IntelliCure

CEC 2025: Team Citadel

Requirements

Model

Classification

- Using a trained classification model, detect if there is a present tumor in uploaded MRI scan images.

Analytics

- In-depth analytics and accurate predictions for a batch of images

Data

Visualization

- Visualize results showing the classification

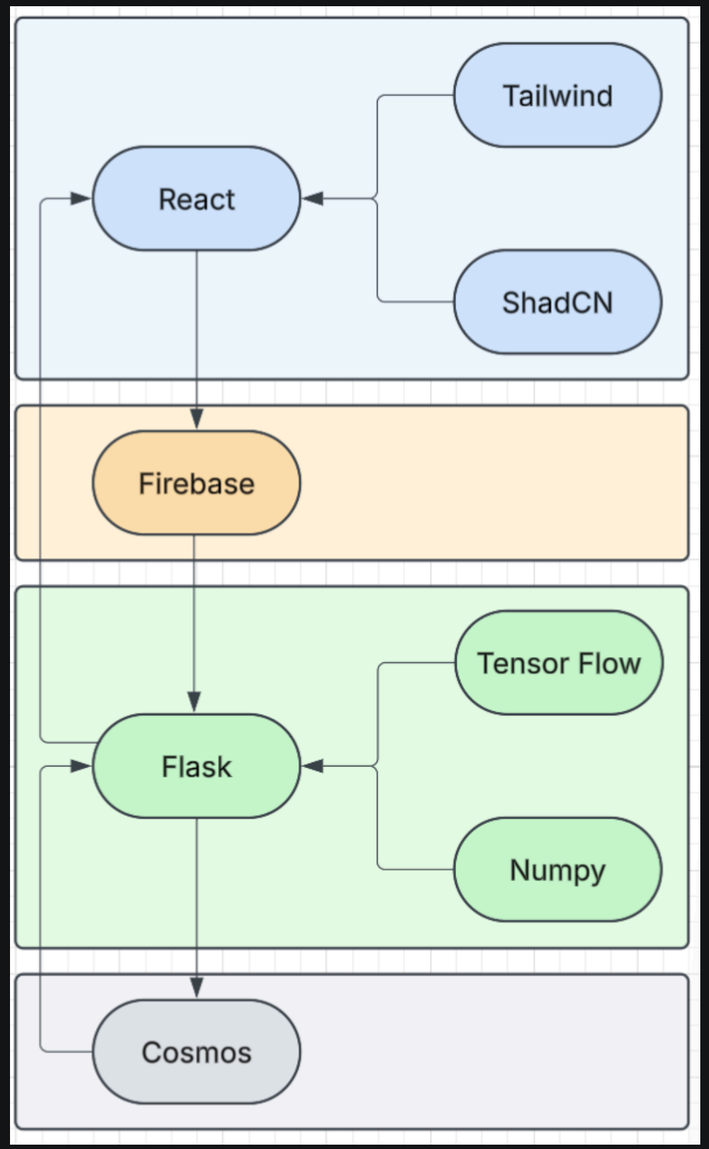
Output

- Download a CSV report file containing the image names and their classification results

Objectives

Accurate Predictions	Custom neural network model fine-tuned for MRI scan classification
Full Stack Application	Fully integrated front-end and back-end to process image uploads, run classifications, and display results
Insightful Analytics	Visualize current model statistics and classification results for performance and trends
Ease of usability	App features a simple and intuitive interface that allows users to effortlessly upload images and view results

