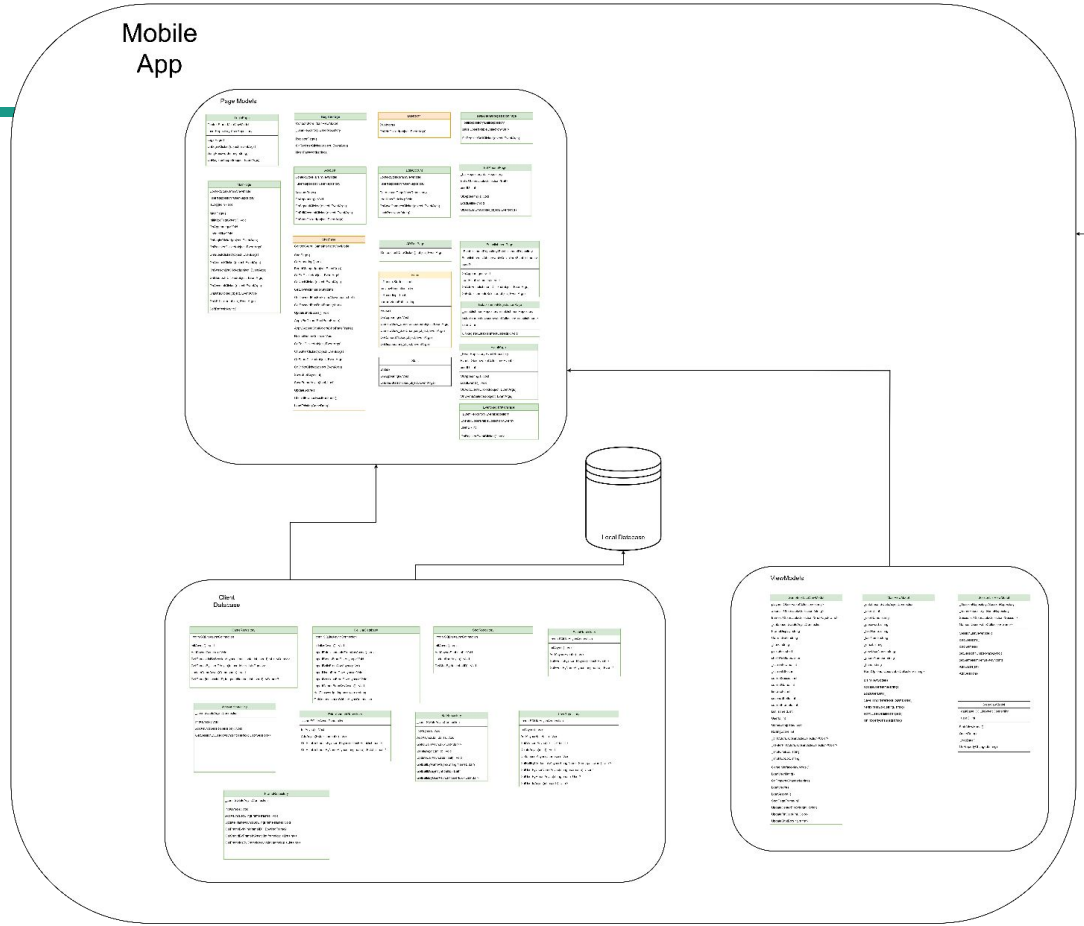
Mobile App (MS1 Achievements)

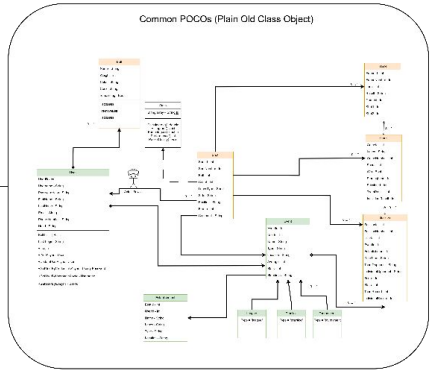
- Events and Sessions
  - New popups for creation
  - Changed navigation from main page to games
- Ball Arsenal
  - Added ball customization for ball color
- CellularCore/Unit Testing
  - DateTimeCalculator
  - StringToColorConverter

Mobile App (Future: MS2)

- Improve Stats Page
  - Add onto bits used for shots
  - Calculate more stats from the database

