BNU

BEACONHOUSE NATIONAL UNIVERSITY

ShotSense

PRJ-F23/337

Design Documentation

EXTERNAL SUPERVISOR

Mirza Danial Masood

INTERNAL SUPERVISORS

Huda Sarfraz

GROUP MEMBERS

Umair Ali Khan	F2020-532
Muhammad Fahad	F2020-157

SCHOOL OF COMPUTER & IT

January 2024

Table of Contents

Introduction

This design document entails the comprehensive overview of the design and system architecture for ShotSense mobile application. The application is a sophisticated platform designed to enhance cricket players' performance by leveraging cutting-edge technologies, including machine learning and computer vision. ShotSense provides users with a holistic experience, from recording and analyzing individual shots to offering personalized insights and recommendations for skill improvement.
In this document, we will delve into the architectural components and the integration of artificial intelligence models that power key features such as shot classification, pose estimation, and ball tracking. The robustness of the design ensures scalability, security, and usability, aligning with the application's core objectives.

Technologies and Development Tools

Front-End

- **Flutter[1]**
  A versatile UI toolkit for building natively compiled applications across mobile, web, and desktop from a single codebase.

- **Dart**
  The programming language optimized for building mobile, desktop, server, and web applications with Flutter.

Back-End

- **Firebase Firestore[12]**
  A NoSQL document database for efficient and scalable data storage, part of the Firebase platform.

- **NodeJS[3]**
  A server-side JavaScript runtime for building scalable network applications.

Storage

- **Google Cloud Storage[7]**
  A scalable and secure object storage solution for storing and retrieving data on Google Cloud Platform.

Machine Learning Technologies

- **TensorFlow[4]**
  An open-source machine learning framework for building and deploying machine learning models.

- **Keras**
  A high-level neural networks API, running on top of TensorFlow, facilitating easy and fast prototyping.

- **YOLOv8[2]**
  You Only Look Once, .An AI model which gives real time pose tracking and ball tracking ability

- **Google Cloud Vertex AI[7]**
  A managed machine learning (ML) platform for building, training, and deploying models on Google Cloud.

- **Google AutoML[11]**
  A user-friendly tool that simplifies the process of training machine learning models without extensive expertise.

Services and Cloud Functions

- **Google Firebase[12]**
  A comprehensive mobile and web application development platform offering various services like authentication, real-time databases, and more.

- **Google Cloud Functions[7]**
  Serverless functions that run in response to events, automatically scaling with the number of executions.

Video Processing for dataset

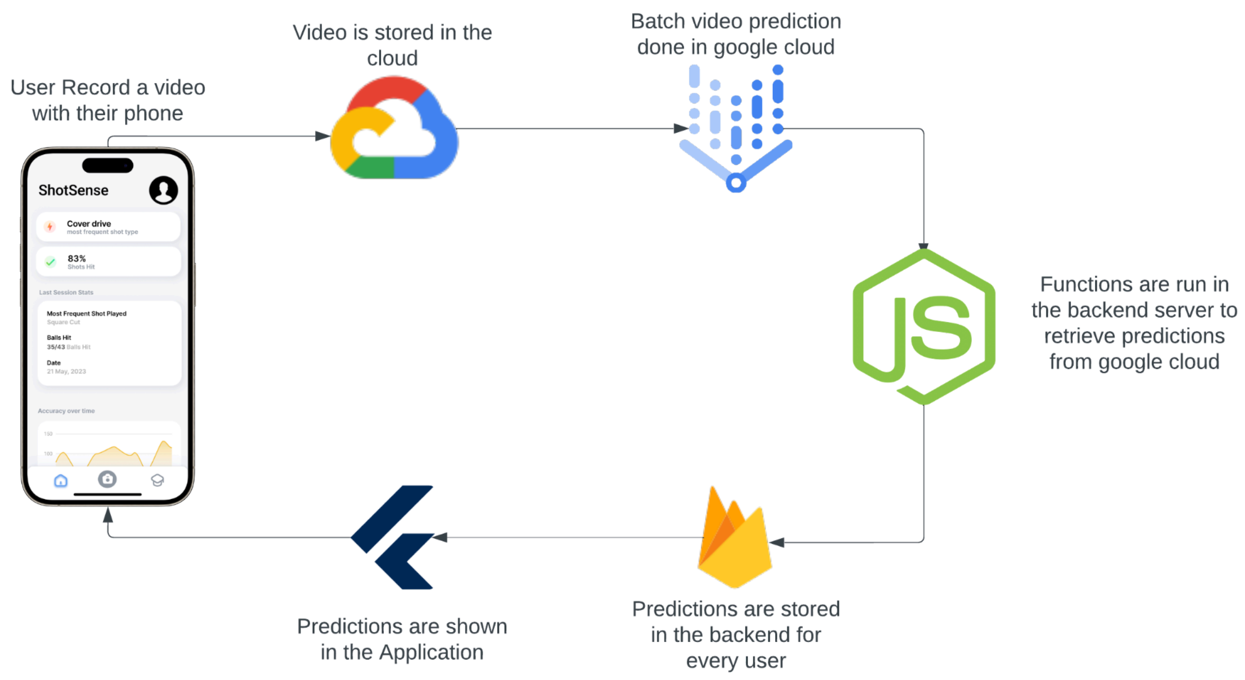
- **OpenCV**
  An open-source computer vision and machine learning software library for image and video processing.

- **Ffmpeg**
  A multimedia processing tool that handles multimedia data such as audio and video

- **Microsoft ClipChamp**
  A video processing tool for editing and compressing videos.

High Level Application Diagram

A High level Diagram showing complete working of the entire application and its related backend through which a user would get classification for their shot played. As a user upload videos of them playing cricket the video gets uploaded to the cloud. Using node server and google's vertex AI we use batch prediction for the videos of complete over(6 balls) recorded by the user. The videos are processed by the shot classification AI and the predictions for each ball is stored in a firebase firestore collection from where the prediction for each shot is displayed to the user in a user-friendly format



User Record a video with their phone
ShotSense
Cover drive
most frequent shot type
83%
Shots Hit
Last Session Stats
Most Frequent Shot Played
Square Cut
Balls Hit
35/43 Balls Hit
Date
21 May, 2023
Accuracy over time
Video is stored in the cloud
Batch video prediction done in google cloud
Functions are run in the backend server to retrieve predictions from google cloud
Predictions are stored in the backend for every user
Predictions are shown in the Application

Figure 1 High Level Diagram showing all the processes when a video is uploaded

Mobile Application Flow Diagram

The application operates seamlessly, offering users a comprehensive platform for analyzing their cricket shots. Upon video upload to the cloud, a robust backend, powered by a Node server and Google's Vertex AI, orchestrates the processing through batch prediction of vertex ai. This AI system discerns shot classifications from videos capturing an entire over (6 balls) played by the user. Using shot classification AI, predictions for each ball are meticulously generated and stored in a Firebase Firestore collection. The user is then presented with a user-friendly display, showcasing detailed shot predictions derived from the processed videos