Tech Stack



shadcn/ui

tailwindcss

React

Firebase

Flask

TensorFlow

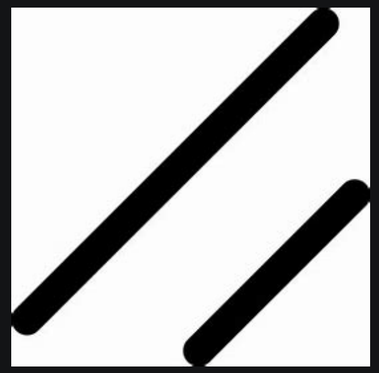
NumPy

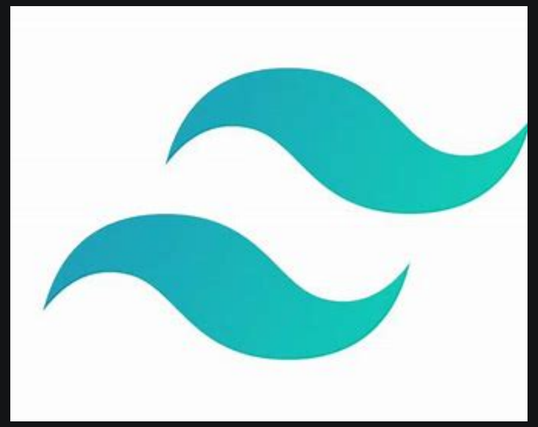
Azure Cosmos

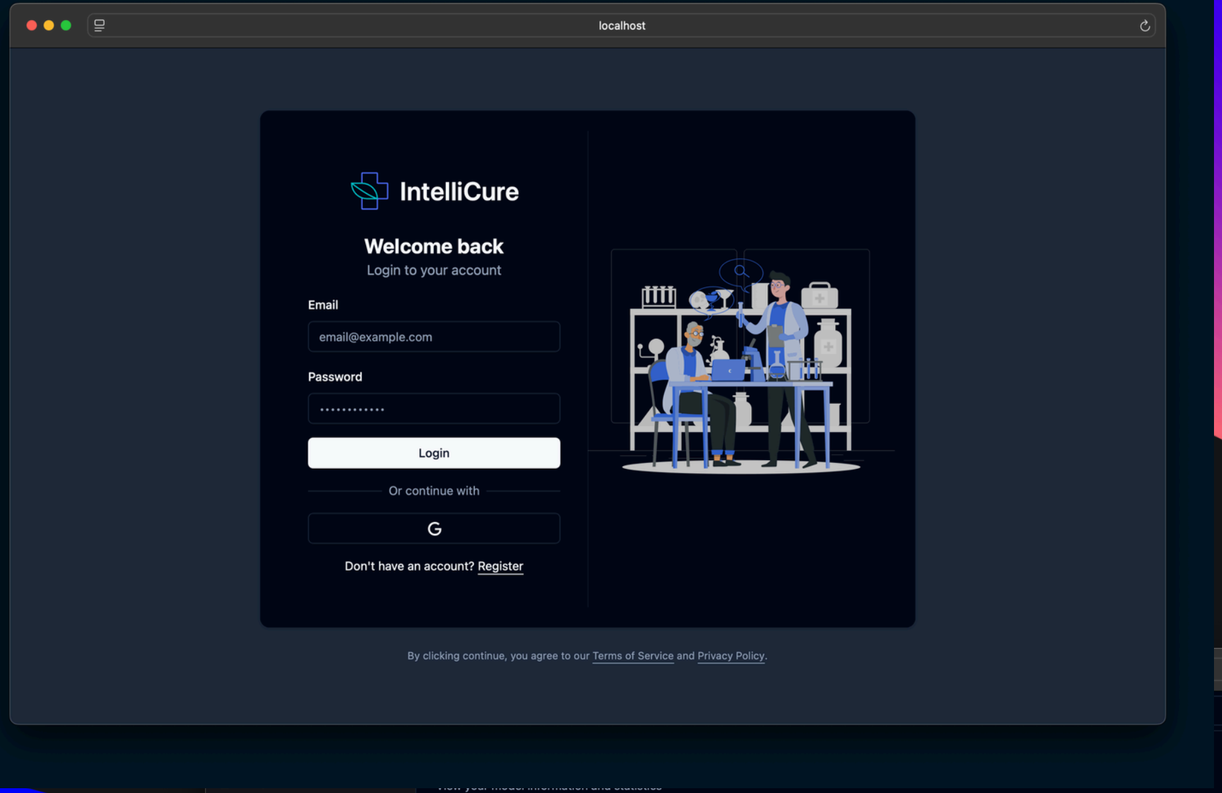
Front-End

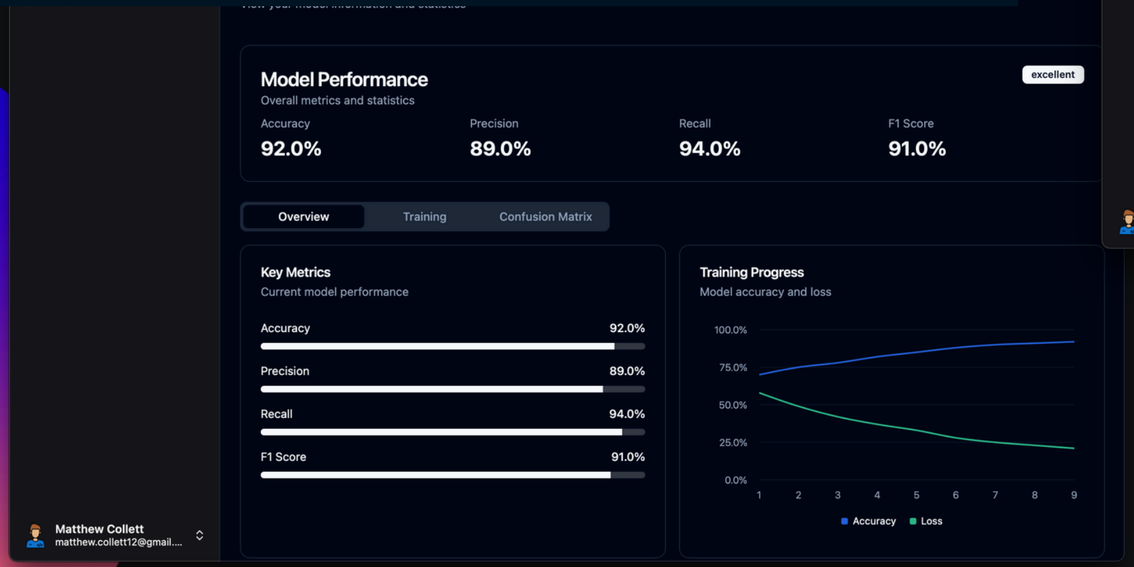
