Modules:

- **Cameraview**
- **Data**
- **Domain**
- **Network**
- **Ui.theme**

Cameraview:

Contains Kotlin files consisting of composable functions related to Android CameraX camera preview and image analysis, overlaying of YOLOv8 bounding box onto the camera preview, and radio buttons

Classes/Files:

- BoundingBoxOverlay.kt
- CameraPreview.kt
- Radiobuttons.kt

BoundingBoxOverlay.kt:

Function: BoundingBoxOverlay composable function

Parameters:
detectionResult: DetectionResult (container class for YOLOv8 output)
Return Type: N/A
Description: Receives a DetectionResult object as an argument and uses the (x,y) coordinate along with the width and height to draw a bounding box over the Android View camera preview in real time. Canvas size matches Camera Preview size exactly.

CameraPreview.kt:

Function: AppFunctions

Parameters:
modifier: Modifier //Modifiers for Compose UI elements.
Return Type: N/A
Description: The AppFunctions composable function organizes the overall user interface related to camera functionalities. It includes radio buttons for selecting operation modes (individual or batch processing), handles the state for detection and OCR results, and manages the activation and deactivation of image analysis and result display through composables like CameraPreview and DisplayResult.

Function: CameraPreview

Parameters:
modifier: Modifier //Modifiers for Compose UI layout and styling.

imageAnalysisExecutor: ExecutorService //Executor service for //running image //analysis //tasks.

detectionResultState: MutableState<DetectionResult?> //Mutable state to store //and update detection //results from image //analysis.

ocrResultState: MutableState<String> //Mutable state to store //OCR results extracted //from detected objects.

isAnalysisActive: MutableState<Boolean> //Controls whether image //analysis should //continue or be paused.

Return Type: N/A
Description:
This composable function configures and displays the camera feed using Android's CameraX library. It sets up a PreviewView for the camera feed and binds lifecycle-aware camera use cases for real-time preview and image analysis.

Image Analysis, inside the Lambda expression, an ImageProxy object is converted to a bitmap and passed to the appropriate CustomObjectDetector class functions for preprocessing and inference. If License Plate is detected, bitmap is cropped to object detection boundaries, and passed to a TextExtraction object where ML Kit Text Recognition inference is done. The results of the analysis are updated in real-time and displayed through a BoundingBoxOverlay.

Radiobuttons.kt

Function: RadioButtons

Parameters:

submitBatch: MutableState<Boolean> //Mutable state to control the //submission of a batch of OCR //results.

clearBatch: MutableState<Boolean> //Mutable state to clear the //accumulated batch data.

plateCount: MutableState<Int> //Mutable state that keeps track of //the number of plates scanned in //batch mode.

radioOptions: List<String> //List of options for radio buttons for //batch scan or individual search

selectedOption: String //Currently selected radio button //option.

onOptionSelected: (String) -> Unit //Callback function that updates the
//selected option based on user
//interaction.

Return Type: N/A

Description:
The RadioButtons composable function provides a user interface component for selecting between different modes of operation (like "Individual Search" and "Batch Scan") and managing batch operations for OCR tasks. It displays radio buttons for each option in radioOptions and updates the interface based on the current selection. It also shows the number of plates scanned when in batch mode. Additional buttons are provided to submit or clear the batch data, invoking the respective states to trigger these actions.

Data:

Contains Kotlin and Java classes that handle the inference process for license plate object detection, storing the results, cropping the image, obtaining ML kit Text Recognition results, and finally displaying final results.

Classes/Files:

- BitmapFunctions.kt
- CustomObjectDetector.kt
- DisplayBatchResult.kt
- DisplayResult.kt
- EnhancedPostOCR.java
- TextExtraction.kt

BitmapFunctions.kt:

Class: BitmapFunctions

Description:
Class containing functions to create a bitmap from an ImageProxy object and to gray out everything outside the object detection bounding box of a bitmap.

Function: imageProxyToBitmap
Parameters:
imageProxy: ImageProxy // image to be processed,
Return Type: Bitmap
Description:
Converts an ImageProxy object to a Bitmap. The function retrieves YUV data from the ImageProxy, converts it into NV21 format, and then uses YuvImage to create a JPEG compressed image which is finally converted to Bitmap format.

