AQUAWISE- by Team BlackGriffin

Problem Statement:

Inaccessible technology prices hinder effective pond management, resulting in productivity loss, heightened disease risk, and environmental hazards. Our mission is to revolutionize this paradigm with an affordable, ML-driven solution

Proposed Solution:

Implementing an **IoT-enabled pond management system** integrating **ML models** for **disease detection** and **water parameters analysis**, catering to both commercial fisheries and personal pond enthusiasts.

Tech Stack

Front End

The frontend is made using NextJS and Tailwind CSS.

Back End

The backend uses Firestore FIrebase for efficient data storage and management.

Model

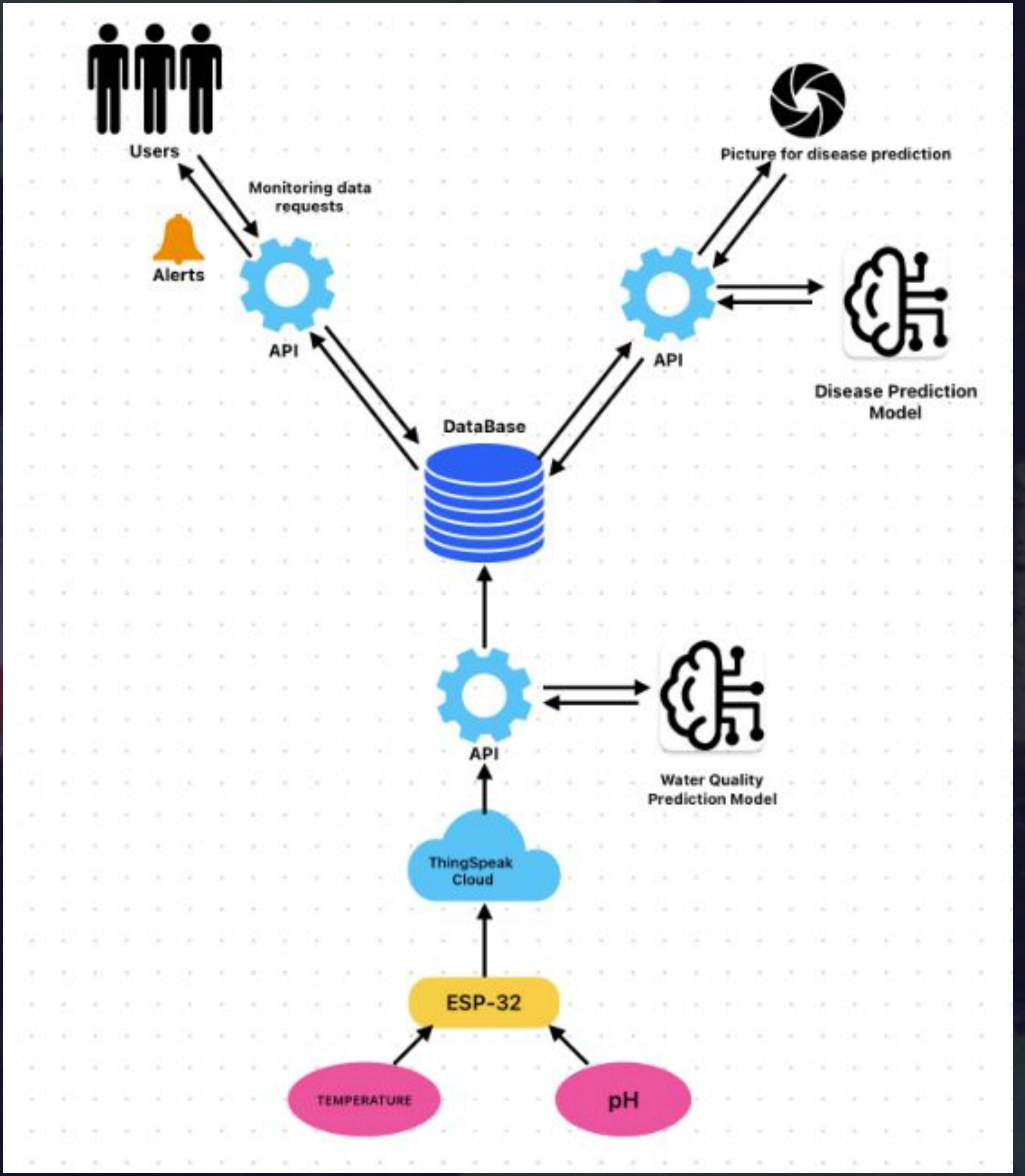
The models are trained using TensorFlow and hosted using Flask.The 2 models used in the solution are for the fish disease prediction and different water parameters analysis.

