PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University Mohamed Khider – BISKRA
Faculty of Exact Sciences, Natural and Life Sciences
Department of Computer Science

Order N°: /M2/2024

Thesis

Presented to obtain the academic master's degree in

Computer Science

Option: Artificial Intelligence.

An Enhanced Intelligent System for Palm trees Mite disease Forecasting

---

Presented by :

Aimen CHAIB

Defended on 25/06/2024 before the members of the jury composed of :

Asma Ammari	Dr. MC-B	Presedent
Salim Bitam	Prof	Supervisor
Adel Abdelli	Dr. MC-B	Examiner

Academic year 2023/2024

Acknowledgements

Praise be to the Almighty God who has given me faith, courage, and patience to carry out this work.

I want to express my deep gratitude to my supervisor Pr. Salim BITAM from Biskra university, for the confidence he has placed in me, through his presence always with me, by his direction, his modesty, his advice, and constructive remarks for the good progress of this work, In one's academic and research activities, having a helpful supervisor like Salim BITAM can make a big impact.

I would like to thank everyone who helps me to improve my work. and who gave me any remark that helped me to perfect this manuscript.

I express my deep gratitude to my parents, my brothers, my sister, and my whole family for their supports and prayers that allowed me to achieve this modest job. I am very grateful for the confidence they have placed in me. Finally, I express my gratitude to all those who have contributed in one way or another to the development of this work.

O Allah, send your blessings on your noble messenger, his family, and companions, and bless us in our life.

AIMEN CHAIB

ملخص

تواجه مزارع النخيل في الجزائر تحديات كبيرة بسبب انتشار آفات النخيل، وأبرزها حشرة بوفروه ، هذه الحشرة تسبب أضراراً جسيمة لمحاصيل النخيل، حيث يمكن أن تؤدي إلى خسائر كبيره من الإنتاج في السنوات الجافة والحارة. تعتمد الحشرة في انتشارها على الظروف البيئية المناسبة، مما يجعل مكافحتها تحدياًا مستمراً للمزارعين.
في الجزائر، يُعتبر النخيل من المحاصيل الزراعية الأساسية التي يعتمد عليها الاقتصاد المحلي بشكل كبير. النخيل ليس فقط مصدراً رئيسياًا للتمور، ولكنه أيضاًا يمثل جزءاً من التراث الثقافي والزراعي للبلاد. ومع ذلك، تواجه هذه المحاصيل تهديدات متزايدة من الآفات الزراعية، مما يؤدي إلى تراجع الإنتاجية وزيادة التكاليف على المزارعين.

تُعد مكافحة حشرة بوفروه أمراً معقداً لعدة أسباب. أولاً، يصعب اكتشاف الإصابة في مراحلها المبكرة باستخدام الفحص البصري التقليدي. تتطور الإصابة بسرعة، مما يجعل من الصعب على المزارعين تحديد الوقت المناسب لاتخاذ إجراءات مكافحة فعالة. ثانياًا، يتطلب التحكم في انتشار الحشرة استخدام مبيدات كيميائية أو بيولوجية، والتي قد تكون مكلفة وصعبة التطبيق على نطاق واسع.

علاوة على ذلك، تؤدي الآفات الزراعية إلى زيادة التكاليف التشغيلية للمزارعين، بما في ذلك تكاليف المبيدات والعمالة اللازمة لتطبيقها. هذه التكاليف المتزايدة تؤثر سلباًا على الربحية الإجمالية للمزارع، مما يزيد من الضغوط الاقتصادية على القطاع الزراعي في الجزائر.
للتصدي لهذه التحديات، جاء مشروعنا كحل مبتكر يعتمد على الذكاء الاصطناعي وإنترنت الأشياء و هندسه البرمجيات. يوفر النظام تنبؤات دقيقة ومبكرة بانتشار حشرة بوفروه، مما يساعد المزارعين على اتخاذ الإجراءات الوقائية المناسبة في الوقت المناسب. يتم جمع البيانات من أجهزة استشعار منتشرة في المزارع، وتُحلل باستخدام نماذج التعلم العميق لتقديم تنبؤات دقيقة.

بفضل هذا النظام، يمكن للمزارعين تقليل الاعتماد على الفحص اليدوي والمعالجات الكيميائية المكلفة. يُسهم النظام في تحسين إدارة المحاصيل، وزيادة الإنتاجية، وتقليل الخسائر الاقتصادية. هذا الحل المتكامل لا يحسن فقط من كفاءة مكافحة الآفات، ولكنه أيضاًا يعزز الاستدامة الزراعية ويساعد في الحفاظ على الاقتصاد المحلي.

كلمات مفتاحية: مركز بيانات، بوفروه، الشبكات العصبيه ،التوقع المبكر، التعلم العميق، إنترنت الأشياء.

Abstract

Algeria's palm plantations face significant challenges due to the spread of palm pests, most notably the mite, which causes serious damage to palm crops, as it can result in significant losses of production in dry and hot years. The insect in its spread depends on appropriate environmental conditions, making its fight a continuing challenge for farmers.

In Algeria, palm is a basic agricultural crop on which the local economy relies heavily. Palms are not only a major source of dates, but also a part of the country's cultural and agricultural heritage. However, these crops face increasing threats from agricultural pests, resulting in reduced productivity and increased costs for farmers.

The fight against the mite is complicated for several reasons. First, it is difficult to detect an injury in its early stages using traditional visual examination. Infection develops rapidly, making it difficult for farmers to determine the right time to take effective control action. Secondly, controlling the spread of the insect requires the use of chemical or biological pesticides, which may be costly and difficult to apply widely.

Furthermore, agricultural pests increase farmers' operational costs, including pesticide and labour costs for their application. These increased costs adversely affect the total profitability of farms, increasing economic pressures on Algeria's agricultural sector.
To address these challenges, our project came as an innovative solution based on AI, IoT and software engineering. The system provides accurate and early predictions of the spread of the mite, helping farmers take appropriate preventive action in a timely manner. Data are collected from sensors scattered on farms, and analyzed using deep learning models to provide accurate predictions.

Thanks to this system, farmers can reduce reliance on manual inspection and costly chemical treatments. The system contributes to improved crop management, increased productivity and reduced economic losses. This integrated solution not only improves the efficiency of pest control, but also promotes agricultural sustainability and helps sustain the local economy.

Key words: *IOT, Deep learning, Forecasting, Data center, Boufaroua.*

Contents

List of Figures

List of Tables

List of Abbreviations

DPM	Date Palm Mite
IOT	Internet Of Things
O. afrastiacus:	Oligonychus Afrastiacus
APIs	Application Programming Interfaces
DHCP	Dynamic Host Configuration Protocol
SQL	Structured Query Language
NoSQL	No Structured Query Language
OLTP	Online transaction processing
JSON	JavaScript Object Notation
DBaas	DataBase As A Service
SOAP	Simple Object Access Protocol
REST	Representational state transfer
RPC	Remote procedure call
XML	Extensible Markup Language
HTTP	Hypertext Transfer Protocol
URLs	Uniform Resource Locator
DNS	Domain Name System
BIND	erkeley Internet Name Domain
NDVI	Normalized Difference Vegetation Index
UML	Unified Modeling Language
MVVM	Model View ViewModel
JWT	Json Web Token

General introduction

Context

Palm date culture is one of the strategic sectors on which Algeria relies as part of the national economic recovery, especially since it has about 20 million productive date palms, producing nearly 11 million quintals of dates; pointing out that the value of national date production has been estimated at more than 487 billion Algerian dinars (equivalent to 3.33 billion dollars) in 2021, or 14% of the value of national agricultural production (RA, 2022) [8].

Adressed problem

In Algeria, one of the most prevalent pests is Palm trees spider mite like (Boufaroua). Early eye detection of mite disease is challenging, and because it spreads quickly, controlling the disease in its advanced stages is also quite challenging. If Boufaroua is not treated in a timely manner, it destroys a lot of dates in a short amount of time. The farmer's ignorance of the best times to apply pesticides or other techniques of bug eradication is the issue at hand.

Objective

Develop an intelligent system using Deep learning, IoT, and Software technologies, our advanced system forecasts the presence of pests and determines the optimal timing for pesticide application and other control methods. By analyzing collected data and real-time sensor information, we empower farmers with proactive alerts, enabling timely intervention.

Manuscript structure

The first chapter.

This will cover the subjects of Back-end development and the IoT. We will discuss the definition, traits, architecture, protocols, and applications of the Internet of Things.Also we will define back-end development and discussing it's components and types.

The second chapter.

The description and discussion of related works

The third chapter.

System design is covered in the third chapter, where we also display the many design diagrams that illustrate the architecture and functionality of our system.

The fourth chapter.

Coding, Experiments and Results in the Fourth Chapter: The software/hardware environment, programming languages, and learning tools will all be covered in this chapter. This chapter also presents the outcomes that were obtained.

Chapter 1

Overview

1.1 Internet of Things

1.1.1 IoT definition

The Internet of Things refers to a network of physical devices, vehicles, appliances, and other physical objects that are embedded with sensors, software, and network connectivity, allowing them to collect and share data [8].

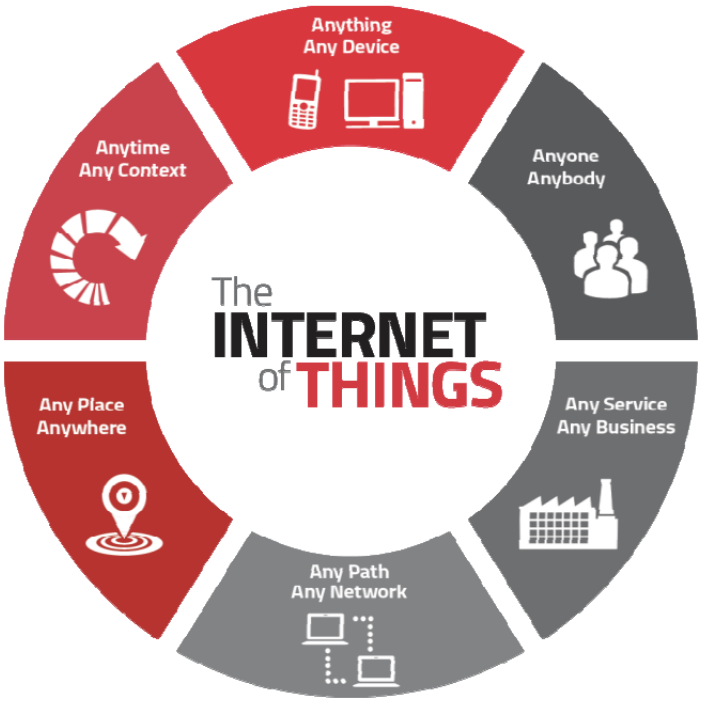


Figure 1.1: The internet of things [9]

1.1.2 IoT Characteristics

1.1.2.1 Interconnectivity

With regard to the IoT, anything can be interconnected with the global information and communication infrastructure [9].

1.1.2.2 Things-related services

The IoT is able of providing thing-related services within the constraints of things, such as privacy protection and semantic consistency between physical things and their associated virtual things. In order to provide thing-related services within the constraints of things, both the technologies in physical world and information world will change [9].

1.1.2.3 Heterogeneity

The devices in the IoT are heterogeneous as based on different hardware platforms and networks. They can interact with other devices or service platforms through different networks. [9]

1.1.2.4 Dynamic changes

The state of devices change dynamically, e.g., sleeping and waking up, connected and/or disconnected as well as the context of devices including location and speed. Moreover, the number of devices can change dynamically [9].

1.1.2.5 Enormous scale

The number of devices that need to be managed and that communicate with each other will be at least an order of magnitude larger than the devices connected to the current Internet [9].

1.1.2.6 Safety

As we gain benefits from the IoT, we must not forget about safety. As both the creators and recipients of the IoT, we must design for safety. This includes the safety of our personal data and the safety of our physical well-being. Securing the endpoints, the networks, and the data moving across all of it means creating a security paradigm that will scale [9].

1.1.2.7 Connectivity

Connectivity enables network accessibility and compatibility. Accessibility is getting on a network while compatibility provides the common ability to consume and produce data [9].

1.1.3 IoT Architecture

IOT architecture consists of different layers of technologies supporting IoT. It serves to illustrate how various technologies relate to each other and to communicate the scalability, modularity and configuration of IoT deployments in different scenarios [9].

1.1.3.1 Perception Layer :

This layer also called as physical layer, whose main functionality is to collect data, information and recognizes the usage of data in the physical world. where all the actuators work according to the information that is collected by the sensors of different object in order to perform specific operations by the corresponding objects with 2-D bar code labels and readers, RFID tags and reader-writers, camera, GPS,sensors,terminals,and sensor network.It acts like the facial skin and the five sense organs of loT, which is mainly identifying objects, gathering information [10].

1.1.3.2 Network Layer

This layer is the middle layer in IoT architecture; where it acts as an interface between application layer and perceptual layer. Its functionality is to process data and information obtained from perception layer and also broadcasting of data and connecting devices in a network [5].It is functioning similarly to the neural network where it includes a convergence network of communication and Internet network, network management center, information center and intelligent processing center, etc [10].

1.1.3.3 Application Layer

It is a layer where in actuality different applications are to be deployed with the usage of IoT services It provides the personalized based services according to user relevant requirements,also creates a link between the users and applications which combines the industry to provide high-level intelligent applications type solutions such as the disaster monitoring, health monitoring, transposition, etc [10].

1.2 Back-end Development

Backend development refers to the server-side programming that powers web applications and websites. It encompasses the creation and maintenance of the software that runs on web servers. Back-end developers are the engineers who build and manage the "behind-the-scenes" logic that determines how a web application functions.

1.2.1 Back-end Development Components

1.2.1.1 Servers

A server, whether it is a computer or software, is designed to listen for and respond to incoming requests. In web development, a server's primary role is to handle HTTP requests. Popular server-side programming languages and frameworks include Node.js, Python (with Flask or Django), Ruby on Rails, and Java (with Spring).

- Types of Servers

a. **Web Servers** These servers handle HTTP requests from clients (such as web browsers) and serve web pages, applications, and other web resources. Examples include Apache HTTP Server, Nginx, Microsoft Internet Information Services (IIS), and LiteSpeed.

b. **File Servers** File servers are dedicated to storing and sharing files over a network. They provide access to files and folders to users or clients through protocols like FTP (File Transfer Protocol), SMB (Server Message Block), NFS (Network File System), or SFTP (SSH File Transfer Protocol).

c. **Database Servers** Database servers are responsible for storing, managing, and providing access to databases. They handle queries and transactions from clients and ensure data integrity, security, and performance. Examples include MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, and MongoDB.

d. **Mail Servers** Mail servers handle the sending, receiving, and storage of email messages. They use protocols like SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol), and IMAP (Internet Message Access Protocol) to manage email communication. Examples include Postfix, Sendmail, Microsoft Exchange Server, and Dovecot.

e. **Proxy Servers** Proxy servers act as intermediaries between clients and other servers. They forward requests from clients to other servers and then relay the re-

sponses back to clients, often providing additional services such as caching, filtering, and anonymization. Examples include Squid, Nginx, and Apache Traffic Server.

f. **DNS Servers** DNS servers translate domain names (e.g., example.com) into IP addresses and vice versa. They help route internet traffic by resolving domain names to their corresponding IP addresses. Examples include BIND, Microsoft DNS Server, and Google Cloud DNS.Server.

g. **DHCP Servers** DHCP servers assign IP addresses and network configuration parameters to devices on a network dynamically. They automate the process of network configuration and help manage IP address allocation. Examples include ISC DHCP, Microsoft DHCP Server, and Cisco DHCP Server.

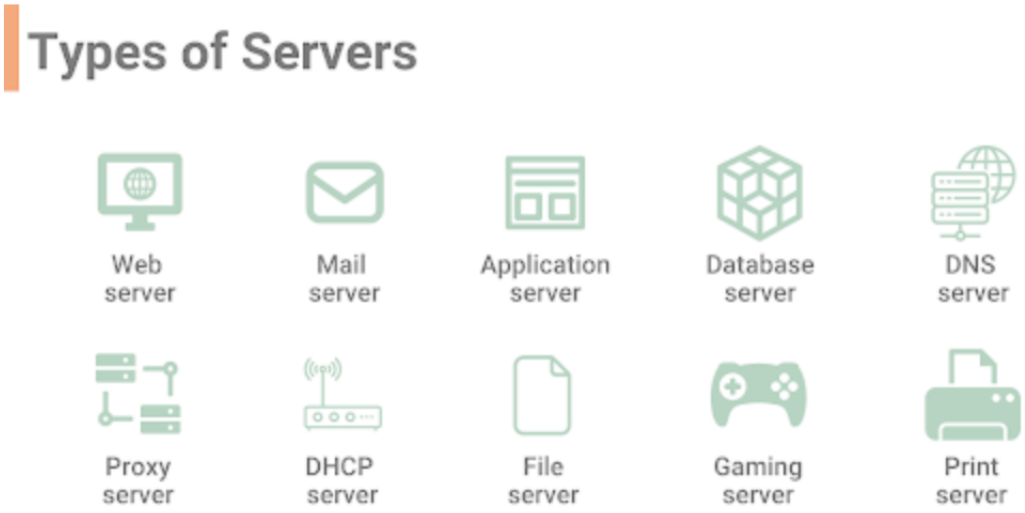


Figure 1.2: Types of Servers [11]

1.2.1.2 Databases :

Databases are used to store and manage application data. They can be relational (e.g., MySQL, PostgreSQL) or NoSQL (e.g., MongoDB, Redis). Backend developers work with databases to store, retrieve, and manipulate data based on application requirements [**?**].

Types of databases

a. **Relational databases** It became dominant in the 1980s. Items in a relational database are organized as a set of tables with columns and rows. Relational database technology provides the most efficient and flexible way to access structured information [12].

b. **NoSQL databases** A NoSQL, or nonrelational database, allows unstructured and semistructured data to be stored and manipulated (in contrast to a relational database, which defines how all data inserted into the database must be composed). NoSQL databases gained popularity when web applications proliferated and more complex [12].

c. **Object-oriented databases** Information in an object-oriented database is represented in the form of objects, as in object-oriented programming [12].

d. **Distributed databases** A distributed database consists of two or more files located in different sites. The database could be dispersed across many networks, housed on several PCs, or in the same physical location. [12].

e. **Data warehouses** A central repository for data, a data warehouse is a type of database specifically designed for fast query and analysis [12].

f. **Graph databases**

- A graph database stores data in terms of entities and the relationships between entities.
- **OLTP databases** An OLTP database is a speedy, analytic database designed for large numbers of transactions performed by multiple users [12].

g. **Cloud databases** A collection of organized or unorganized data that is housed on a private, public, or hybrid cloud computing platform is known as a cloud database. Cloud database models come in two flavors: traditional and database as a service (DBaaS). With DBaaS, a service provider handles maintenance and administrative duties. [12].

h. **Document/JSON database** Document databases are a contemporary alternative to rows and columns for storing data since they are made for storing, retrieving, and managing document-oriented information. [12].

1.2.1.3 APIs

A software application's various components can communicate with one another through APIs. APIs are used in web development to facilitate communication between an application's client-side (front end) and server-side (back end). APIs can be built using different protocols, GraphQL, or RESTful. [?].

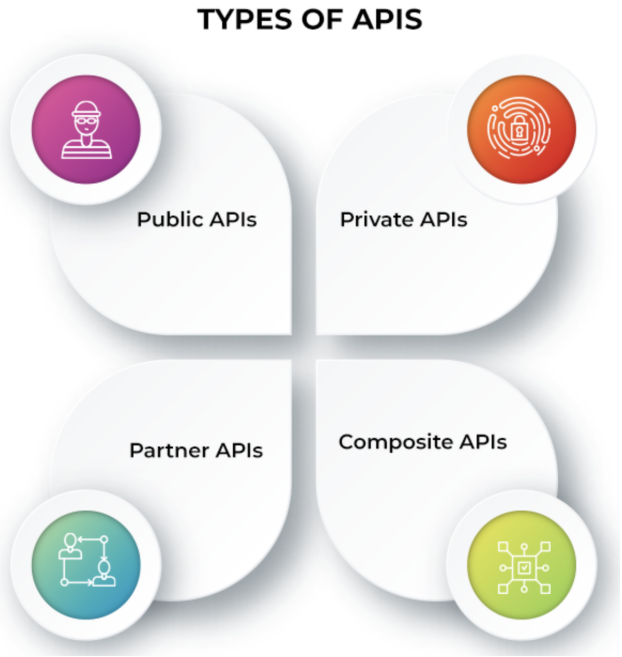


Figure 1.3: Types of APIs [13]

A - Types of APIs

a. **Open APIs** Are open source application programming interfaces you can access with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats [14].

b. **Partner APIs** Connect strategic business partners. Typically, developers access these APIs via a public API developer portal in a self-service manner. Nevertheless, in order to use partner APIs, they must finish an onboarding procedure and obtain login credentials. [14].

c. **Internal APIs** Remain hidden from external users. These private APIs aren't accessible to individuals outside of the organization and are meant to enhance productivity and communication across different internal development teams [14].

d. **Composite APIs** Combine multiple data or service APIs. They allow programmers to access several endpoints in a single call. Composite APIs are useful in microservices architecture where performing a single task may require information from several sources [14].

B - API protocols

a. **SOAP** a protocol that facilitates data sharing between client and server apps using the XML format. They come with an envelope tag that summarizes the contents of the message. The message body and headers provide related information. Since it isn't dependent on any one programming language, it is adaptable. Instead, it permits software programs created in several programming languages. Rather, it allows software applications written in different programming languages to communicate by adhering to the SOAP protocol rules [15].

b. **REST** The most popular architectural approach to building APIs. It utilizes the standard HTTP methods and resource-based URLs to perform operations. REST APIs are stateless, which means that they do not store any client-related information. The resource's state representation is transferred to the server by the client through a request. Typically, it is sent in one of the following formats: plain text, XML, or JSON. [15].

c. **RPC** Invokes functions or procedures on a remote system. It follows a client-server model where the client initiates a call, and the server executes the procedure requested. The client invokes the remote procedure call using a normal function syntax abstracting the complexities of network communication. It facilitates communication between dispersed systems by calling processes as though they were local calls. Popular remote procedure call (RPC) frameworks include XML RPC, JSON RPC, Apache Thrift, gRPC, and others. [15].

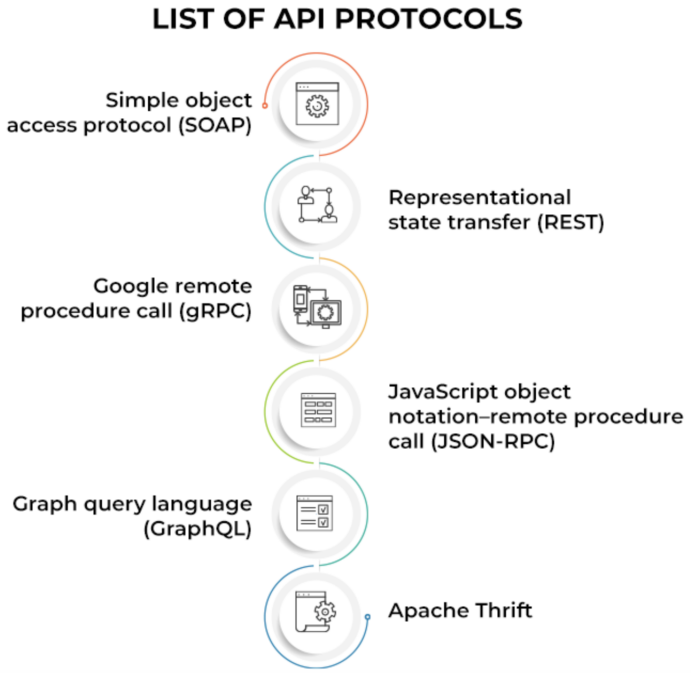


Figure 1.4: APIs Protocols [13]

1.3 O. afrasiaticus -Boufaroua-

O. afrasiaticus (McGregor 1939), called Boufaroua in Algeria (often this term refers to dust), is one of the pests that have spread remarkably in oasis agrosystems; it has become widespread in the Biskra region over the last twenty years (FACI, 2021). This date palm spider deposits its eggs on dates where they are firmly adhered to and shielded by a rather thick, white or greyish silky web secreted by the adult; the eggs hatch into light green, oval larvae that are roughly 0.15 millimeters long after two to three days. O. afrasiaticus typically has a life cycle of two to three weeks; the development cycle takes 12 to 18 days from egg to imago stage. It takes two to three days for the Boufaroua population on date palms to double in size. The number of people per population increases to six, which makes it extremely competitive for farmers who see their harvests run out. [10].

All palm groves in North Africa and the Middle East are home to Boufaroua, a member of the Tetranychidae family that was first identified and documented in Algeria by Marc André in 1932 (there was a misidentification between Oligonychus afrasiaticus and Paratetranychus simpliplex). This mite is roughly 0.22-0.44 mm long and 0.17-0.20 mm wide; the female is larger than the male. It is almost undetectable to the human eye. [10].

Figure 1.5: O. afrasiaticus male (top) and female (bottom) observed under optical microscope (x 100) [16]

1.3.1 The Period of appearance of the Boufaroua

The two major outbreaks of this pest were observed from the second dekad of June and the first dekad of August (Fig. 1.6). However, a small proliferation was recorded at the end of September, but it did not cause much damage [10].

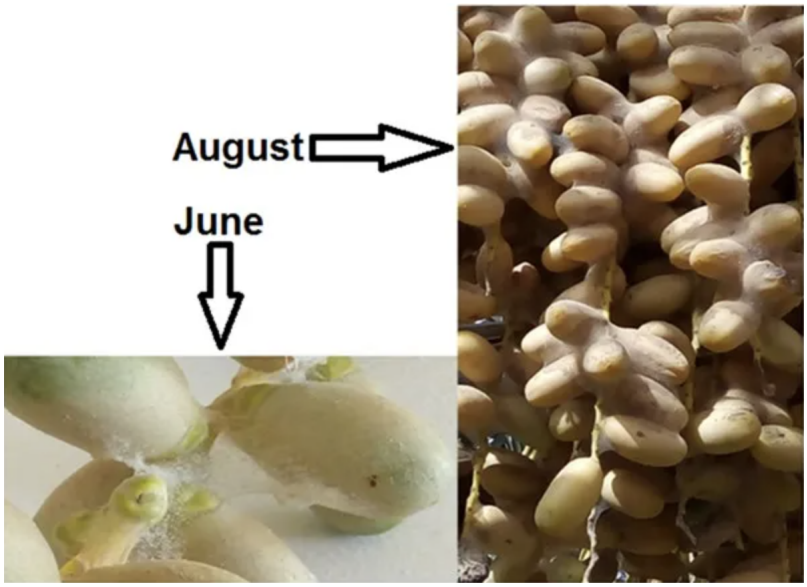


Figure 1.6: Installation of webs of Boufaroua on dates (photo credit: Mohammed FACI, 2021) [10].

1.3.2 The weather factors of appearance of the DPM -Tolga-

[10] Farmers surveyed in the field claimed that there had never been much of this date pest in the past. The expansion of Boufaroua became more apparent at the start of the millennium, albeit no substantial harm was reported. On the other hand, the damage was greater and the pest's spread was more obvious in 2021. The following factors have contributed to this pest's spread:

1.3.2.1 Recording of high air temperatures and low-rate relative air humidity inside palm groves [10]

The Boufaroua mite needs about two weeks to reach the adult stage; therefore, the averages of the 15 days preceding the start of the two waves of the mite's spread are (Fig. 1.7): - first wave (from 15 June 2021); minimum temperature 26.97 °C, maximum temperature 35.73 °C and relative humidity 34.63 %. - second wave (from 05 August 2021); minimum temperatures were 31.76 °C, maximum 41.38 °C, while humidity was 24.58 % [10].

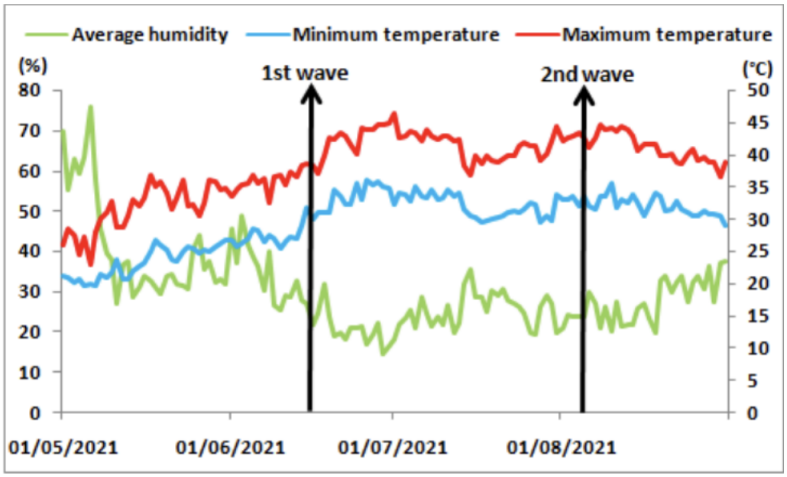


Figure 1.7: Daily evolution of temperature and relative air humidity during the period of the Boufaroua propagation in Tolga (May2021 - august2021) [10].

1.3.2.2 Surface temperature and soil moisture

- In May 2021, the low surface temperature class ranged between 28.9 to 33.8 °C and occupied 5717.5 hectars, mainly in areas covered by date palm and agricultural lands, medium temperatures class ranged between 33 °C and 39 °C and occupied nearly 22597 hectares. On the peripheral area of agricultural lands, high temperatures ranged between 39.3 °C and 44.8 °C, on bare ground, where vegetation is sparse, and on rocky outcrops. During the same period, low soil moisture values varied between 0.11 and 0.16 , on bare soils outside the agricultural lands, and occupied 9379.5 ha (nearly 22 % of the study area),

medium values ranged between 0.27 and 0.41,on rangelands and peripheral agricultural lands. High values are between 0.42 to 0.55 and occupy 17.62 % of the study area, these values represent palm trees and irrigated agricultural lands.

- In July 2021, the low-temperature class values raised to range between 32.8 °C and 35 °C and occupied 5712 ha of palm trees and vegetation cover, the medium temperature class ranged between 35 to 40.1 °C and occupied nearly 22584 ha, and high temperatures ranged between 40.1 °C and 42.6 °C on bare soils and also on the peripheral areas of agricultural lands. The moisture values raised vary between 0.25 to 0.44 for low values, while medium values increased to vary between 0.45 and 0.61, these classes occupied 23578.25 ha, and high values increased up to 0.62 and 0.78 and occupied 14.38 % of the study area.