Clean & Functional

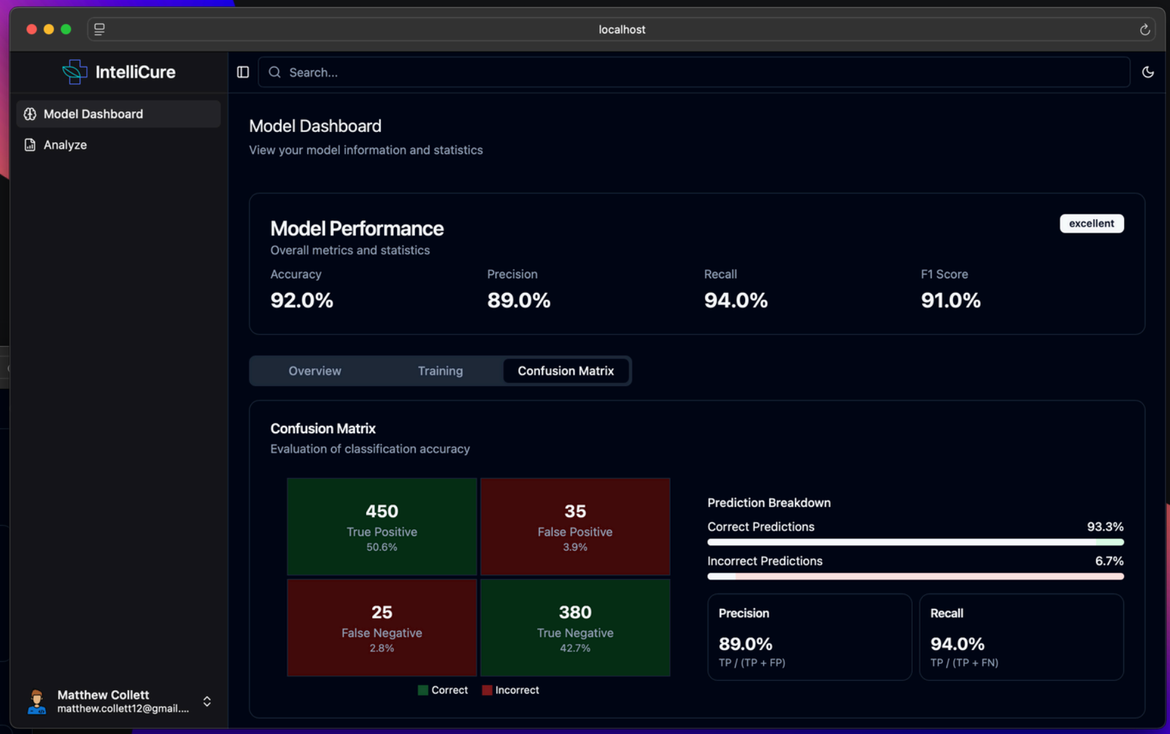
- Simple & aesthetic UI for non-technical users
- Displays data in a simple way for understandability
- Query backend to perform all data processing
- Easily can extend application with new charts and analytics