Hardware

The hardware components include Temperature and pH sensors, ESP32Wroom as microcontroller to enable real-time monitoring and data collection from the ponds.

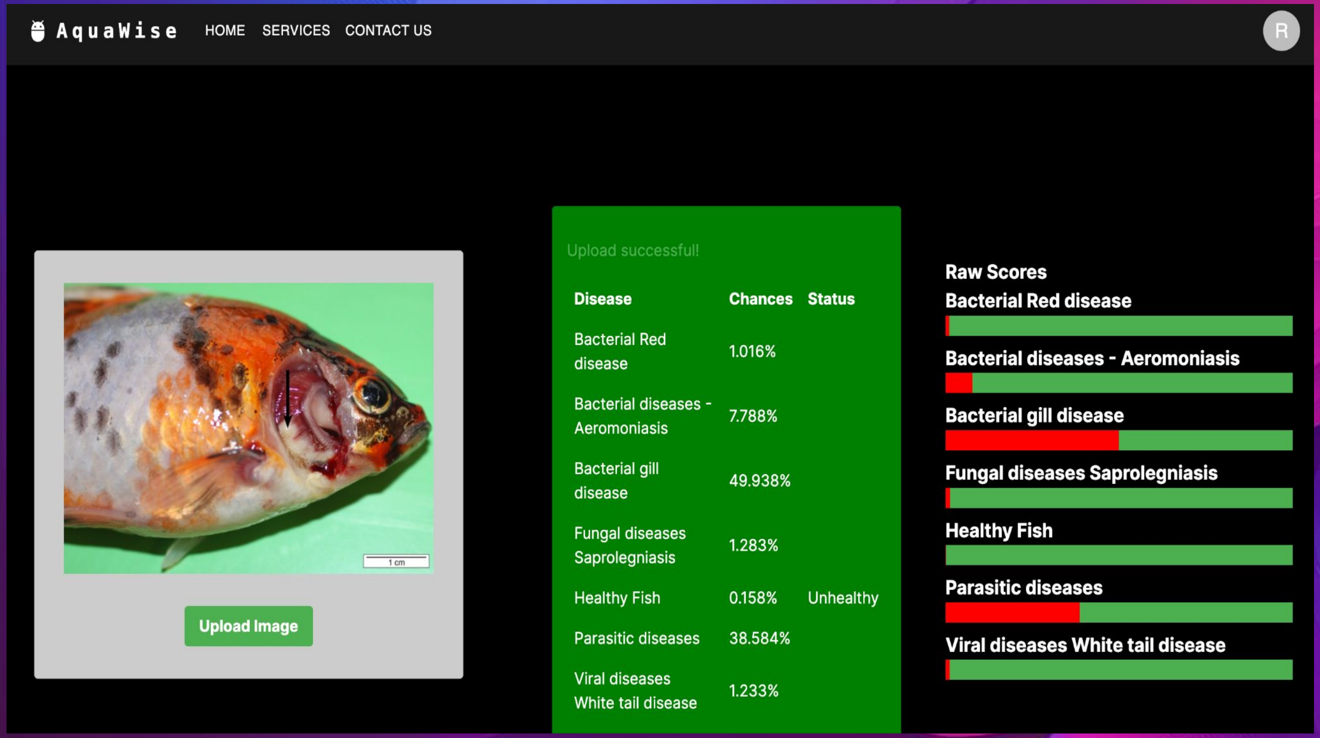
Workflow

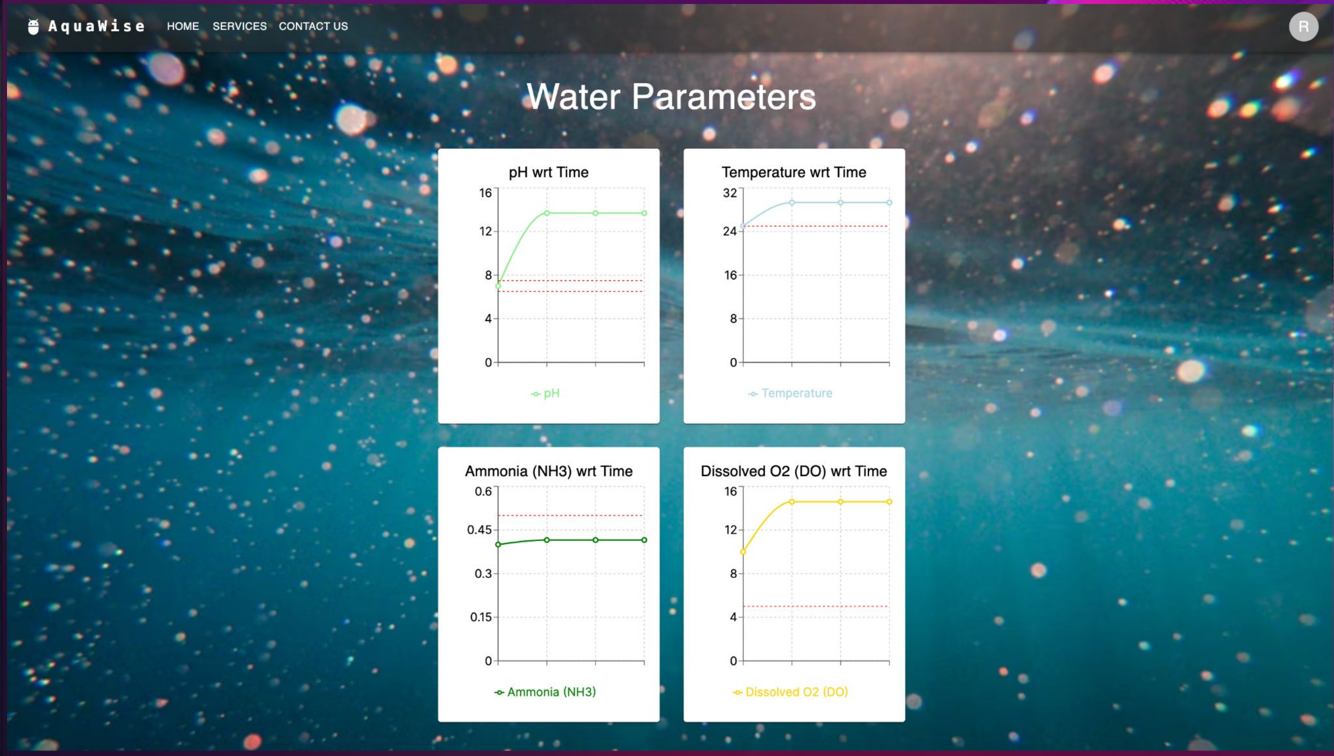
1. Users request monitoring data (temperature, pH) from ESP-32 and ThingSpeak.
2. Data is stored in a database for real-time monitoring and predictive analysis.
3. An API enables data exchange between the user interface, database, and predictive models.
4. API predicts diseases based on the monitoring data, specifically the picture provided.
5. Water Quality Prediction Model predicts water quality based on the monitoring data.
6. Alerts are sent to users if any anomalies or potential issues are detected.