- In August 2021, The low-temperature class values raised significantly to range between 37 °C and 40.3 °C and occupied 6584.3, the medium temperature class raised to range between 40.3 to 42.1 °C and occupied nearly 22514 ha, and high temperatures ranged between 43.6 °C and 47.4 °C on bare soils and also on the peripheral areas of agricultural lands.The soil humidity raised significantly and clearly, value classes raised to 0.62 to 0.99 for the highest values and occupied 8419 hectares [10].

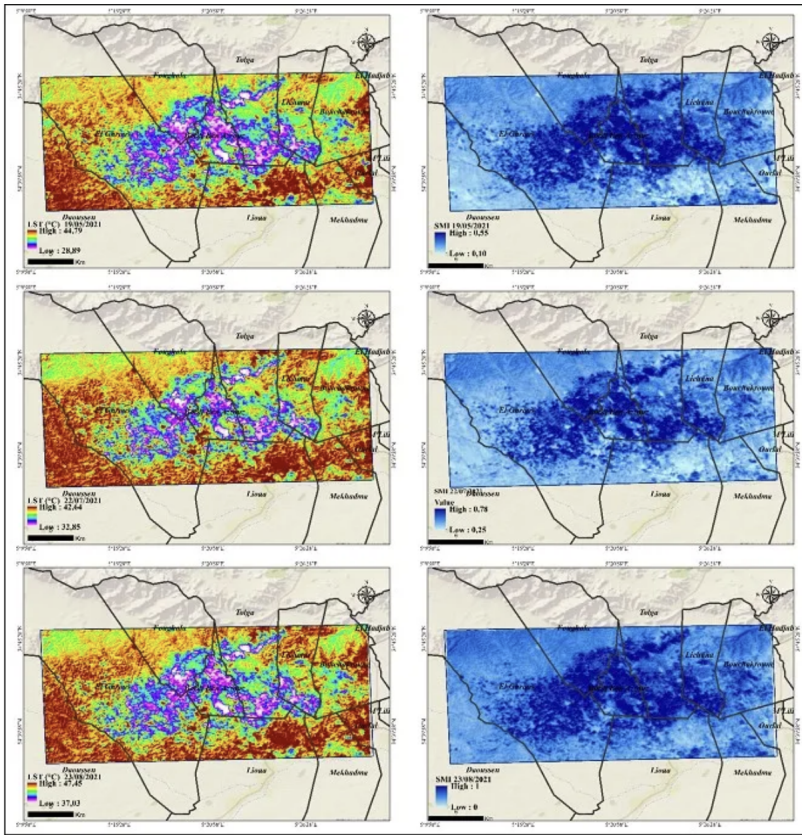


Figure 1.8: Estimation of surface temperature and soil moisture before the multiplication of Boufaroua in the Tolga region in 2021 [10].

1.3.2.3 Lack of rainfall and long-duration drought sequences

The study conducted on the O. afrasiaticus mite in Tunisian oases highlighted the significance of weather conditions, particularly rainfall and dry conditions, in the mite's seasonal occurrence. The research emphasized that the annual average rainfall in the Tunisian oases is approximately 96 mm, indicating a relatively dry climate conducive to the mite's presence.
Additionally, the study noted that the mite prefers dryer areas, which aligns with its infestation patterns starting around mid-May to June when it builds silken webs on date bunches in dry conditions, creating a suitable environment for reproduction and survival. This preference for dry conditions and the impact of rainfall on the mite's activity underscore the importance of these weather factors in influencing the appearance and population dynamics of the O. afrasiaticus mite in date palm orchards [17].

1.3.2.4 Dew points

The dew point, a critical metric in understanding atmospheric moisture levels, is the temperature at which air becomes saturated with water vapor, leading to the formation of dew. This fundamental meteorological parameter is not only determined by temperature but also heavily influenced by humidity levels. As temperature increases, the air's capacity to hold moisture expands, resulting in a higher dew point. Similarly, elevated humidity levels contribute to an increase in the dew point, as the air becomes closer to saturation. Therefore, the dew point is a product of the intricate interplay between temperature and humidity, reflecting the atmospheric conditions conducive to dew formation.
Here's the calculation formula of the dew points :

$$Td = T - ((100 - RH)/5.)$$

Where Td is dew point temperature (in degrees Celsius), T is observed temperature (in degrees Celsius), and RH is relative humidity (in percent). Apparently this relationship is fairly accurate for relative humidity values above 50% [18].

1.3.2.5 Dust, accentuated especially by the development of the road network.

The proliferation of dust, exacerbated by the expansion of road networks, poses significant challenges to environmental and public health. Wind speed emerges as the primary catalyst for the dispersal of dust particles, exacerbating air quality concerns. Beyond its role in dust dissemination, wind speed serves as a crucial agent for the transportation of minute organisms, such as mites, across vast distances. As gusts propel dust-laden air masses, they carry with them a myriad of microscopic inhabitants, including mites,

facilitating their dispersal and colonization in new habitats. Thus, wind speed not only exacerbates dust pollution but also plays a pivotal role in the dispersal dynamics of mites, underscoring its multifaceted impact on environmental health.

1.3.2.6 Lack of maintenance inside the palm groves and invasion of weeds

The attack of the Boufaroua can become widespread, with several generations succeeding over time. What are the refuge plants for this mite? Common reed Diss grass is the main host plants of this mite.

Figure 1.9: Weed (Diss) in the study area [19].

1.4 Conclusion

In conclusion, the fusion of IoT, artificial intelligence, and back-end development offers innovative solutions for agricultural challenges, notably in combating pests like Oligonychus afrasiaticus, a significant threat to date palm cultivation in North Africa and the Middle East. These technologies enable the collection of environmental data influencing pest appearance, facilitating targeted treatments and predictive modeling. Beyond pest management, IoT and AI enhance various agricultural processes, optimizing resource use and improving yields. Robust back-end systems support seamless data integration and analysis, underpinning the transformative potential of these technologies in agriculture. Ultimately, the integration of IoT, AI, and back-end development empowers farmers to make informed decisions, mitigate risks, and sustainably enhance productivity to ensure food security and environmental sustainability.

Chapter 2

Related Work

2.1 Date palm spider mite(Oligonychus afrasiaticus McGregor) forecasting and monitoring system

This study, proposed in [20], aimed to develop a decision making system for integrated pest management of the spider mite, a significant pest of date palms. The research involved sampling date palm trees, evaluating pest damage, and creating forecasting and monitoring models based on climatic and geostatistical data.

2.1.1 Material and Methods

2.1.1.1 Forecasting models

The forecasting models were simulated using Andrewartha and Brich's model, with correlation coefficients calculated between damage and weather data. The model coefficients were estimated in different months during the pest hibernation, and multiple stepwise regression analysis was performed to reduce the number of model factors. The forecasting models were simulated for each area separately, using parameters such as average spring temperature, average summer temperature, number of months with rainfall, average humidity of summer, and average moisture in April.

2.1.1.2 Monitoring model

Geostatistical methods were used to simulate monitoring models based on the spatial variables theory. The spatial correlation between samples was described as a mathematical model known as the variogram, If it is assumed that the total number of each pair of

samples N (h) located at the distance h, then :

$$y(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} [Z(xi) - Z(xi+h)]^2$$

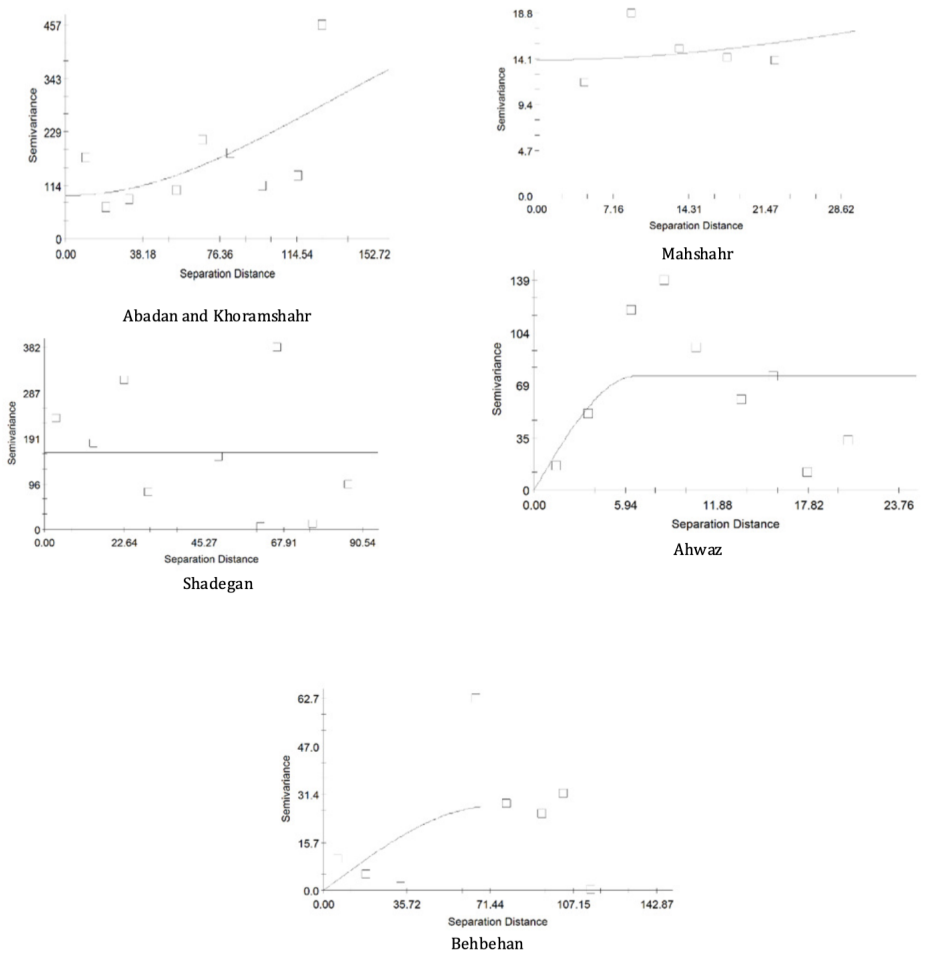


Figure 2.1: Variugarms of Date palm spider mite injury distribution in different regions of Khuzestan province.

The variogram model parameters can be used to estimate the distribution of insects. All interpolation algorithms estimate the value at a given location as a weighted sum of data values at surrounding locations. Almost all assign weights according to functions that give a decreasing weight within creasing separation distance. Kriging is a commonly used method of interpolation (prediction)for spatial data. The data are a set of observations of some variable(s) of interest, with some spatial correlation present(Katheriene, 2001)[20].

2.1.2 Results

The pest injury has started around June every year, and its severity was increased gradually by warming weather. So that the pest injury increased to a maximum on July and

August and then reduced gradually. The injury was stopped in September at the same time that fruit ripening occurs and the fruits enter the phonological date stage. The relationships between climatic factors and date spider mite injury as correlation coefficients can be seen in Table 1 [20].

Regions	Climatic parameters	t(N-1)	Coefficient of correlation	Significant Level
Ababan and Khoramshahr	X_1	-1	-0.5	0.39
	X_2	1.29	0.6	0.28
	X_{20}	0.75	-0.4	0.50
	X_{30}	-1.29	-0.6	0.28
	X_{33}	-0.35	-0.52	0.74
Shadegan	X_1	0.54	-0.53	0.62
	X_2	1	0.5	0.39
	X_{20}	0.54	-0.41	0.62
	X_{30}	1.29	-0.6	0.28
	X_{33}	-0.17	-0.54	0.87
Mahshahr	X_1	-1	-0.5	0.39
	X_2	0.08	0.51	0.93
	X_{20}	0.19	-0.51	0.86
	X_{30}	0.54	-0.63	0.62
	X_{33}	1.69	-0.7	0.18
Ahwaz	X_1	0.35	-0.52	0.75
	X_2	-0.75	0.4	0.50
	X_{20}	-0.75	-0.54	0.50
	X_{30}	0.35	-0.62	0.74
	X_{33}	0.17	-0.71	0.87
Behbehan	X_1	0.86	-0.57	0.45
	X_2	0.86	0.45	0.45
	X_{20}	0.86	-0.49	0.45
	X_{30}	1.56	-0.67	0.21
	X_{33}	0.17	-0.51	0.87

Figure 2.2: Table 1: The correlation analysis between Date palm spider mite injury and climatic parameters [20]

The date spider mite injury forecasting models were simulated as follows : [20]
- Abadan and Khoramshahr:

$$I = 3.58X_{20} - 0.43X_{30} + 1.23X_{33} - 94.96$$

- Shadegan:

$$I = 1.5X_{20} - 0.18X_{30} + 0.04X_{33} - 16.66$$

- Behbehan:

$$I = 13.6X_{20} - 0.96X_{30} + 2.57X_{33} - 27.66$$

- Mahshahr:

$$I = 0.97X_{20} - 0.24X_{30} + 0.42X_{33} - 35.65$$

- Ahwaz:

$$I = 5.88X_{20} - 0.51X_{30} + 0.25X_{33} - 29.76$$

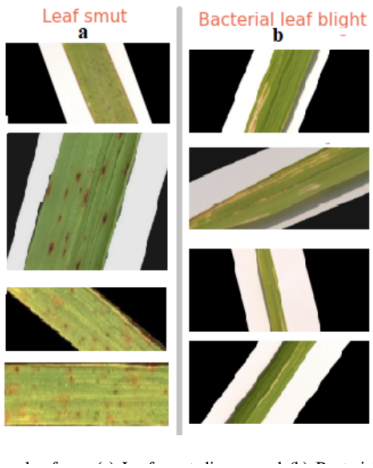
The results of multiple stepwise regression models were registered to Table 2 as below :

Regions	Parameters	Coefficient	Standard Error
Ababan and Khoramshahr	Intercept	-94.96	0.08
	X20	3.58	0.49
	X30	-0.42	0.28
	X33	1.23	0.74
Shadegan	Intercept	-16.66	0.66
	X20	1.05	0.44
	X30	-0.18	0.07
	X33	0.04	0.03
Mahshahr	Intercept	-35.65	0.62
	X20	0.97	0.38
	X30	-0.24	0.36
	X33	0.42	0.25
Ahwaz	Intercept	-29.75	0.51
	X20	5.88	0.53
	X30	-0.51	0.55
	X33	0.25	0.40
Behbehan	Intercept	-27.66	0.81
	X20	13.6	0.65
	X30	-0.96	0.12
	X33	2.57	0.28

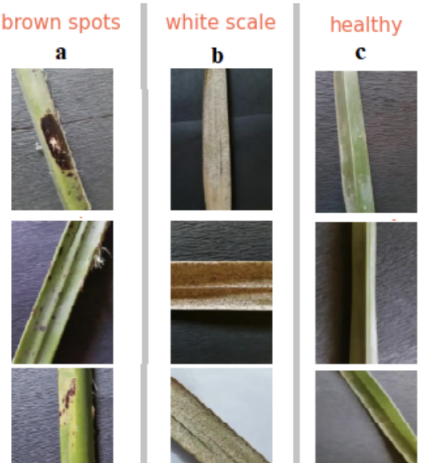
Figure 2.3: Table2: Regression analysis of Date palm spider mite injury forecasting models[20]

2.2 Classification of Palm Trees Diseases using Convolution Neural Network

In order to identify and classify the four common diseases that now affect palms—bacterial leaf blight, brown spots, leaf smut, white scale, and healthy leaves, the authors of [21] developed a CNN model. Broadly speaking, bacterial leaf blight, brown spots, leaf smut, and white scale are prevalent palm diseases. Because the nature and symptoms of these illnesses vary in terms of how they manifest, where they appear, and how widely they affect palm trees, new methods are required to identify them before they seriously harm palm trees.



(a) Samples from: (a) Leaf smut disease and (b) Bacterial leaf blight disease [6]



(b) Samples from Brown spots disease. b)Samples from White scale disease. c) Samples from healthy leaves [6]

Figure 2.4: four common diseases and healthy leaves

2.2.1 Materials and Methods

During data preparation, the authors initially scaled each image to 60 × 60 resolution to ensure consistency. To reduce data correlation and improve the visibility of crucial features, including curves and edges, so that CNN can detect them more readily, they use picture whitening based on Zero Component Analysis (ZCA). The pixel values of each image are then normalized so that they all lie between 0 and 1. The proposed model consists of four convolutional layers, a fully connected layer, and softmax as the output layer. This model's accuracy was 99.10% (compared to 99.56% for MobileNet and 99.35% for VGG16). [6]

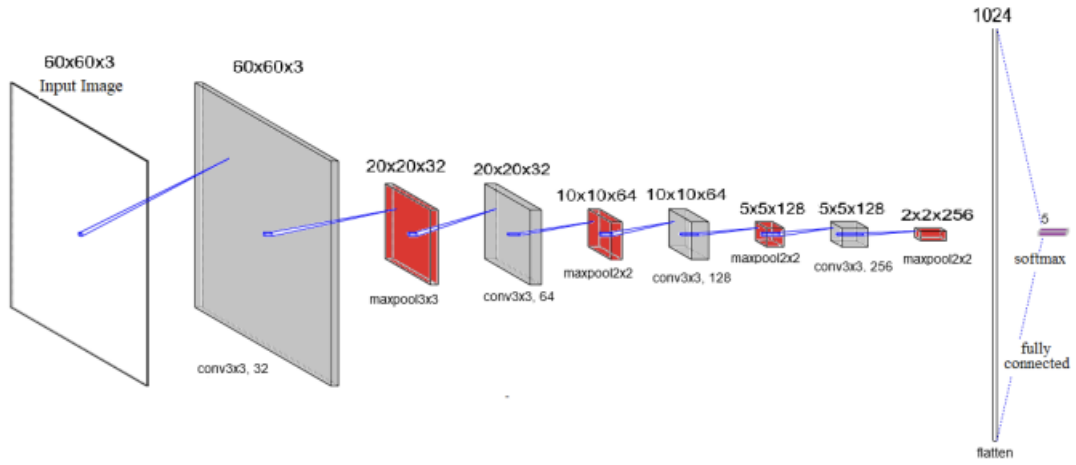


Figure 2.5: The architecture of proposed CNN [6]

In this work, an effective CNN model for identifying and classifying common palm tree diseases—such as brown spots, bacterial leaf blight, leaf smut, and white scale—was proven. This method is related to both VGG-16 and MobileNet, and it has an easy-to-understand structure and good accuracy. Furthermore, it has a lower computational training cost (20.048s/epoch) than other models (343.85s/epoch and 2559s/epoch for MobileNet and VGG-16, respectively). But this model can't pinpoint exactly which parts of the palm tree are affected. [6].

Table 2.1: The performance of each model based on accuracy [6]

Disease	Accuracy (%)		
	Proposed Model	MobileNet Model	Vgg16 Model
Brown Spots	99.55	100.00	100.00
White Scale	99.35	99.75	99.35
Bacterial Leaf Blight	98.79	98.98	98.99
Leaf Smut	99.20	98.88	99.90
Healthy	98.90	99.90	99.90
Average	**99.10**	**99.56**	**99.35**

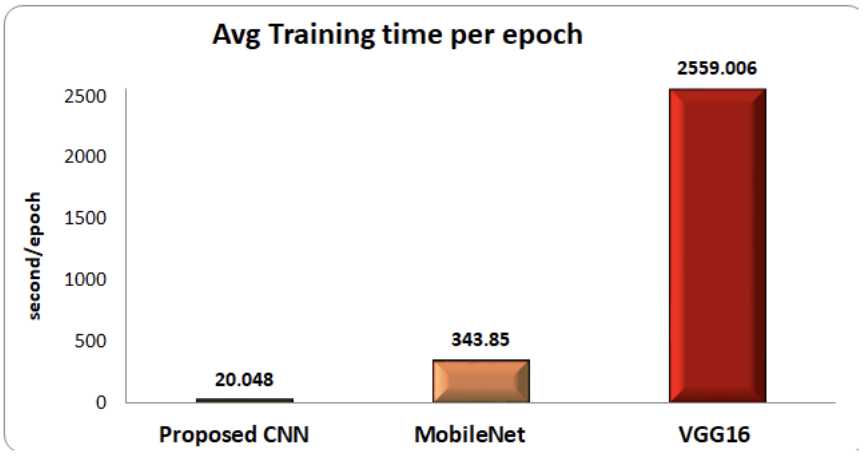


Figure 2.6: Avg training time per epoch for each model [6]

2.3 Development and Validation of Innovative Machine Learning Models for Predicting Date Palm Mite Infestation on Fruits

Authors created a precise method for monitoring and forecasting DPM infestations through decision-making.

2.3.1 Materials and Methods

The present study [21] was performed in an arid region at the experimental farm of Date Palm Research Center of Excellence (latitude: 25°16′4.3″ N, longitude: 49°42′30.1″ E) at the Agricultural Training and Research Station, King Faisal University (KFU), Saudi Arabia. The experiment was performed during two successive seasons (2021 and 2022).

In each experimental season, the study ran from 1 April (one month after fruit set) until the end of August. To train and test machine learning regression models for predicting DPM infestation on date palm fruits, the study area was sampled for two fully grown date palm cultivars (Khalas and Barhee). The meteorological variable data and fruit properties at various development stages, i.e., hababook, kimri, khalal, rutab, and tamr, were collected. Additionally, an Internet of Things-based weather station set up at the study area by Mohammed et al. collected data on the primary meteorological variables, such as the maximum temperature, minimum temperature, average temperature, maximum relative humidity, minimum relative humidity, average relative humidity, average wind speed, maximum solar radiation, and average daily solar radiation. [28] [21].

In order to identify the best model for predicting the DPM on date palm fruits based on three input variables, two regression algorithms—linear regression (LR) and decision forest regression (DFR)—were developed, assessed, and validated using Microsoft Azure Machine Learning (ML) Studio. (1) The study area's meteorological variables, such as the maximum temperature (TMax), minimum temperature (TMin), average temperature (TAvg), maximum relative humidity (RHMax), minimum relative humidity (RHMin), average relative humidity (RHAvg), average wind speed (WSAvg), maximum solar radiation (SRMax), and average daily solar radiation (DSRAvg); (2) The date fruits' physical and chemical characteristics at different phases of development, include fruit weight (FW), fruit firmness (FF), fruit moisture content (FMC), total soluble solids (TSS), total sugar (TS), and tannin content (TC); and (3) The combination data of meteorological variables and physicochemical properties. The following is a description of the regression algorithms used within Azure ML in this study [28]. They use Microsoft Azure Machine Learning to Building Predictive Models (Data Acquisition, Data Analysis, Models Training, and Models Performance) [28]. After they created and evaluated a predictive model suitable for predicting DPM infestations, we deployed it using Azure ML to make it easier to use as a predictive Azure ML web service on the Azure cloud platform to make it easier to use as a web service. The predictive experiment was automatically obtained after selecting the train model module for the best developed model. Figure shows the predictive experiment of the web service that was created automatically for the prediction in the Azure ML studio platform web service [28] [21].

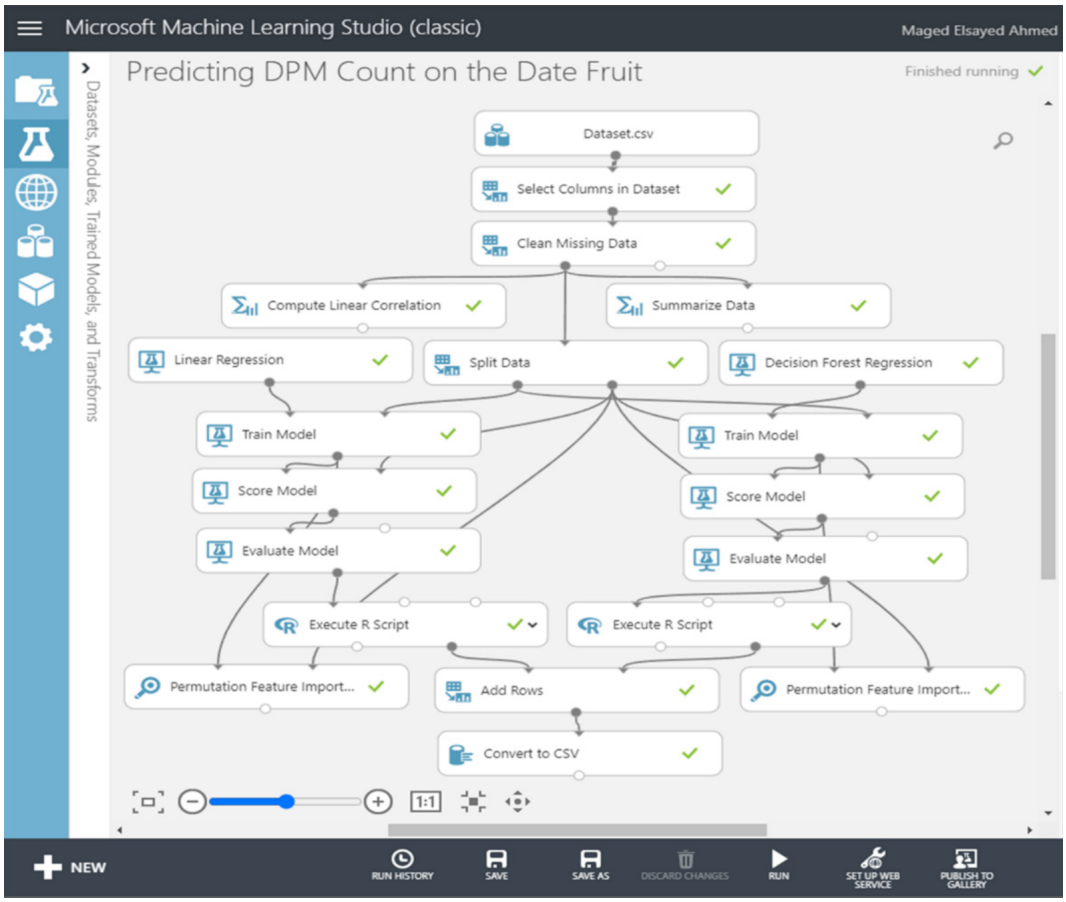


Figure 2.7: A screenshot for an experiment of the data analysis and building the predictive models for predicting DPM infestation on the date fruit using Microsoft Azure Machine Learning [21].

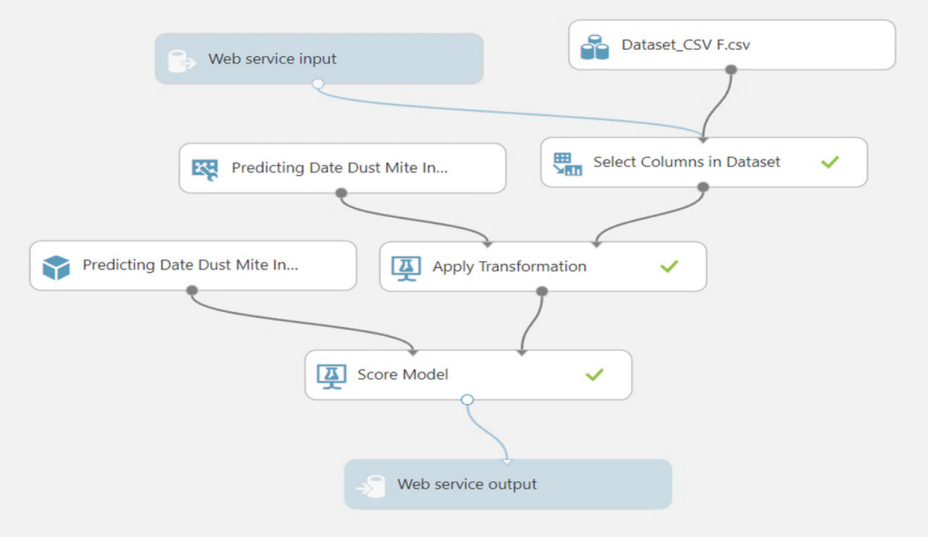


Figure 2.8: A screenshot for the experiment of the web service that was created for the prediction model in Azure Machine Learning [21].

2.4 Relationship of Date Palm Tree Density to Dubas Bug Ommatissus lybicus Infestation in Omani Orchards

2.4.1 Study area

The study area of this research [7] was in the north of Oman, Ad'Dakhiliyah Governorate, in WilayatSamail (23°14′ – 23°5′ N, 57°52′ – 57°51′ E) (Figure 1). From Saygja Village in the north (430 m) to Al Falajain Village in the south (675 m), the altitude rises. The minimum and maximum temperatures in 2017 were 14 ◦ C and 29 ◦ C, respectively, with January and June having the lowest and June having the highest values. Between January and August, the annual relative humidity varied from 0% to 75%. There was 3 mm of rain in October and 15 mm in February. Sunlight duration during the day varied from 10 hours and 42 minutes in December to 13 hours and 34 minutes in June [32] [7].

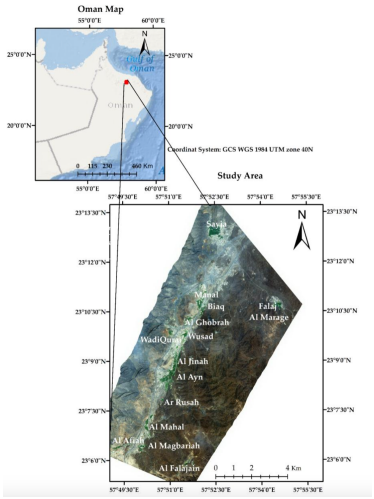


Figure 2.9: Study area location in the north of Oman [7].

To find the trees in the picture, they first employ local maxima algorithms, which find the brightest pixel inside a given frame. The pixel with the highest reflectance relative to the other pixels in the same window is assessed as a likely tree site. Three window scales were tried (Figure 2. 13): 3 m (6 × 6 pixels), 5 m (10 × 10 pixels), and 7 m (14 × 14 pixels) in order to discover which windows worked best. To distinguish between vegetation and non-vegetation, the image's Normalized Difference Vegetation Index (NDVI) was computed. After that, the date palm trees were separated from other types of vegetation using the Maximum Likelihood classification algorithm[27].