Mobile UML Overview



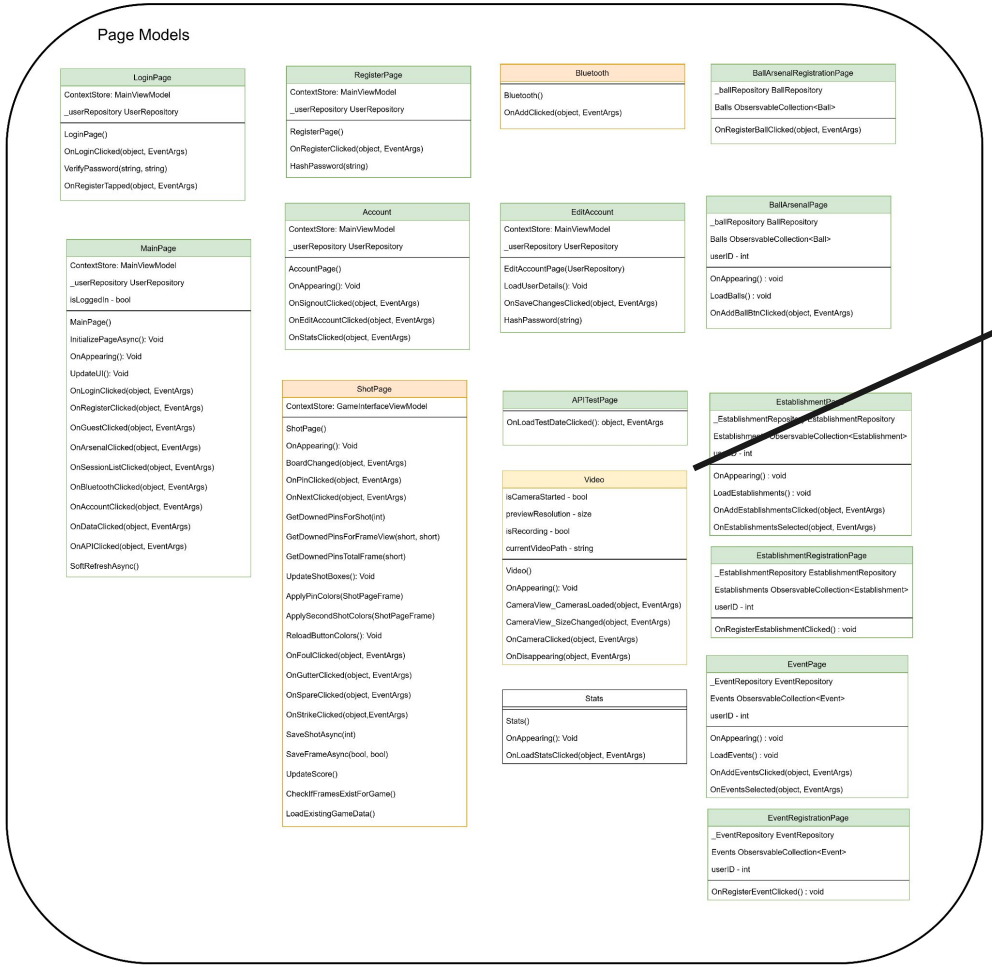


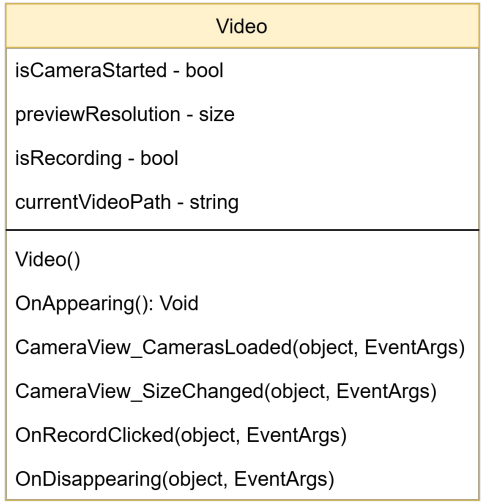
Implemented

Partially Implemented

Not Yet Implemented

Mobile UML - Page Models



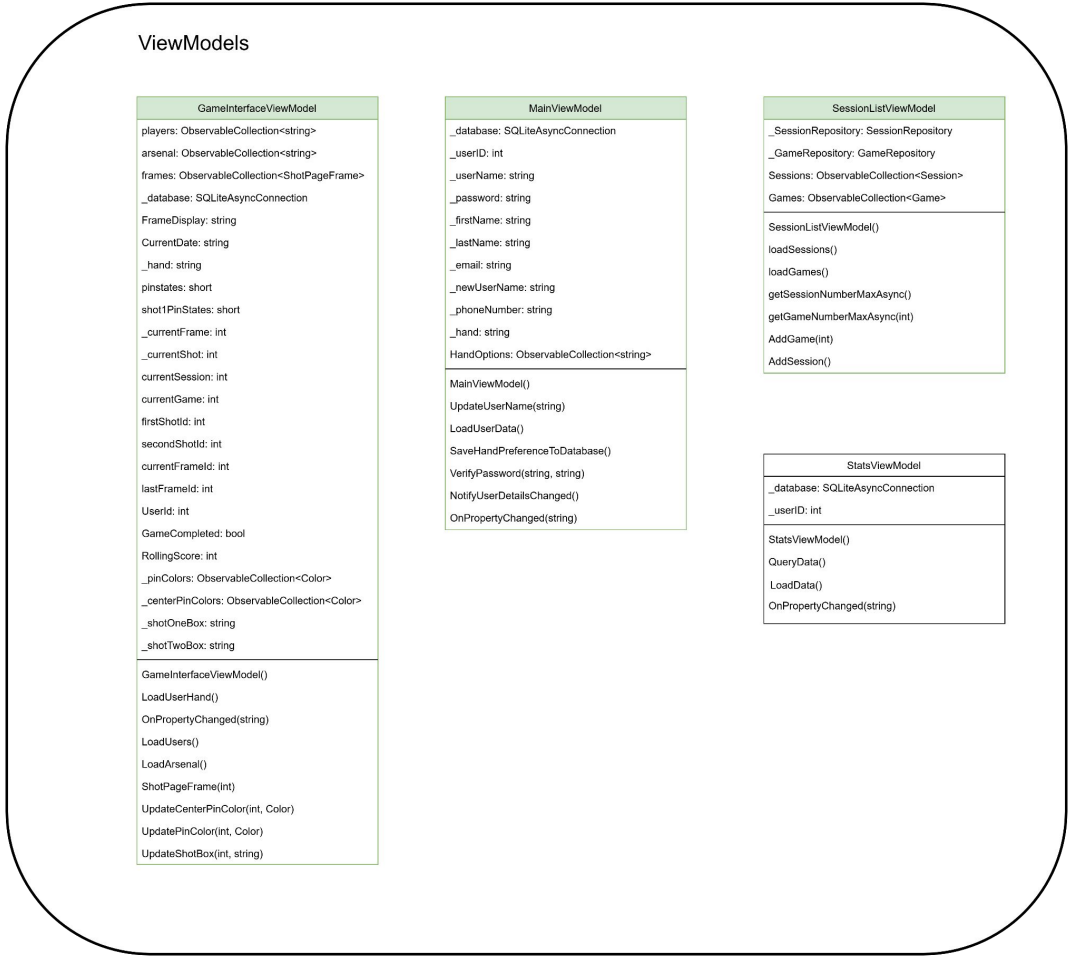