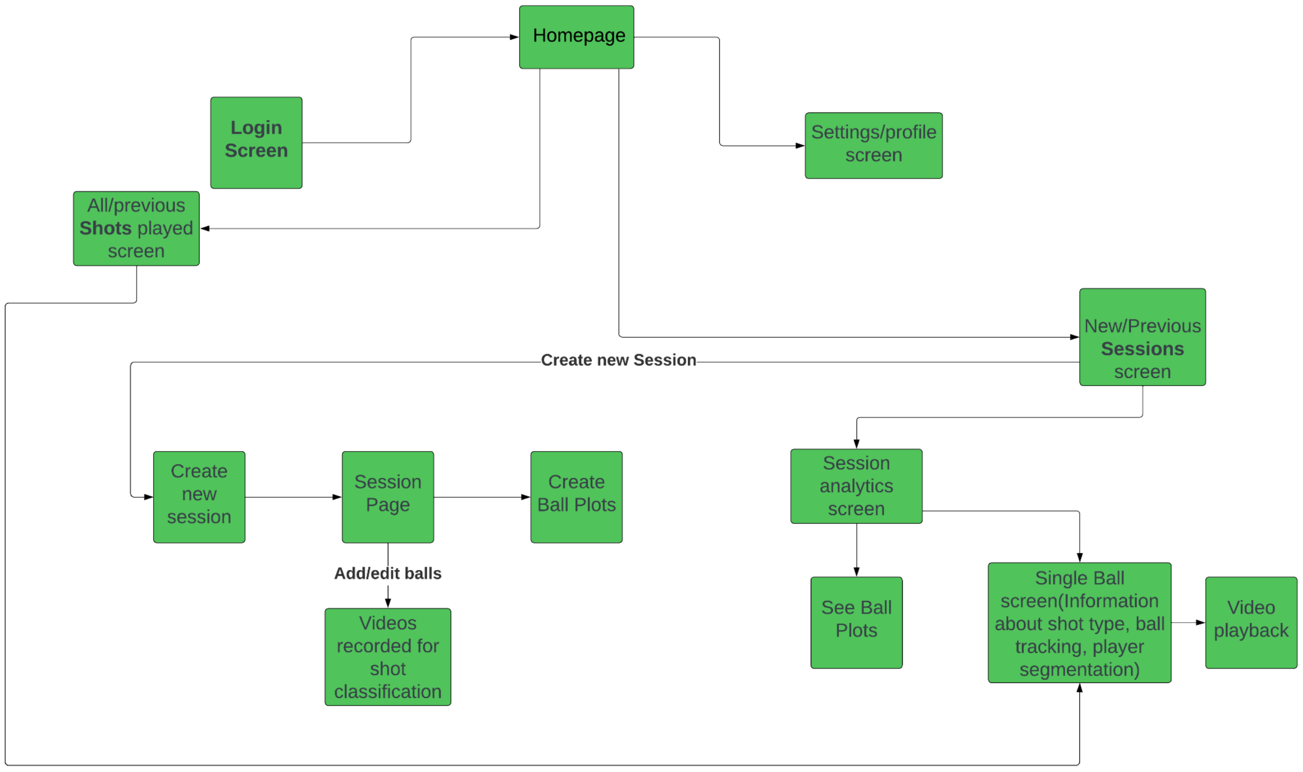

Homepage
Login Screen
Settings/profile screen
All/previous Shots played screen
New/Previous Sessions screen
Create new Session
Create new session
Session Page
Create Ball Plots
Add/edit balls
Videos recorded for shot classification
Session analytics screen
See Ball Plots
Single Ball screen(Information about shot type, ball tracking, player segmentation)
Video playback

Figure 2 Application Flow Diagram

System Architecture (C4 Diagram)

The C4 architecture diagram[8] breaks down our project into three main levels: software system, container, and components. Looking at the software system level, we have a snapshot of the entire project, mainly divided into two important parts: the core application and the Backend for AI.

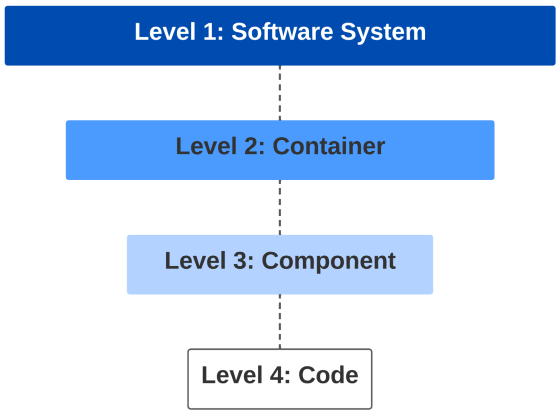


Figure 3

Level 1: Software System

Software system shows an overview of the project separated by application. The Backend for AI acts like a bridge, making sure the AI model and the main application can work together smoothly. This diagram helps us see how everything fits together in shotsense.

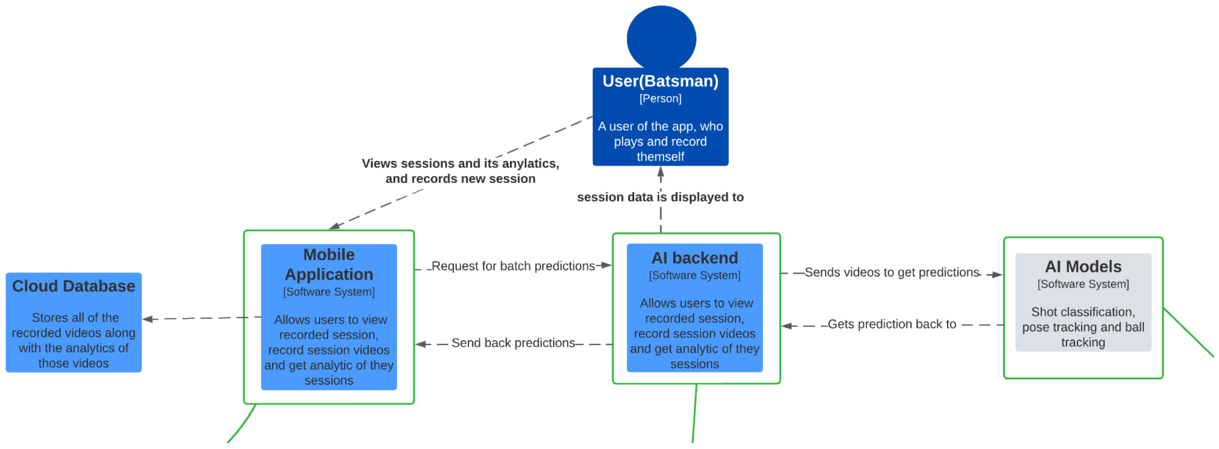

User(Batsman)
[Person]
A user of the app, who plays and record themself
Views sessions and its anylatics, and records new session
session data is displayed to
Cloud Database
Stores all of the recorded videos along with the analytics of those videos
Mobile Application
[Software System]
Allows users to view recorded session, record session videos and get analytic of they sessions
Request for batch predictions
Send back predictions
AI backend
[Software System]
Allows users to view recorded session, record session videos and get analytic of they sessions
Sends videos to get predictions
Gets prediction back to
AI Models
[Software System]
Shot classification, pose tracking and ball tracking

Figure 4 The software system

Level 2: Containers

The container level of our system architecture provides insight into the internal mechanisms of the mobile application, AI backend, and AI models.