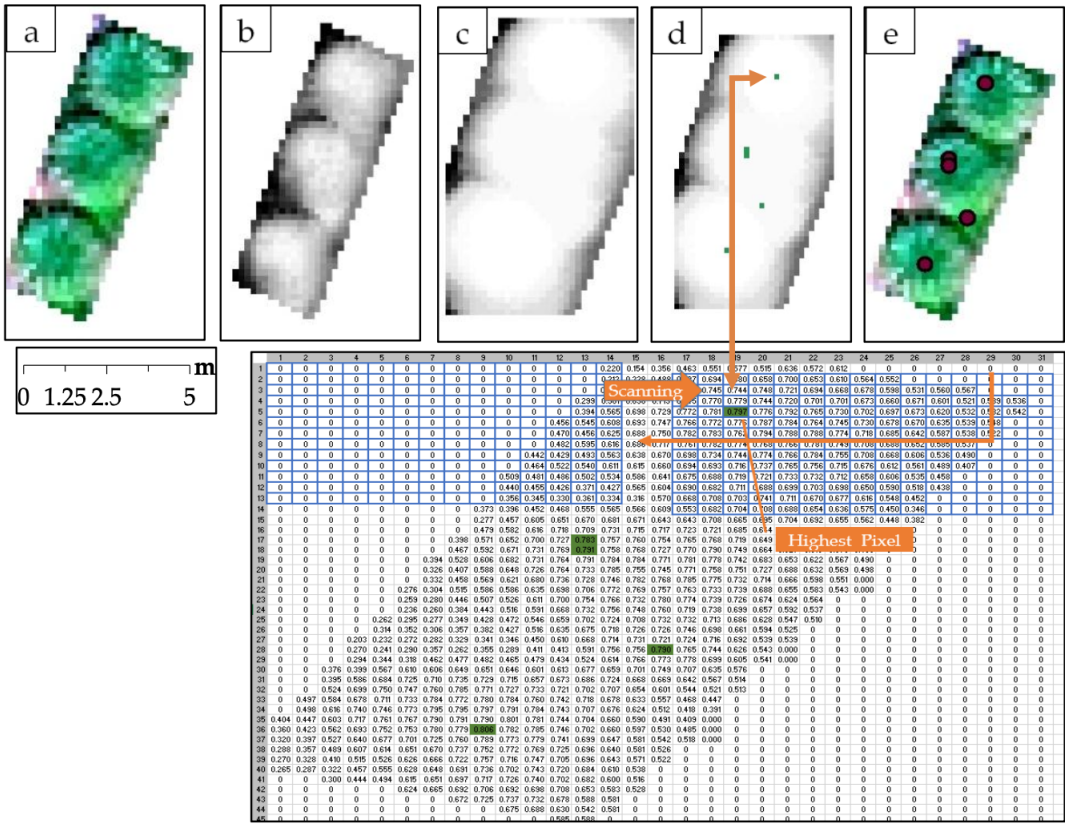


Figure 2.10: Local Maxima illustration process figure of windows 7 [7].

Next, the infestation data was prepared in an Excel spreadsheet, including the tree coordinate location and infestation levels of each tree, and saved as a comma delimited (CSV) file and imported into ArcMap as a point layer. The ordinary least square (OLS) regression was used to test the global correlation and Geographic Weight Regression (GWR) was used to find the local spatial relationship [27] [7].

The tree counting results from different windows by local maxima, we can see in the table 7m radius window was the most suitable window (error of 11.8%) overall accuracy 88.2%. The problem of local maxima is error increased with very dense and unsystematic planting patterns [7].

The current study indicates that density is one factor that contributes to the severity of the O. lybicus infestation. but this is not enough to study the O. lybicus infestation. They need to add other factors [7].

Table 2.2: Counting Date Palm Trees by Local Maxima Accuracy [27] [7].

Village	**FieldCount**	**LocalMaxima**	**Prediction Estimation Error**			
			Window 3	Window 5	Window 7	
Al'Afi'ah	282	529	371	244	87.6%	31.6%
Al Ayn	298	484	380	323	62.4%	27.5%

Al Falajain	302	400	400	268	32.5%	32.5%
Al Ghobrah	179	984	553	180	449.7%	208.9%
Al Jinah	271	457	347	240	68.6%	28.0%
Al Mahal	550	852	667	493	54.9%	21.3%
Al Magbariah	382	634	420	295	66.0%	9.9%
Falaj AlMaraga	338	578	397	275	71.0%	17.5%
Biaq	376	786	567	365	109.0%	50.8%
Manal	249	342	248	178	37.3%	-0.4%
Sayja	354	591	370	273	66.9%	4.5%
WadiQurai	582	760	591	458	30.6%	1.5%
Wusad	474	842	577	450	77.6%	21.7%

2.5 Design of a new Intelligent System for Early Prediction of Date palm spider mite (Boufaroua)

The Author of this study [1] mentioned :

2.5.1 Overall system architecture

Their architecture is composed of two entities: In the first entity (deep learning): they prepared data, build their models, train and test them (the learning phase) and save them. The second entity (IOT): this section is reserved for the prediction in two elements, part put in a palm tree composed of sensors and a communication module that sends the gathered data to Raspberry Pi 4. This part can use one or many depending on the size of the farm. Part two is Raspberry Pi 4 and screen to show results.
Next, they put models in Raspberry Pi 4. So, that it automatically makes a prediction. The figure below represents the overall architecture of our system [1].

2.5.2 Overall system operation

The working principle of our system is to use Deep Learning and IoT to predict the problem of DPM. As shown in the figure below, the user will launch the program that is in the Raspberry Pi, next it collects data for sensors and makes predictions [1].

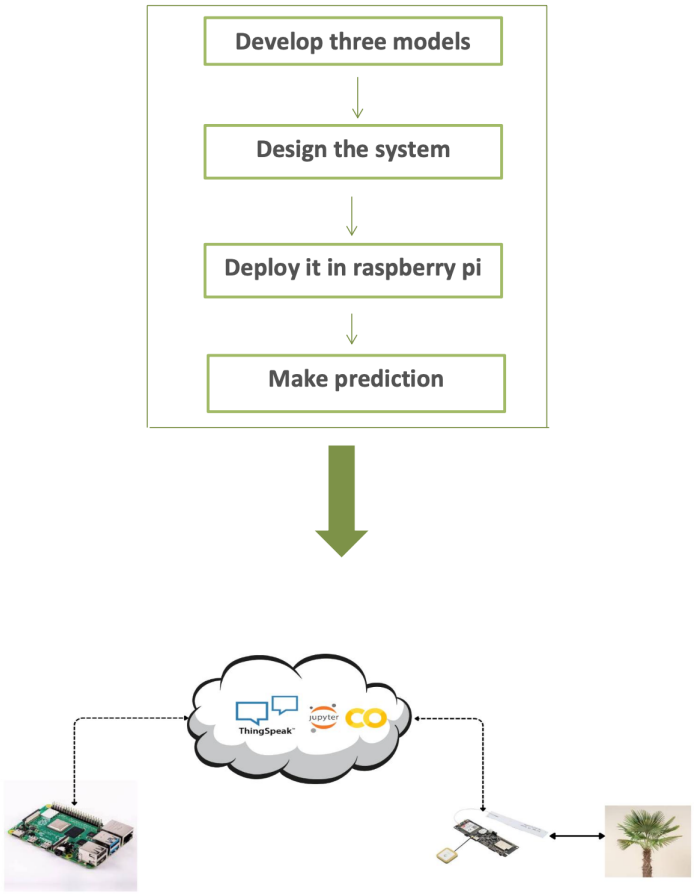


Figure 2.11: Global architecture of our system [7].

2.5.3 Methodology

In this work the author [1] expose the methodologies to realize his system. First, he builds early prediction models for DPM. Then, he present the deployment of models in the Raspberry Pi. To better understand his approach, he explains each part separately .

Table 2.3: Layers of our 90 days model.

Layer type	**Output Shape**	**Activation**
GRU	128∗90	Relu
Dropout	128∗90	-
GRU	128∗90	Relu
Dropout	128∗90	-
GRU	12890	Relu
Dropout	12890	-
GRU	128	Relu
Dense	4	Softmax

Table 2.4: Layers of our 60 days model (Table 2).

Layertype	**OutputShape**	**Activ**
Normalization	14∗60	-
LSTM	40∗60	Relu
Dropout	40∗60	-
LSTM	40∗60	Tanh
Dropout	40∗60	-
LSTM	40∗60	Relu
Dropout	40∗60	-
LSTM	40∗60	Tanh
Dropout	40∗60	-
LSTM	40∗60	Relu
Dropout	40∗60	-
LSTM	40∗60	Tanh
Dropout	40∗60	-
LSTM	40∗60	Relu
Dropout	40∗60	-
LSTM	40∗60	Tanh
Dropout	40∗60	-
LSTM	40	Relu
Dropout	40	-
Dense	4	softmax

Table 2.5: Layers of our 30 days model (Table 1).

Layer type	**Output Shape**	**Activation**
Normalization	14∗30	-
LSTM	20∗30	Relu
Dropout	20∗30	-
LSTM	20∗30	Relu
Dropout	20∗30	-
LSTM	20∗30	Relu
Dropout	20∗30	-
LSTM	20∗30	Relu
Dropout	20∗30	-
LSTM	20	Relu
Dropout	20	-
Dense	4	softmax

Train and validation of all modules will be 80% of the dataset for the train and 20% for validation:

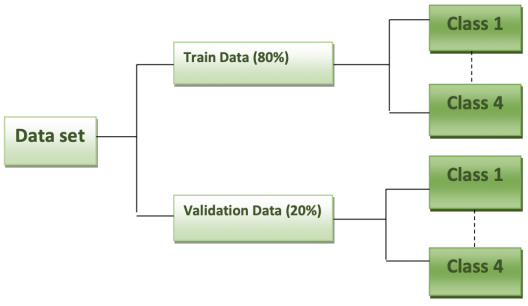


Figure 2.12: Data splitting [1].

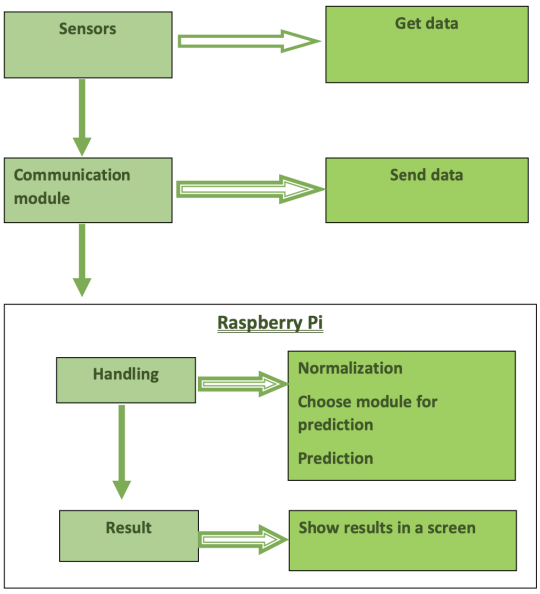


Figure 2.13: Steps of develop the device [1].

2.6 Conclusion

In conclusion, in the first article's methodology lacks detail, potentially compromising reproducibility. Insufficient discussion on model limitations and validation raises doubts about the study's reliability and real world applicability, the Second article's main objective was to create a Convolutional Neural Network model that could be used to classify and diagnose prevalent diseases that impact palm plants. The suggested model demonstrated excellent accuracy and structural simplicity, however it was unable to correctly identify the tree sections that were impacted. The third article used machine learning models to categorize DPM infestations on fruits. The study created regression models using data on fruit qualities and weather conditions, however, there was no discussion of how accurate each model was compared. The significance of early DPM infestation prediction was emphasized by the researchers, as opposed to only counting the mites.
The fourth article examined the connection between Ommatis lybicus, or the Dubas bug, infestation in Omani orchards and the density of date palm trees. The study indicated the necessity to take other factors into account in addition to the finding that tree density affects the severity of infestation. In the fifth thesis they predict the class of the mite for the current day which means only classification not early prediction before some days, Also the predicted model depend on 2 factors only (temperature and humidity).

Our proposal, presented in the next chapter, is to suggest deep learning to predict and forecast the DPM before appearing in real world, to do that, we take into account other factors such as wind speed, rain drop, soil moisture, and dew points.

Chapter 3

Design of the Suggested System

3.1 Introduction

This chapter provides an illustration of the design of our suggested system through the many design diagrams that explain how our system works. We first give a synopsis of our system's overall architecture and methods before going into detail on each component.

3.2 Context and objective

DPM, in particular, pose a serious threat to date palm harvests due to their rapid spread and potential to cause stunted growth and disastrous output reductions. Additionally, the bulk of traditional procedures are costly and take a long time because they mostly rely on experts and manuals. To make up for the lost time, they also employ a lot of chemicals. In order to intelligently monitor farms, we propose a new method in this study that is built on the ideas of fusing Iot with deep learning.
This system has the capacity to execute an expert task automatically to address this issue and anticipate the illness before it manifests. The suggested concept is to assist farmers by providing the DPM prediction period and risk level before many days of appearing in the farms.

3.3 Overall system architecture

Our architecture is composed of four entities:

3.3.1 IoT Dispositif (Station)

Data Collection Network: We will set up a network of IoT devices to collect meteorological and environmental data. This includes a station equipped with sensors and a communication module that sends the gathered data to a centralized system. Depending on the size of the farm, multiple such setups may be used.

3.3.2 Data Center and Hosted Servers

Data Management: We will establish a data center to manage and analyze the information collected by the IoT devices. The collected data will be saved on hosted servers, ensuring efficient storage and retrieval for further processing.

3.3.3 AI Models

Forecasting: Our AI models will be trained and optimized to predict the appearance of the date palm mite over an extended period. These models will process the data from the data center and provide forecasts, allowing for early intervention and prevention strategies.

3.3.4 Mobile Application (alert system)

Alert System and Visualization: We will develop a mobile application that serves as an alert system. The app will visualize data, show the progression of mite infestations through charts, and display historical data. This will enable users to monitor the situation in real-time and take preventive measures when necessary.

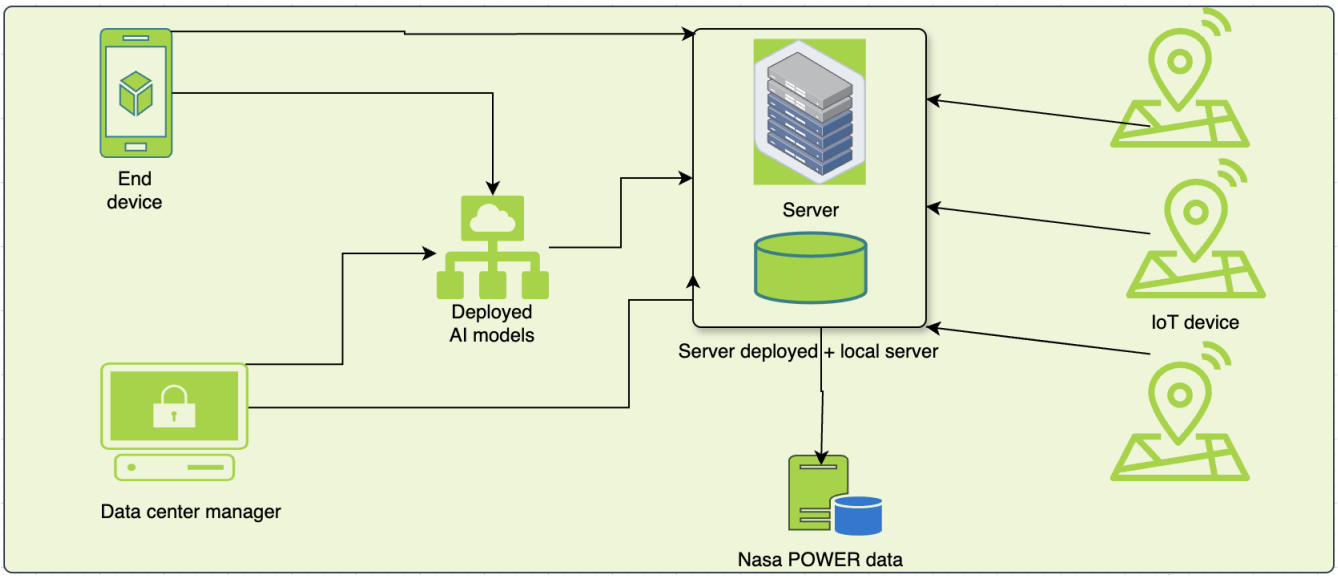


Figure 3.1: Overall system architecture.

3.4 Overall system preparation

Our system preparation involves setting up a network of IoT devices to collect environmental data, which is then managed in a centralized data center. We ensure efficient data processing with hosted servers and develop AI models for accurate forecasting. Additionally, we prepare a mobile app for real-time alerts and visualizations.

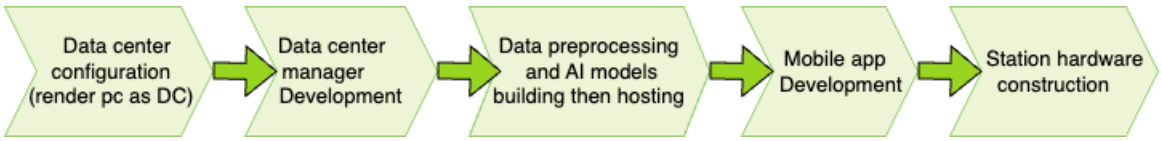


Figure 3.2: overall system preparation.

3.5 Methodology

3.5.1 Data center configuration (Packtriot)

We utilize Packtriot to configure our setup as a data center due to its robust capabilities in managing and optimizing large-scale storage solutions. Our infrastructure, characterized by extensive storage capacity, demands a system that can efficiently handle vast amounts of data while ensuring reliability, security, and seamless scalability. Packtriot meets these requirements with its advanced storage management features, high availability, and support for diverse data storage needs, making it the ideal choice to support our data-intensive operations.

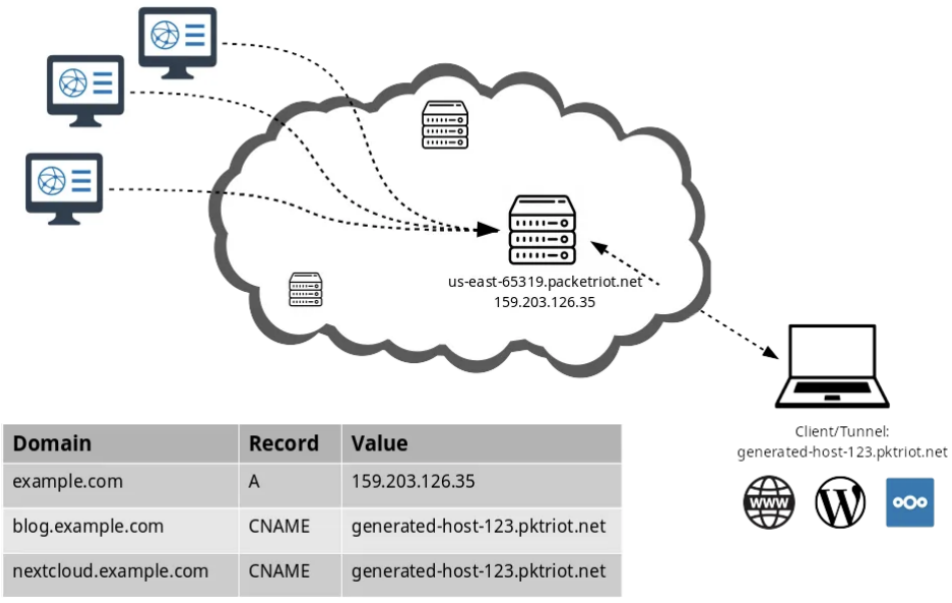


Figure 3.3: Packtriot working architecture. [2]

3.5.2 Data center manager

Our server manager is designed to efficiently handle a variety of requests, providing tailored responses based on the specific needs of each request. It supports multiple functionalities, ensuring that data is processed and returned accurately and promptly.

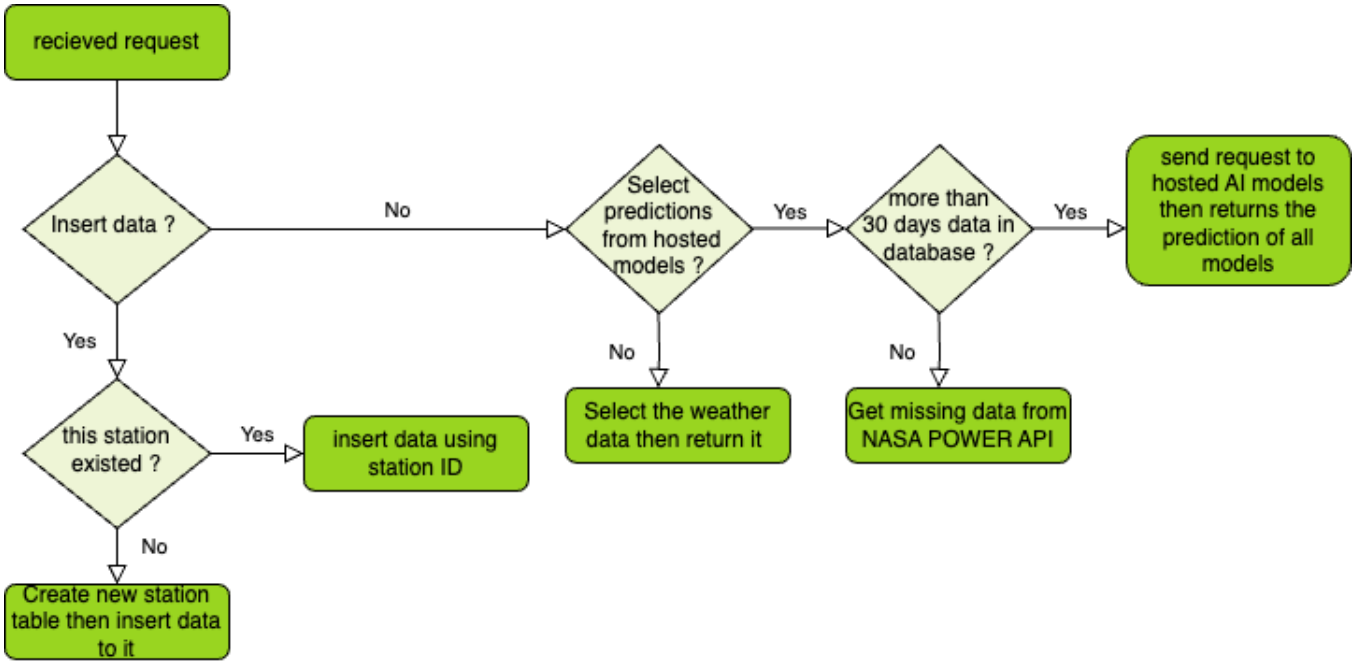


Figure 3.4: Server requests.

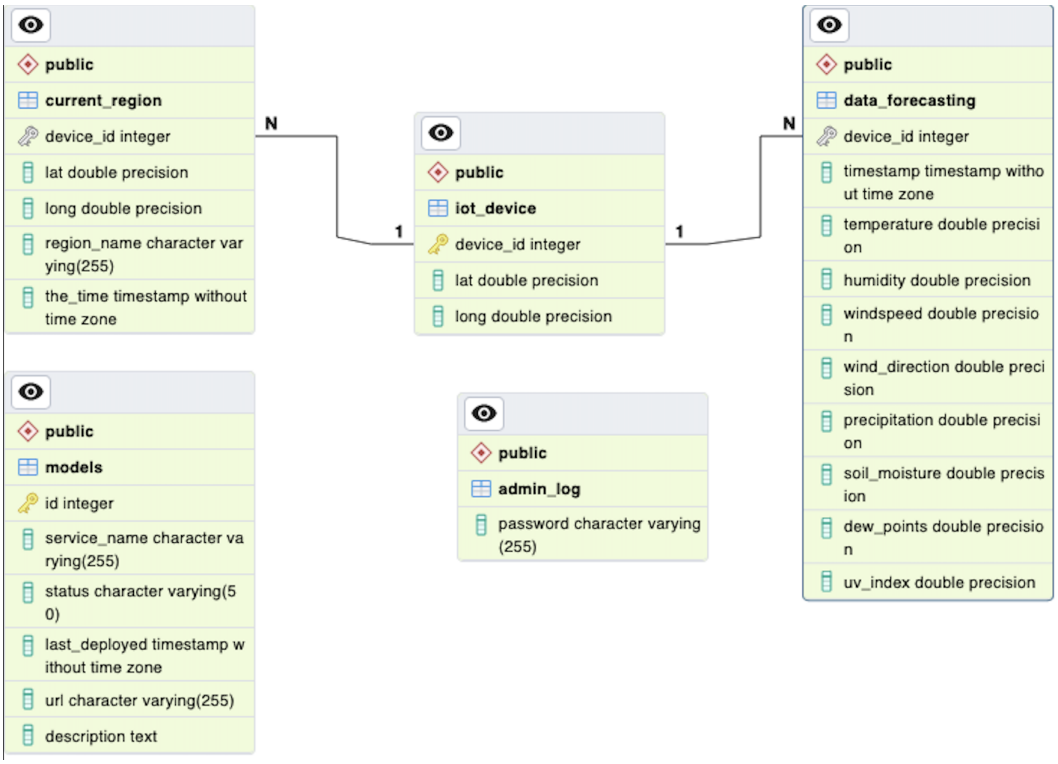


Figure 3.5: Data center manager Entity Relational Diagram.

3.5.3 Models development

Here's the steps of develop our suggested models :

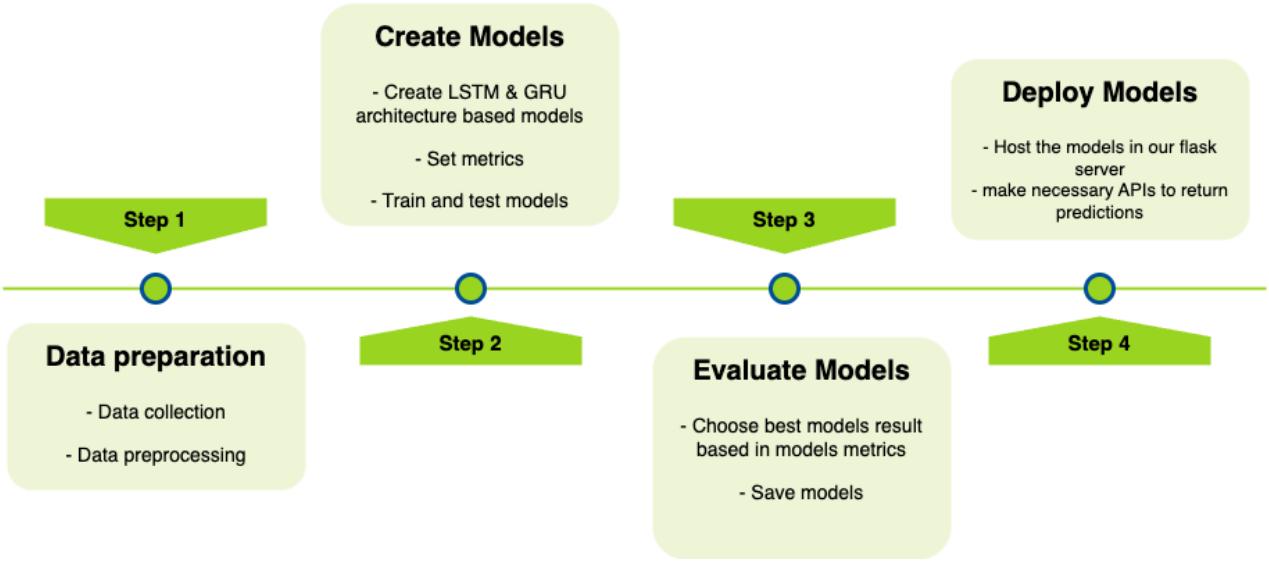


Figure 3.6: Models development.

3.5.3.1 Data preparation

a) **Data collection**

We collect our data from the NASA POWER API, and in cases where data is missing for less than 30 days from our own station, we fill in the missing data using this API, ensuring comprehensive and accurate information , here's the NASA POWER service API architecture :

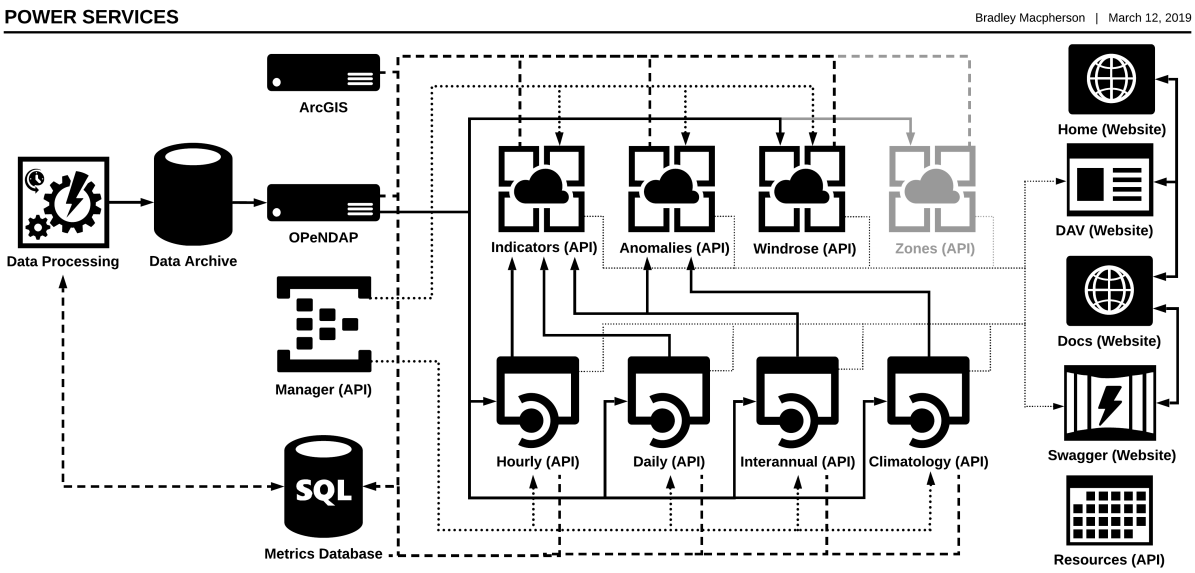


Figure 3.7: NASA POWER service API architecture.

b) **Data Preprocessing**

Preprocessing, which includes a variety of methods for getting raw data ready for analysis, is an essential stage in creating reliable AI models. In order to maintain uniformity, this process entails cleaning the data to remove inconsistencies, addressing missing values, and normalizing or scaling characteristics.