Function: grayOutBitmapOutsideBoundingBox (Companion Object)
Parameters:
source: Bitmap //Source bitmap image

detectionResult: DetectionResult //Detection results containing //normalized coordinates (x, y) and //dimensions (width, height) of the //detected object.

rotation: Int //The degree of rotation needed to //correct the orientation of the //image.

Return Type: Bitmap object

Description:
Modifies the source bitmap by graying out all areas except for the detected object. The function first adjusts the orientation of the bitmap based on the provided rotation angle. It then calculates the bounding box using the detection results and uses a Canvas to paint the regions outside the bounding box in gray. This function is essentially used to crop the license plate after object detection.

Private Function: rotateBitmap (Companion Object)

Parameters:
source: Bitmap //The source bitmap to rotate.
rotationDegrees: Int //Rotation angle in degrees.
Return Type: Bitmap
Description:
Rotates the source bitmap by the specified angle. This is used to adjust the orientation of an image to the correct viewing angle before further processing.

CustomObjectDetector.kt:

Class: CustomObjectDetector

Description:
The CustomObjectDetector class integrates TensorFlow Lite to deploy a custom-trained YOLOv8 object detection model within the application. It handles the loading of the model, preprocessing of images, and running inference to detect objects with their bounding boxes.

Function: loadModelFile

Parameters: None
Return Type: N/A
Description:
Loads the TensorFlow Lite model from the application's assets. The model file is mapped into memory for fast access by the TensorFlow Lite interpreter.

Function: preprocessImage

Parameters:
bitmap: Bitmap //Bitmap image to preprocess
Return Type: ByteBuffer
Description:

Prepares the input image for inference by the TensorFlow Lite model. This includes rotating the bitmap to the correct orientation, resizing it to the dimensions expected by the model, and normalizing pixel values. The processed image data is returned as a ByteBuffer, ready for inference.

Function: runInference

Parameters:

inputData: ByteBuffer // Preprocessed image data

Return Type: DetectionResult?

Description:

Conducts object detection on the preprocessed image. It feeds the input data to the TensorFlow Lite interpreter, which returns the detection results. The function evaluates each detection, checking it against a confidence threshold to determine if it should be considered. The most confident detection result is returned, encapsulating the coordinates and dimensions of the detected object, along with its confidence score.

Private Function: rotateBitmap

Parameters:

source: Bitmap //The source bitmap to rotate

rotationDegrees: Int // Rotation angle in degrees

Return Type: Bitmap

Description:

Rotates the source bitmap by the specified angle. This function is used to adjust the orientation of an image to the correct viewing angle before preprocessing it for model input.

DisplayBatchResult.kt:

Composable Function: DisplayBatchResult

Parameters:

plates: MutableList<String> // A list of license plate numbers to be queried

states: MutableList<String> //Corresponding list of states for each plate
//number

platesApi: PlatesAPI // API interface for querying plate information

apiKey: String //API key for authentication with the plates API

onClose: () -> Unit //Callback function to be invoked when the user
//wishes to close the results display.

Return Type: N/A

Description:

This composable function manages the display and querying of a batch of license plates using a provided API. It performs API calls for each plate and state combination, handling and displaying the results or errors within a Compose UI. If a plate is not registered, it collects such plates and their corresponding states

to display them separately. The function also includes a button to close the results view, which triggers the provided onClose callback.

DisplayBatch.kt:

Composable Function: DisplayResult

Parameters:

ocrResultState: MutableState<String> //A mutable state holding the OCR
//result as a combination of license
//plate and state separated by an
//underscore.

platesApi: PlatesAPI //API interface for querying license
//plate information.

apiKey: String //API key for authentication with the
//plates API.

onClose: () -> Unit //Callback function invoked when the
//user closes the results display.

Return Type: N/A

Description:

This composable function provides the user interface to enter and search for a license plate through an API using Jetpack Compose. It allows modifying or correcting the detected OCR license plate and state data, making a call to the provided API to fetch the relevant information. The results are presented in the UI after querying, and the user can manually select the state from a dropdown if necessary.

EnhancedPostOCR.java:

Class: EnhancedPostOCR

Description:

The EnhancedPostOCR class is designed to enhance and correct Optical Character Recognition (OCR) results specifically for the extraction of license plate numbers and associated state name. It uses heuristic methods alongside the Levenshtein distance algorithm to refine OCR outputs, primarily to correct misrecognitions and filter out irrelevant text.