The Mobile Application Container

The Mobile Application (refer to Figure 5) functions as the user interface, allowing users to record and upload videos. It captures video data from the user's device, forwards it to the AI backend for processing, and stores the predictions in the Firebase database.

The AI Backend Container

The AI Backend (see Figure 6) is composed of a node server that handles requests from the mobile application. It creates a JSON Lines file, containing URIs of all the videos uploaded to cloud storage. This file is then utilized to generate predictions for the requested videos.

AI Models Container

AI Models (depicted in Figure 7) showcase various models integral to the application. Pose tracking enhances video playback by illustrating body movements of the player, complemented by player segmentation (refer to the implementation document for details). Ball tracking is employed to generate ball plots and heatmaps for session analysis.

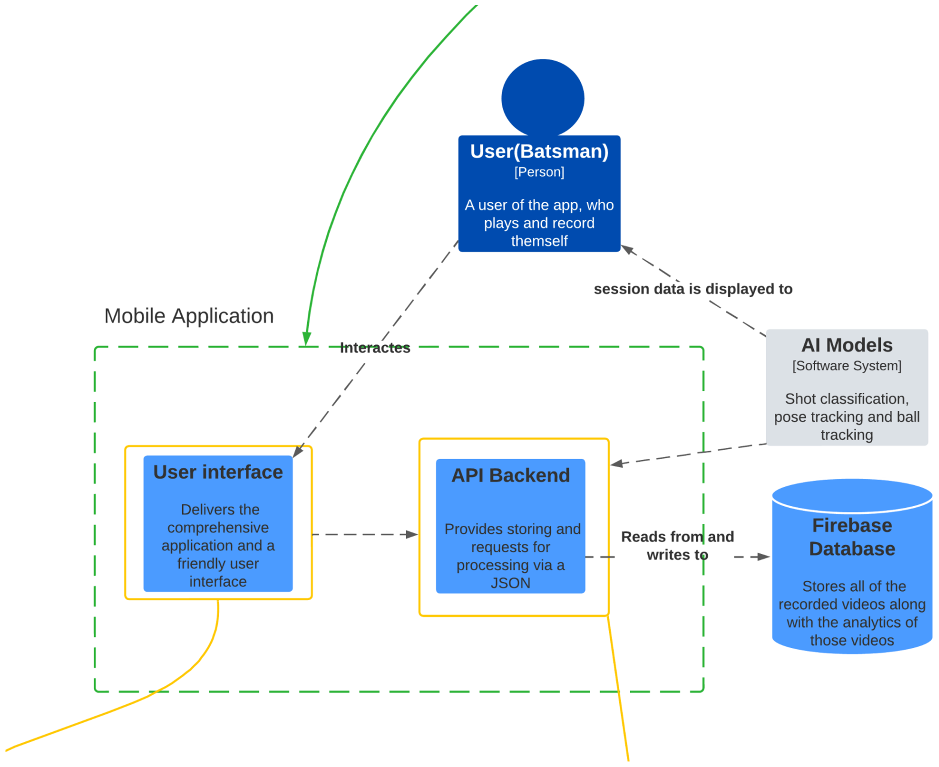

User(Batsman)
[Person]
A user of the app, who plays and record themself
session data is displayed to
Mobile Application
Interactes
AI Models
[Software System]
Shot classification, pose tracking and ball tracking
User interface
Delivers the comprehensive application and a friendly user interface
API Backend
Provides storing and requests for processing via a JSON
Reads from and writes to
Firebase Database
Stores all of the recorded videos along with the analytics of those videos

Figure 5 Mobile Application Container

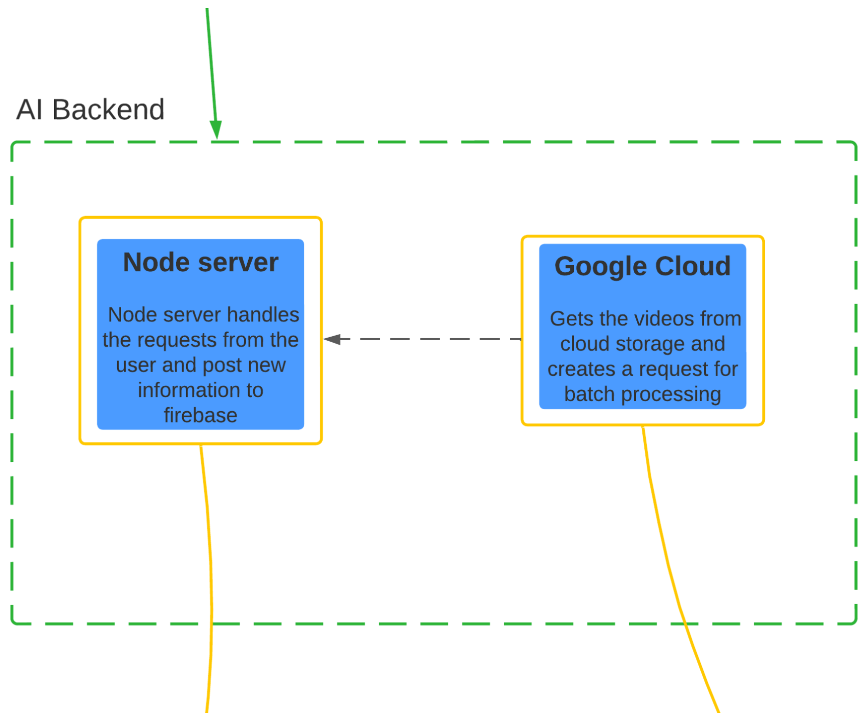

AI Backend
Node server
Node server handles the requests from the user and post new information to firebase
Google Cloud
Gets the videos from cloud storage and creates a request for batch processing