Also in the preprocessing phase of our AI project, we meticulously prepared the time series data by dividing it into segments to optimize model training. Specifically, we created three distinct models, each utilizing different data splitting strategies: the first model trained on 30-day intervals with targets set 7 days ahead, the second on 30-day intervals with 15-day targets, and the third on 30-day intervals with 30-day targets.

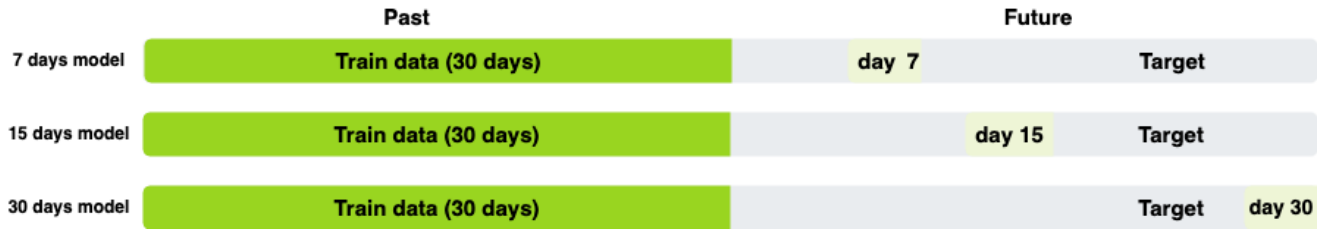


Figure 3.8: A Single Segment for every model.

3.5.3.2 Create models network architectures

We have made 3 network architectures based on the previous 30 days in reason of life cycle of this insect after discussing with an expert in the field of palm trees insects ,first model for 7 days early prediction , and another for 15 days ahead , and the last 30 days far.

However, this data may not be available for some farms who using late this device.

Therefore, we used NASA POWER API for recuperation data recuperation.

We use Bi-LSTM in the model of 7 days early prediction because Bi-LSTM can deal with time-series data .

But for the model of 15 days we used Stacked-GRU also good for time series data and 30 days early prediction we used Bi-GRU which can deal with time-series data too and they are computationally more efficient than LSTM networks.

Table 3.1: Layers of our 7 days early prediction model.

Layer type	**Output Shape**	**Activation**
Normalization	30∗14	-
Bi-LSTM	180∗30	-
Dropout	180∗30	-
Bi-LSTM	180∗30	-
Dropout	180∗30	-
Bi-LSTM	90∗30	-
Dropout	90∗30	-
Bi-LSTM	90	-
Dense	1	-

Table 3.2: Layers of our 15 days early prediction model.

Layer type	**Output Shape**	**Activation**
Normalization	30∗14	-
GRU	96∗30	-
Dropout	96∗30	-
GRU	96∗30	-
Dropout	96∗30	-
GRU	96∗30	-
Dropout	96∗30	-
GRU	96	-
Dense	1	-

Table 3.3: Layers of our 30 days early prediction model.

Layer type	Output Shape	Activation
Normalization	30∗14	-
BiGRU	120∗30	-
Dropout	120∗30	-
BiGRU	120∗30	-
Dropout	120∗30	-
BiGRU	60∗30	-
Dropout	60∗30	-
BiGRU	60	-
Dense	1	-

LSTMs are a unique class of neural networks that outperform RNNs in terms of performance, while also addressing some of the major issues with RNNs related to vanishing gradients and long-term dependencies. The best use case for LSTMs is long-term dependency; you will see later how they get around the issue of vanishing gradients.

[3].

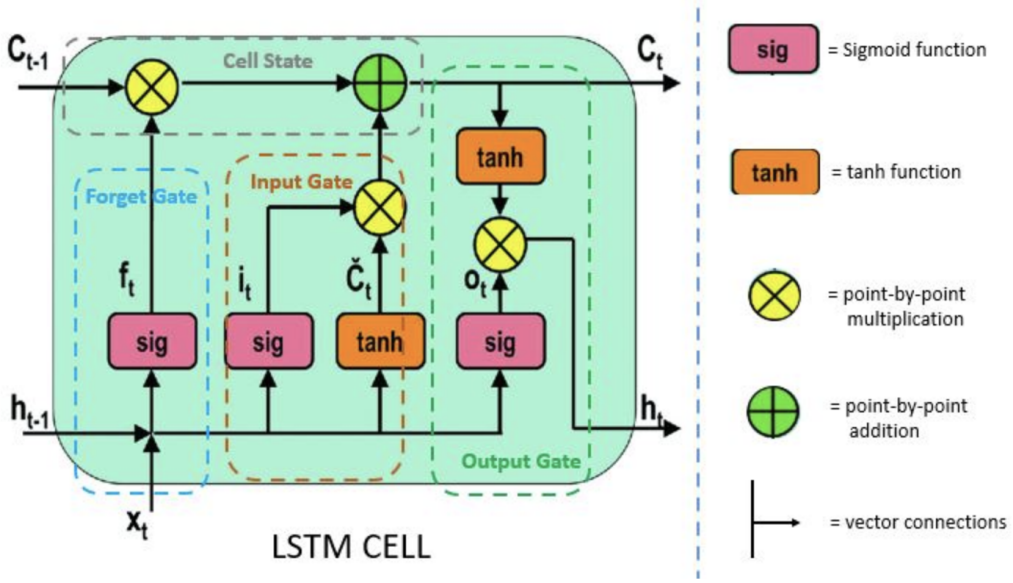


Figure 3.9: General LSTM architecture [3].

The forget gate decides which information needs attention and which can be ignored. The information from the current input Xt and hidden state H_{t-1} are passed through the sigmoid function. Sigmoid generates values between 0 and 1. It concludes whether the part of the old output is necessary (by giving the output closer to 1). This value of Ft will later be used by the cell for point-by-point multiplication [3].

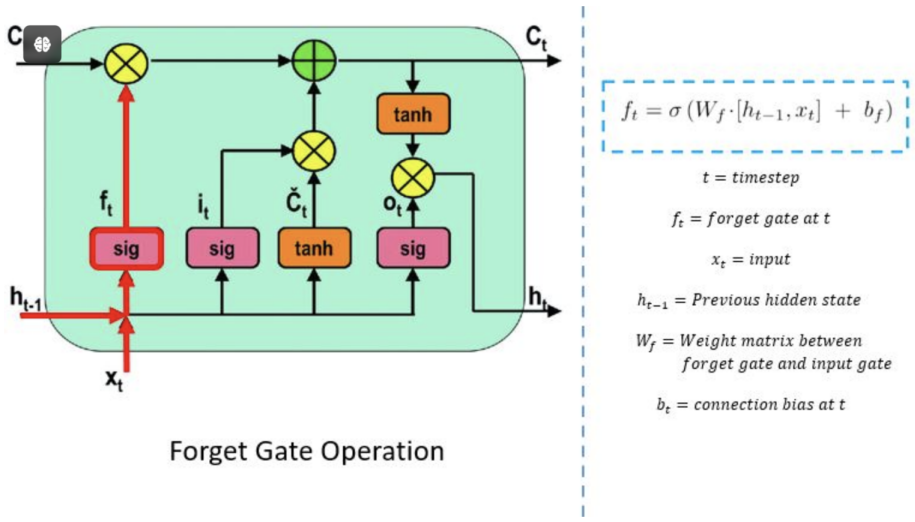


Figure 3.10: Forget gate [3].

The input gate performs the following operations to update the cell status.

First, the current state Xt and previously hidden state H_{t-1} are passed into the second sigmoid function. The values are transformed between 0 (important) and 1 (not-important). Next, the same information of the hidden state and current state will be passed through the tanh function. To regulate the network, the tanh operator will create a vector $\tilde{C}_t$ with all the possible values between -1 and 1. The output values generated form the activation functions are ready for point-by-point multiplication [3].

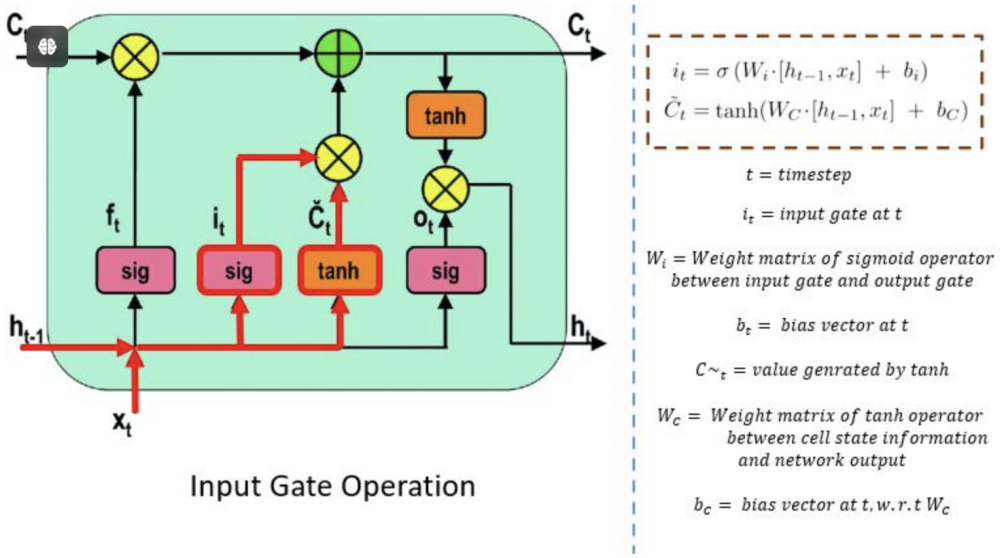


Figure 3.11: input gate operation. [3].

The network has enough information form the forget gate and input gate. The next step is to decide and store the information from the new state in the cell state. The previous cell state C_{t-1} gets multiplied with forget vector f_t . If the outcome is 0, then

values will get dropped in the cell state. Next, the network takes the output value of the input vector i_t and performs point-by-point addition, which updates the cell state giving the network a new cell state C_t [3].

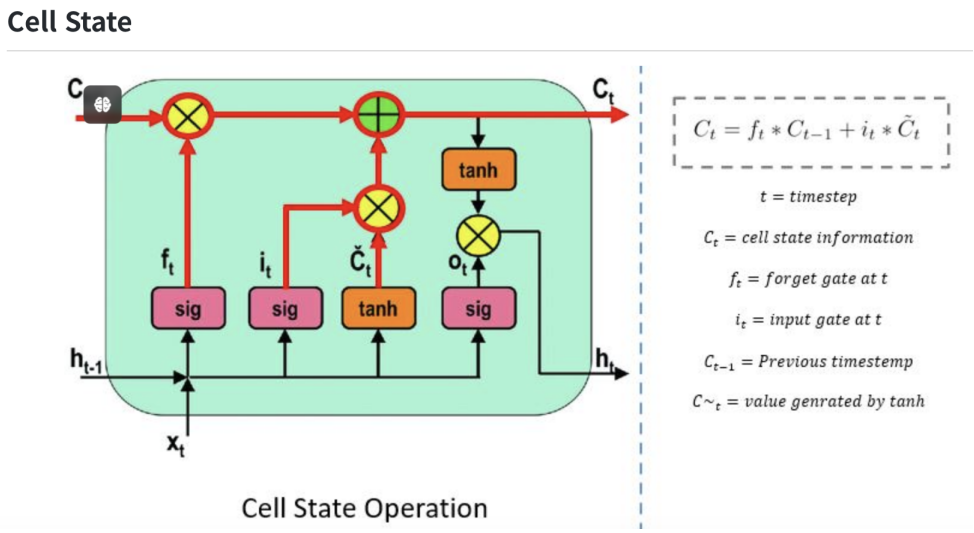


Figure 3.12: Cell state.

The value of the subsequent hidden state is determined by the output gate. Previous input information is contained in this state. The third sigmoid function receives the values of the previous concealed state and the current state first. After that, the tanh function is applied to the newly created cell state that resulted from the cell state. The multiplicity of these two outputs is done point-by-point. The network determines what data the hidden state should contain based on the final value. The purpose of this hidden state is prediction.

Finally, the new cell state and new hidden state are carried over to the next time step.

The new concealed state and new cell state are then carried over to the following time step. In summary, the forget gate decides whatever pertinent data from the earlier stages is required. The output gates complete the next hidden state, while the input gates determine what pertinent data can be supplied from the previous stage. [3].

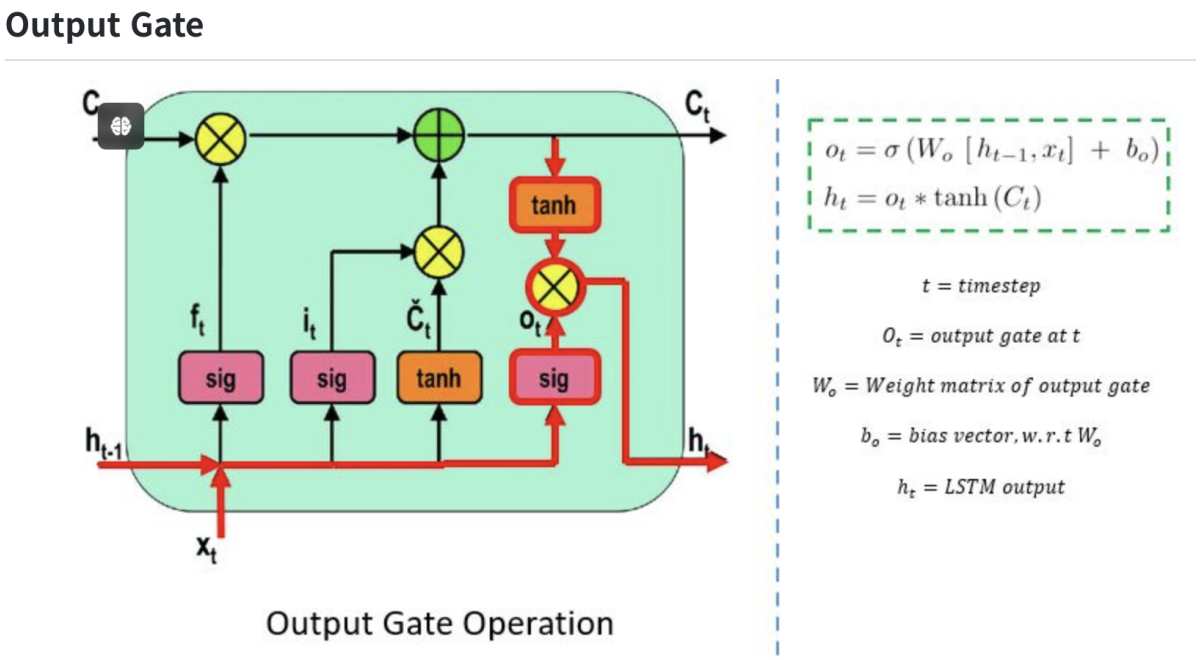


Figure 3.13: Output gate equation.

- Gated Recurrent Units are similar to the LSTM networks. GRU is a kind of newer version of RNN [3].

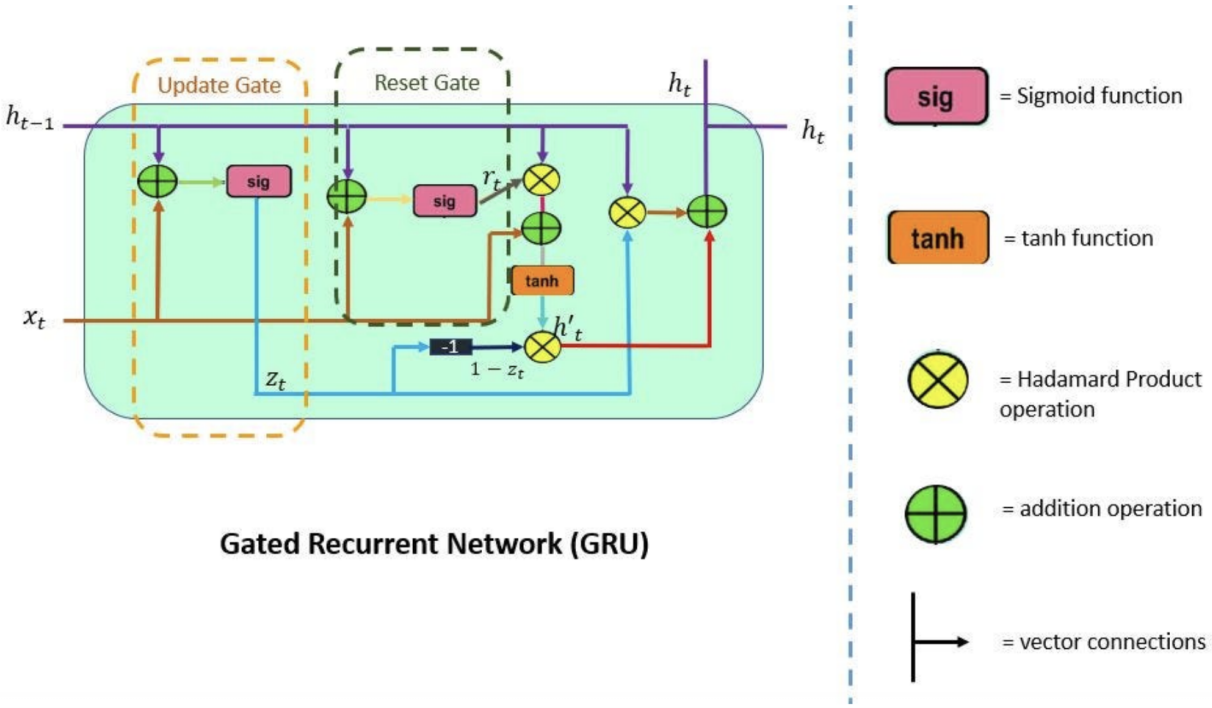


Figure 3.14: GRU Architecture.

- Update Gate is a combination of Forget Gate and Input Gate. Forget gate decides what information to ignore and what information to add in memory [3].

- Reset Gate Resets the past information in order to get rid of gradient explosion. Reset Gate determines how much past information should be forgotten [3].

- Paradigms of various LSTM networks

a. **Simple LSTM network (Vanilla)** The Simple LSTM can be thought of as a basic LSTM network architecture where only a single hidden LSTM Layer is being used followed by a standard feed forward output layer. [5].

b. **Stacked LSTM Model** Stacked LSTM architectures with multiple hidden layers can build up progressively higher level of abstractions. They provide a more ideal representation of sequence data, and thus, work more effectively. In Stacked LSTM architecture, current LSTM hidden layer is fed with input from previous LSTM hidden layer. This can substantially increase the performance of neural networks [4].

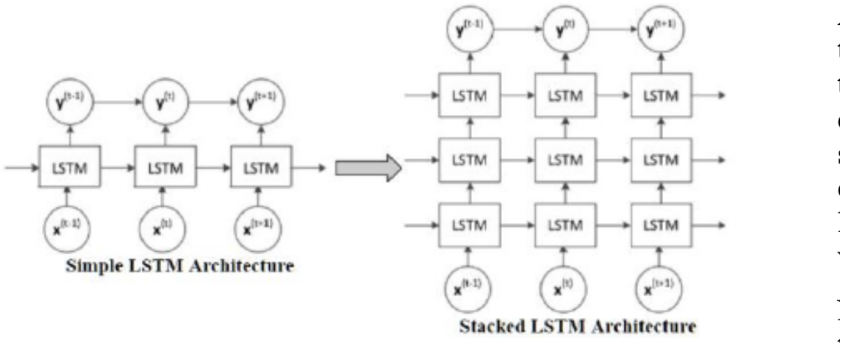


Figure 3.15: Simple lstm vs stacked lstm [4].

c. **Bi-directional LSTM (BiLSTM)** is a deep learning algorithm used for time series data forecasting. BiLSTM enhances the traditional LSTM algorithm by incorporating hidden states from both forward and backward layers, allowing it to process information in both directions. This makes BiLSTM particularly effective in contexts where input context is crucial. It is extensively used in tasks such as text classification .[5].

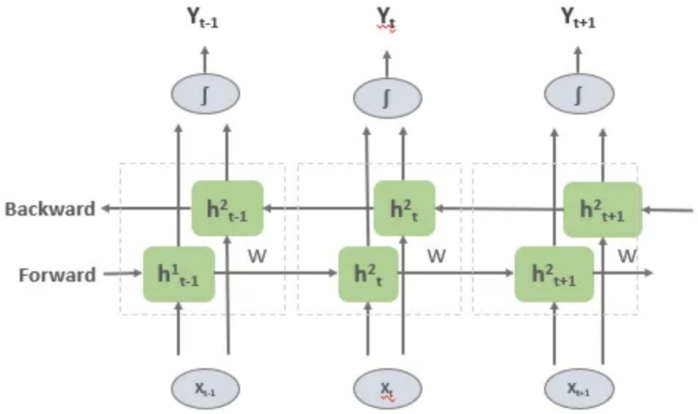


Figure 3.16: Bi-LSTM architecture [5].

3.5.3.3 Models deployment and actions.

After creation our 3 models we deploy then on Render cloud and we intended to separate our two servers from individually for better practice.

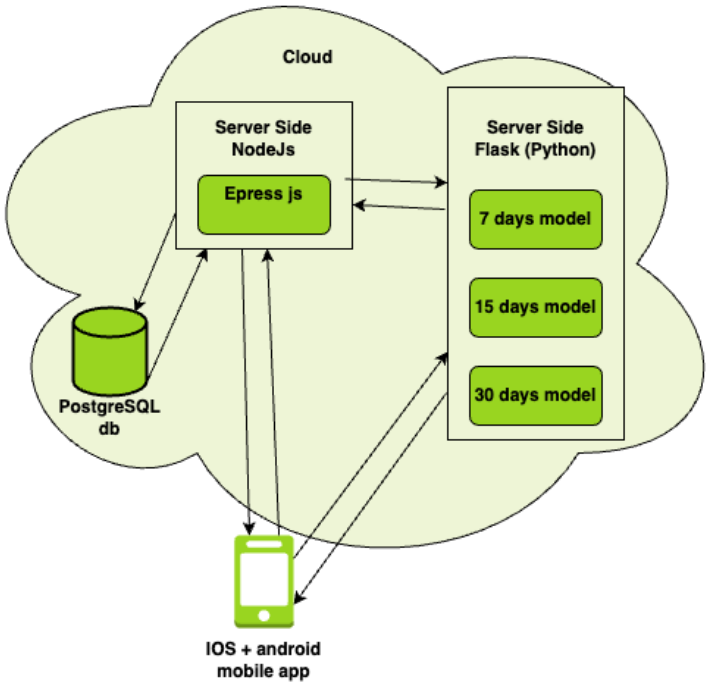


Figure 3.17: models deployment.

3.5.4 Mobile application development

To facilitate data visualization and provide timely alerts to farmers, including notifications about material issues, we developed an efficient mobile application designed to enhance their decision making processes.

3.5.4.1 Framework used to build our mobile application

We used Flutter which is an open-source UI software development toolkit created by Google. It enables developers to build natively compiled applications for mobile, web, and desktop from a single codebase. With its rich set of pre-designed widgets, fast performance powered by the Dart language, and a robust architecture for state management, Flutter allows for high productivity and expressive, flexible UIs. It streamlines development processes by facilitating seamless cross-platform app creation, ensuring consistent performance and a unified user experience across different devices.

3.5.4.2 Architecture used to develop our mobile application

In our Flutter app, we implemented the MVVM architecture, which is derived from Clean Architecture principles. Given the app's relatively small size, MVVM provided a streamlined and efficient solution without the additional complexity of full Clean Architecture.

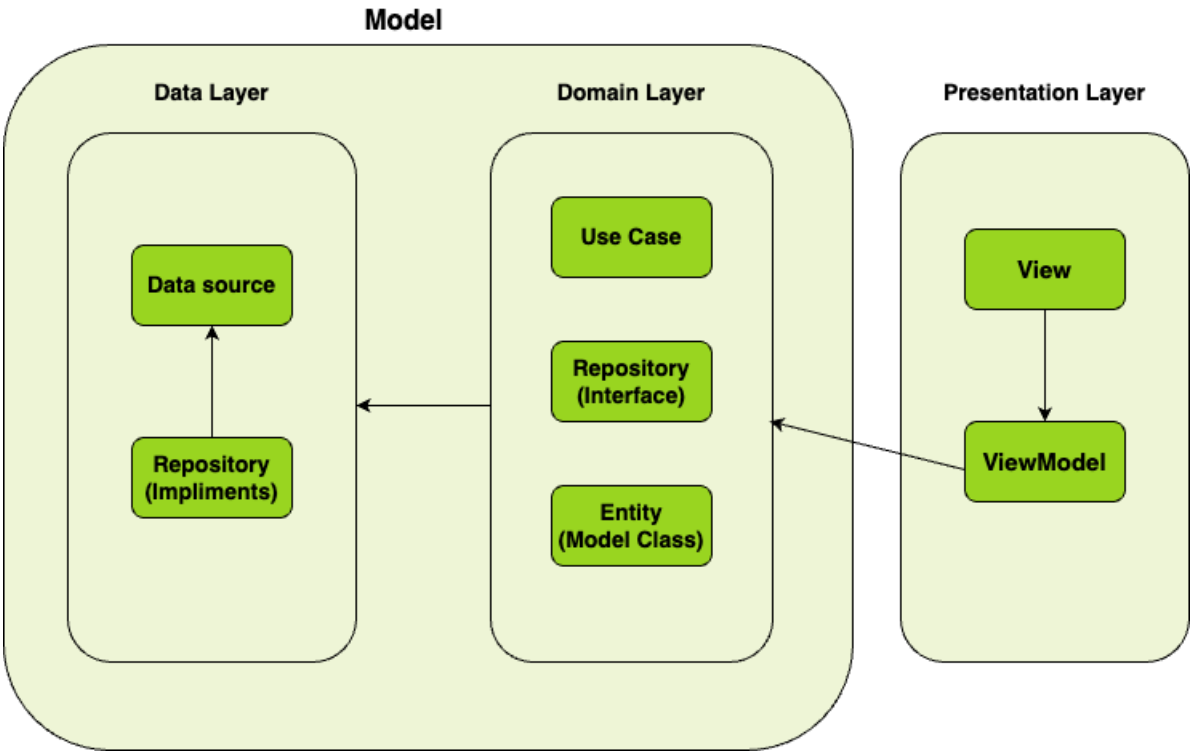


Figure 3.18: MVVM with Clean Architecture.

In our Presentation Layer in ViewModel phase we used Bloc as a state management in our application.

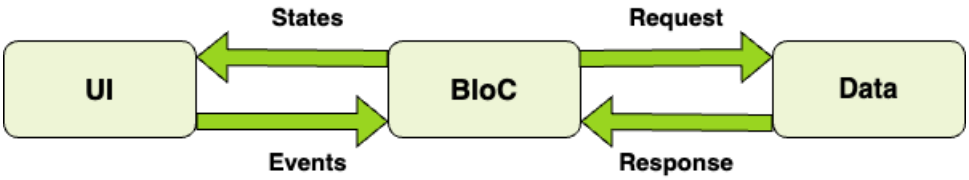


Figure 3.19: BloC state management Architecture.

3.5.5 Hardware construction (Our stations)

All our IoT stations, autonomously powered by a battery and solar panel, includes temperature, humidity, wind speed,wind direction, anemometer, soil moisture sensors, a GSM module, and GPS to get the current coordinates (latitude and longitude). It diligently collects all data, which is transmitted over time with precision to our server via an HTTP request over the internet provided by a SIM card.

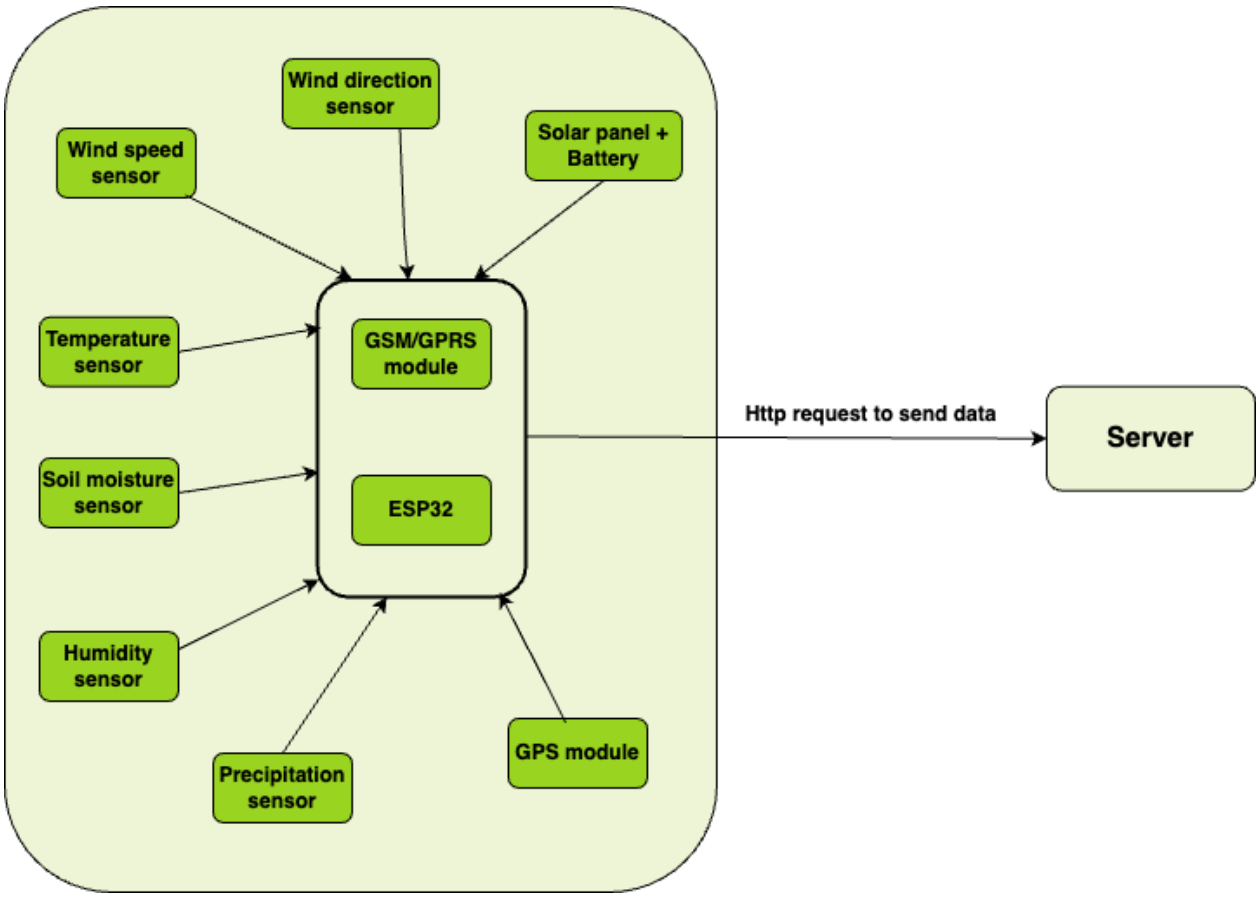


Figure 3.20: Station Architecture.