localhost
IntelliCure
Welcome back
Login to your account
Email
email@example.com
Password
Login
Or continue with
G
Don't have an account? Register
By clicking continue, you agree to our Terms of Service and Privacy Policy.

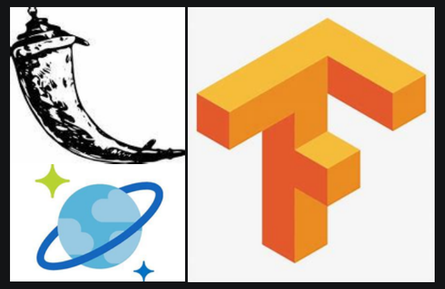

Model Performance
Overall metrics and statistics
excellent
Accuracy
92.0%
Precision
89.0%
Recall
94.0%
F1 Score
91.0%
Overview
Training
Confusion Matrix
Key Metrics
Current model performance
Accuracy 92.0%
Precision 89.0%
Recall 94.0%
F1 Score 91.0%
Training Progress
Model accuracy and loss
100.0%
75.0%
50.0%
25.0%
0.0%
1 2 3 4 5 6 7 8 9
Accuracy
Loss
Matthew Collett
matthew.collett12@gmail....


localhost
IntelliCure
Search...
Model Dashboard
Analyze
Model Dashboard
View your model information and statistics
Model Performance
Overall metrics and statistics
excellent
Accuracy
92.0%
Precision
89.0%
Recall
94.0%
F1 Score
91.0%
Overview
Training
Confusion Matrix
Confusion Matrix
Evaluation of classification accuracy
450
True Positive
50.6%
35
False Positive
3.9%
25
False Negative
2.8%
380
True Negative
42.7%
Correct
Incorrect
Prediction Breakdown
Correct Predictions 93.3%
Incorrect Predictions 6.7%
Precision
89.0%
TP / (TP + FP)
Recall
94.0%
TP / (TP + FN)
Matthew Collett
matthew.collett12@gmail....