Figure 6 AI Backend Container

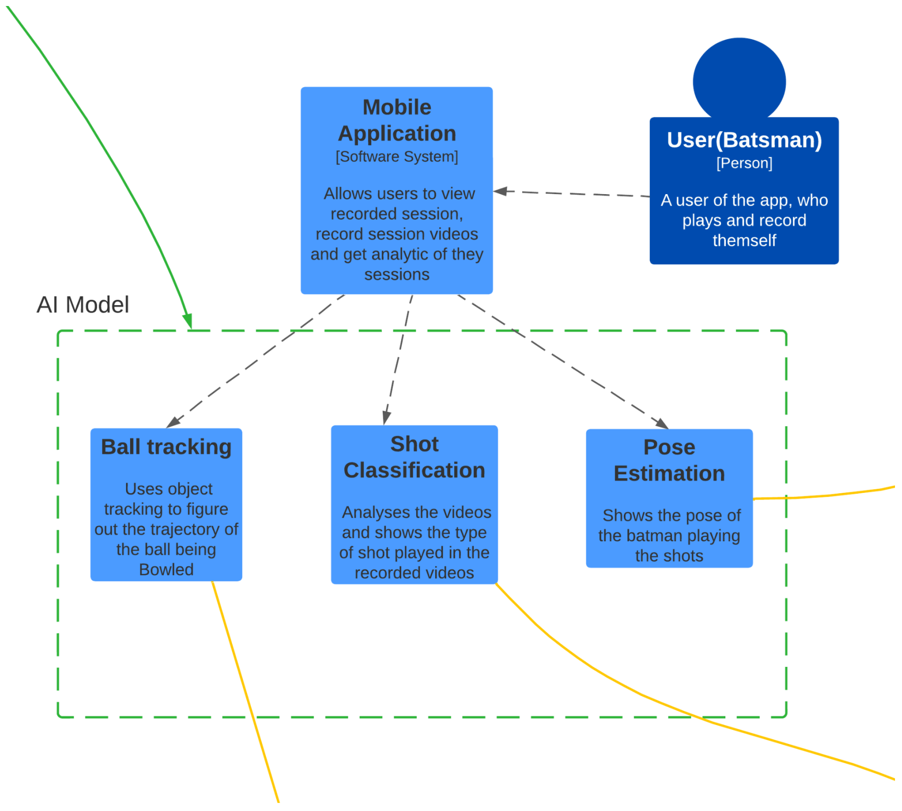

Mobile Application
[Software System]
Allows users to view recorded session, record session videos and get analytic of they sessions
User(Batsman)
[Person]
A user of the app, who plays and record themself
AI Model
Ball tracking
Uses object tracking to figure out the trajectory of the ball being Bowled
Shot Classification
Analyses the videos and shows the type of shot played in the recorded videos
Pose Estimation
Shows the pose of the batman playing the shots

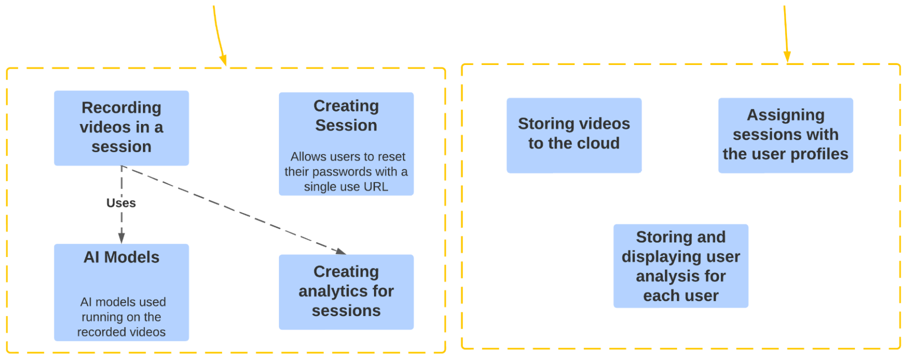
Figure 7 AI Model Container

Level 3: Components

Components delve into the intricacies of each container, shedding light on the fundamental functionalities that define their core operations. This detailed exploration aims to provide a comprehensive understanding of the specific roles and interactions within the Mobile Application, AI Backend, and AI Models containers.

Components of Mobile Application

Figure 7 shows how videos recorded are processed through the AI models to then show the results in each session. Mobile application also houses the process of storing user stats in the database

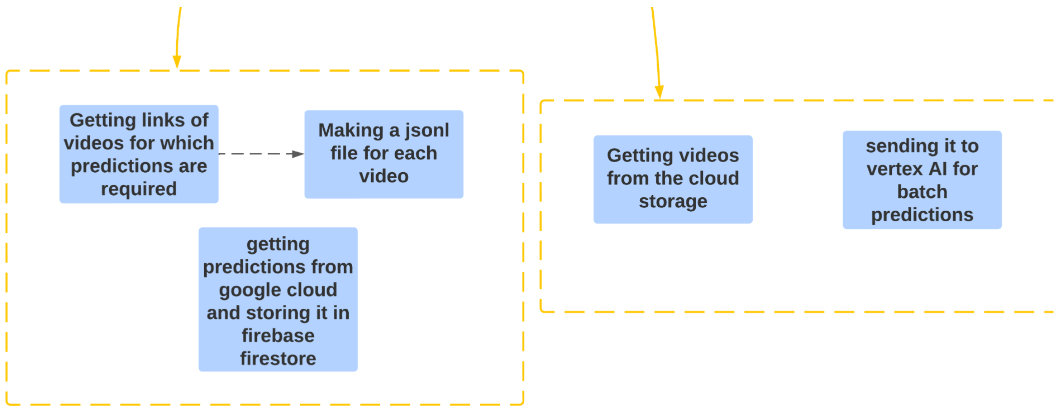


Recording videos in a session
Creating Session
Allows users to reset their passwords with a single use URL
Uses
AI Models
AI models used running on the recorded videos
Creating analytics for sessions
Storing videos to the cloud
Assigning sessions with the user profiles
Storing and displaying user analysis for each user

Figure 8 Components of Mobile Application Container

Components of AI Backend

The figure below shows the process of getting URIs and creating JSON Lines file for the requested videos and eventually getting the batch predictions from vertex AI and sending the predictions back to the firebase database



Getting links of videos for which predictions are required
Making a jsonl file for each video
getting predictions from google cloud and storing it in firebase firestore
Getting videos from the cloud storage
sending it to vertex AI for batch predictions

Figure 9 Components of AI Backend Container

Components of AI Models

Show all the AI models in shotsense and use the videos uploaded creating pose tracking and player segmentation. Another AI model showing the shot predictions and finally a model that uses ball tracking to create ball plots for a session