3.6 UML diagrams

3.6.1 Class diagrams

3.6.1.1 Model class diagram

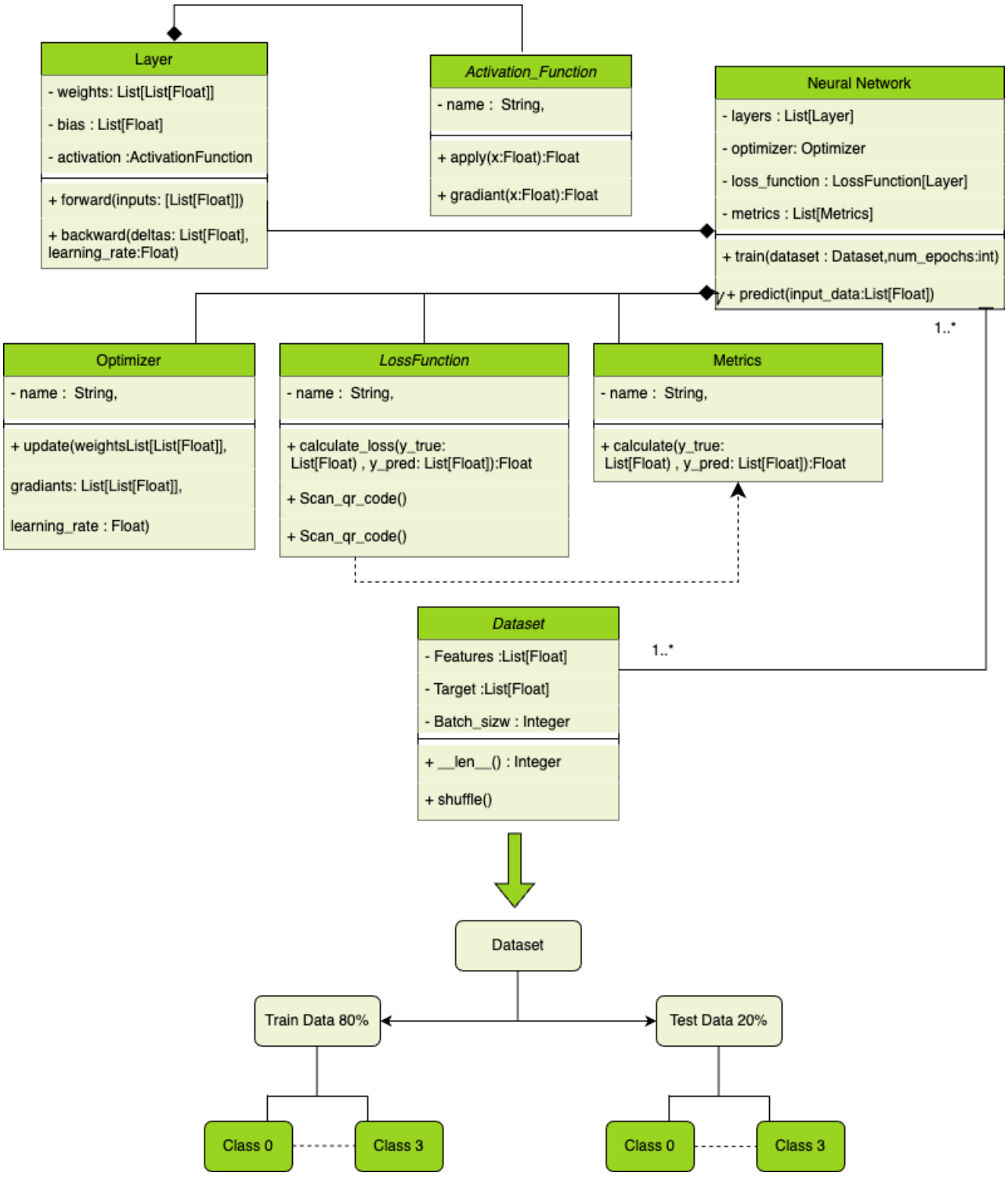


Figure 3.21: Our model class diagram.

3.6.1.2 System's general class diagram

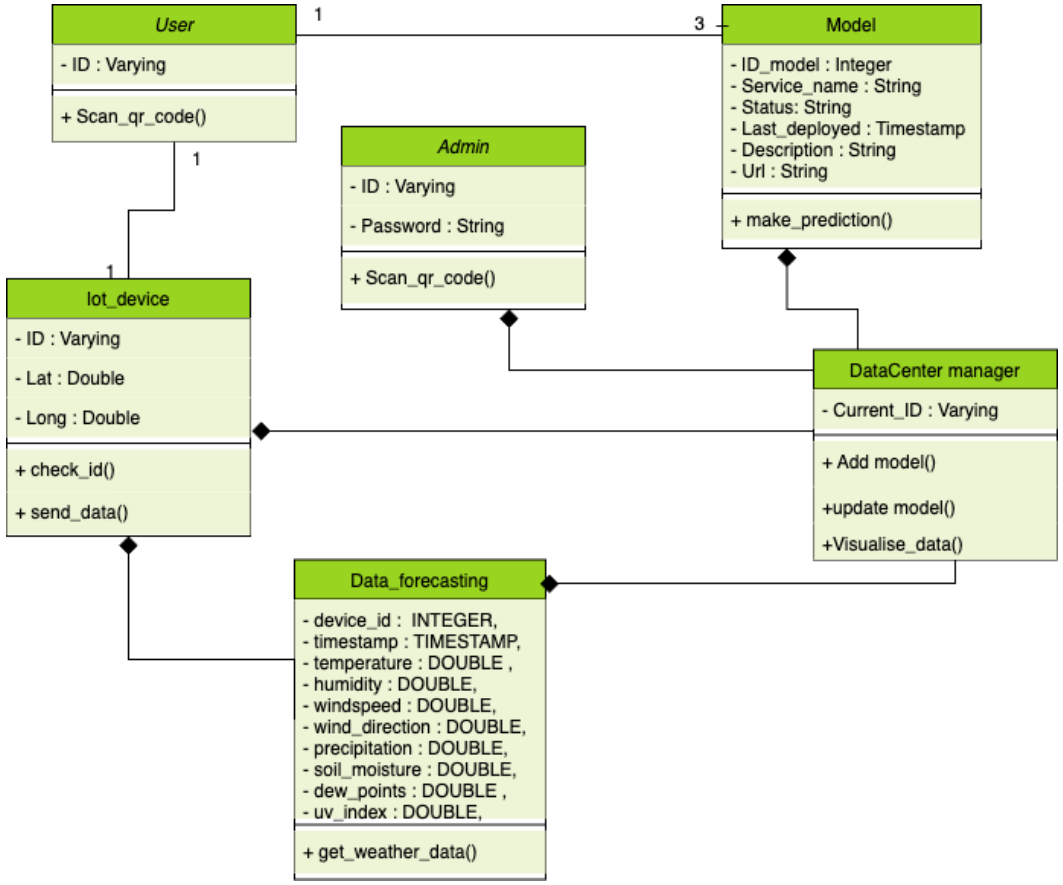


Figure 3.22: System's general class diagram.

3.6.2 Sequence diagrams

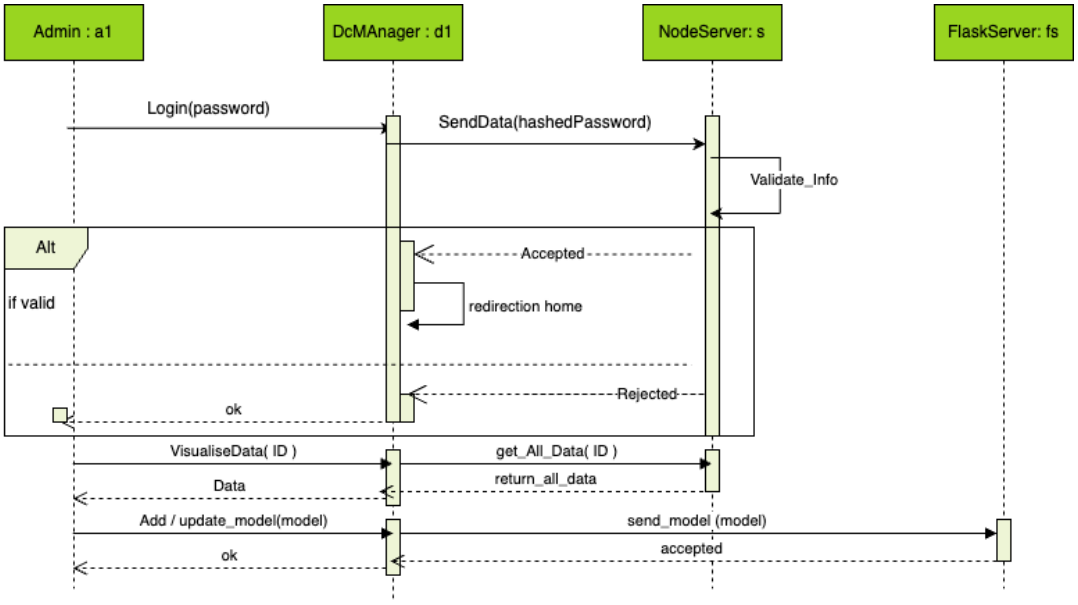


Figure 3.23: Data center manager admin's sequence diagram.

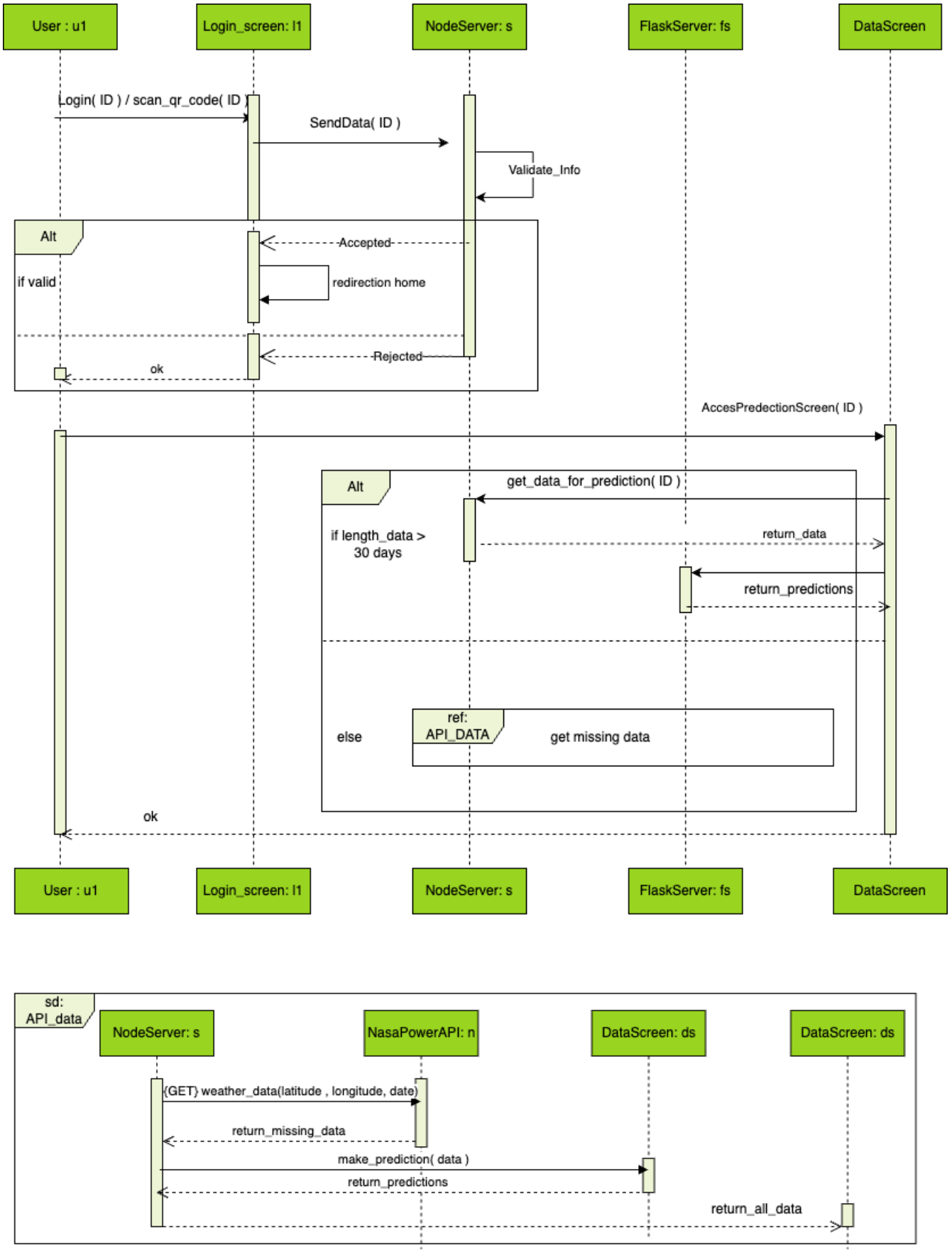


Figure 3.24: System's sequence diagram.

3.6.3 Use case diagram

Figure 3.25: System's use case diagram.

3.7 Conclusion

This chapter has covered the design of our system, including the use case, sequence, and class diagrams, as well as the overall architecture and functioning of the system. We have also outlined the approach used for the many components that comprise the system.
Our system's implementation will be covered in the upcoming chapter. The mobile app, software, hardware, AI models environment, and implementation strategies will all be covered in detail.

Chapter 4

Coding, Experiments and Results

4.1 Introduction

This chapter aims to show you how we put our early prediction system into practice.
First, we will describe the environment components of our suggested hardware and software, followed by a description of the dataset and the details of the development of our models.
In the next section we will shows the deployments details of our software and AI models.
This chapter will conclude with a comparison of the system's execution results to those of other systems and a display of the results.

4.2 Software tools

In our system, we utilized HTML, CSS, JavaScript, and Bootstrap for the data center manager interface. For the backend, we implemented Node.js and PostgreSQL to manage our database operations.
Our AI models were developed using Python, TensorFlow, and Keras, and deployed on a specialized server using Flask.
We leveraged GitHub and Render services for deploying the data center manager and PostgreSQL database.
Packtriot was employed to configure the data center on a local laptop. For our mobile application, we used the Flutter framework with Dart.

4.2.1 Programming languages, libraries and frameworks

In our system we used as a programming languages, frameworks and libraries :

4.2.1.1 JavaScript

JavaScript is an interpreted language, not a compiled language. Developers use to make interactive webpages. From refreshing social media feeds to displaying animations and interactive maps, JavaScript functions can improve a website's user experience [22].

Figure 4.1: JavaScript logo.

4.2.1.2 Node.js

Node.js ® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts [22].

Figure 4.2: Node JS logo.

4.2.1.3 Bootstrap

Bootstrap is an open-source front-end framework for creating responsive, mobile-first web applications. It includes pre-designed HTML, CSS, and JavaScript components that streamline development and ensure consistent design across devices.

Figure 4.3: Bootstrap logo.

4.2.1.4 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together [23].

Figure 4.4: Python logo.

4.2.1.5 Keras

Keras is a deep learning API written in Python and capable of running on top of either JAX, TensorFlow, or PyTorch.
Keras is simple , flexible , powerful [24].

Figure 4.5: Keras logo.

4.2.1.6 Flask

Python Flask is a web framework. Because it contains an integrated database and doesn't require any specific tools or libraries, it falls within the micro framework category [25].

Figure 4.6: Flask logo.

4.2.1.7 Flutter & Dart

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. Flutter is fask , productive , flexible [26].
Flutetr based on Dart which is an approachable, portable, and productive language for high-quality apps on any platform [27].

Figure 4.7: Flutter and Dart logo.

4.2.2 Databases

In our system we used local and global databases sush as :

4.2.2.1 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads [28].

Figure 4.8: PostgreSQL logo.

4.2.2.2 SqfLite

Sqflite is a popular package for implementing SQLite databases in Flutter. It provides a simple and efficient way to store and retrieve data from local storage, making it ideal for mobile applications that require offline data access [**?**].

4.2.3 Cloud and deployment

In our system we used some platforms to deploy our servers and models:

4.2.3.1 Github

GitHub is a web-based hosting service that allows developers to create, host, and share their computer code.

Figure 4.9: Github logo.

4.2.3.2 Render

Render is a unified cloud to build and run all your apps and websites with free TLS certificates, global CDN, private networks and auto deploys from Git [**?**].

Figure 4.10: Render logo.

4.2.3.3 Packtriot

Packetriot provides a cloud-based edge network that clients connect to using a secure reverse tunneling protocol that can connect applications or devices on local or private networks to the Internet [29].

Figure 4.11: Packtriot logo.

4.3 Hardware

4.3.1 LILYGO T-SIMA7670SA R2 ESP32

The LILYGO T-SIMA7670SA R2 ESP32 board is a versatile development board designed for IoT applications, combining the power of the ESP32 microcontroller with LTE connectivity provided by the SIMA7670SA module.

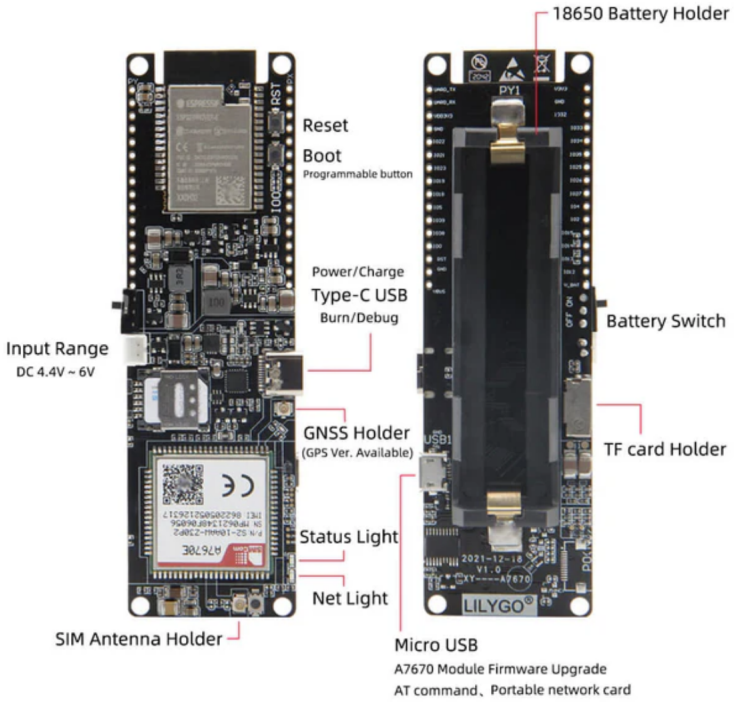


Figure 4.12: LILYGO T-SIMA7670SA R2 ESP32.

- The Features :

- Supply voltage: 3.3V DC or 5V DC
- ESP32 chip (WROVER-B Module) (240MHz dual-core processor)
- Flash memory: 4MB
- PSRAM: 8MB
- SRAM: 520KB
- Built-in Wi-Fi
- Built-in Bluetooth
- USB to serial converter: CP2104 or CH9102 (drivers)
- Built-in SIMA7670SA module
- Built-in nano SIM card slot
- Built-in SIM antenna slot
- Built-in GPS antenna slot
- Built-in Li-ion/Li-Po battery charging circuit:

o DW01A battery protection IC o CN3065 solar energy charging interface for 4.4-6.8V solar panel o Built-in 1x 18650 battery holder o Built-in solar panel connector 2p JST-PH

- Built-in Micro SD card slot
- Built-in on/off switch.

4.3.2 DHT22 sensor

The DHT22 sensor is a popular digital sensor used for measuring temperature and humidity. It is widely used in various electronics and IoT projects due to its accuracy and ease of use. Below is a detailed overview of the DHT22 sensor, this sensor is capable of measuring temperature with a range of -40°C to +80°C and accuracy of ±0.5°C

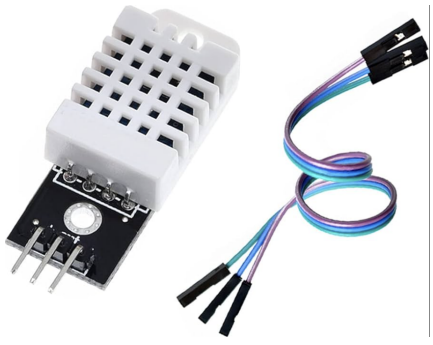


Figure 4.13: DHT22 sensor .

4.3.3 Module Sensor rain water drop detector Pic Arduino FC-37 YL-83

The FC-37 (also known as YL-83) is a rainwater drop detection sensor module commonly used with microcontrollers like Arduino. This sensor can detect the presence of water droplets, making it useful for weather monitoring, irrigation systems, and other environmental sensing applications.

- To get the rain-drop in Milliliter: rainSensorValue= 0 , dryValue= 0 , wetValue= 1023 , maxMilliliters = 100

$$\text{rainMilliliters} = \left(\frac{wetValue - dryValue}{rainSensorValue - dryValue} \right) maxMilliliters$$

Figure 4.14: Module Sensor rain water drop detector .

4.3.4 Capacitive soil moisture sensor v1.2

This sensor used to measure the moisture level in soil. Unlike resistive soil moisture sensors, which can corrode over time due to direct exposure to soil, the capacitive sensor uses capacitive sensing to measure the dielectric permittivity of the surrounding medium, which is correlated with soil moisture.

Figure 4.15: Capacitive soil moisture sensor.

4.3.5 Module GPS GY-NEO6MV2 V2

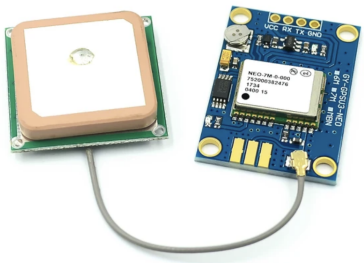


Figure 4.16: Module GPS GY-NEO6MV2 V2.

4.3.6 Solar panel 6v 0.8w and 3.7 V 2200 mAh rechargeable lithium-ion battery

Figure 4.17: Solar panel 6v 0.8w

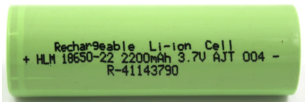


Figure 4.18: Caption for the right image

Figure 4.19: 3.7 V 2200 mAh rechargeable lithium-ion battery

- For **wind speed** and **wind direction**, we utilized the OpenWeatherMap API to fetch current data. this involved sending an HTTP request with the latitude and longitude of the location to obtain the current values.

4.4 Implementation

We will discuss the implementation of each part of our system in details :

4.4.1 Data center configuration and its UI manager and deployment

- First we started configuring data center using patriot by making in our laptop

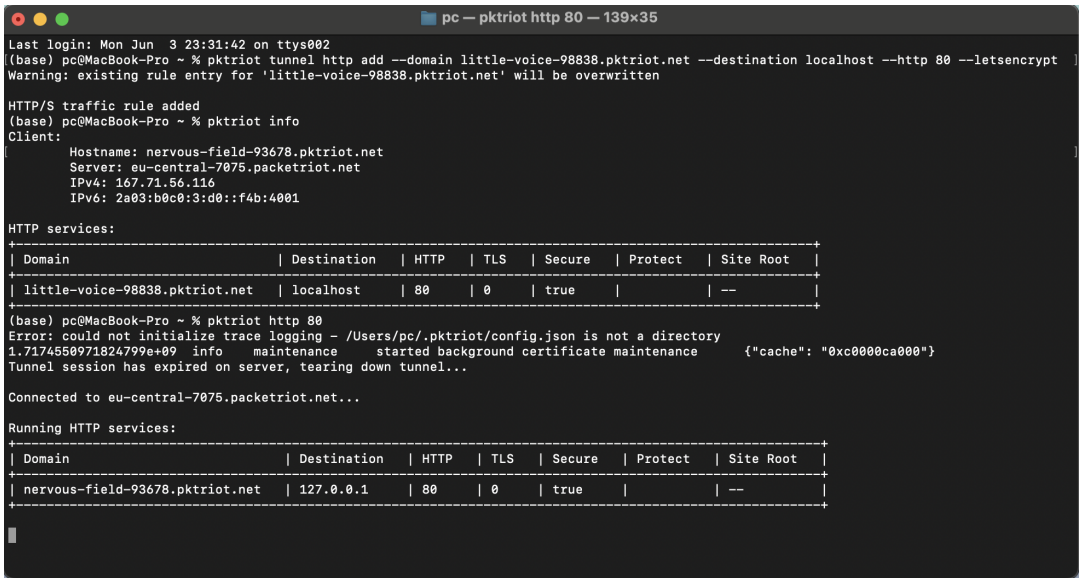


Figure 4.20: Packtriot configuration .

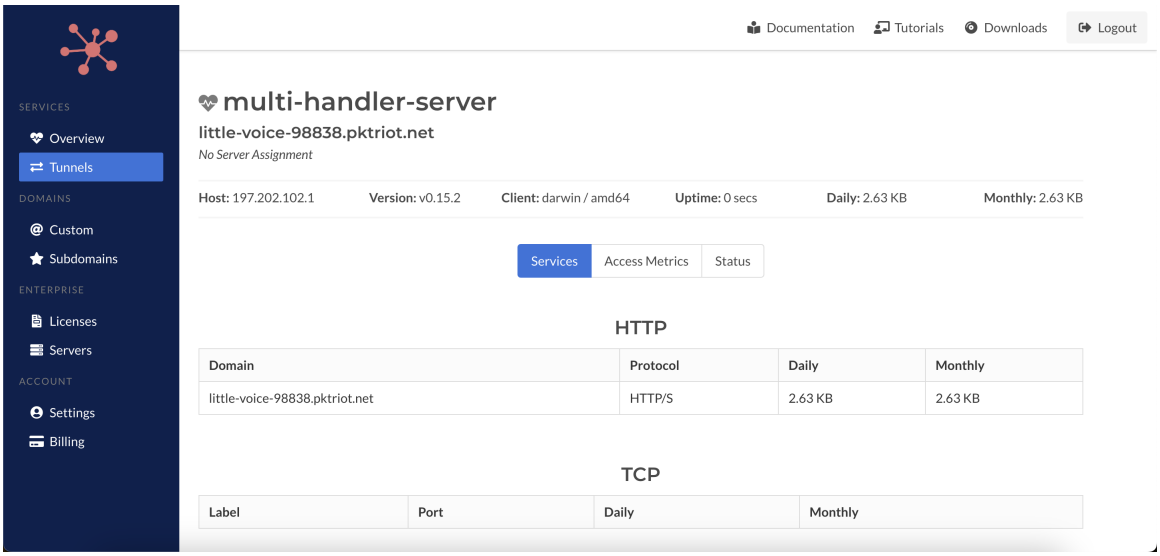


Figure 4.21: Packtriot dashboard .

- We reliate our node JS server with packtriot configuration, the link : **"nervous-field-93678.pktriot.net"** is used to visit our data center manager

Figure 4.22: Node server .

Here's our PostgreSQL database configuration , which has admin log table which has admins informations ,and iot device table which has all the informations of every iot device , and current region table which has the historic of every iot device and current placement , and data table which has all the weather data of each iot device , also we have models table, which has the all information about the hosted AI models and the server linked to it.

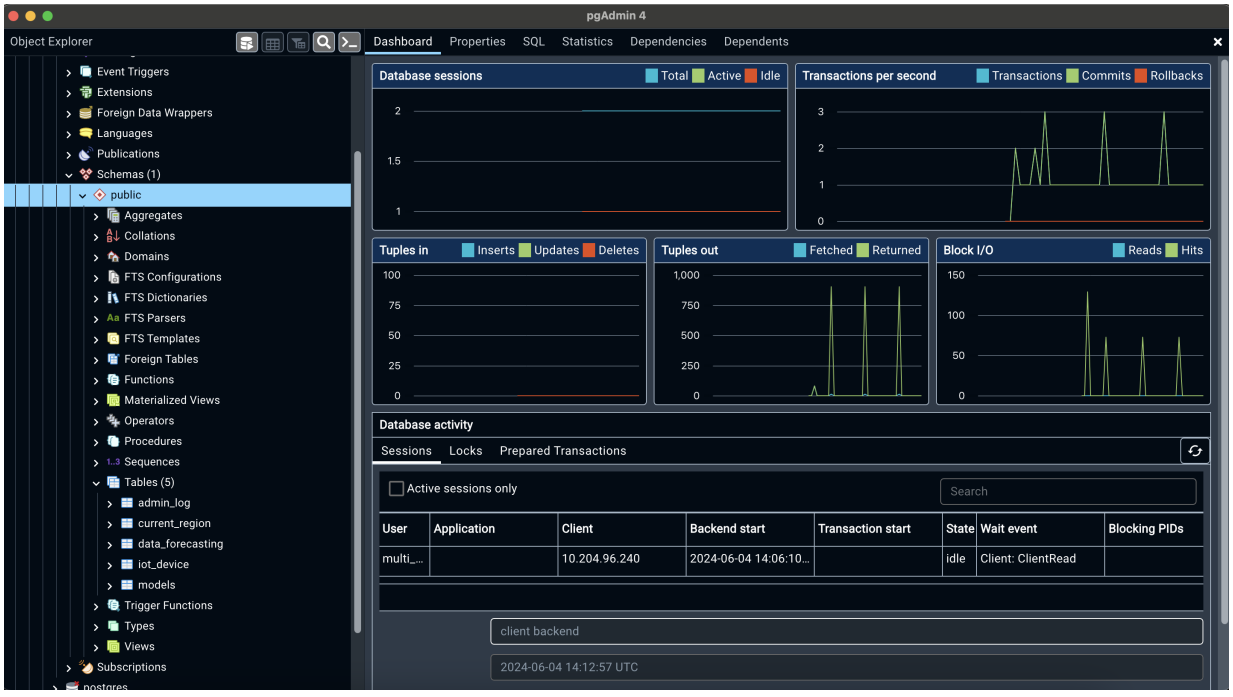


Figure 4.23: Postgresql admin config .