Implemented

Partially Implemented

Not Yet Implemented

Mobile UML - View Models



Implemented

Partially Implemented

Not Yet Implemented

Mobile UML - Database

Client
Database

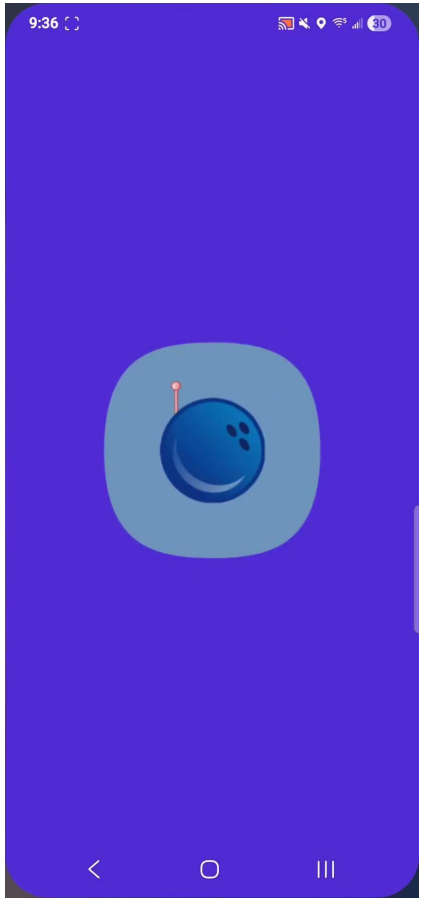