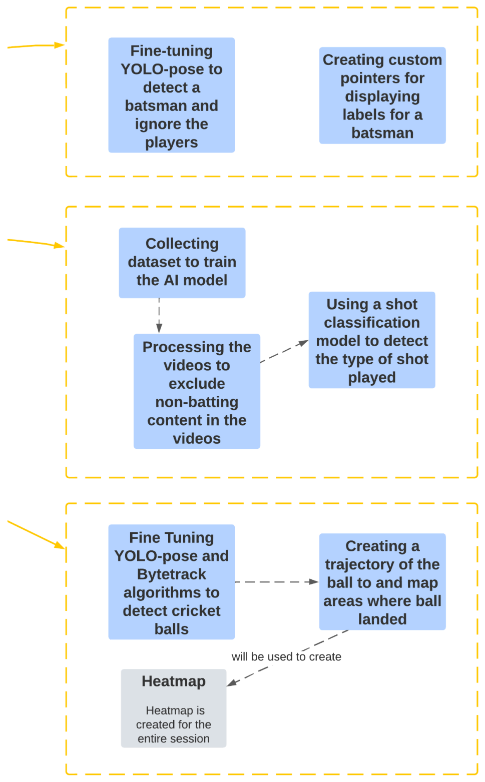
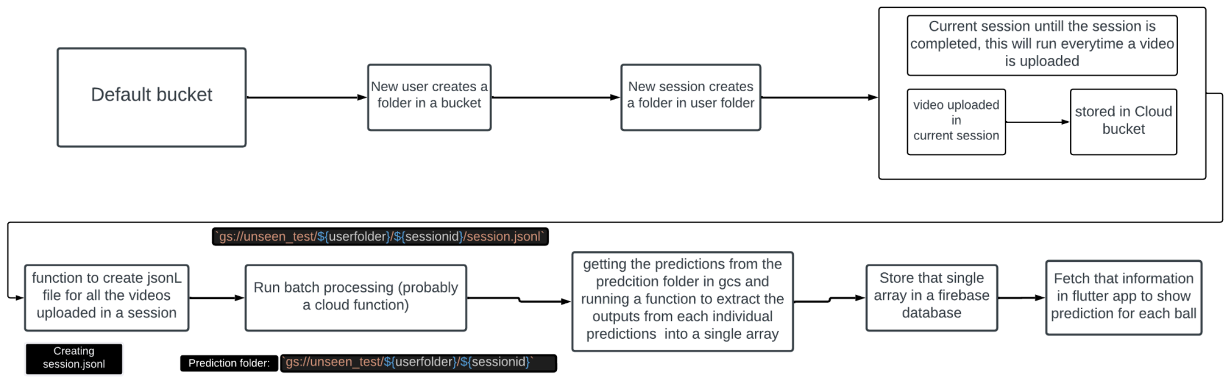

Fine-tuning YOLO-pose to detect a batsman and ignore the players
Creating custom pointers for displaying labels for a batsman
Collecting dataset to train the AI model
Processing the videos to exclude non-batting content in the videos
Using a shot classification model to detect the type of shot played
Fine Tuning YOLO-pose and Bytetrack algorithms to detect cricket balls
Creating a trajectory of the ball to and map areas where ball landed
will be used to create
Heatmap
Heatmap is created for the entire session

Figure 10 Components of AI Models Container

Backend/API Design

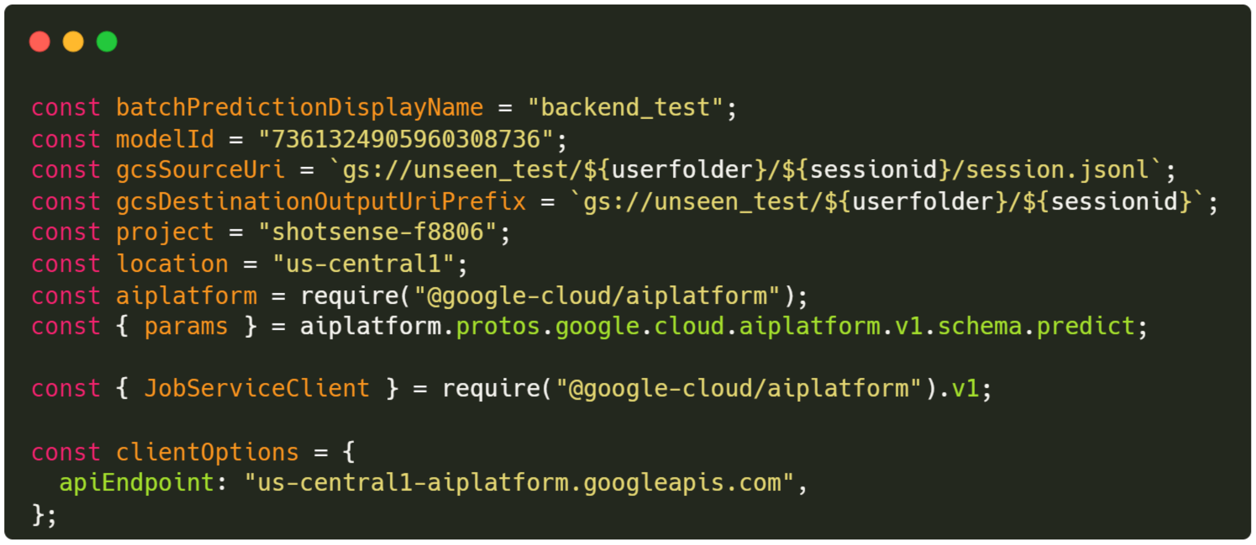
Batch Predictions to Flutter, Backend Design[9]

The backend design for the shot classification model involves a sophisticated sequence of interactions across various project components, seamlessly integrating storage, databases, AI models, and the mobile application interface. At the heart of this architecture, a main bucket serves as the project's core repository, while individual folders in that bucket are dynamically generated for each user and their sessions within Google Cloud Storage during the signup process. Upon video uploads, a dedicated cloud function is triggered, producing JSON Lines files for each video, subsequently transmitted to Google's Vertex AI for processing. The AI model generates predictions, organized into a predictions folder in JSON Lines format. Another cloud function is then invoked to persist these predictions into a Firebase Firestore collection specific to each user. Finally, this data is presented in the Flutter app, delivering a seamless and informative experience to users keen on exploring and understanding their cricket shot predictions.