Function: getOutPutText

Parameters:

result: Text //The OCR result object containing recognized text blocks.

Return Type: String

Description:

Extracts the license plate number by identifying the text line within the largest bounding box, presumed to be the license plate. It also searches for and validates state names within the OCR results, applying a series of corrections and adjustments to enhance

accuracy. The function returns a string concatenating the refined license plate and the identified state abbreviation.

Function: checkStateAccuracy
Parameters:
element: String //A text element extracted from the OCR results.
Return Type: LevenshteinResult //container class
Description:
Evaluates the accuracy of state name recognition by comparing the text element against a predefined list of state names using the Levenshtein distance. It filters out irrelevant text based on predefined criteria to focus on potential state names.

Function: excludeFromStateName
Parameters:
element: String //A text element extracted from the OCR results.
Return Type: Boolean
Description:
Determines whether a text element should be excluded from the state name correction process, based on a predefined list of common non-state text found on license plates.

Function: getStateCode
Parameters:
state: String //A full state name.
Return Type: String
Description:
Converts full state names into their corresponding two-letter abbreviations. If no matching state is found, it returns an empty string.

TextExtraction.kt:

Class: TextExtraction

Description:
The TextExtraction class utilizes Google's ML Kit to perform OCR on bitmap images. It processes images to extract textual content, specifically targeting license plate numbers and state names. The class is designed to handle preprocessing of images (converting to grayscale) to improve OCR accuracy and implements post-processing to extract and refine the relevant text.

Function: processImage
Parameters:
bitmap: Bitmap // The image on which OCR needs to be performed.
onResult: (String) -> Unit //Callback function to handle the OCR result.
onError: (Exception) -> Unit // Callback function to handle errors during OCR
//processing.
Return Type: N/A
Description:

Converts the input bitmap to grayscale and processes it using the configured TextRecognizer. This function handles success and failure during text recognition, invoking the respective callback functions with the results or error information.

Function: toGrayscale
Parameters:
bitmap: Bitmap //The original color bitmap image.
Return Type: Bitmap
Description:
Converts the input bitmap image to a grayscale bitmap. This conversion is used to enhance the OCR process, as grayscale images tend to give better results in text recognition due to reduced noise and color variability.

Function: getOutPutText
Parameters:
result: Text //Text recognition result object from ML Kit.
Return Type: String
Description:
Analyzes the recognized text to extract the most relevant information—typically the license plate number and the state name. This function identifies the text line with the largest bounding box as the likely plate number and uses a combination of heuristic checks and Levenshtein distance calculations to confirm the state name.

Domain:

This module contains data classes relevant to the logic or core domain of the application.

Classes/Files:

- DetectionResult.kt
- LevenshteinResult.kt

DetectionResult.kt:

Class: DetectionResult

Description:

The DetectionResult data class encapsulates the output information provided by the YOLOv8 object detection model. It specifically targets the detection of license plates, storing the relevant data such as bounding box coordinates, dimensions, and confidence scores.

Attributes:

x: Float
Description:
The x-coordinate of the bounding box's center. This coordinate is relative to the image on which detection is performed.

y: Float
Description:

The y-coordinate of the bounding box's center, also relative to the image.

width: Float
Description:
Represents the width of the detected bounding box, providing an idea of the detected object's horizontal size.

height: Float
Description:
Represents the height of the detected bounding box, giving a measure of the object's vertical size.

confidence: Float
Description:
The confidence score associated with the detected object. It quantifies how sure the model is that the detected object is a license plate, ranging from 0.0 (no confidence) to 1.0 (maximum confidence).

LevenshteinResult.kt:

Class: LevenshteinResult

Description:
The LevenshteinResult data class stores the results of Levenshtein distance-based spell checking used to verify and correct state names obtained through OCR (Optical Character Recognition). It retains the original inference, the corrected state name, and the computed Levenshtein distance between them.

Attributes:

distance: Int
Description:
The computed Levenshtein distance between the original OCR inference and the identified state name. This distance measures the number of single-character edits required to transform the original inference into the corrected state name, indicating the correction accuracy.

state: String
Description:
The name of the state identified after Levenshtein distance-based correction. This field holds the final corrected state name resulting from the spell check process.

originalInference: String
Description:
The original OCR inference for the state name before any spell-check correction. This value represents the raw output as interpreted by the OCR system.