Mobile App - Event and Session Creation

- Josh put stuff here - Josh

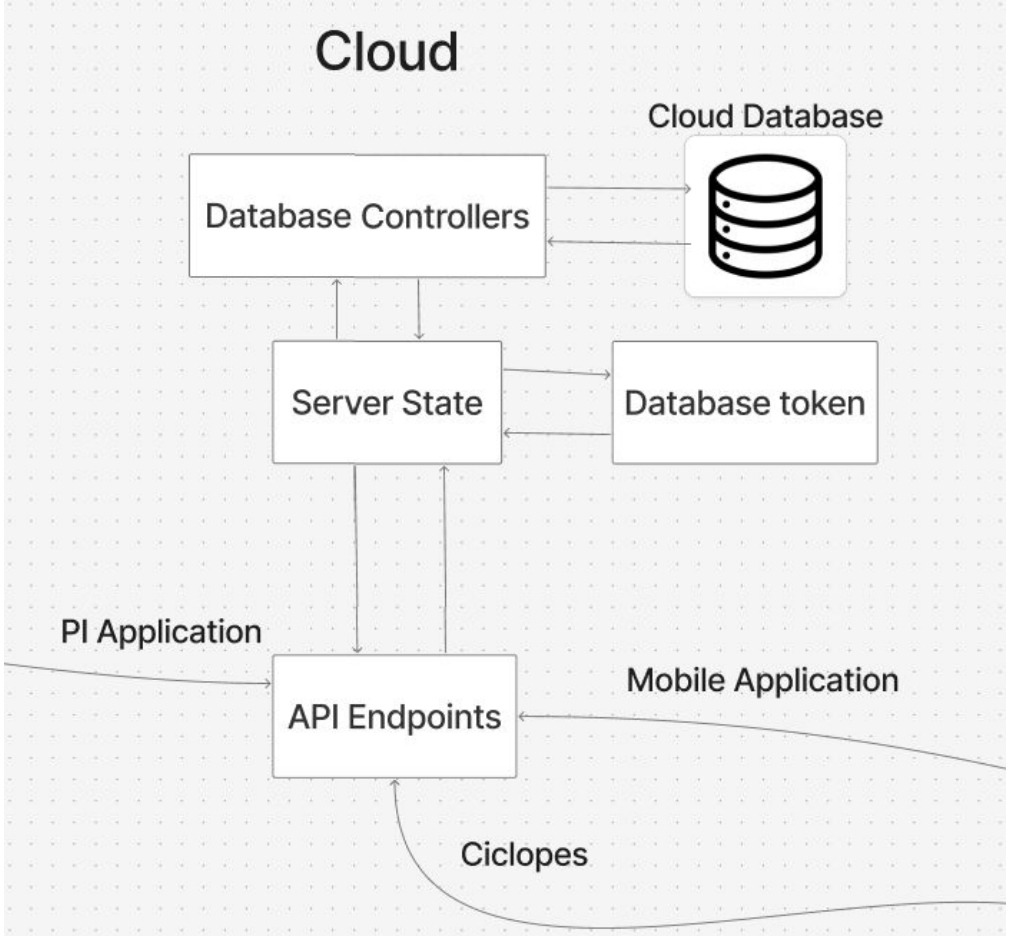
Mobile App - Ball Arsenal Updated

- Josh put stuff here - Josh

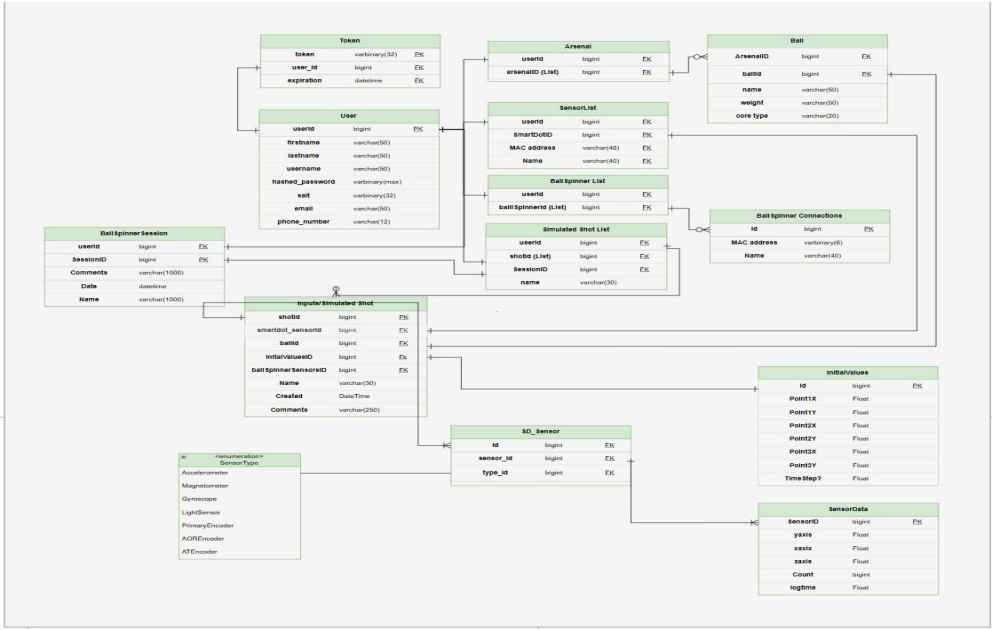
Mobile App - Demo!