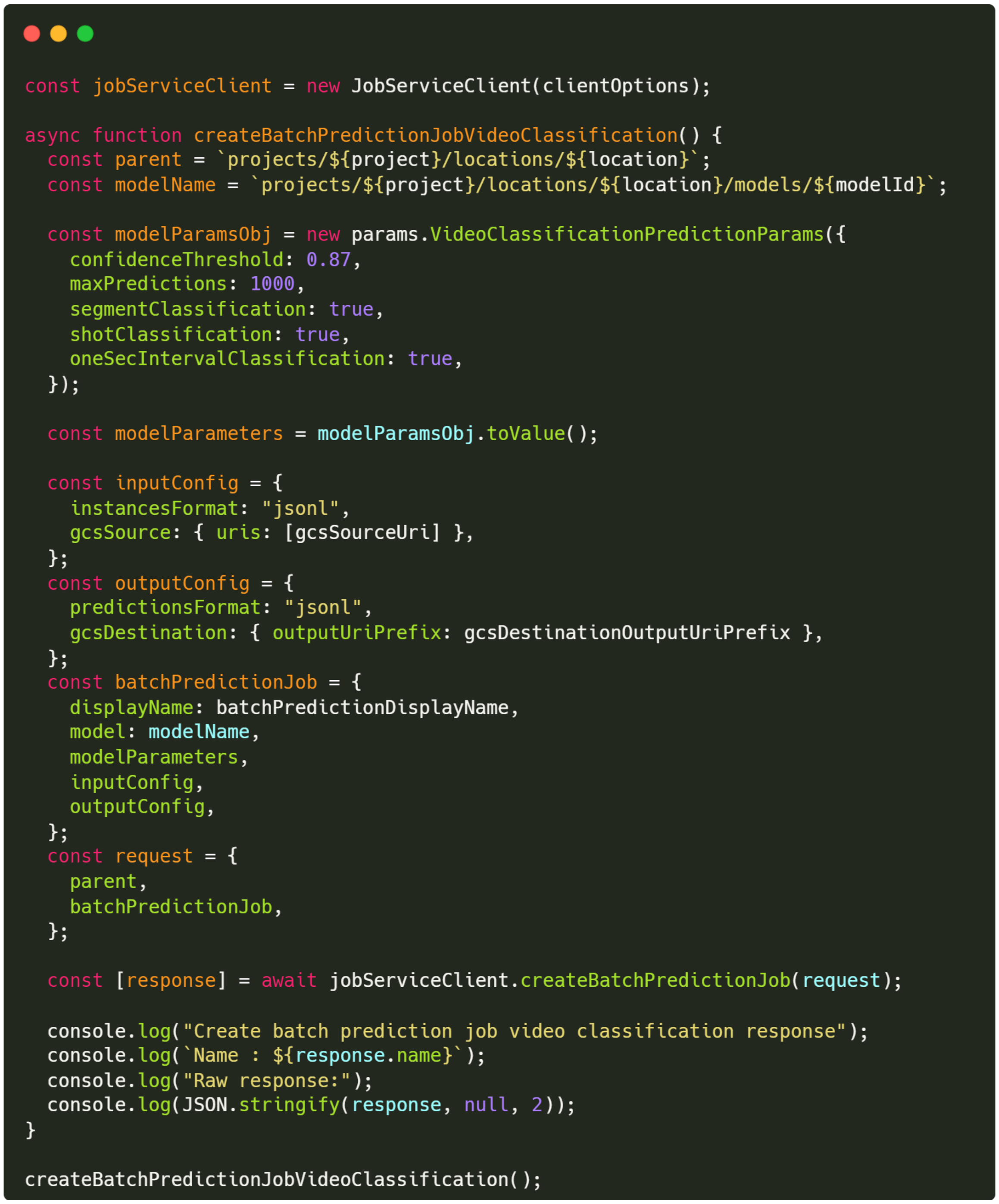
Default bucket
New user creates a folder in a bucket
New session creates a folder in user folder
Current session untill the session is completed, this will run everytime a video is uploaded
video uploaded in current session
stored in Cloud bucket
`gs://unseen_test/${userfolder}/${sessionid}/session.jsonl`
function to create jsonL file for all the videos uploaded in a session
Run batch processing (probably a cloud function)
getting the predictions from the predcition folder in gcs and running a function to extract the outputs from each individual predictions into a single array
Store that single array in a firebase database
Fetch that information in flutter app to show prediction for each ball
Creating session.jsonl
Prediction folder:
`gs://unseen_test/${userfolder}/${sessionid}`

Figure 11 Complete backend flow diagram from a video being uploaded to getting predictions



```
const batchPredictionDisplayName = "backend_test";
const modelId = "7361324905960308736";
const gcsSourceUri = `gs://unseen_test/${userfolder}/${sessionid}/session.jsonl`;
const gcsDestinationOutputUriPrefix = `gs://unseen_test/${userfolder}/${sessionid}`;
const project = "shotsense-f8806";
const location = "us-central1";
const aiplatform = require("@google-cloud/aiplatform");
const { params } = aiplatform.protos.google.cloud.aiplatform.v1.schema.predict;

const { JobServiceClient } = require("@google-cloud/aiplatform").v1;

const clientOptions = {
  apiEndpoint: "us-central1-aiplatform.googleapis.com",
};
```

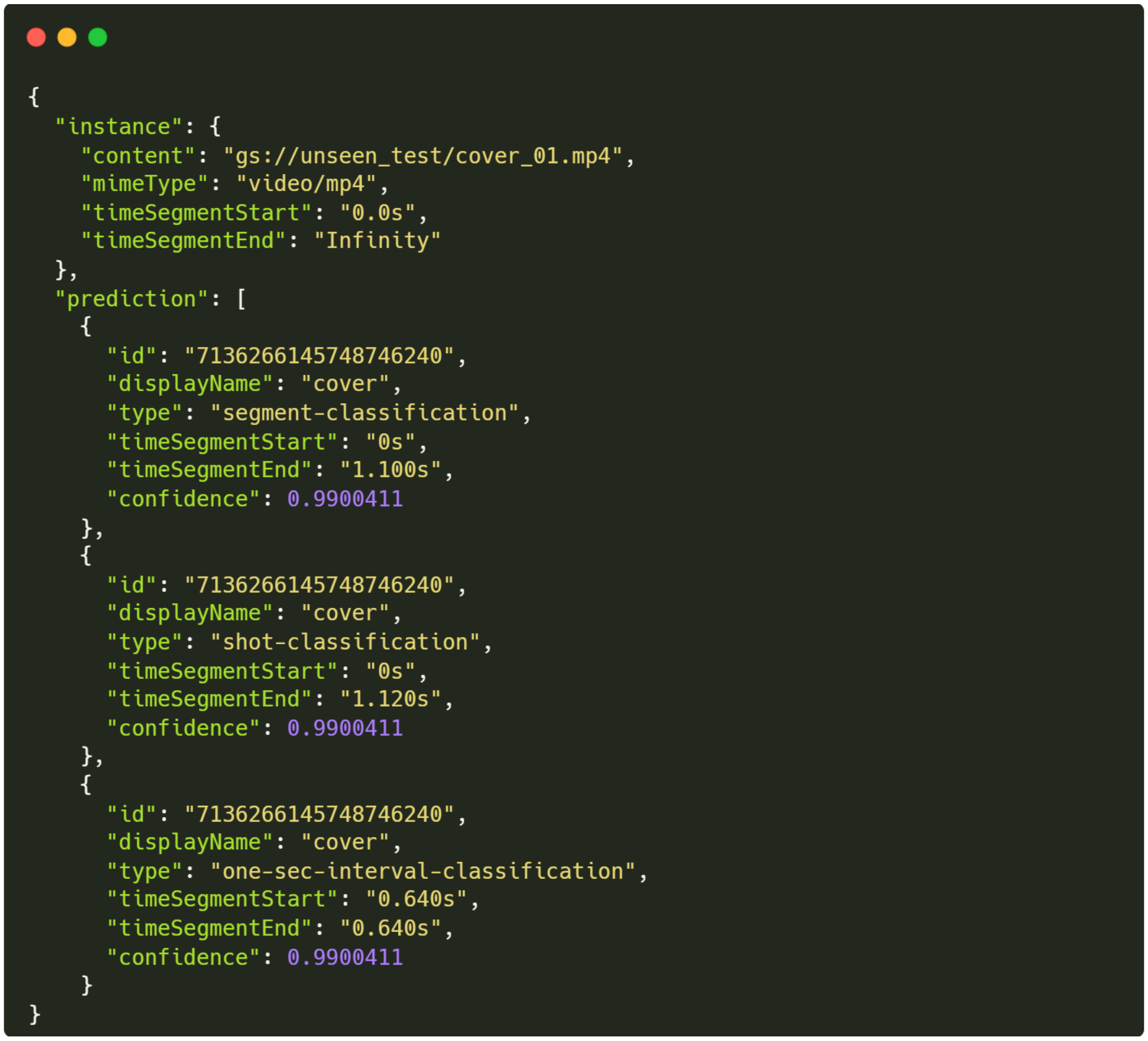
Figure 12 Defining parameters for batch prediction



```
const jobServiceClient = new JobServiceClient(clientOptions);

async function createBatchPredictionJobVideoClassification() {
  const parent = `projects/${project}/locations/${location}`;
  const modelName = `projects/${project}/locations/${location}/models/${modelId}`;

  const modelParamsObj = new params.VideoClassificationPredictionParams({
    confidenceThreshold: 0.87,
    maxPredictions: 1000,
    segmentClassification: true,
    shotClassification: true,
    oneSecIntervalClassification: true,
  });

  const modelParameters = modelParamsObj.toValue();

  const inputConfig = {
    instancesFormat: "jsonl",
    gcsSource: { uris: [gcsSourceUri] },
  };
  const outputConfig = {
    predictionsFormat: "jsonl",
    gcsDestination: { outputUriPrefix: gcsDestinationOutputUriPrefix },
  };
  const batchPredictionJob = {
    displayName: batchPredictionDisplayName,
    model: modelName,
    modelParameters,
    inputConfig,
    outputConfig,
  };
  const request = {
    parent,
    batchPredictionJob,
  };

  const [response] = await jobServiceClient.createBatchPredictionJob(request);

  console.log("Create batch prediction job video classification response");
  console.log(`Name : ${response.name}`);
  console.log("Raw response:");
  console.log(JSON.stringify(response, null, 2));
}

createBatchPredictionJobVideoClassification();
```