Frontend

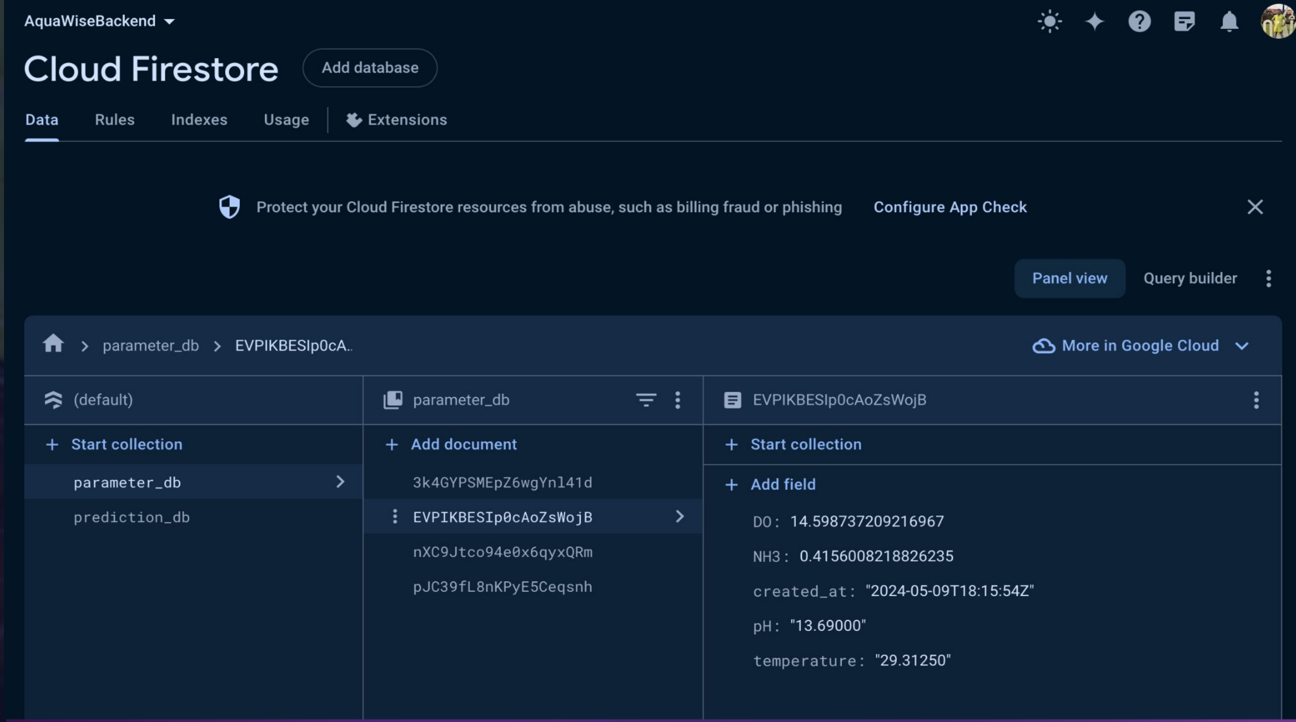
We have leveraged Next.js and Tailwind CSS for creating a seamless UI, complemented by intuitive charts for visualizing data. Our platform features dedicated pages for water parameter analysis and fish disease prediction, providing users with real-time database updates for informed decision-making.





Backend

The backend comprises two databases: one stores fish prediction model outputs, while the other logs water parameter data obtained via sensors and IoT technologies, facilitating comprehensive analysis and management.We have used Firestore for this purpose.We also implemented user authentication using Firebase.




```python
def preprocess_image(img):
    img_array = tf.image.resize(img, (224, 224))
    img_array = tf.keras.preprocessing.image.img_to_array(img_array)
    img_array = img_array / 255.0
    img_array = np.expand_dims(img_array, axis=0)
    return img_array

@app.route('/api/disease_prediction', methods=['POST'])
def image_size():
    # Check if the request contains a file
    if 'file' not in request.files:
        return jsonify({'error': 'No file provided'})

    file = request.files['file']

    # Check if the file is empty
    if file.filename == '':
        return jsonify({'error': 'No selected file'})

    img = tf.image.decode_image(file.read(), channels=3)
    img_array = preprocess_image(img)

    input_details = model.get_input_details()
    output_details = model.get_output_details()

    model.set_tensor(input_details[0]['index'], img_array)
    model.invoke()

    prediction = model.get_tensor(output_details[0]['index'])
    score = tf.nn.softmax(prediction)

    predict = {label: {'raw_score': float(prediction[0][i]), 'probability': float(score[0][i])} for i, label

    return jsonify({'predictions': predict})

if __name__ == '__main__':
```

Model

- Two models are trained i.e. the parameter prediction model and the disease detection model
- Water Parameter Analysis Model: We have trained two models i.e. Gradient Boosting Machines and Random Forest on the data. The average of these two model's output is shown to the end user.
- Disease Prediction Model: A classification model based on CNN is trained on a Disease dataset using the Tensorflow library and hosted using Flask.

```
X = df[['PH','Temperature (C)']]
y = df['Ammonia(g/ml)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#random-forest model
rf_regressor = RandomForestRegressor(n_estimators=20, random_state=42)
rf_regressor.fit(X_train, y_train)
predictions = rf_regressor.predict(X_test)
rmse = mean_squared_error(y_test, predictions,squared=False)
rmse
```

```
/opt/homebrew/lib/python3.11/site-packages/sklearn/metrics/_regression.py:483: FutureWarning: 'squared' is deprecat
version 1.4 and will be removed in 1.6. To calculate the root mean squared error, use the
function'root_mean_squared_error'.
  warnings.warn(

0.24681008738018678
```

```
X_train = np.array(X_train)
X_test = np.array(X_test)
#gradient boosting machines
gbm_model = GradientBoostingRegressor()
gbm_model.fit(X_train, y_train)
y_pred3 = gbm_model.predict(X_test)
rmse3 = mean_squared_error(y_test, y_pred3,squared=False)
rmse3
```