Backend

COSMOS DATABASE

- Scaleable
- High performance
- NoSQL



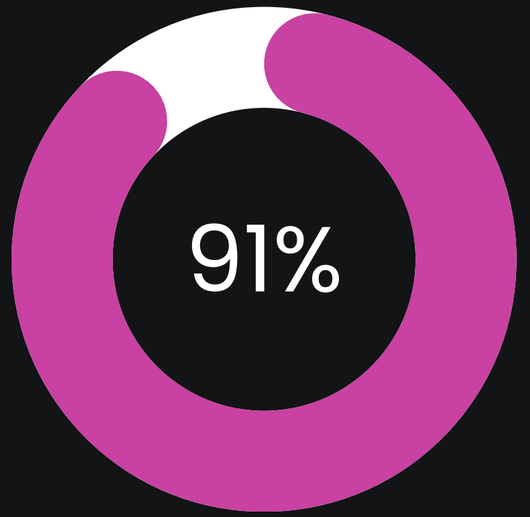
AUTHENITCATION: FIREBASE

- Efficient Computation
- Scalability
- Google login Integration

FLASK REST API

- Routing capabilities
- Easy integration

CNN Model



No feature extraction

- Automatically find patterns in the images

Detail Focus

- Scans small sections of the image, focusing on details in specific areas

Pattern Detection

- Understand how parts of the image relate to each other, helping them detect complex patterns

Hypertuned Model

Image Resizing

- Structuring data for proper model ingestion
- Images resized to 128x128

Data

- 80-20 training/testing split

Parameters

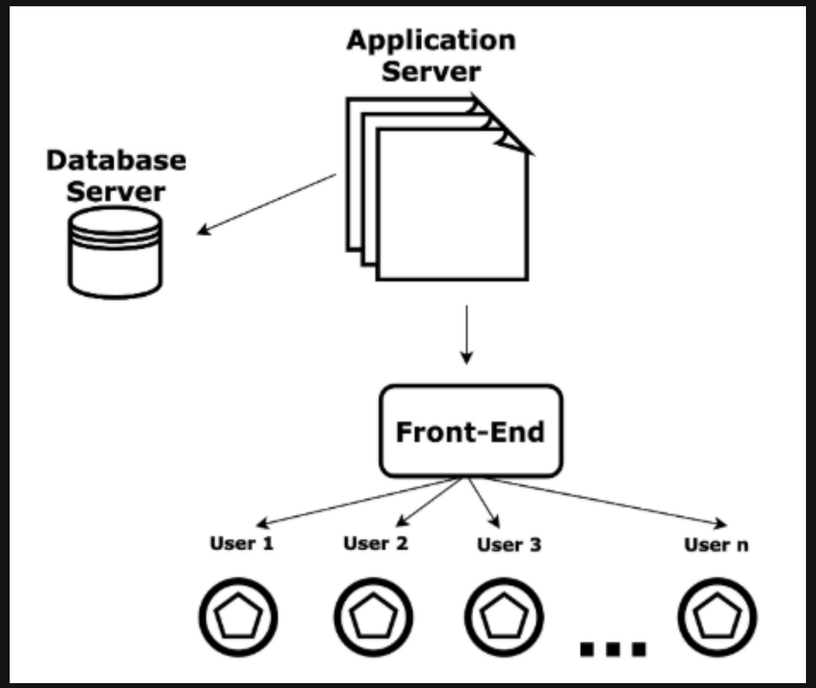
- Hypertune parameters - Number of filters, size of filters, activation function, pooling layers

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

def build_model():
    """ Define the CNN architecture for 128x128 images. """
    model = Sequential([
        Conv2D(64, (3,3), activation='relu', input_shape=(128,128,1)),  # Updated size
        Conv2D(64, (3,3), activation='relu'),
        MaxPooling2D(2,2),
        Conv2D(128, (3,3), activation='relu'),
        Conv2D(128, (3,3), activation='relu'),
        MaxPooling2D(2,2),
        Flatten(),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer="adam",
                  loss="binary_crossentropy",
                  metrics=["accuracy"])
    return model
```

Architecture

Mono-Repo 3-Tier Architecture

- Single repository structure for all components
- Clear Separation
- Presentation Layer: React UI
- Application Layer: Flask handles logic
- Data Access Layer: Database operations



Code Structure

Modular Design

- separate files and folders to keep the logic clean and reusable

High Cohesion & Low Coupling

- module handles a specific task and interact through modules

Scalability & Maintainability

- structured the project with clear functions and classes, easy to modify, improve, and extend the model in the future.

Moving Forward

Utilize Existing Models

- Explore pre-existing CNN models

Architecture vs Performance

- Trade-off between model performance and architecture during training

Extend Functionality

- Enhance model to pinpoint abnormalities in MRI scans

Thank You