In our data center manager we have 16 pages ,all pages informations related to one iot device (station),so this is a dynamic platform , it can be used in several plant researches and disease where its data can be used for statistics and studies and building AI models and so on ...

- Let's start with Admin login page : we used JWT to insure that only admins can access to other pages.

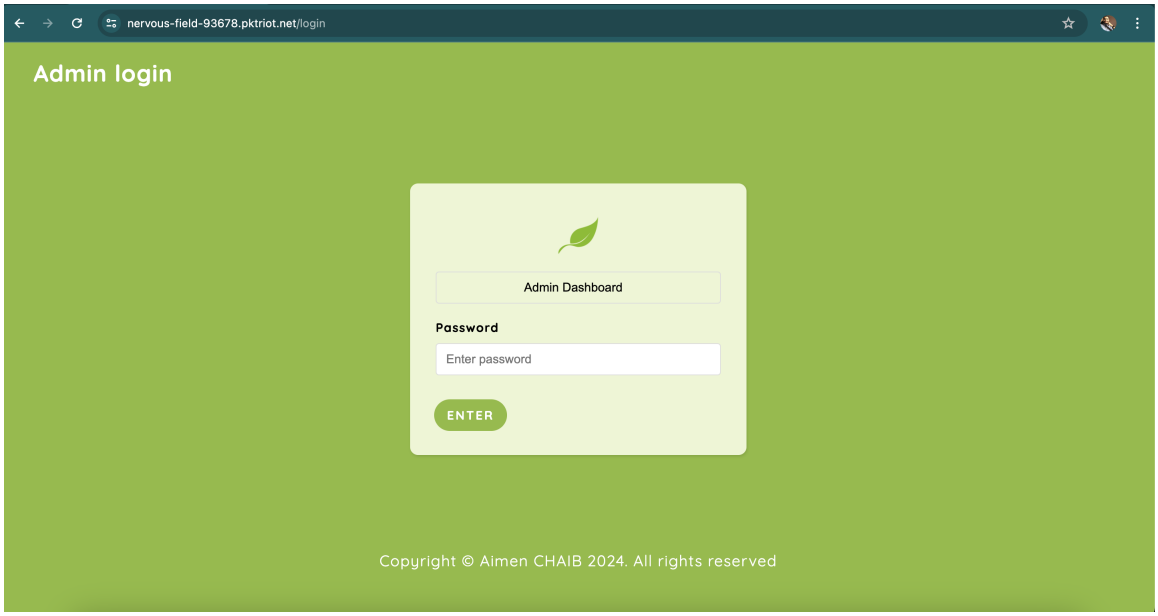


Figure 4.24: Admin login page .

- In the home page we have map which has all coordinates of the implemented Iot devices , and its informations , it's selectable:

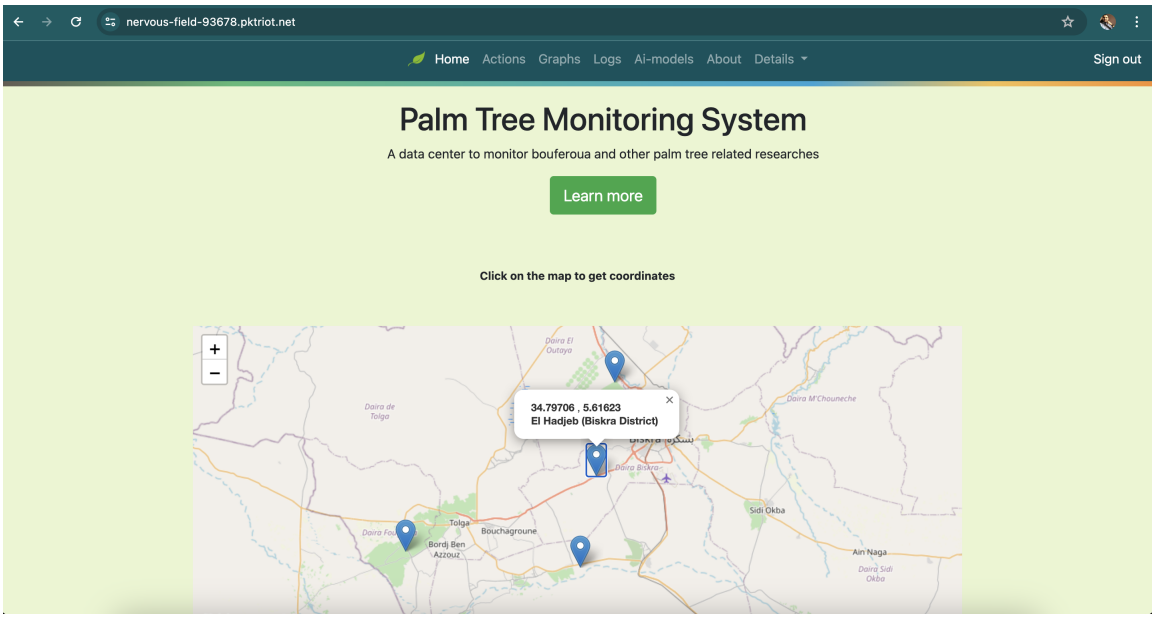


Figure 4.25: Home page .

Here you can search for any Iot device in the datacenter then you can select the devices you want to refresh the datacenter manager to bring all data of it , the data distributed automatically in all pages.

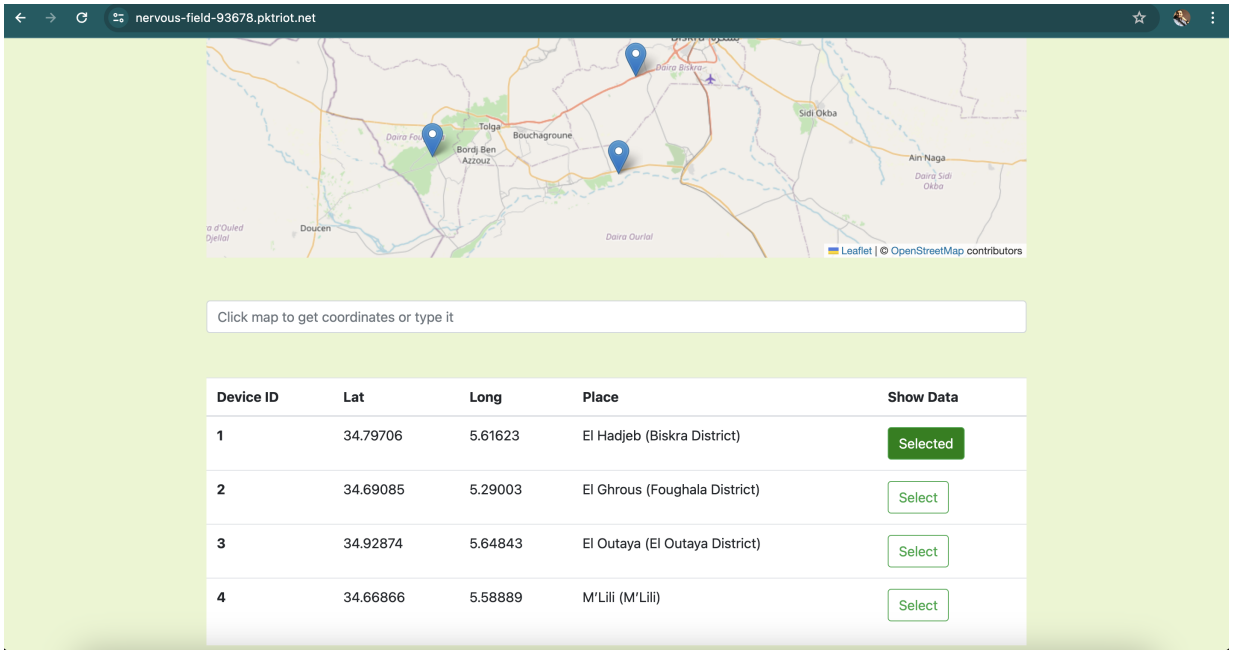


Figure 4.26: Home page (follows) .

- Here you can see all current hardware status related to selected Iot device

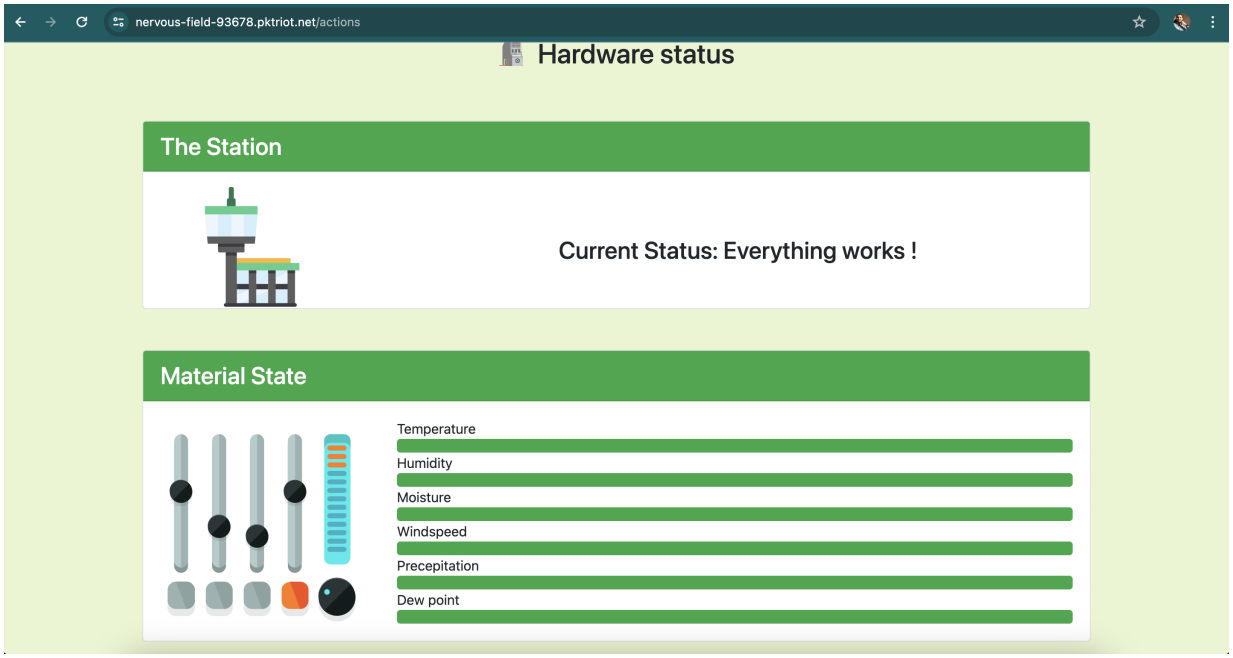


Figure 4.27: Hardware status page .

- In the graphs page you can see the latest values recorded by the remote

weather station are used to plot the graph. The graph is updated every 2 hours to reflect the most recent state of the system.

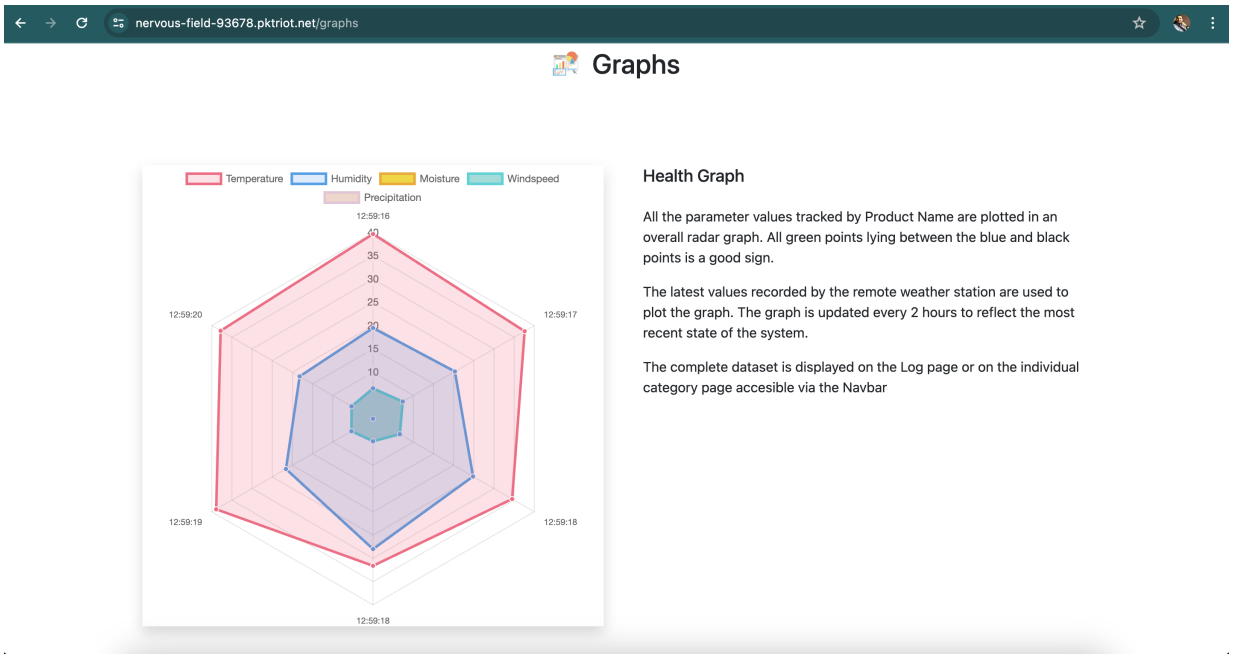


Figure 4.28: Health graph .

Also there's a graph for each factor for all day data trace.

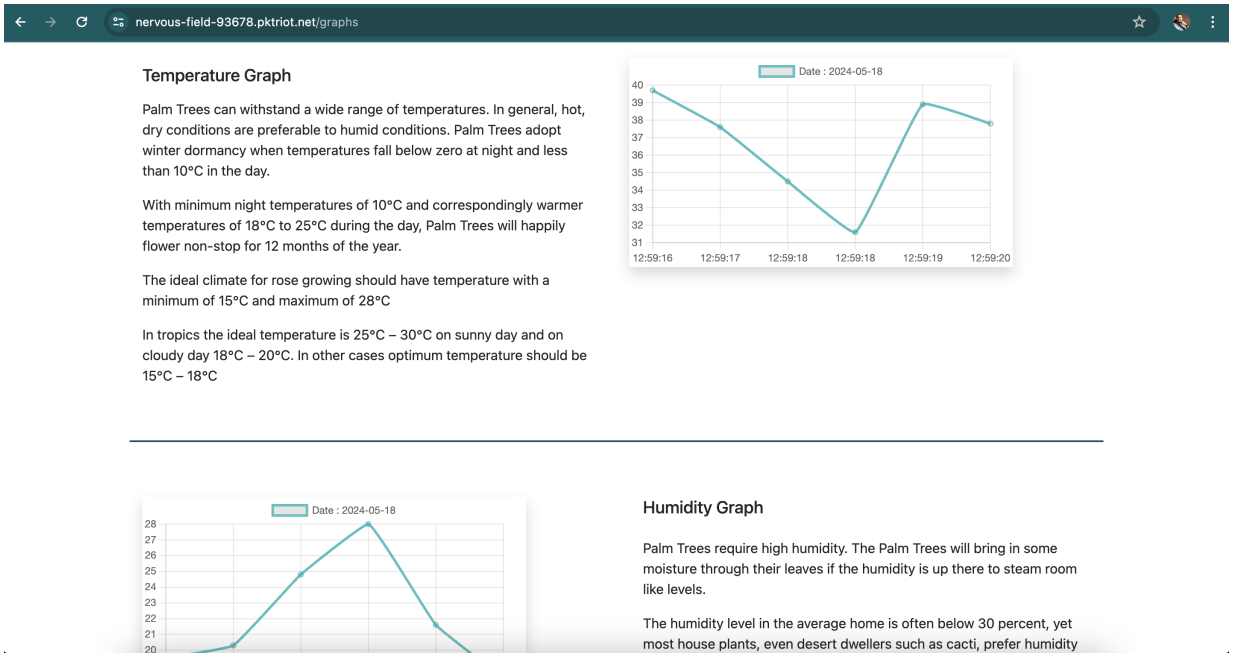
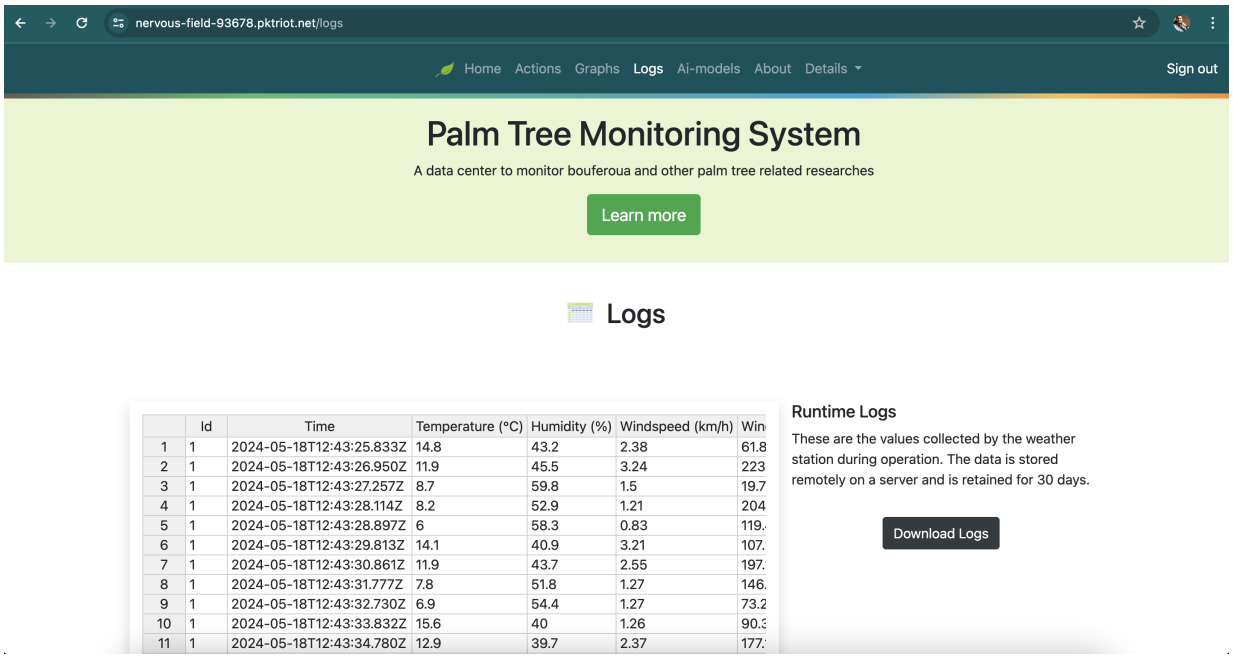


Figure 4.29: Graphs .

In our logs page appears all weather data of this station where you can press download button to download it in a csv file



	Id	Time	Temperature (°C)	Humidity (%)	Windspeed (km/h)	Win
1	1	2024-05-18T12:43:25.833Z	14.8	43.2	2.38	61.8
2	1	2024-05-18T12:43:26.950Z	11.9	45.5	3.24	223
3	1	2024-05-18T12:43:27.257Z	8.7	59.8	1.5	19.7
4	1	2024-05-18T12:43:28.114Z	8.2	52.9	1.21	204
5	1	2024-05-18T12:43:28.897Z	6	58.3	0.83	119.
6	1	2024-05-18T12:43:29.813Z	14.1	40.9	3.21	107.
7	1	2024-05-18T12:43:30.861Z	11.9	43.7	2.55	197.
8	1	2024-05-18T12:43:31.777Z	7.8	51.8	1.27	146.
9	1	2024-05-18T12:43:32.730Z	6.9	54.4	1.27	73.2
10	1	2024-05-18T12:43:33.832Z	15.6	40	1.26	90.3
11	1	2024-05-18T12:43:34.780Z	12.9	39.7	2.37	177.

Figure 4.30: Logs Page .

- This is our AI models page where you can view or add new or update AI models for specific researches or study where you can define the URL of your server related to your hosted server , in our case we deploy our Boufaroua models in our Flask server

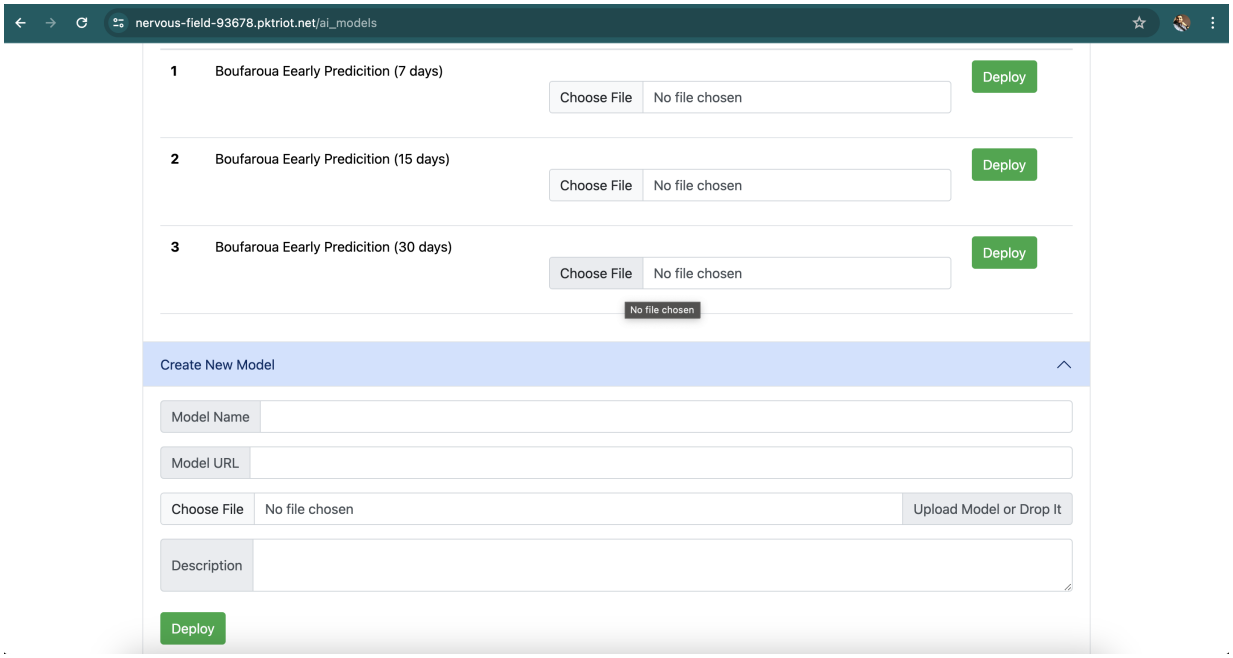
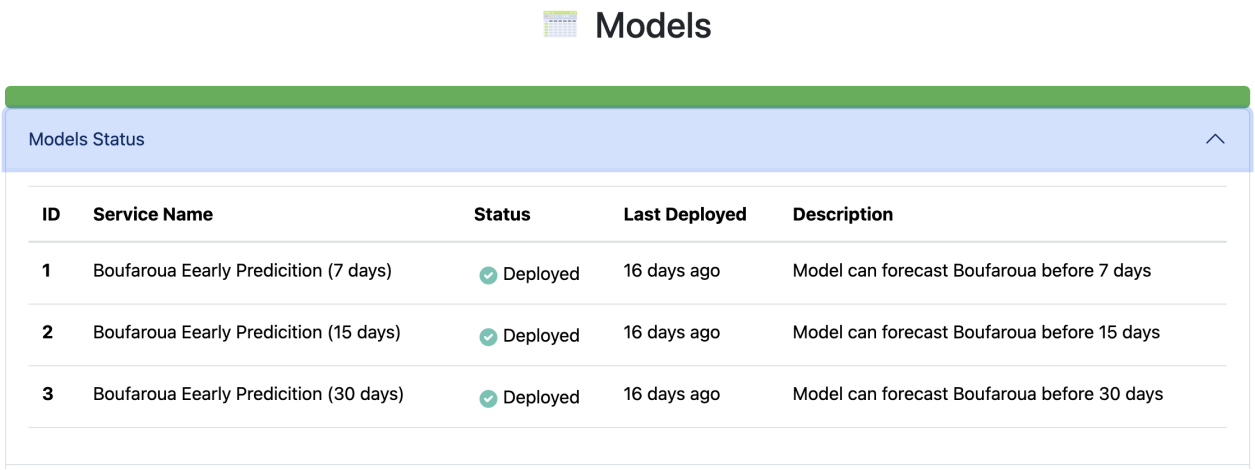


Figure 4.31: Models Deployment Page .

- Here's an example of our deployed 3 models :



ID	Service Name	Status	Last Deployed	Description
1	Boufaroua Eearly Predicition (7 days)	Deployed	16 days ago	Model can forecast Boufaroua before 7 days
2	Boufaroua Eearly Predicition (15 days)	Deployed	16 days ago	Model can forecast Boufaroua before 15 days
3	Boufaroua Eearly Predicition (30 days)	Deployed	16 days ago	Model can forecast Boufaroua before 30 days

Figure 4.32: Deployed models.

- Also you can download logs for every weather factor and see its graph , here's an example of temperature factor :

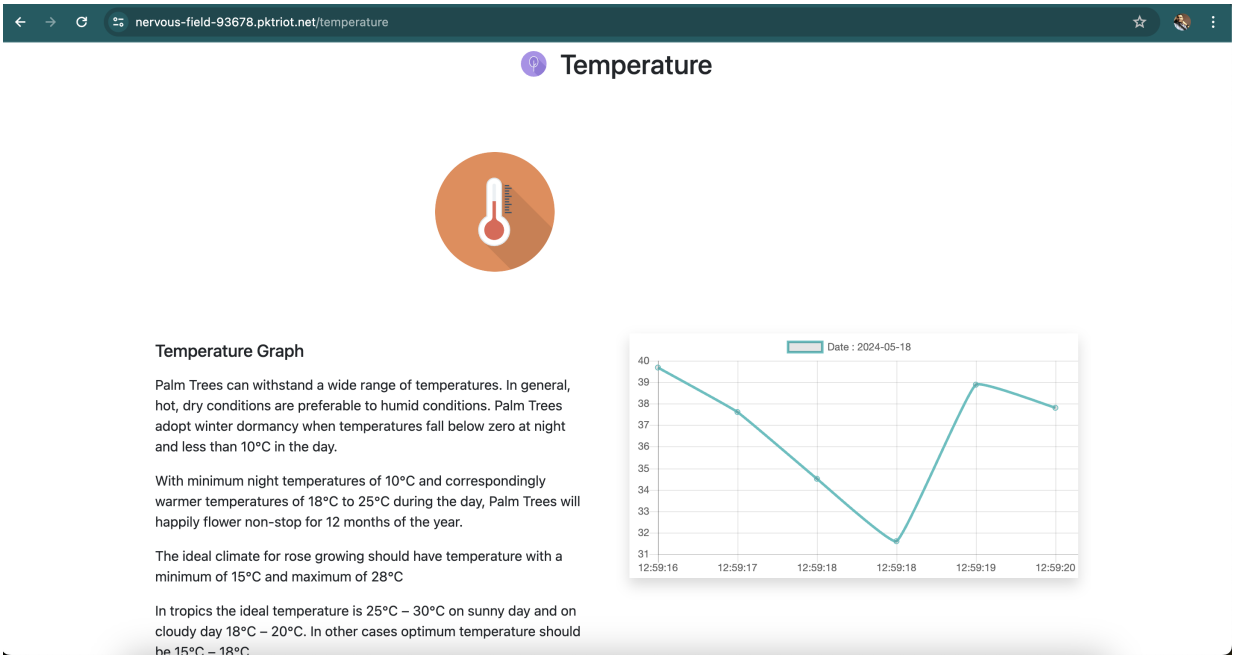


Figure 4.33: Temperature graph on its page.

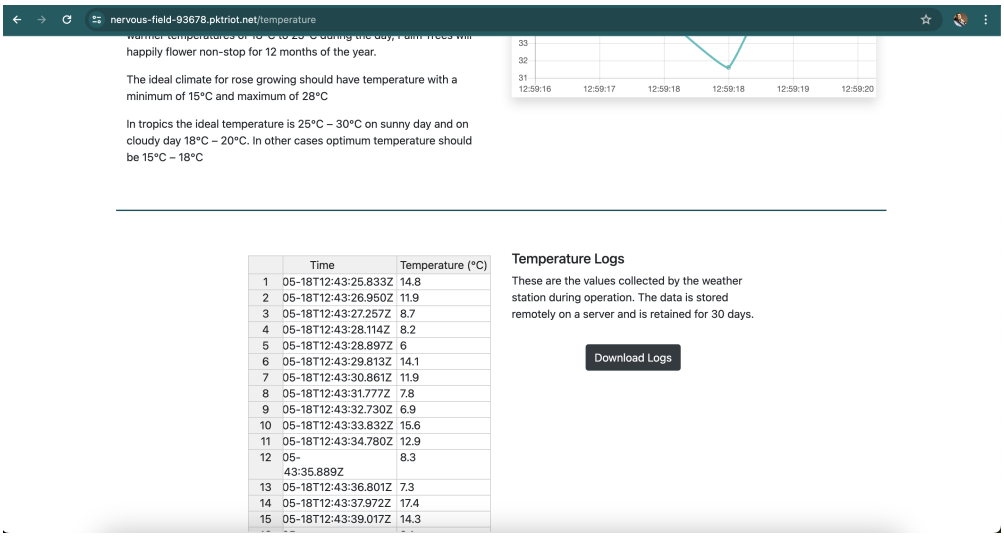


Figure 4.34: Temperature logs on its page.

In addition to that , we host our platform and database in a private server (Render) to insure data redundancy to prevent loss data in case one source fails , after upload all the source code in GitHub , i relate this final to Render server .