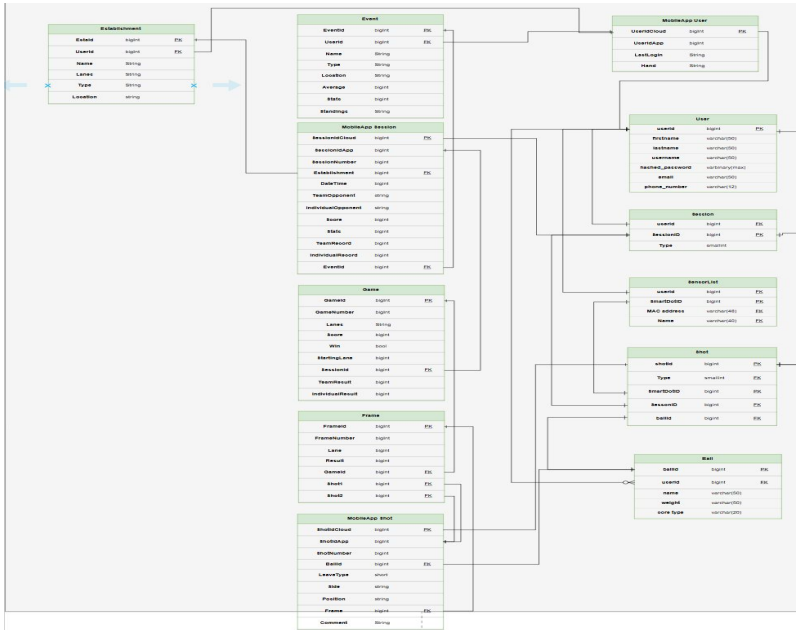


Backend/Cloud High-Level Overview

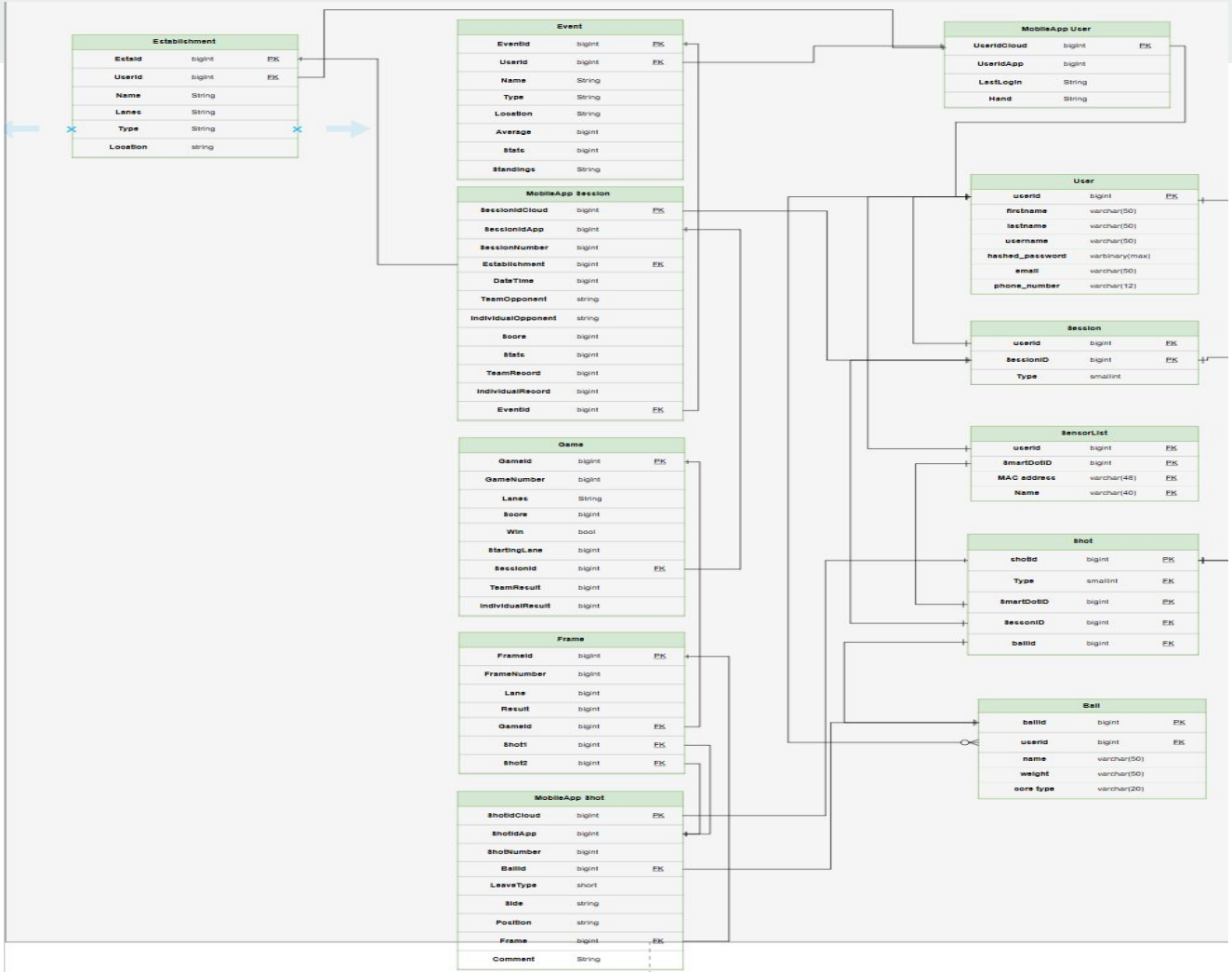


Backend/Cloud Database Schema

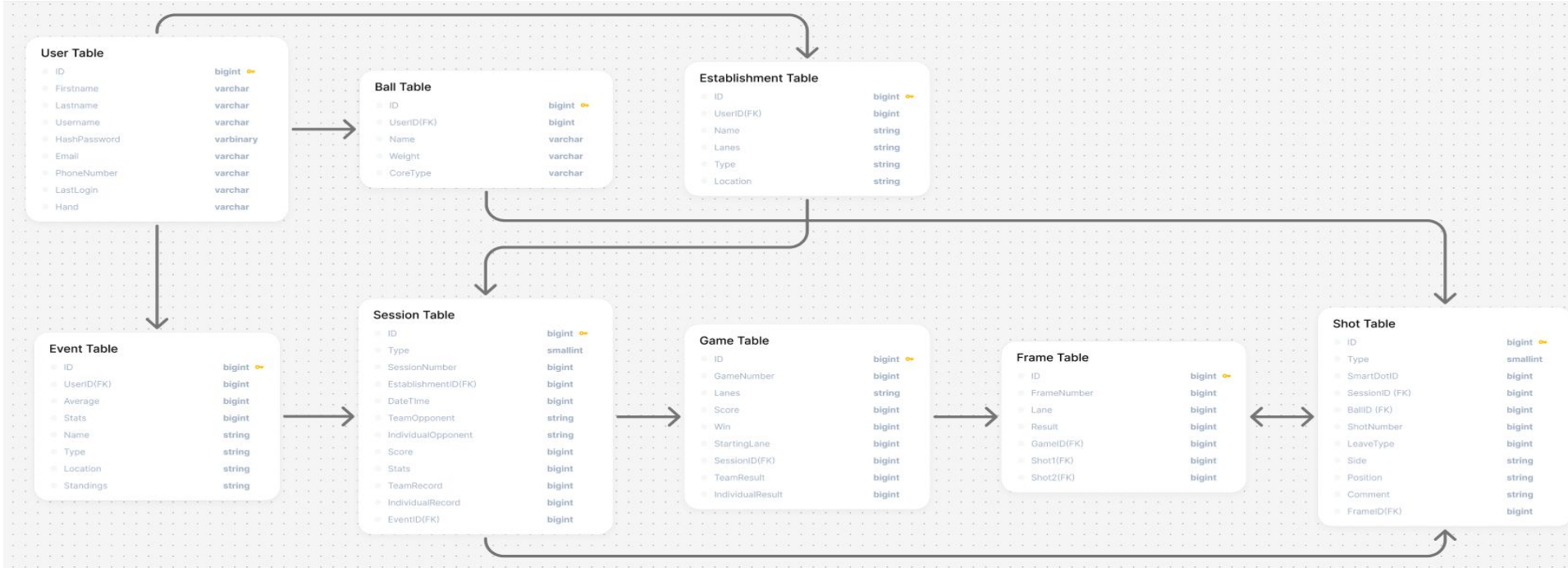




Old Schema


Establishment
EstabId bigint PK
UserId bigint FK
Name String
Lanes String
Type String
Location string
Event
EventId bigint PK
UserId bigint FK
Name String
Type String
Location String
Average bigint
Static bigint
Standings String
MobileApp Session
SessionIdCloud bigint PK
SessionIdApp bigint
SessionNumber bigint
Establishment bigint FK
DateTime bigint
TeamOpponent string
IndividualOpponent string
Score bigint
Static bigint
TeamRecord bigint
IndividualRecord bigint
EventId bigint FK
Game
GameId bigint PK
GameNumber bigint
Lanes String
Score bigint
Win bool
StartingLane bigint
SessionId bigint FK
TeamResult bigint
IndividualResult bigint
Frame
FrameId bigint PK
FrameNumber bigint
Lane bigint
Result bigint
GameId bigint FK
Shot1 bigint FK
Shot2 bigint FK
MobileApp Shot
ShotIdCloud bigint PK
ShotIdApp bigint
ShotNumber bigint
BallId bigint FK
LeaveType short
Side string
Position string
Frame bigint FK
Comment String
MobileApp User
UserIdCloud bigint PK
UserIdApp bigint
LastLogin String
Hand String
User
userid bigint PK
firstname varchar(50)
lastname varchar(50)
username varchar(50)
hashed_password varbinary(max)
email varchar(50)
phone_number varchar(12)
Session
userid bigint FK
SessionID bigint PK
Type smallint
SensorList
userid bigint FK
SmartDotID bigint FK
MAC address varchar(48) FK
Name varchar(40) FK
Shot
shotId bigint PK
Type smallint FK
SmartDotID bigint PK
SessonID bigint FK
ballId bigint FK
Ball
ballId bigint PK
userid bigint FK
name varchar(50)
weight varchar(50)
core type varchar(20)

New Schema


User Table
ID bigint
Firstname varchar
Lastname varchar
Username varchar
HashPassword varbinary
Email varchar
PhoneNumber varchar
LastLogin varchar
Hand varchar
Ball Table
ID bigint
UserID(FK) bigint
Name varchar
Weight varchar
CoreType varchar
Establishment Table
ID bigint
UserID(FK) bigint
Name string
Lanes string
Type string
Location string
Event Table
ID bigint
UserID(FK) bigint
Average bigint
Stats bigint
Name string
Type string
Location string
Standings string
Session Table
ID bigint
Type smallint