Figure 13 Batch Prediction Node.js Client

Utilizing the Node.js client library, the script configures and submits the job, specifying a pre-trained model identified by modelId. Parameters for the video classification model, including confidence threshold and maximum predictions, are set in modelParamsObj. Input data, formatted as a JSON Lines file (JSONL) in a Google Cloud Storage bucket, undergoes classification, and the results are stored in a designated location.



```
{
  "instance": {
    "content": "gs://unseen_test/cover_01.mp4",
    "mimeType": "video/mp4",
    "timeSegmentStart": "0.0s",
    "timeSegmentEnd": "Infinity"
  },
  "prediction": [
    {
      "id": "7136266145748746240",
      "displayName": "cover",
      "type": "segment-classification",
      "timeSegmentStart": "0s",
      "timeSegmentEnd": "1.100s",
      "confidence": 0.9900411
    },
    {
      "id": "7136266145748746240",
      "displayName": "cover",
      "type": "shot-classification",
      "timeSegmentStart": "0s",
      "timeSegmentEnd": "1.120s",
      "confidence": 0.9900411
    },
    {
      "id": "7136266145748746240",
      "displayName": "cover",
      "type": "one-sec-interval-classification",
      "timeSegmentStart": "0.640s",
      "timeSegmentEnd": "0.640s",
      "confidence": 0.9900411
    }
}
```

Figure 14 Prediction in form of JSON output for single video

The JSON output from the createBatchPredictionJobVideoClassification function provides predictions for a video instance named "cover_01.mp4." The predictions include multiple classifications, such as segment classification, shot classification, and one-second interval classification, all indicating the presence of a "cover" shot type. The predictions include specific time segments within the video, their corresponding confidence scores, and the unique identifier for each classification. The high confidence scores, particularly around 99%, signify the model's strong confidence in identifying the "cover" shot type during various intervals throughout the video. These detailed predictions offer valuable insights into the model's performance and contribute to the comprehensive shot analysis within the ShotSense application.

ML Design for Shot Classification

In the development of our application, the initial focus was on implementing the shot classification model, a pivotal component of our machine learning architecture. This model is designed to analyze videos, determining and classifying the specific type of shot depicted. The emphasis during the implementation phase has been on ensuring optimal performance through careful attention to the preparation of both training and testing datasets.

Model Development

In the model development phase, our primary goal is to create a reliable and effective system for categorizing cricket shots based on user-uploaded videos. This fundamental aspect of the application allows users to gain valuable insights into their gameplay by understanding the types of shots they play. The following sections provide an in-depth exploration of the specific machine learning techniques employed, including the utilization of pre-trained models such as VGG16 and Google's AutoML. The focus remains on delivering a practical and accurate shot classification model that enhances the overall user experience.

VGG-16

Leveraging pre-trained models like VGG16[10] is a strategic choice in our pursuit of shot classification accuracy. VGG16, having excelled in image recognition on ImageNet, provides a robust foundation for understanding intricate features in diverse datasets. By incorporating this proven architecture into our model, we tap into its ability to recognize patterns, streamlining the learning process for shot-specific nuances.

To tailor VGG16 to cricket-shot dynamics, we introduced a personalized touch. Our model enhancement involves integrating a custom fully connected layer and opting for a Gated Recurrent Unit (GRU) layer over the conventional LSTM. This modification caters to the unique demands of cricket shots, ensuring that the shot classification model not only benefits from VGG16's pre-learned features but also aligns optimally with the complexities of cricket gameplay, delivering precise insights efficiently.