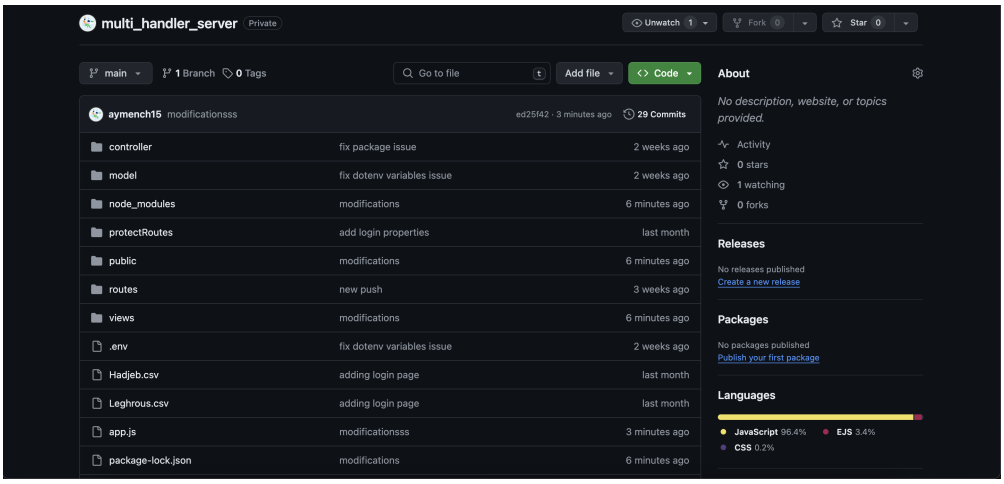


Figure 4.35: Upload server in GitHub.

Figure 4.36: Linked Render to GitHub .

- Our server and database started and hosted successfully on Render

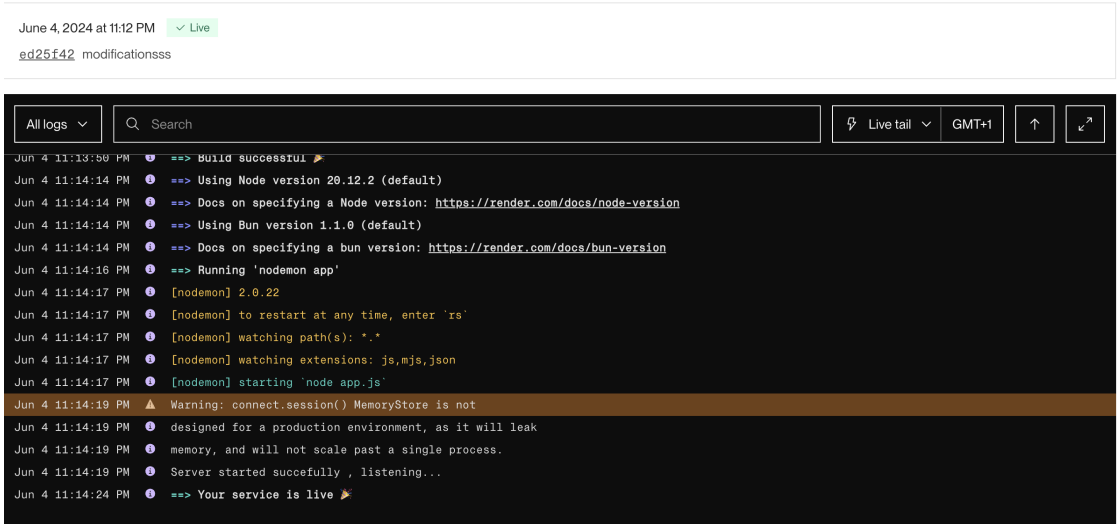


Figure 4.37: Starting server on Render succefusslly .

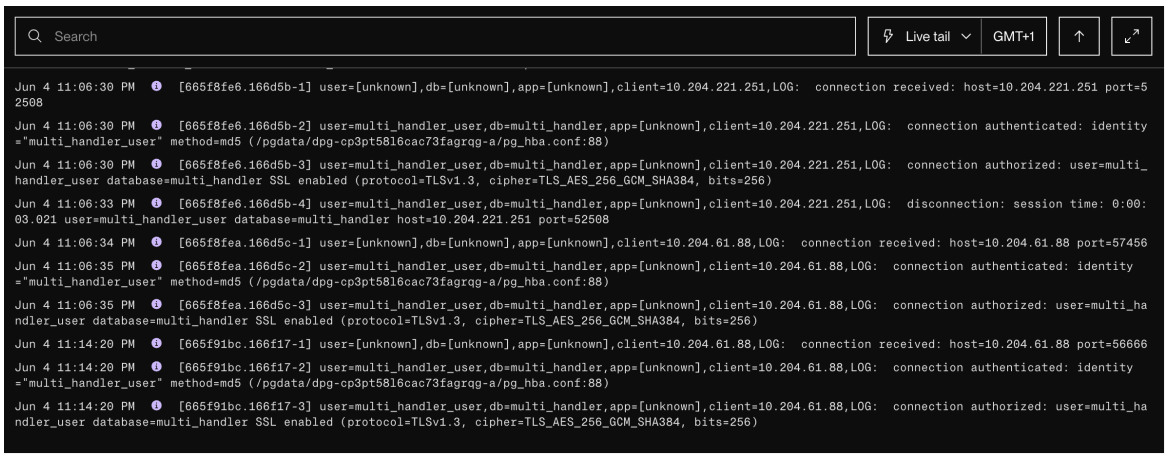


Figure 4.38: Starting database on Render succefusslly .

- We can see them active on Render

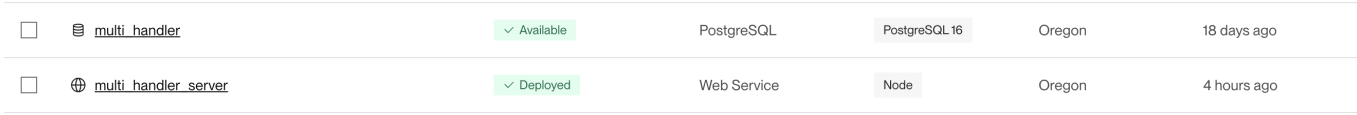


Figure 4.39: Both server and database deployed and active on Render.

- The link to the platform : "https://multi-handler-server.onrender.com"

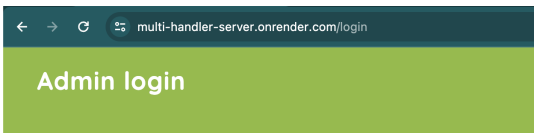


Figure 4.40: Browsing the platform online .

4.4.2 AI models details and deployment

4.4.2.1 Dataset description

We got the data from NASA POWER API

YEAR	MO	DY	HR	TEMPERATURE	RELATIVE-HUM	DEW-POINT	WINSSPEED2	WINDIRECTION	WINDSPEED10	WINDIRECTION	PRECIPITATION	SOILMOISTURE	UVINDEX
2021	2	15	1	04.01	80.94	01.05	1.5	19.75	2.87	19.72	0.0	0.1	0.0
2021	2	15	7	5.27	58.5	-2.15	2.38	61.84	03.09	61.42	0.0	0.0	0.12
2021	2	15	13	12.75	35.69	-2.02	3.24	223.73	3.97	223.88	0.0	0.0	3.0
2021	2	15	19	5.37	56.19	-2.6	1.21	204.08	2.63	203.82	0.0	0.0	0.0
2021	2	16	1	3.73	56.0	-4.11	0.83	119.48	1.29	119.05	0.0	0.0	0.0
2021	2	16	7	4.11	49.81	-5.23	3.21	107.29	4.27	107.14	0.0	0.7	0.15
2021	2	16	13	13.12	26.25	-5.6	2.55	197.1	03.06	197.23	0.0	0.4	3.27
2021	2	16	19	5.38	47.56	-4.7	1.27	146.8	2.6	145.88	0.0	0.0	0.0
2021	2	17	1	2.49	54.44	-5.55	1.27	73.23	2.58	73.82	0.0	0.0	0.0
2021	2	17	7	7.71	43.19	-3.93	1.26	90.36	2.26	89.41	0.0	0.0	0.15
2021	2	17	13	17.65	18.94	-6.03	2.37	177.17	2.84	177.16	0.0	0.0	
2021	2	17	19	09.08	35.69	-5.14	1.32	140.54	2.3	141.07	0.0	0.0	0.0
2021	2	18	1	4.19	45.88	-6.18	1.49	82.46	03.07	82.98	0.0	0.0	0.0

Figure 4.41: Raw data from NASA POWER API .

4.4.2.2 Data preparation

We check if there's a missing data , in this API sometimes the missing values wrote as -999 and sometimes -1 or NULL so that's why we did :

```
all_data.isna().sum()
```

```
Y                0
YEAR             0
MONTH            0
DAY              0
TEMP_MIN         0
TEMP_MAX         0
TEMP_AVG         0
HUM_MIN          0
HUM_MAX          0
HUM_AVG          0
DEWP_MIN         0
DEWP_MAX         0
DEWP_AVG         0
WIND2_MIN        0
WIND2_MAX        0
WIND2_AVG        0
WINDSPEED10_MIN  0
WINDSPEED10_MAX  0
WINDSPEED10_AVG  0
PRECIP           0
SOILMOIS         0
UVIDX            0
dtype: int64
```

Figure 4.42: Missing values NULL.

```
columns_list = ['YEAR','MONTH','DAY','TEMPERATURE_MIN','TEMPERATURE_MAX','TEMPERATURE_AVG','HUMIDITY_MIN','HUMIDITY_MAX'
(all_data[columns_list] == -1).sum().sum()
```

```
0
```

Figure 4.43: Missing values -1 .

```
columns_list = ['YEAR','MONTH','DAY','TEMPERATURE_MIN','TEMPERATURE_MAX','TEMPERATURE_AVG','HUMIDITY_MIN','HUMIDITY_MAX'
(all_data[columns_list] == -999).sum().sum()
```

```
0
```

Figure 4.44: Missing values -999 .

Then we change the data structure to be min value and max value then the average value for each factor in the reason of determine the true value , example : min temperature 10 and max temperature 20 so the average is (min temp + max temp) /2 = 15 we can get the same value if the min temperature where 12 and max temperature where 18 so the average is also 15 ,so we can't depend in only the average values .

	TEMP_MIN	TEMP_MAX	TEMP_AVG	HUM_MIN	HUM_MAX	HUM_AVG	DEWP_MIN	DEWP_MAX	DEWP_AVG	WIND2_MIN	WIND2_MAX	WIND2_AVG	PRECIP	SOILMOIS	UVIDX
0	20.72	31.58	26.15	33.25	61.62	47.44	13.11	14.97	14.04	0.29	2.11	1.20	0.00	0.36	3.70
1	21.48	34.13	27.80	25.88	58.50	42.19	11.49	15.23	13.36	0.21	3.52	1.86	0.00	0.35	3.77
2	22.47	37.24	29.86	24.31	51.06	37.69	10.42	17.73	14.07	0.83	4.82	2.83	0.00	0.34	4.03
3	25.08	36.61	30.84	24.38	57.88	41.13	12.79	16.80	14.79	0.55	2.85	1.70	0.00	0.34	3.62
4	24.01	36.43	30.22	19.94	65.81	42.88	9.62	17.23	13.43	0.41	3.07	1.74	0.33	0.34	3.54
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4378	26.23	39.29	32.76	18.88	40.88	29.88	9.69	14.23	11.96	0.75	4.71	2.73	0.00	0.11	3.67
4379	26.81	38.73	32.77	18.94	44.06	31.50	9.85	15.93	12.89	1.00	5.57	3.29	0.00	0.11	4.71
4380	25.67	37.90	31.79	15.88	37.81	26.85	6.90	12.08	9.49	0.48	4.09	2.29	0.00	0.11	3.46
4381	27.31	37.41	32.36	16.75	34.06	25.41	7.44	12.32	9.88	0.60	4.12	2.36	0.00	0.11	3.55
4382	26.58	39.95	33.27	14.62	42.38	28.50	8.09	13.44	10.77	1.75	5.94	3.85	0.28	0.11	4.10

4383 rows × 15 columns

Figure 4.45: All data cleaned.

- LSTM and GRU uses sigmoid and tanh that are sensitive to magnitude so values need to be normalized in all models:

```
t = train_columns[1:]
print(t)
scaler = StandardScaler()
scaler = scaler.fit(all_data[t])
all_data_scaled = scaler.transform(all_data[t])
merged_array = np.concatenate((all_data['Y'].values.reshape(-1, 1), all_data_scaled), axis=1)
```

Figure 4.46: Data normalisation.

After adding our target column (Y) we split for each model the data into blocs depend on the future day can this model forecast where **nfuture** is the number of days we want to look into the future based on the past days, and **npast** Number of past days we want to use to predict the future.

```
X_train = []
Y_train = []

n_future = 7
n_past = 30

for i in range(n_past, len(all_data[train_columns]) - n_future +1):
    X_train.append(merged_array[i - n_past:i, 0:all_data_scaled.shape[1]])
    Y_train.append(merged_array[i + n_future - 1:i + n_future,0])
```

Figure 4.47: Data splitting into blocs for our 7 days model.

```
X_train = []
Y_train = []

n_future = 15
n_past = 30

for i in range(n_past, len(all_data[train_columns]) - n_future +1):
    X_train.append(merged_array[i - n_past:i, 0:all_data_scaled.shape[1]])
    Y_train.append(merged_array[i + n_future - 1:i + n_future,0])
```

Figure 4.48: Data splitting into blocs for our 15 days model.

```
X_train = []
Y_train = []

n_future = 30
n_past = 30

for i in range(n_past, len(all_data[train_columns]) - n_future +1):
    X_train.append(merged_array[i - n_past:i, 0:all_data_scaled.shape[1]])
    Y_train.append(merged_array[i + n_future - 1:i + n_future,0])
```

Figure 4.49: Data splitting into blocs for our 30 days model.

Then we split the obtained data equally among the class :

```
x_train,x_valid,y_train,y_valid = train_test_split(X_train,Y_train,test_size=0.2,stratify=Y_train,shuffle=True)
x_train.shape,y_train.shape,x_valid.shape,y_valid.shape,
```

```
((3471, 30, 14), (3471, 1), (868, 30, 14), (868, 1))
```

Figure 4.50: Data splitting .

Here's our models architecture diagrams :

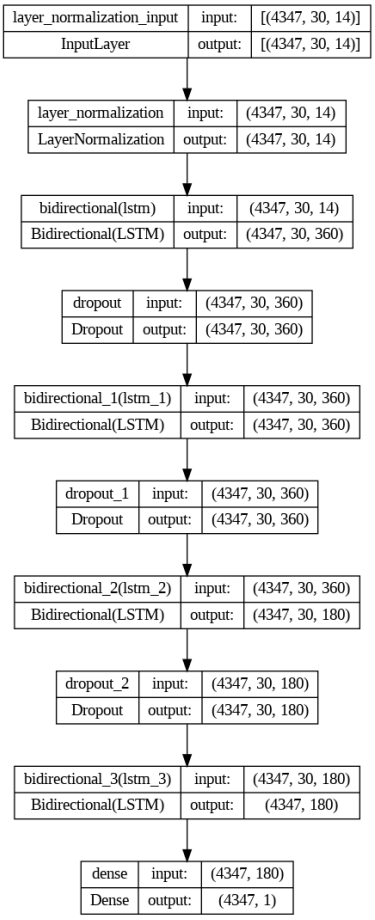


Figure 4.51

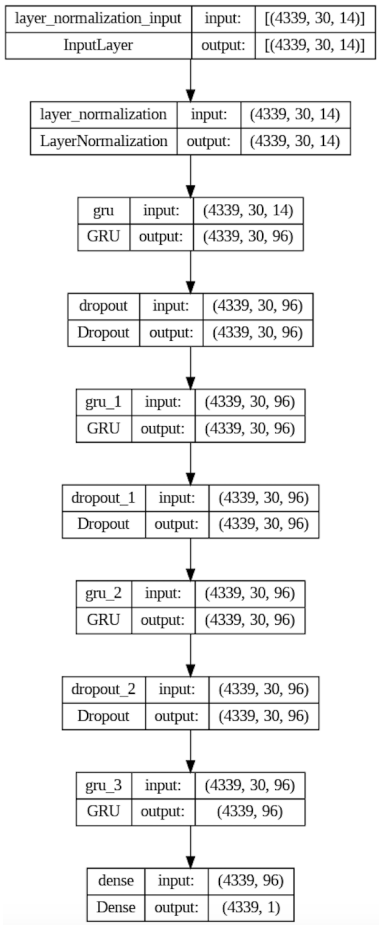


Figure 4.52: 15 days Model Architecture Diagram

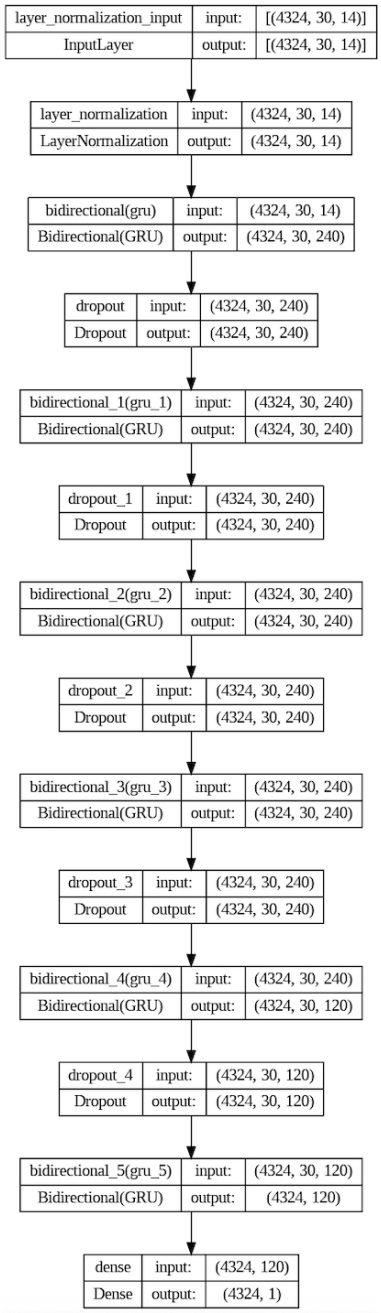


Figure 4.53: 30 days Model Architecture Diagram

We used Keras to build our neural network , here's the code of our three models :

```
callback = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

model=keras.Sequential([
    keras.layers.LayerNormalization(axis=1 , center=True , scale=True),
    keras.layers.Bidirectional(GRU(180,return_sequences=True,input_shape=(X_train.shape[1], X_train.shape[2]))),
    keras.layers.Dropout(0.18),
    keras.layers.Bidirectional(GRU(180,return_sequences=True)),
    keras.layers.Dropout(0.18),
    keras.layers.Bidirectional(GRU(90,return_sequences=True)),
    keras.layers.Dropout(0.18),
    keras.layers.Bidirectional(GRU(90)),
    keras.layers.Dropout(0.18),
    keras.layers.Dense(1)
])
opt = keras.optimizers.Adam(learning_rate=0.0002)
model.compile(
    optimizer= opt,
    loss="mse",
    metrics=["accuracy"]
)
```

Figure 4.54: Our 7 days model

- 7 days model training

```
Epoch 17/80
109/109 [==============================] - 53s 488ms/step - loss: 0.0080 - accuracy: 0.9370 - val_loss: 0.0064 - val_accuracy: 0.9345
Epoch 18/80
109/109 [==============================] - 55s 503ms/step - loss: 0.0029 - accuracy: 0.9379 - val_loss: 0.0083 - val_accuracy: 0.9368
Epoch 19/80
109/109 [==============================] - 56s 511ms/step - loss: 0.0038 - accuracy: 0.9385 - val_loss: 0.0059 - val_accuracy: 0.9368
```

Figure 4.55: 7 days model training

```
callback = EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)
model=keras.Sequential(
    [
    keras.layers.LayerNormalization(axis=1 , center=True , scale=True),
    keras.layers.GRU(96,return_sequences=True,input_shape=(X_train.shape[1], X_train.shape[2]),
                     use_bias=True,bias_initializer='zeros'),
    keras.layers.Dropout(0.2),
    keras.layers.GRU(96,return_sequences=True),
    keras.layers.Dropout(0.2),
    keras.layers.GRU(96,return_sequences=True),
    keras.layers.Dropout(0.2),
    keras.layers.GRU(96),
    keras.layers.Dense(1)
    ]
    )
opt = keras.optimizers.Adam(learning_rate=0.0003)
model.compile(
    optimizer= opt,
    loss="mse",
    metrics=["accuracy"]
)
```

Figure 4.56: Our 15 days model

- 15 days model training

```
poch 17/80
99/109 [==============================] - 11s 101ms/step - loss: 0.0133 - accuracy: 0.9358 - val_loss: 0.0131 - val_accuracy: 0.9355
poch 18/80
99/109 [==============================] - 11s 100ms/step - loss: 0.0130 - accuracy: 0.9360 - val_loss: 0.0192 - val_accuracy: 0.9332
poch 19/80
99/109 [==============================] - 10s 87ms/step - loss: 0.0115 - accuracy: 0.9369 - val_loss: 0.0177 - val_accuracy: 0.9355
poch 20/80
99/109 [==============================] - 11s 100ms/step - loss: 0.0091 - accuracy: 0.9360 - val_loss: 0.0117 - val_accuracy: 0.9332
poch 21/80
99/109 [==============================] - 11s 102ms/step - loss: 0.0070 - accuracy: 0.9369 - val_loss: 0.0296 - val_accuracy: 0.9297
poch 22/80
99/109 [==============================] - 9s 83ms/step - loss: 0.0087 - accuracy: 0.9375 - val_loss: 0.0201 - val_accuracy: 0.9320
poch 23/80
99/109 [==============================] - 11s 100ms/step - loss: 0.0071 - accuracy: 0.9372 - val_loss: 0.0160 - val_accuracy: 0.9332
poch 24/80
99/109 [==============================] - 11s 99ms/step - loss: 0.0106 - accuracy: 0.9366 - val_loss: 0.0348 - val_accuracy: 0.9355
```

Figure 4.57: 15 days model training

```
callback = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
model=keras.Sequential([
    keras.layers.LayerNormalization(axis=1 , center=True , scale=True),
    keras.layers.Bidirectional(GRU(120,return_sequences=True,
              input_shape=(X_train.shape[1], X_train.shape[2]),use_bias=True,bias_initializer='zeros')),
    keras.layers.Dropout(0.2),
    keras.layers.Bidirectional(GRU(120,return_sequences=True)),
    keras.layers.Dropout(0.2),
    keras.layers.Bidirectional(GRU(120,return_sequences=True)),
    keras.layers.Dropout(0.2),
    keras.layers.Bidirectional(GRU(120,return_sequences=True)),
    keras.layers.Dropout(0.2),
    keras.layers.Bidirectional(GRU(60,return_sequences=True)),
    keras.layers.Dropout(0.2),
    keras.layers.Bidirectional(GRU(60)),
    keras.layers.Dense(1)

])
opt = keras.optimizers.Adam(learning_rate=0.0002)
model.compile(
    optimizer= opt,
    loss="mse",
    metrics=["accuracy"]
)
```

Figure 4.58: Our 30 days model

- 30 days model training

```
217/217 [==============================] - 77s 353ms/step - loss: 0.0415 - accuracy: 0.9136 - val_loss: 0.0465 - val_accuracy: 0.9006
Epoch 5/100
217/217 [==============================] - 73s 335ms/step - loss: 0.0322 - accuracy: 0.9176 - val_loss: 0.0503 - val_accuracy: 0.9133
Epoch 6/100
217/217 [==============================] - 76s 351ms/step - loss: 0.0225 - accuracy: 0.9193 - val_loss: 0.0272 - val_accuracy: 0.9179
Epoch 7/100
217/217 [==============================] - 73s 336ms/step - loss: 0.0244 - accuracy: 0.9219 - val_loss: 0.0487 - val_accuracy: 0.9121
Epoch 8/100
217/217 [==============================] - 71s 328ms/step - loss: 0.0184 - accuracy: 0.9222 - val_loss: 0.0497 - val_accuracy: 0.9145
Epoch 9/100
217/217 [==============================] - 79s 364ms/step - loss: 0.0205 - accuracy: 0.9231 - val_loss: 0.0136 - val_accuracy: 0.9202
Epoch 10/100
217/217 [==============================] - 74s 342ms/step - loss: 0.0174 - accuracy: 0.9231 - val_loss: 0.0343 - val_accuracy: 0.9179
Epoch 11/100
217/217 [==============================] - 75s 347ms/step - loss: 0.0149 - accuracy: 0.9240 - val_loss: 0.0193 - val_accuracy: 0.9191
Epoch 12/100
217/217 [==============================] - 74s 343ms/step - loss: 0.0171 - accuracy: 0.9240 - val_loss: 0.0193 - val_accuracy: 0.9191
```

Figure 4.59: 30 days model training

4.4.2.3 Discussion results

We will go over some of the metrics we used in this part to evaluate our models:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where:

n is the number of observations

y_i is the actual value

$\hat{y}_i$ is the predicted value

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

n is the number of observations

y_i is the actual value

$\hat{y}_i$ is the predicted value

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

n is the number of observations

y_i is the actual value

$\hat{y}_i$ is the predicted value

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where:

n is the number of observations

y_i is the actual value

$\hat{y}_i$ is the predicted value

$\bar{y}$ is the mean of the actual values

MSE , MAE , RMSE, and R^2 Score are appropriate and commonly used for evaluating continuous value models, including models like GRU and LSTM networks.

- 7 days model

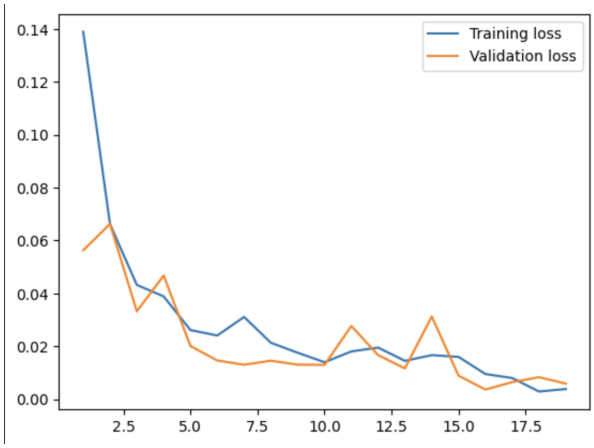


Figure 4.60: Train-validation loss curve

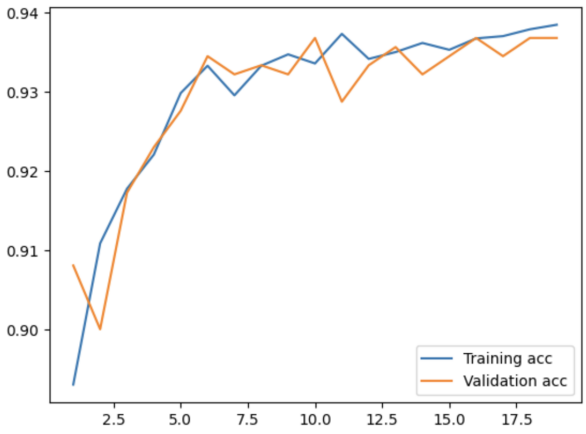


Figure 4.61: Train-validation acc curve

The graph illustrates the training and validation accuracy over 19 epochs with a batch size of 32. Both accuracies increase sharply, with training (blue) and validation (orange) accuracies stabilizing around 93.85%. The close alignment between the curves indicates minimal overfitting and good generalization to unseen data. The model demonstrates high performance and robustness in forecasting data correctly, achieving approximately 94% accuracy in both training and validation.

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_valid, pred)
mse
```

```
0.008175874207838205
```

(a) Mse for 7 days model

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_valid, pred)
mae
```

```
0.03963469816130939
```

(b) Mae for 7 days model

```
rmse = np.sqrt(mse)
rmse
```

```
0.09042054085128116
```

(c) Rmse for 7 days model

```
from sklearn.metrics import r2_score
r2 = r2_score(y_valid, pred)
r2
```

```
0.9798615661871635
```

(d) R2 score for 7 days model

Figure 4.62: The metrics used to evaluate the model

The metrics used to evaluate the forecasting model indicate strong performance and accurate predictions. The MSE is low at 0.0081, MAE is 0.03, RMSE is 0.09, and the R^2 value is high at 0.9798. These results suggest that the model effectively captures the underlying patterns in the data, making it highly reliable for forecasting.

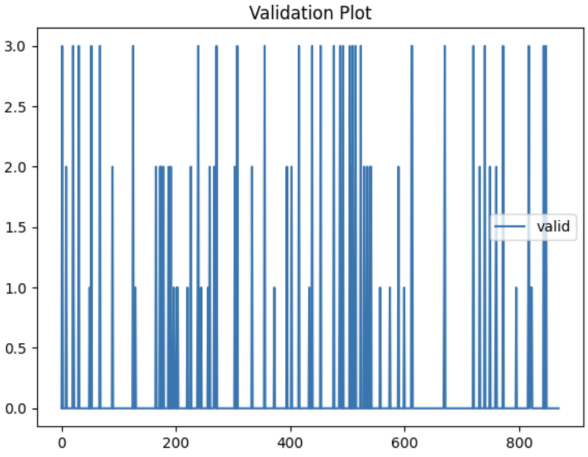


Figure 4.63: Validation plot for 7 days model

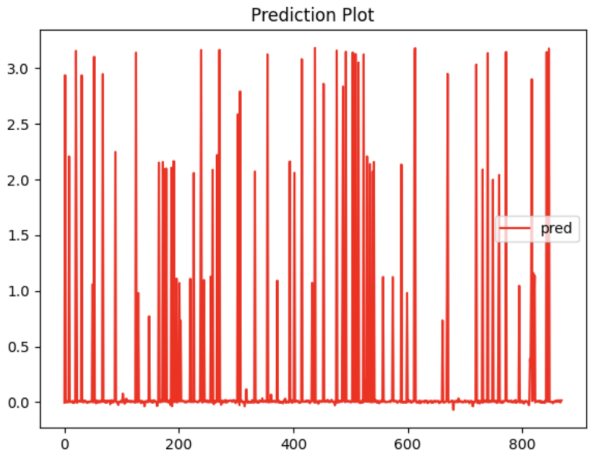


Figure 4.64: Prediction plot for 7 days model

The real data plot and the prediction plot are very similar, indicating that the model's predictions closely match the actual data.