SessionNumber bigint
EstablishmentID(FK) bigint
DateTime bigint
TeamOpponent string
IndividualOpponent string
Score bigint
Stats bigint
TeamRecord bigint
IndividualRecord bigint
EventID(FK) bigint
Game Table
ID bigint
GameNumber bigint
Lanes string
Score bigint
Win bigint
StartingLane bigint
SessionID(FK) bigint
TeamResult bigint
IndividualResult bigint
Frame Table
ID bigint
FrameNumber bigint
Lane bigint
Result bigint
GameID(FK) bigint
Shot1(FK) bigint
Shot2(FK) bigint
Shot Table
ID bigint
Type smallint
SmartDotID bigint
SessionID (FK) bigint
BallID (FK) bigint
ShotNumber bigint
LeaveType bigint
Side string
Position string
Comment string
FrameID(FK) bigint

Cloud (MS1 Goals)

- Implement Automated Migrations
- API Improvements/Additions

Automated Migrations

Before:

- Writing raw SQL in DB changelog to add new tables
- Needed to write SQL to modify tables and keep track of what modifications are being made
- Difficult to have multiple developers working on adding/removing/modifying DB as changelog is a single file

Now:

- Migrations are managed by Microsoft EF Core SQL tools
- C# classes instead of writing SQL
- Migration generation and script file writing
- Multiple people can safely create/modify DB without interfering

Example C# DB Object Class:

```
public class EstablishmentTable
{
    [Key]
    public int ID { get; set; }
    [Required]
    public string Name { get; set; }
    [Required]
    public string Lanes { get; set; }
    [Required]
    public string Type { get; set; }
    [Required]
    public string Location { get; set; }
}
```

Auto-generated SQL script code for same class:

```
BEGIN TRANSACTION;
GO

IF NOT EXISTS(SELECT * FROM [__EFMigrationsHistory] WHERE [MigrationId] = N'20251029015335_UsersEstabGamesSeshShots')
BEGIN
    CREATE TABLE [combinedDB].[Establishments] (
        [ID] int NOT NULL IDENTITY,
        [Name] nvarchar(max) NOT NULL,
        [Lanes] nvarchar(max) NOT NULL,
        [Type] nvarchar(max) NOT NULL,
        [Location] nvarchar(max) NOT NULL,
        CONSTRAINT [PK_Establishments] PRIMARY KEY ([ID])
    );
END;
GO
```

API Improvements/Additions

- 18 new endpoints added
- All endpoints for create/retrieval
- All new DB objects have corresponding endpoints