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(7, activation='softmax')  # Assuming you have 7 classes
])
```

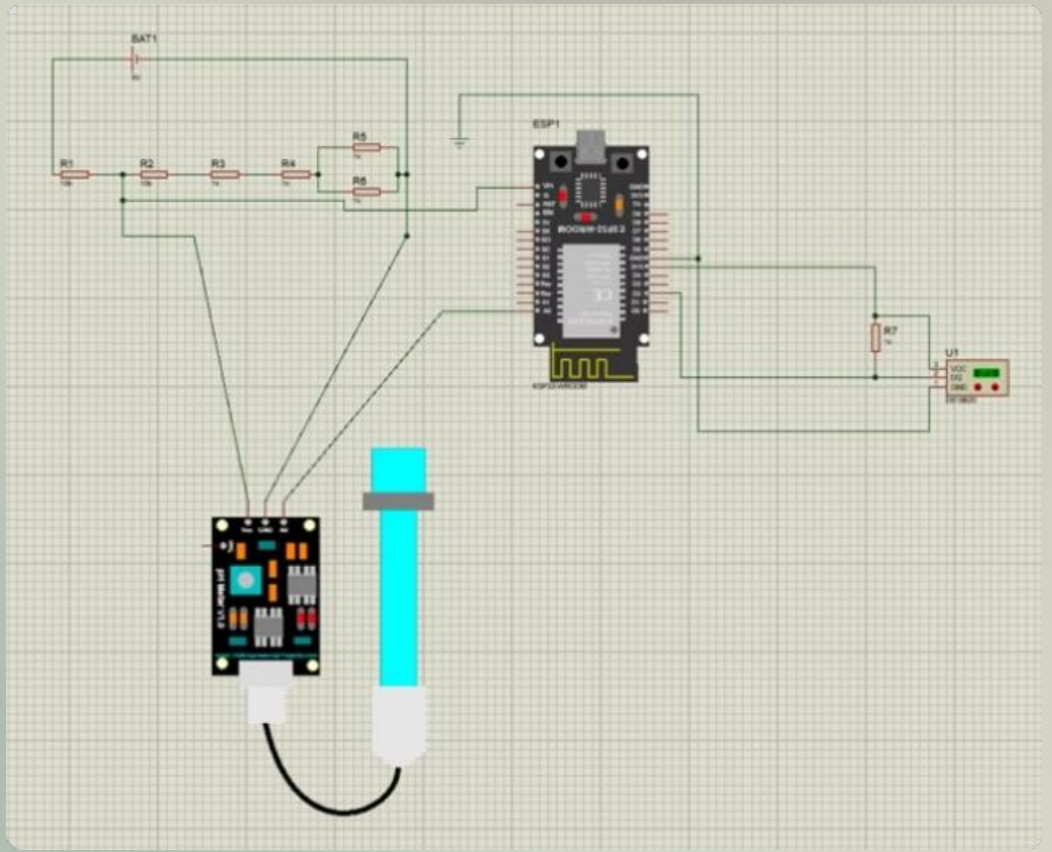
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
# Set device placement to CPU
with tf.device('/CPU:0'):
    # Create TensorFlow Datasets
    train_dataset = tf.data.Dataset.from_tensor_slices((image_train, train_labels_encoded)).batch(32)
    test_dataset = tf.data.Dataset.from_tensor_slices((image_test, test_labels_encoded)).batch(32)

    # Training
    model.fit(train_dataset, epochs=10, validation_data=test_dataset)
```