- 15 days models

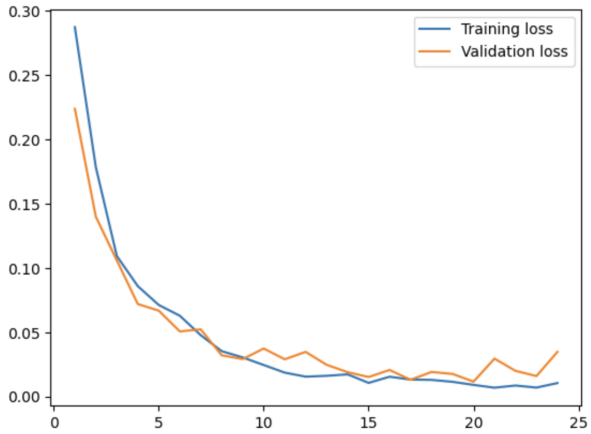


Figure 4.65: Train-validation loss curve

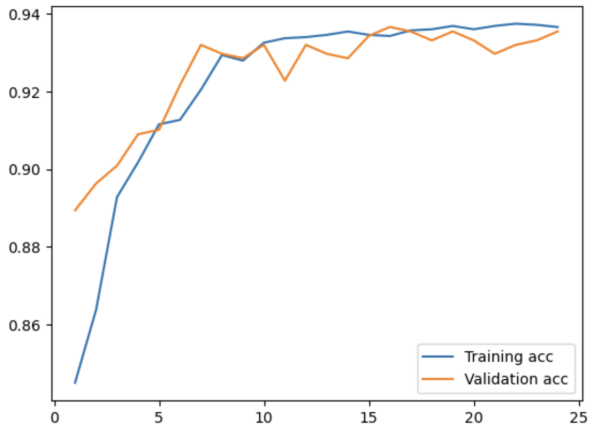


Figure 4.66: Train-validation acc curve

The graph illustrates the training and validation accuracy over 25 epochs with a batch size of 32. Both accuracies increase sharply, with training (blue) and validation (orange) accuracies stabilizing around 93.55%. The close alignment between the curves indicates minimal overfitting and good generalization to unseen data. The model demonstrates high performance and robustness in forecasting data correctly, achieving approximately 94% accuracy in both training and validation.

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_valid, pred)
mse
```

```
0.011660314854156677
```

(a) Mse for 15 days model

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_valid, pred)
mae
```

```
0.03905958155908274
```

(b) Mae for 15 days model

```
rmse = np.sqrt(mse)
rmse
```

```
0.10798293779184134
```

(c) Rmse for 15 days model

```
from sklearn.metrics import r2_score
r2 = r2_score(y_valid, pred)
r2
```

```
0.9713393087674395
```

(d) R2 score for 15 days model

Figure 4.67: The metrics used to evaluate the model

The metrics used to evaluate the forecasting model indicate strong performance and accurate predictions. The MSE is low at 0.0117, MAE is 0.0391, RMSE is 0.1080, and the R^2 value is high at 0.9713. These results suggest that the model effectively captures the underlying patterns in the data, making it highly reliable for forecasting.

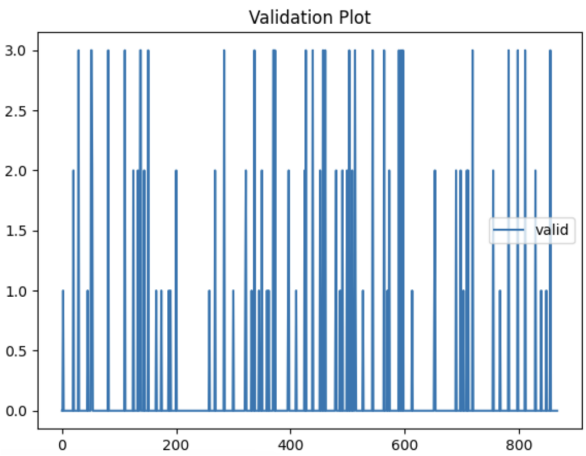


Figure 4.68: Validation plot for 15 days model

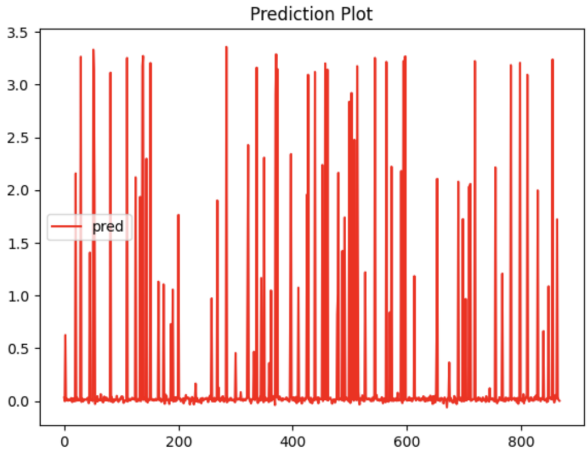


Figure 4.69: Prediction plot for 15 days model

The real data plot and the prediction plot are very similar, indicating that the model's predictions closely match the actual data.

- 30 days models

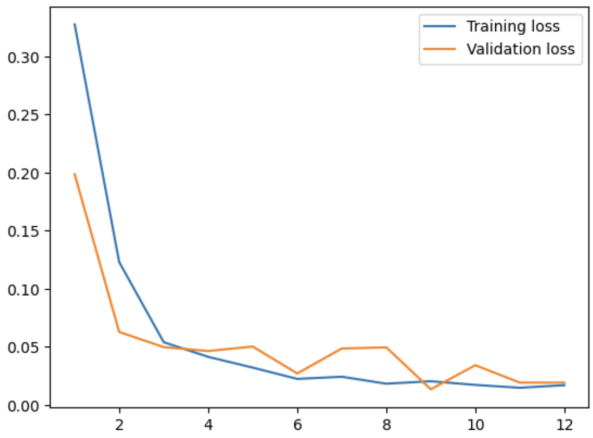


Figure 4.70: Train-validation loss curve

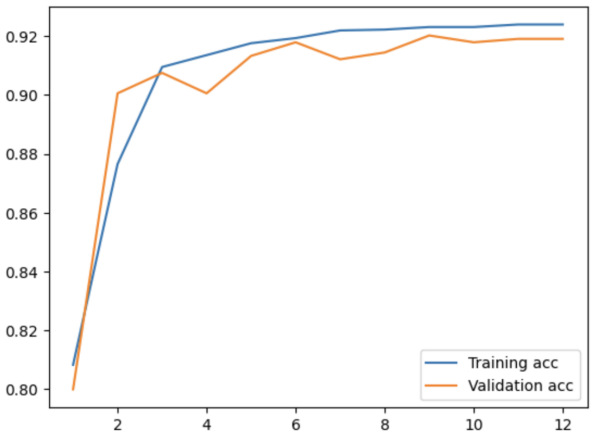


Figure 4.71: Train-validation acc curve

The graph illustrates the training and validation accuracy over 12 epochs with a batch size of 32. Both accuracies increase sharply, with training (blue) and validation (orange) accuracies stabilizing around 92%. The close alignment between the curves indicates minimal overfitting and good generalization to unseen data. The model demonstrates high performance and robustness in forecasting data correctly, achieving approximately 92% accuracy in both training and validation.

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_valid, pred)
mse
```

```
0.013572976593325004
```

(a) Mse for 30 days model

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(y_valid, pred)
mae
```

```
0.04455748067339721
```

(b) Mae for 30 days model

```
rmse = np.sqrt(mse)
rmse
```

```
0.11650311838455228
```

(c) Rmse for 30 days model

```
from sklearn.metrics import r2_score
r2 = r2_score(y_valid, pred)
r2
```

```
0.9700446561280238
```

(d) R2 score for 30 days model

Figure 4.72: The metrics used to evaluate the model

The metrics used to evaluate the forecasting model indicate strong performance and accurate predictions. The MSE is low at 0.0135, MAE is 0.0445, RMSE is 0.116, and the R^2 value is high at 0.9700. These results suggest that the model effectively captures the underlying patterns in the data, making it highly reliable for forecasting.

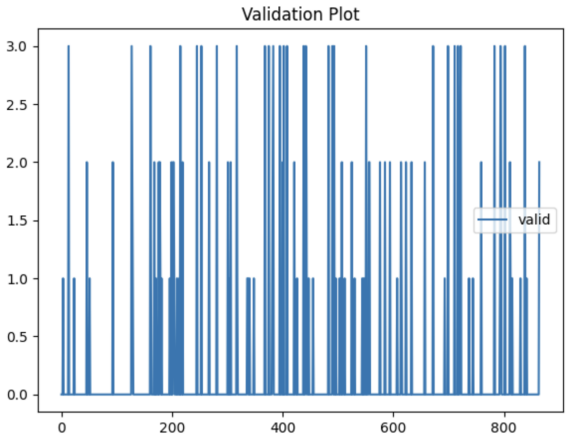


Figure 4.73: Validation plot for 30 days model

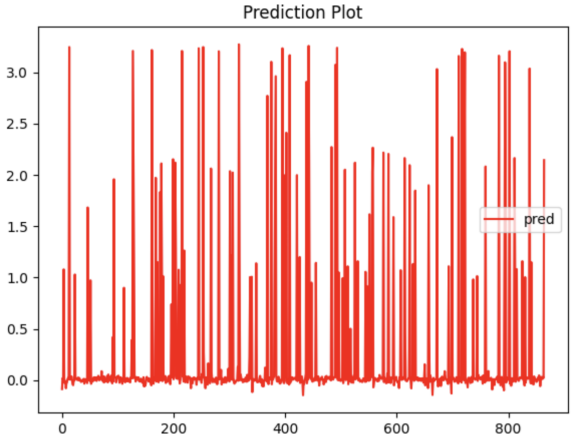


Figure 4.74: Prediction plot for 30 days model

Also here the real data plot and the prediction plot are very similar, indicating that the model's predictions closely match the actual data.

Table 4.1: Comparative between our model and previous similar study.

Aspect	**Classification Model (Prev study)**	**Forecasting model**
Model Type	LSTM,GRU	Bi-LSTM , GRU , Bi-GRU
Prediction Horizon	1 day	7 , 15 , 30 days in future
Evaluation Metrics	Confusion Matrix,Precision,F1-Score	MSE, MAE, RMSE, R^2 Score
Accuracy	High	High despite prediction horizon
Model Complexity	Moderate	Higher (due to long pred horizon)

4.4.3 Mobile application

Here's some application's screens:

4.4.3.1 Login screen

In this screen the user either type the unique ID of his prototype or scan the QR code where located on the station's outer cover.

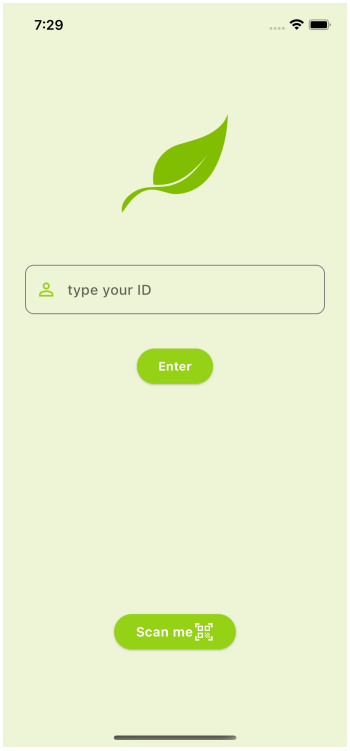
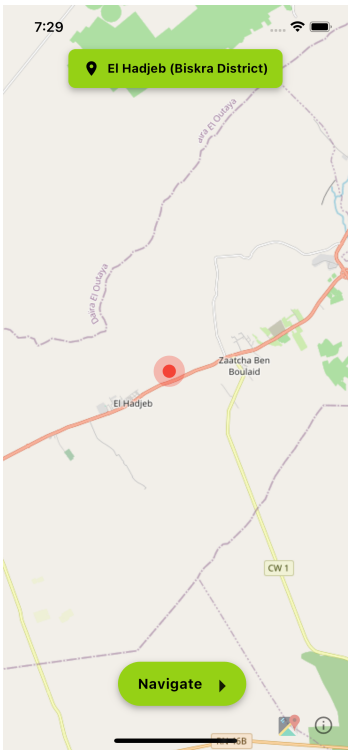


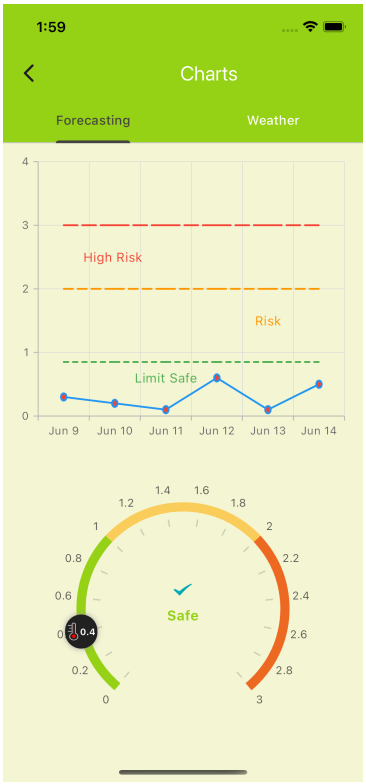
Figure 4.75: Login screen.

4.4.3.2 Map screen



This screen shows the exactly location of the station in real time , and a button the navigate the predictions and weather screens

4.4.3.3 Predictions screen



This screen shows the predictions of stages where we have Three cases :
- Safe state indicates stage 0 where it means there's no mite exist.
- Risk state indicates stage 1 where it means the mite is starting appearing and may being adult.
- Hight Risk state is final stage where indicates that the mite is spread widely on the farm.

4.4.3.4 Current weather screen

This screen shows a current weather data

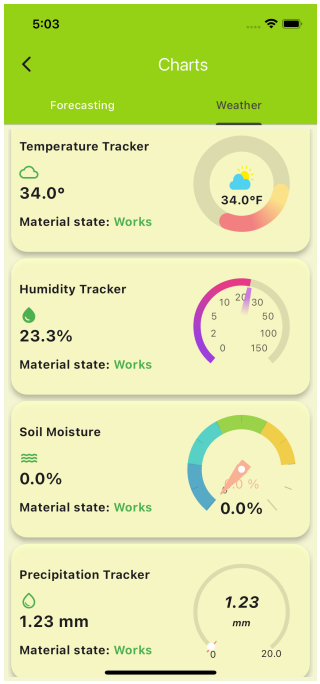


Figure 4.76: Weather screen.

4.4.4 Hardware implementation

We use a LilyGO T-SIMA7670sa development board, which incorporates the SIMA7670s module for GSM communication. This board is equipped with an external antenna to enhance signal reception. In addition to its GSM capabilities, the system includes a DHT22 sensor for precise temperature and humidity measurements. A solar panel is employed to capture solar energy, which powers the system. For energy storage, a lithium-ion battery is integrated, ensuring uninterrupted operation even without direct sunlight. The device can transmit data over the GSM network and is powered by solar energy and a rechargeable battery. Furthermore, we have integrated an FC-37 YL-83 raindrop sensor, a capacitive soil moisture sensor v2.0, and a GPS antenna to acquire coordinates. We

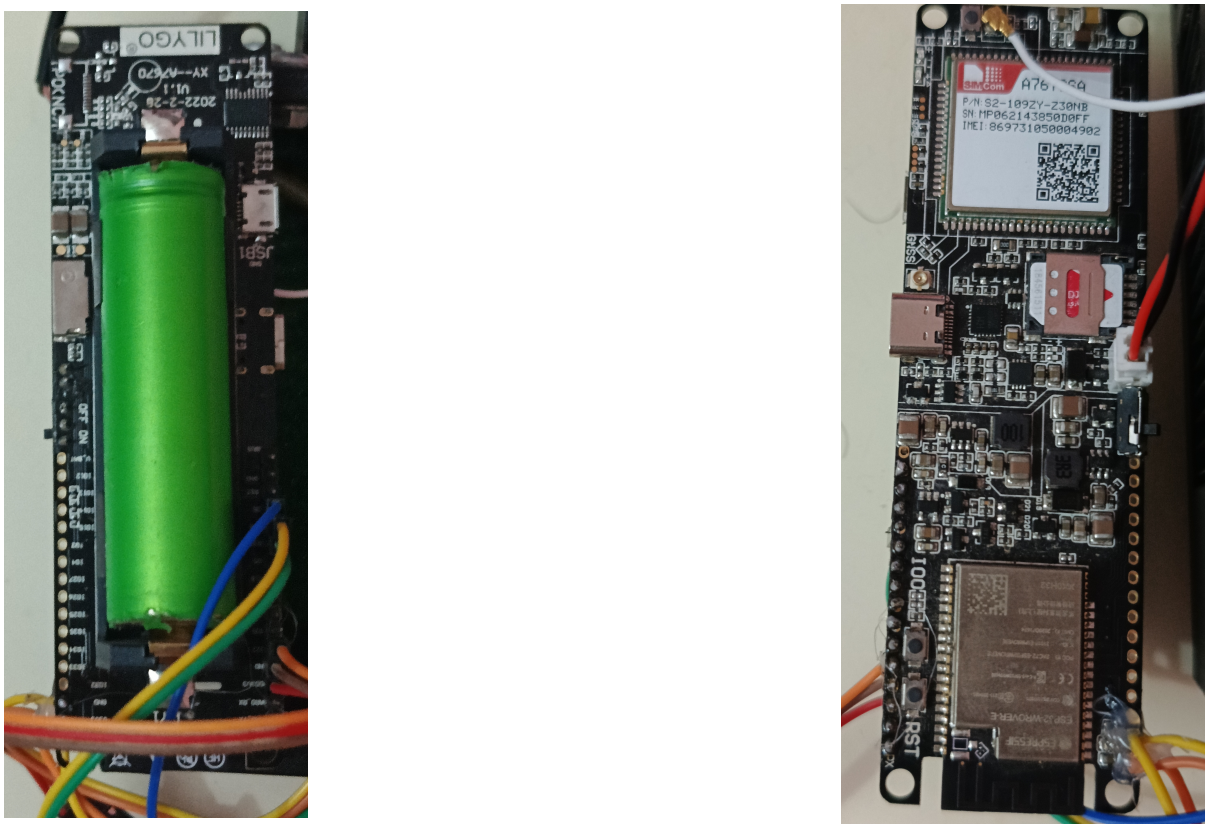


Figure 4.77: Device mounting

positioned a SIM card and GPS and solar panel in the designated areas before attaching a DHT22 sensor to the required components. The sensor was connected to the GND (ground), GPIO 22 (general-purpose input/output), and VCC 3v3 (power supply) pins , for rain drop sensor GPIO 36 ,for soil moisture sensor GND , GPIO 0 , VCC 3v3 pins

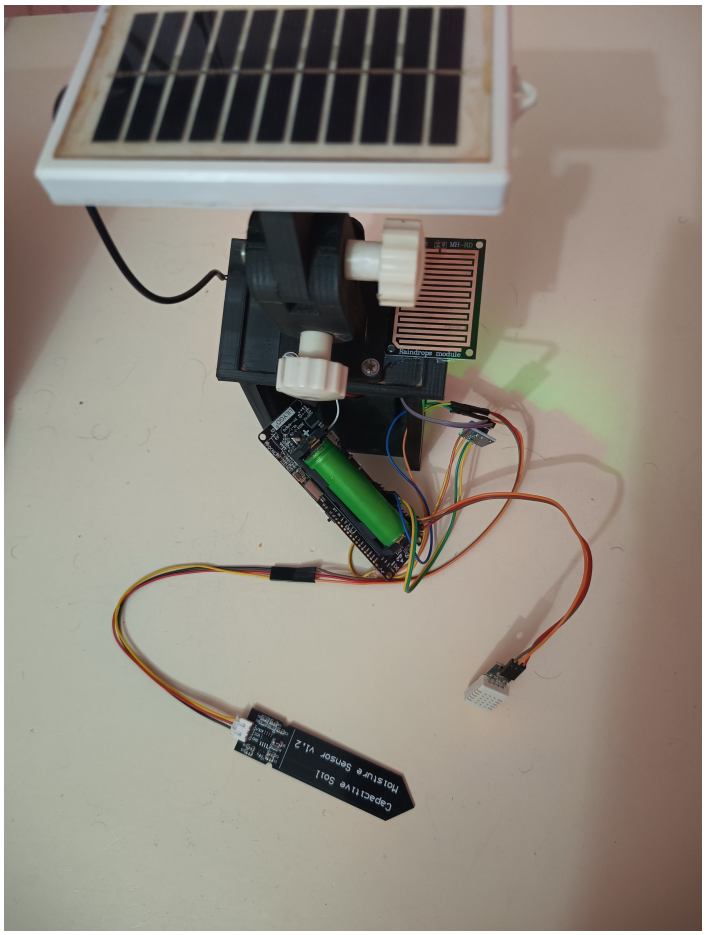


Figure 4.78: Mounted sensors .

We used Arduino IDE to get all necessary libraries and then programe our device (LilyGO T-SIMA7670 esp32).
We send the data to both our hosted server and our data center using an Http request.

timestamp timestamp without time zone	temperature double precision	humidity double precision	windspeed double precision	wind_direction double precision	precipitation double precision	soil_moist double pre
2024-06-20 15:01:46.594994	41.3	13.3	31.1	102	0	
2024-06-20 15:02:31.769376	41.2	13.2	31.4	101	0	
2024-06-20 15:02:37.789055	41.2	13.2	31.4	101	0	

Figure 4.79: Received data from our device.

Here's the last shape of our station :

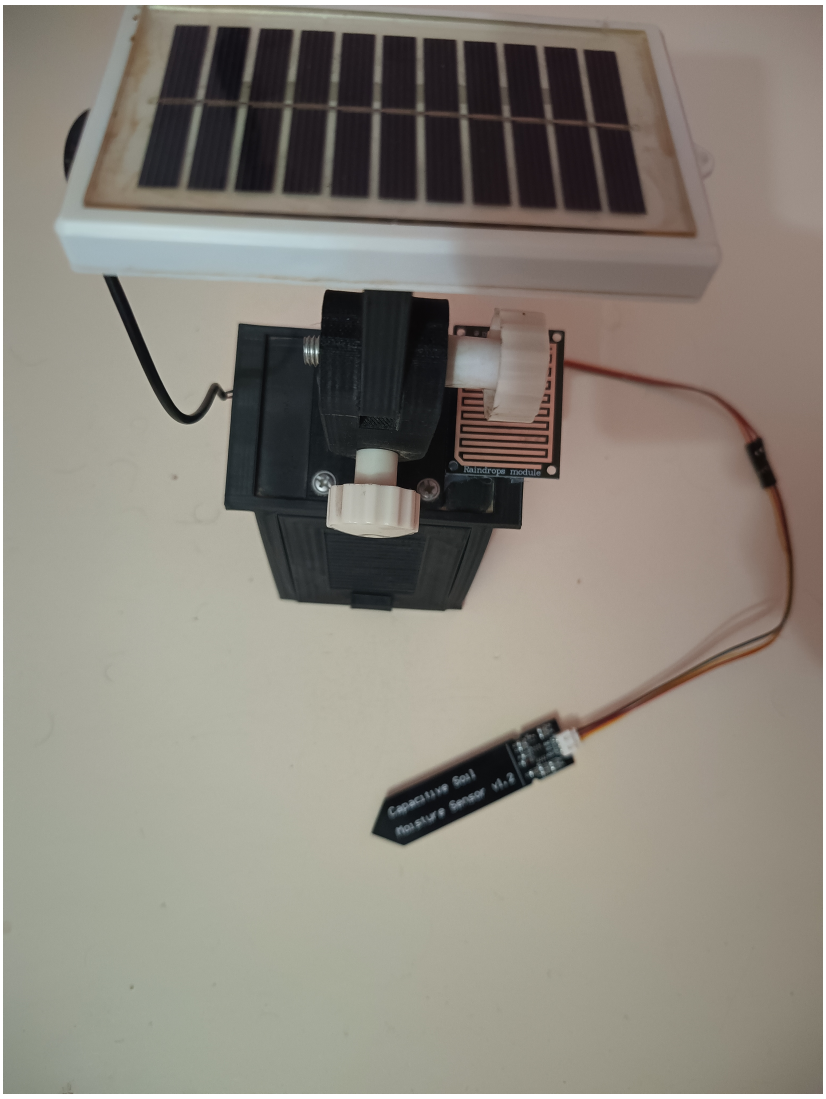


Figure 4.80: Station shape.

4.5 Conclusion

In the final chapter, we detailed the implementation of our system, covering both hardware and software components, as well as the AI models used. We then analyzed the results, demonstrating the system's efficiency and effectiveness. Ultimately, our findings confirm that the system achieves high performance and meets our objectives successfully.

General conclusion

Boufaroua and other diseases poses a severe threat to date palm productivity, with infection rates increasing under conditions of insufficient irrigation and neglect of maintenance. During dry and warm years, the impact of these ailments can significantly reduce crop yield, posing a threat to overall production, making it essential to address this issue promptly. However, these diseases is difficult to detect in its early stages visually, and controlling its spread becomes increasingly challenging as it progresses rapidly. Farmers often struggle to determine the appropriate timing for implementing chemical treatments or other preventive measures to combat this pest effectively.

In response to this critical challenge, we have successfully implemented an early forecasting system for this palm disease, We have detailed the data set used for training our models and the data preparation steps involved to ensure accurate and reliable predictions. We developed three different models based on future forecasting stages , we used Bi-LSTM to forecast the stage in 7 days in the future and Stacked-GRU to forecast the stage in 15 days in future , Bi-GRU to forecast the stage in 30 days in the future.

Our project's outcomes show great promise in tracking palm trees, identifying illnesses early, and assisting farmers promptly. By integrating IoT, deep learning, and software technologies, we have created a system that reduces the need for costly chemical treatments and physical inspections, enabling farmers to make informed decisions and take preventive actions against palm disease.

Looking forward, there are several perspectives for future development:

- Emphasizing the need for a large amount of diverse data to continually enhance and refine our models, making them more robust and accurate over time.
- Extending the system to detect and manage a wide range of palm tree diseases, ensuring comprehensive protection for date palms and other varieties.
- Utilizing and adapting new AI techniques based on the extensive data we collect, further improving the system's predictive capabilities and responsiveness.

Bibliography

[1] Abdelhak Heroucha. Design of a new intelligent system for early prediction of date palm spider mite boufaroua. Master's thesis, University of Mohamed Khider Biskra, 2023.

[2] https://packetriot.com/. Accessed: May 25, 2024.

[3] https://wikidocs.net/166318. Accessed: June 06, 2024.

[4] Ayesha Sahar and Dongsoo Han. An lstm-based indoor positioning method using wi-fi signals. pages 1–5, 08 2018.

[5] Hanuman Verma, Saurav Mandal, and Akshansh Gupta. Temporal deep learning architecture for prediction of covid-19 cases in india. *Expert Systems with Applications*, 195:116611, 02 2022.

[6] Marwan Abu-zanona, Said Elaiwat, Shayma'a Younis, Nisreen Innab, and MM Kamruzzaman. Classification of palm trees diseases using convolution neural network. *International Journal of Advanced Computer Science and Applications*, 13, 01 2022.

[7] Rashid Al Shidi, Lalit Kumar, Salim Al-Khatri, Malik Albahri, and Mohammed Alaufi. Relationship of date palm tree density to dubas bug ommatissus lybicus infestation in omani orchards. *Agriculture*, 8:64, 04 2018.

[8] https://www.ibm.com/topics/internet-of-things. Accessed: April 4, 2024.

[9] Sabah Abdulazeez, Abbas Nawar, Nawar Banwan, and Imad Tareq. Internet of things: Architecture, technologies, applications, and challenges. 2:36–52, 03 2024.

[10] Mohammed Faci, Meriem Boultif, Mohamed Matallah, Billal Nia, and Mohammed Mesnoua. Boufaroua (oligonychus afrasiaticus) in tolga palm groves in 2021. 6:8–24, 03 2023.

[11] https://www.spiceworks.com/tech/tech-general/articles/what-is-a-server/. Accessed: April 4, 2024.

[12] https://www.oracle.com/database/what-is-database/. Accessed: April 4, 2024.

[13] https://www.spiceworks.com/tech/devops/articles/application-programming-interface/. Accessed: April 5, 2024.

[14] https://www.ibm.com/topics/api. Accessed: April 5, 2024.

[15] https://www.educative.io/answers/what-are-api-protocols. Accessed: April 5, 2024.

[16] Chaker Bennour, Ali ben belgacem, Hamza Hammadi, and Hmed Nasr. Oligonychus afrasiaticus (mcgregor) : Problématiques d'infection du palmier dattier et son contrôle : Révision de littérature oligonychus afrasiaticus (mcgregor): Problem of date palm infection and its management: A review. 2:20–24, 02 2022.

[17] Sameh Chaaban, Chermiti Brahim, and Serge Kreiter. Oligonychus afrasiaticus and phytoseiid predators' seasonal occurrence on date palm phoenix dactylifera (deglet noor cultivar) in tunisian oases. *Bulletin of Insectology*, 64:15–21, 11 2011.

[18] https://iridl.ldeo.columbia.edu/dochelp/QA/Basic/dewpoint.html. Accessed: April 5, 2024.

[19] Abdelghani SAOUDI MABROUK. Etude de la cartographie du boufaroua, oligonychus afrasiaticus dans les palmeraies des ziban ; etude cas la région de sidi okba. Master's thesis, University of Mohamed Khider Biskra, 2022.

[20] Masoud Latifian. Date palm spider mite (oligonychus afrasiaticus mcgregor) forecasting and monitoring system. *WALIA journal*, 30:79–85, 01 2014.

[21] Maged Mohammed, Hamadttu Elshafie, and Muhammad Munir. Development and validation of innovative machine learning models for predicting date palm mite infestation on fruits. *Agronomy*, 13:494, 02 2023.

[22] https://nodejs.org/en. Accessed: June 02, 2024.

[23] https://www.python.org/doc/essays/blurb/. Accessed: June 02, 2024.

[24] https://keras.io/about/. Accessed: June 02, 2024.

[25] Mandeep Singh, Ayushi Verma, Aashwaath Parasher, Nidhi Chauhan, and Gaurav Budhiraja. Implementation of database using python flask framework. 12 2019.

[26] https://flutter.dev/. Accessed: June 02, 2024.

[27] https://dart.dev/. Accessed: June 02, 2024.

[28] https://www.postgresql.org/about/. Accessed: June 02, 2024.

[29] https://packetriot.com/. Accessed: June 03, 2024.

[30] Pragati Thawani. Understanding architecture of internet of things. 08 2022.

[31] https://power.larc.nasa.gov/docs/services/api/. Accessed: May 27, 2024.

[32] https://render.com/. Accessed: June 03, 2024.