GET /api/gets/GetAppUsers

GET /api/gets/GetAppShots

GET /api/gets/GetAppSessions

GET /api/gets/GetAppGames

GET /api/gets/GetAppEstablishments

GET /api/gets/GetBallsByUsername

GET /api/gets/GetEventsByUsername

GET /api/gets/GetFramesByGameId

GET /api/gets/GetShotsByUsername

POST /api/posts/PostBalls

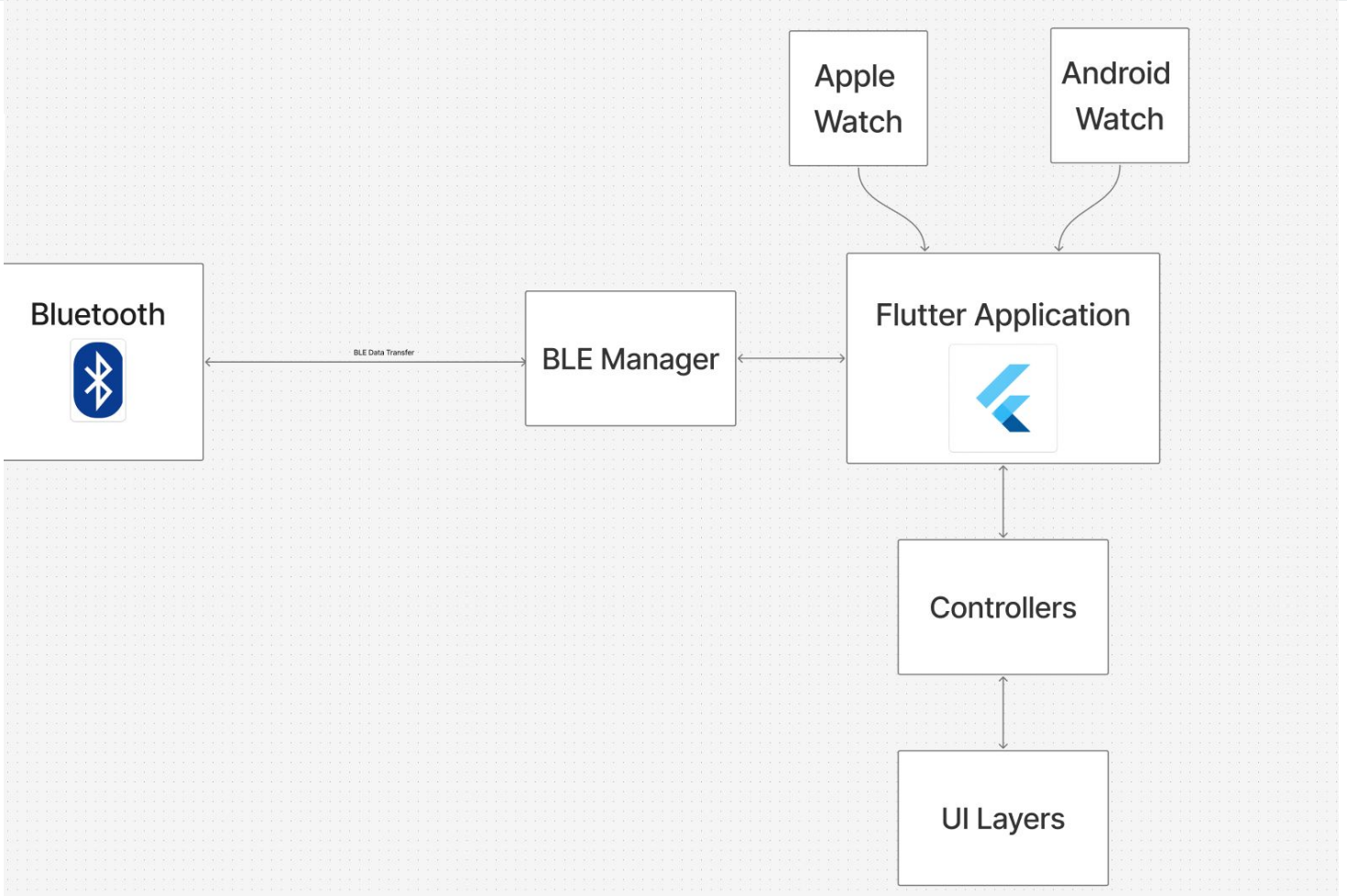
POST /api/posts/PostEvent

POST /api/posts/PostFrames

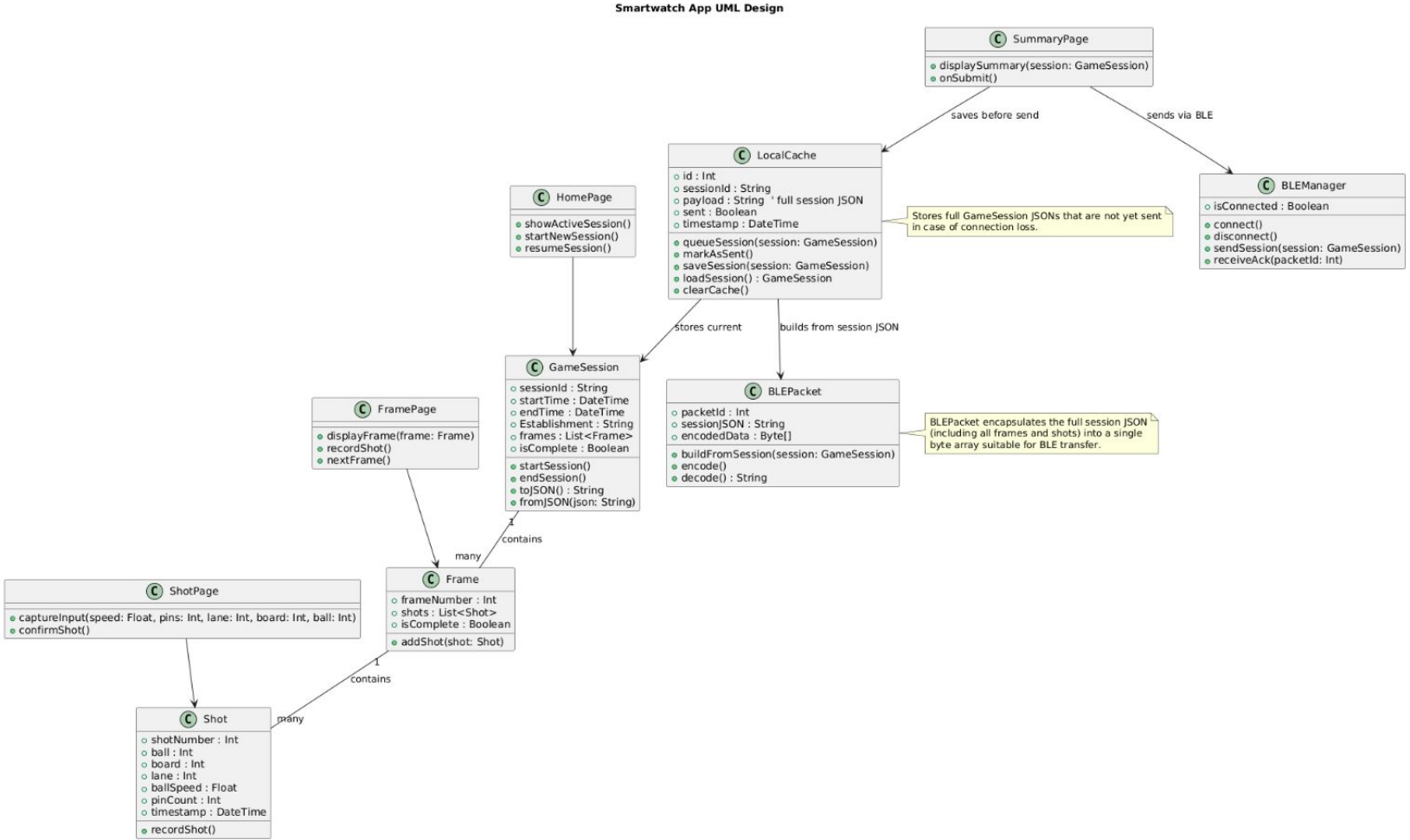
Cloud (Future: MS2)