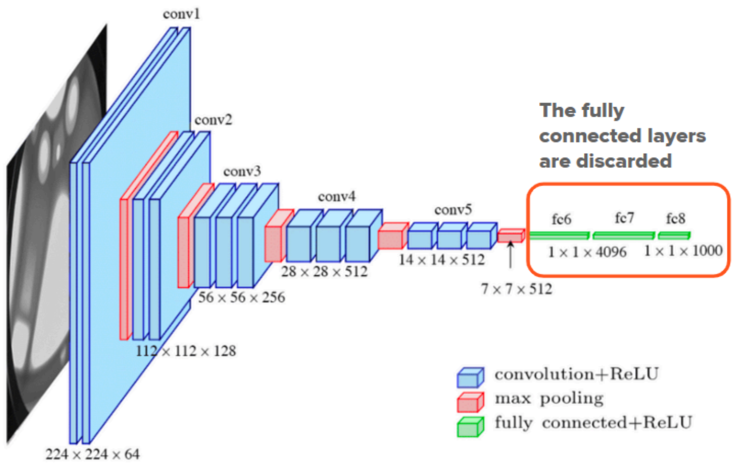
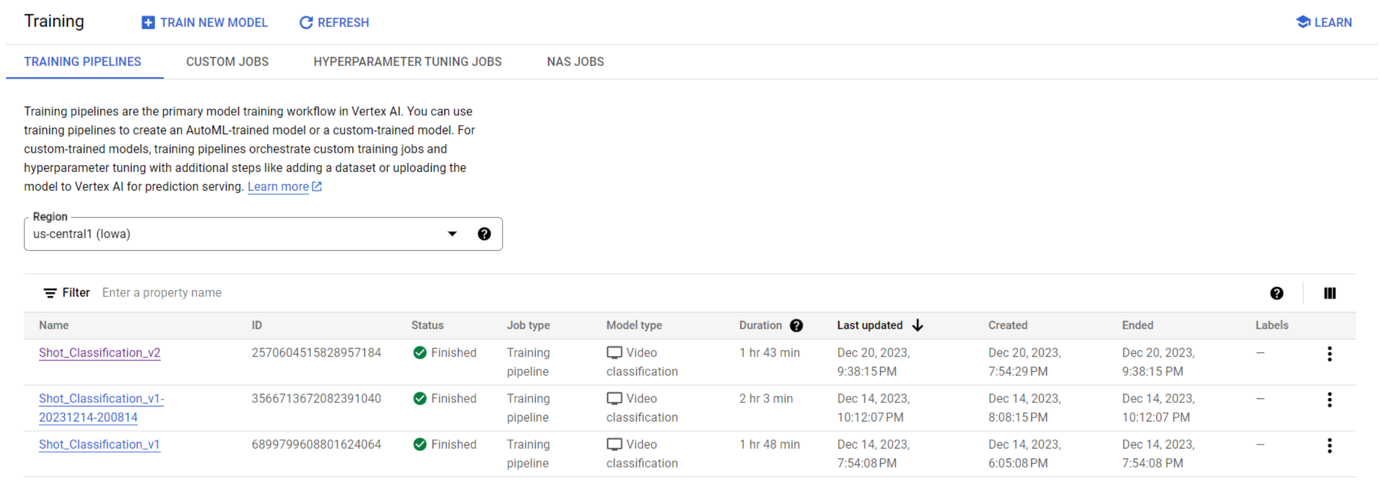


Figure 15 VGG-16 with its hidden layers. FC was discarded for our customized layers

Google AutoML

Google AutoML[11] serves as an invaluable tool, offering an accessible and efficient platform for training machine learning models without the need for extensive expertise. By leveraging AutoML's user-friendly interface, we seamlessly integrated our dataset, allowing for straightforward customization of model parameters and training processes. This not only accelerates the development lifecycle but also democratizes machine learning capabilities, enabling us to harness the power of sophisticated models with ease. With Google AutoML, our shot classification model benefits from advanced training techniques and optimizations, fostering a user-centric approach to model development and achieving reliable outcomes in the dynamic context of cricket shot analysis.



Training pipelines are the primary model training workflow in Vertex AI. You can use training pipelines to create an AutoML-trained model or a custom-trained model. For custom-trained models, training pipelines orchestrate custom training jobs and hyperparameter tuning with additional steps like adding a dataset or uploading the model to Vertex AI for prediction serving. Learn more

Region: us-central1 (Iowa)

Filter: Enter a property name

Name	ID	Status	Job type	Model type	Duration	Last updated	Created	Ended	Labels
Shot_Classification_v2	2570604515828957184	✓ Finished	Training pipeline	Video classification	1 hr 43 min	Dec 20, 2023, 9:38:15 PM	Dec 20, 2023, 7:54:29 PM	Dec 20, 2023, 9:38:15 PM	–
Shot_Classification_v1-20231214-200814	3566713672082391040	✓ Finished	Training pipeline	Video classification	2 hr 3 min	Dec 14, 2023, 10:12:07 PM	Dec 14, 2023, 8:08:15 PM	Dec 14, 2023, 10:12:07 PM	–
Shot_Classification_v1	6899799608801624064	✓ Finished	Training pipeline	Video classification	1 hr 48 min	Dec 14, 2023, 7:54:08 PM	Dec 14, 2023, 6:05:08 PM	Dec 14, 2023, 7:54:08 PM	–

Figure 16 Training pipeline for AutoML model

Conclusion

In summary, this design document has provided a comprehensive overview of the ShotSense mobile application's system architecture, emphasizing its use of advanced technologies like machine learning and computer vision to enhance cricket players' performance. The architectural components, including shot classification, pose estimation, and ball tracking, have been intricately woven together to ensure scalability, security, and usability. Technologies such as Flutter, Dart, Firebase, Node Js, and machine learning frameworks like TensorFlow, Keras, YOLOv8, Google Cloud Vertex AI, and Google AutoML form the backbone of the application, facilitating a seamless user experience from shot recording to personalized skill improvement insights.

The integration of high-level application and mobile application flow diagrams, along with the C4 architecture diagram, elucidates the cohesive functioning of the mobile application, AI backend, and AI models. The backend/API design underscores a sophisticated flow of data, seamlessly connecting storage, databases, AI models, and the mobile application interface. In the ML design for shot classification, the strategic use of pre-trained models like VGG16 and Google AutoML demonstrates a commitment to accuracy, with VGG16 customized for cricket-specific nuances. Overall, this design document serves as a clear blueprint, outlining the architectural decisions and progress made in shaping ShotSense into an advanced, user-centric platform for cricket enthusiasts.

References

1. Flutter documentation. (n.d.). Flutter. https://docs.flutter.dev/

2. Ultralytics. (n.d.). Pose. Ultralytics YOLOv8 Docs. https://docs.ultralytics.com/tasks/pose/

3. Index | Node.js v21.6.0 Documentation. (n.d.). https://nodejs.org/docs/latest/api/

4. Video classification. (n.d.). TensorFlow. https://www.tensorflow.org/lite/examples/video_classification/overview

5. FFMpeg Documentation. (n.d.). https://ffmpeg.org/documentation.html

6. OpenCV. (n.d.). OpenCV. https://forum.opencv.org/

7. Google Cloud documentation | Documentation. (n.d.). Google Cloud. https://cloud.google.com/docs

8. The C4 model for visualizing software architecture. (n.d.). https://c4model.com/

9. Get predictions from a video classification model. (n.d.). Google Cloud. https://cloud.google.com/vertex-ai/docs/video-data/classification/get-predictions#aiplatform_create_batch_prediction_job_video_classification_sample-nodejs

10. Rohini G, 'Everything you need to know about VGG16', https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

11. AutoML Google Cloud. https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide

12. Documentation. (n.d.). Firebase. https://firebase.google.com/docs