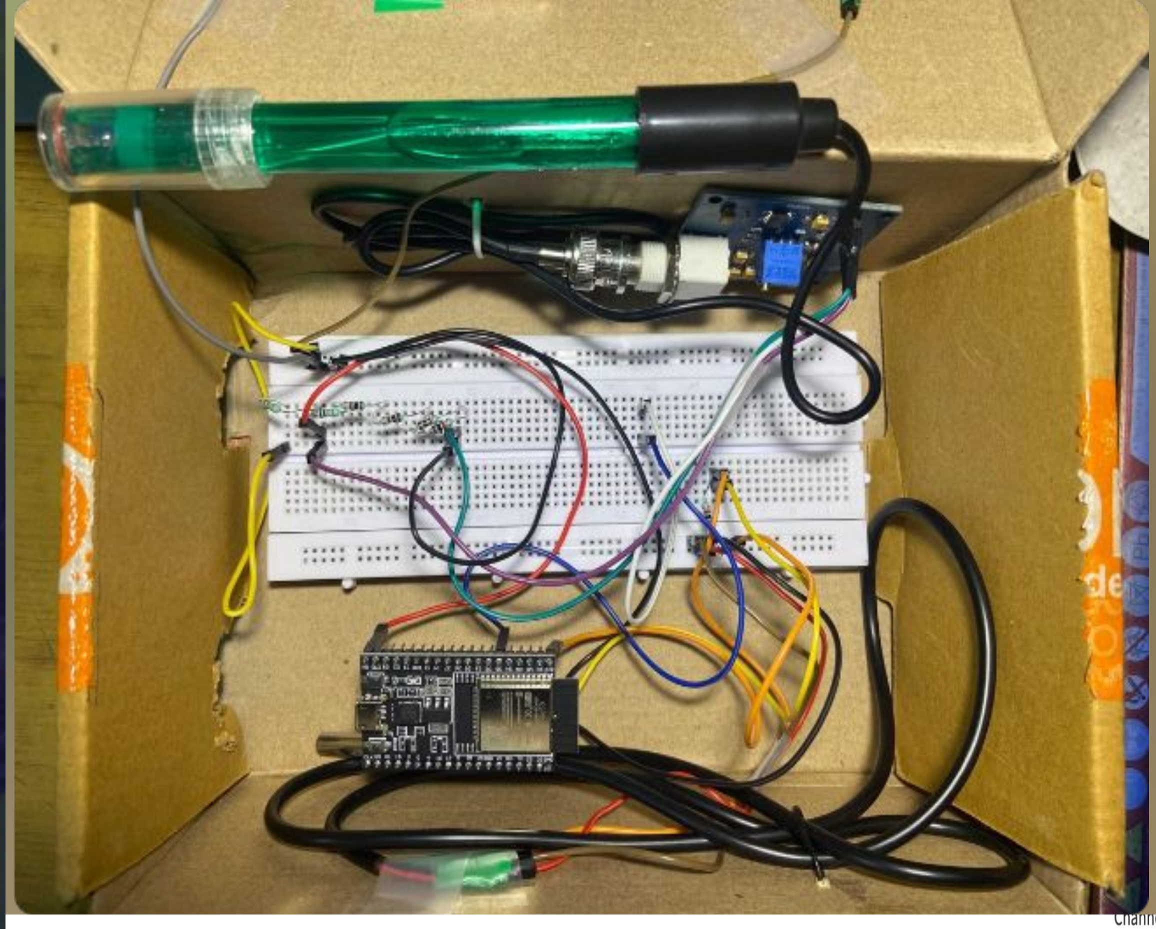
Hardware

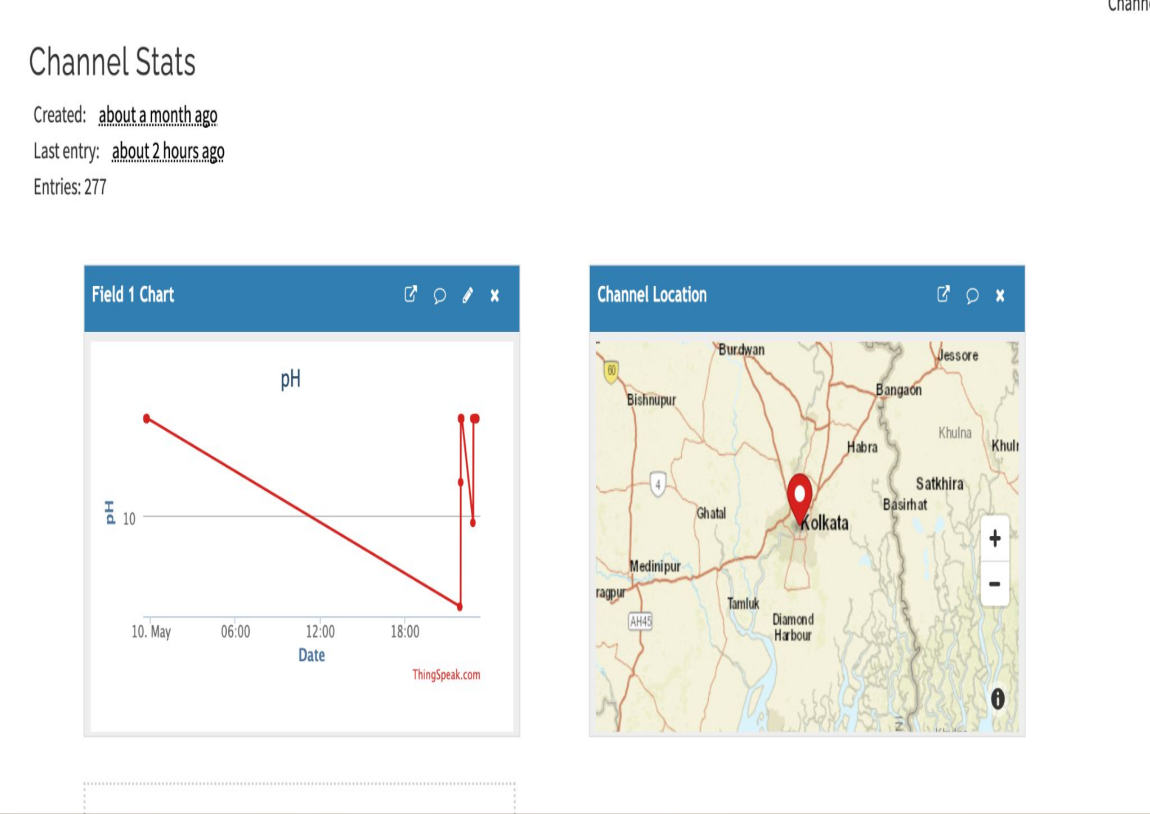
Hardware plays a vital role in the project, and its schematic is its heart. As per the aim of our project, we tried to create a hardware system to feed the database. Given the schematic, we derived the required voltage source from the given DC power source considering the availability and used ESP32 Wroom Microcontroller as the governing module of the circuit interconnected with the sensors i.e. DS18B20 and Analog pH sensor that fetches pH and temperature values respectively.



Hardware

Implementation of the hardware for real life solutions being the ultimate aim along with easy availability of the tools, we implemented the hardware using the above mentioned sensors and microcontrollers, which was programmed using Arduino IDE and the data were sent and synced with the Thingspeak Cloud which acted as a moderator database to our ML model.





Business Model

Key Partners
- Hardware Manufacturer
- Hosting Service Provider
- Delivery service providers

Key Activities
- Real time updates on Monitoring platform
- Disease prediction and trend analysis

Key Resources
- Sensors and Microcontrollers
- Hosting Services

Value Proposition
- Offers cheaper IoT monitoring devices.
- Fish Disease Prediction using Images

Customer Relationships
- Personalized customer experience according to scale
- Swift complaint management system

Channels
- Website and E-Commerce sites.

Customer Segments
- Fisheries
- Fish Breeders
- Hobbyists

Cost Structure
- Hardware Costs
- Marketing Expenses
- Database and Frontend Hosting Charges
- Shipment and Godown Rental Costs

Revenue Streams
- Selling of Hardware and Software solutions
- Newsletter Subscription fee

Future Prospect

- **Enhanced ML Models:** We aim to gather more data in collaboration with fisheries and research teams to make the models more accurate and catered to different geographic locations and target region specific diseases.
- **Improved Alert Systems:** We aim to develop a mobile application and integrate better database hosting services for enhanced alerting systems.
- **Compact and Efficient Hardware:** We aim to make the hardware more curated, customized and compact with the same efficiency by looking ahead for more compact micro-processors integrated with SOCs.

Ma

Kinjal Bhattacharyya

Team Leader

ML model development and Integration

Arya Bhattacharyya

Member

Frontend and Database Development

Mokshyada Mishra

Member

Hardware and Backend Integration.

Sophomores at National Institute Of Technology Rourkela