- **Implement Pi Cloud Database**

Smart Watch High-Level Overview

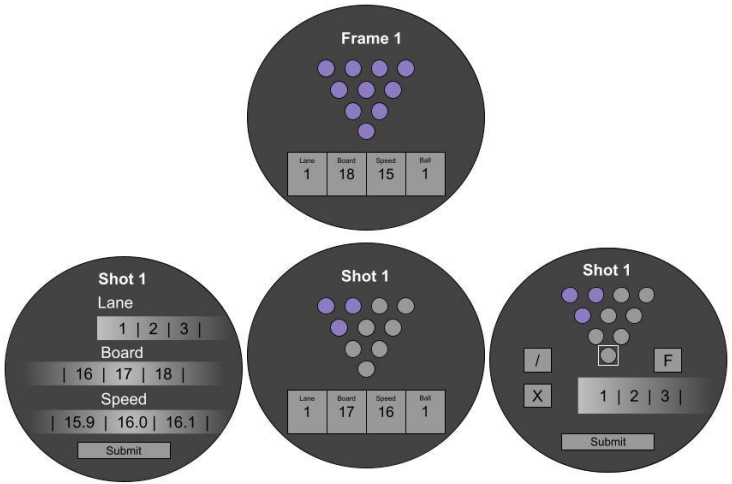


Smart Watch UML



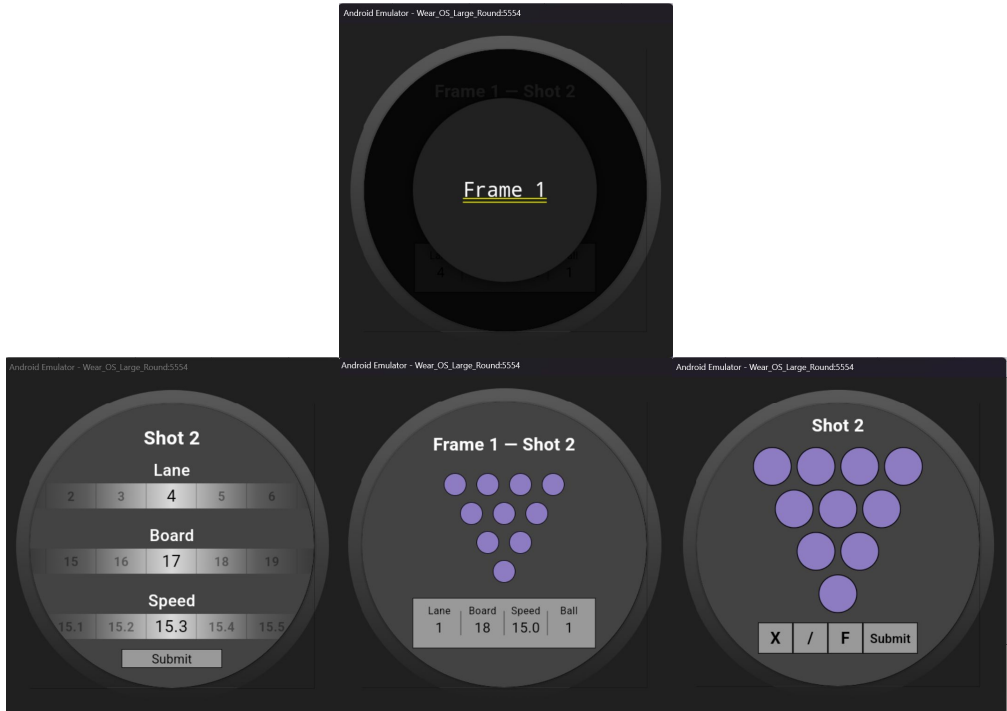
Smart Watch (MS1 Goals)

- Research and select env/language
  - Kotlin MP/React Native/Flutter
- Create basic navigation/codebase in selected language
- Refine sketches/functions
- Find Bluetooth libraries
- Understand watch storage/best practices



Smart Watch (MS1 Achievements)

- Flutter/dart
- Android watch emulator
- Created a starting platform
  - Basic navigation
  - Starting UI/functionality
  - Architecture from UML
- Found existing libraries/open source apps
  - BLE
  - Samsung Health
- Implemented Unit Tests



Smart Watch (Future MS2)

- Object based data storage/manipulation/passing
  - Game ->Frames->Shots
  - Provides dynamic interaction
  - Less overhead
- Fully functioning shot input and additional data
- Bluetooth connection

Smartwatch Demo