Network:

This module handles the communication with external services and APIs.

Classes/Files:

- DriverInfo.java
- PlatesAPI.java
- RetrofitClient.java

DriverInfo.java:

Class: DriverInfo

Description:
The DriverInfo class represents a data model for driver information. This class provides the structure for how driver details are serialized and deserialized via the network, with the help of the Gson library. It is designed to store essential data such as driver's identification number, license plate information, vehicle details, and contact information.

Attributes:

idNum: int
Description:
Represents the unique identification number for the driver. This field is annotated with SerializedName("IDnum") to map the JSON field correctly.

name: String
Description:
The driver's full name. This field is annotated with SerializedName("Name").

plateNum: String
Description:
The driver's license plate number. This field is annotated with SerializedName("PlateNum").

plateState: String
Description:
The state abbreviation where the license plate is registered. This field is annotated with SerializedName("PlateState").

make: String
Description:
The make of the driver's vehicle. This field is annotated with SerializedName("Make").

model: String
Description:

The model of the driver's vehicle. This field is annotated with SerializedName("Model").

personType: String
Description:
Indicates the driver's type or category (e.g., student, faculty). This field is annotated with SerializedName("PersonType").

email: String
Description:
The driver's email address, annotated with SerializedName("Email").

Methods:

getIdnum(): int
Description:
Returns the driver's unique identification number.

getPlateNum(): String
Description:
Returns the driver's license plate number.

toString(): String
Description:
Provides a formatted string representing the driver's information, including their ID, name, vehicle details, and contact information.

PlatesAPI.java:

Interface: PlatesAPI

Description:
The PlatesAPI interface defines the structure for interacting with a RESTful API to query information based on a vehicle's license plate. It utilizes Retrofit annotations to map requests and parameters, allowing for seamless integration with external services.

Methods:

queryLicensePlate(@Query("state") String state, @Query("plate") String licensePlate, @Header("api-key") String api_key): Call<DriverInfo>
Description:
This method queries the external API for driver information using the provided license plate number and state abbreviation. It returns a Call object that can be executed to obtain a DriverInfo object with the relevant details.

Parameters:

state: String

Represents the state abbreviation (e.g., "PA" for Pennsylvania) where the license plate is registered.

licensePlate: String
The actual license plate number to be queried.

api_key: String
The API key required for authorization with the external service.

RetrofitClient.java:

Class: RetrofitClient

Description:
The RetrofitClient class is a singleton that provides a configured instance of Retrofit for making HTTP requests. It uses an interceptor to log network request and response data, and the Gson converter for JSON serialization.

Method:

getClient(String baseUrl): Retrofit
Description:
This method returns a single Retrofit instance configured with a base URL and logging interceptor for debugging purposes. If the instance doesn't already exist, it initializes it with the provided base URL, attaches the OkHttp client, and adds the Gson converter factory.
Parameters:
baseUrl: String
The base URL of the API to be used in requests.
Return Type: Retrofit Object

Ui.theme:

This module hanldes app permissions and the user login as well as setting up the homescreen of the app.

Classes/Files:

- MainActivity.kt

MainActivity.java:

Class: MainActivity

Description:
The MainActivity class serves as the primary activity for the Parking Permit application, handling the user interface and permissions necessary for app functionality. It manages the sign-in process and transitions to the main operational screens upon successful authentication.

Function: onCreate

Parameters: savedInstanceState: Bundle
Return Type: void
Description:
Sets up the user interface and initializes the sign-in process within a Material-themed layout. This method also prepares the application for user input and interaction.

Function: requestCameraPermission
Parameters: None
Return Type: void
Description:
Requests camera permission at runtime to ensure the application has the necessary access to perform its operations. This is crucial for functionalities that involve accessing the device's camera.

Composable Function: SignInScreen
Parameters: onSignInClicked: (username: String, password: String) -> Unit
Return Type: N/A
Description:
Renders the sign-in user interface allowing users to enter their credentials. It validates these credentials against predefined values and provides feedback on authentication success or failure.

Composable Function: HomeScreen
Parameters: None
Return Type: N/A
Description:
Displays the main screen of the application after a successful sign-in. This screen includes key functionalities and access to various parts of the app.