TED UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CMPE491 - HIGH LEVEL DESIGN REPORT

Project Title

ALSApp (Agriculture and Livestock Support Application)

Project Supervisor

Venera ADANOVA

Project Juries

Ayşe Yasemin Seydim

Tansel Dökeroğlu

Project Members

Engin Samet Dede

Sevgi Dilay Demirci

Gizem Ünsal

27/12/2024

Table Of Contents

1. Introduction **3**
1.1 Project Description 3
1.2 Purpose of the system 3
1.3 Design goals 4
1.4 Overview 5
2. Current software architecture **6**
3. Proposed software architecture **13**
3.1 Overview 13
3.2 Subsystem Decomposition 15
3.3 Hardware/software mapping 19
3.4 Persistent Data Management 22
3.5 Access Control and Security 26
3.7 Boundary Conditions 30
4. Subsystem Services **32**
5. Glossary **38**
6. References **41**

1. Introduction

1.1 Project Description

The goal of our project is to develop an application that will make tracking easier for farmers in a variety of ways. Furthermore, this platform combines all the necessary resources to improve livestock performance and farming activities. Users will be able to enter their animals' health data and receive relevant information, track their crops digitally for optimal efficiency, and track the growth and development of their crops and animals in real time using an easy-to-use interface that is favourable to both farmers and livestock. By consolidating all agricultural and livestock management needs onto a single platform, this system enables users to manage their operations in a more organized, efficient, and planned manner. It also provides instant updates on industry developments, helping users stay informed and responsive to industry changes. They will be aware of government support, mainly offered by the Ministry of Agriculture and Forestry, and benefit from funding opportunities. In addition, the following process can also be used by the Ministry of Agriculture and Forestry staff to keep track of the reliability and accuracy of support.

1.2 Purpose of the system

The ALSApp seeks to create an integrated digital platform that is built for the farmer and livestock breeder to simplify the management & monitoring of their farm and livestock operations. The application brings all the defined key features into a single application like crop and livestock tracking, real time weather update and easy access to the government support programs.

The system enables users to develop digital literacy and address the issues of digital resource operational inefficiencies by providing intuitive tools. Farmers can get notified if something is wrong with their crops and livestock, monitor, get the latest government support information and so forth. Additionally, Ministry of Agriculture and Forestry staff can also monitor the reliability and accuracy of the support delivered via the platform.

Through the ALSApp we strive to bridge the gap between modern digital solutions and traditional farming practices to increase productivity, sustainability and increase awareness of governmental assistance programs. This system design, however, telegraphs that it's easy to use, scalable, and secure for farmers as well as ministry officials, providing a robust solution.

1.3 Design goals

The design of ALSApp is guided by the goals that are mentioned below:

Usability: It was designed to be easy to use by farmers and livestock breeders with differing levels of digital literacy. Users won't have a large learning curve when it comes to using the interface.

Functionality: Core functionalities include crop and livestock tracking, real time weather updates and easy one stop shop of government support, all provided by ALSApp. These features are designed to address directly the special requirements of the agricultural and livestock sectors.

Reliability: It is crucial to ensure that data management is true and dependable. It can allow un consistent stream of weather, government and resources that will help during elimination of errors and downtime.

Scalability: The system is built to sustain high growing user bases as well as increasing data loads without sacrificing performance. It allows ALSApp to be useful as it grows.

Security: Strong measures of authentication, such as two factor verification, protect sensitive information and ensure user data security, and comply with all prevailing data protection standards.

Integration: ALSApp will simply work with external systems such as real time weather services and the Ministry of Agriculture and Forestry database. The availability of accurate and up to date information will be guaranteed with utilizing this ability.

Maintainability: It is modular architecture that makes the application easy to update and to enhance. The trouble this also affords is that it allows easy future improvements, bug fixes, and scalability upgrades without a lot of disruption.

Efficiency: The ALSApp is optimized performance and involved in minimizing the response time to some of the critical operations like data retrieval, and notifications, etc.

These goals make sure that ALSApp delivers on its mission of giving farmers, livestock breeders, and ministry officials a reliable, open, and accessible tool for farmers and livestock management.

1.4 Overview

ALSApp (AgriLivestock Support Application) is an inclusive digital platform for farmers and livestock breeders where traditional systems are complementary with modern technology. This document describes the goals, implementation details, and design of ALSApp, a product to improve productivity, and sustainability, and to increase the accessibility of resources. Key sections include:

Project Introduction: It explains why ALSApp is needed and its usefulness in informing farm and livestock management with digital solutions.

System Architecture: It highlights the modularity and scalability of the system, with a robust performance bound and scalability for future adaptability.

Core Features: Real time tracking of crops and livestock, weather alerts, government support integration, and AI-powered analytics were detailed functionalities.

Design Principles: It stresses low cost, ease of activities, security, scalability and maintainability to cater for the varying needs of these diverse users, ministry officials.

This paper is a foundational document describing what ALSApp is capable of and its vision

to transform agricultural and livestock management.

2. Current software architecture

The software architecture in our application is not yet fully completed, but has been advanced to a certain point. In order to give an idea about the entire software architecture we have developed up to this point, the user interfaces of the application, which started to be developed with flutter, are added to this section.

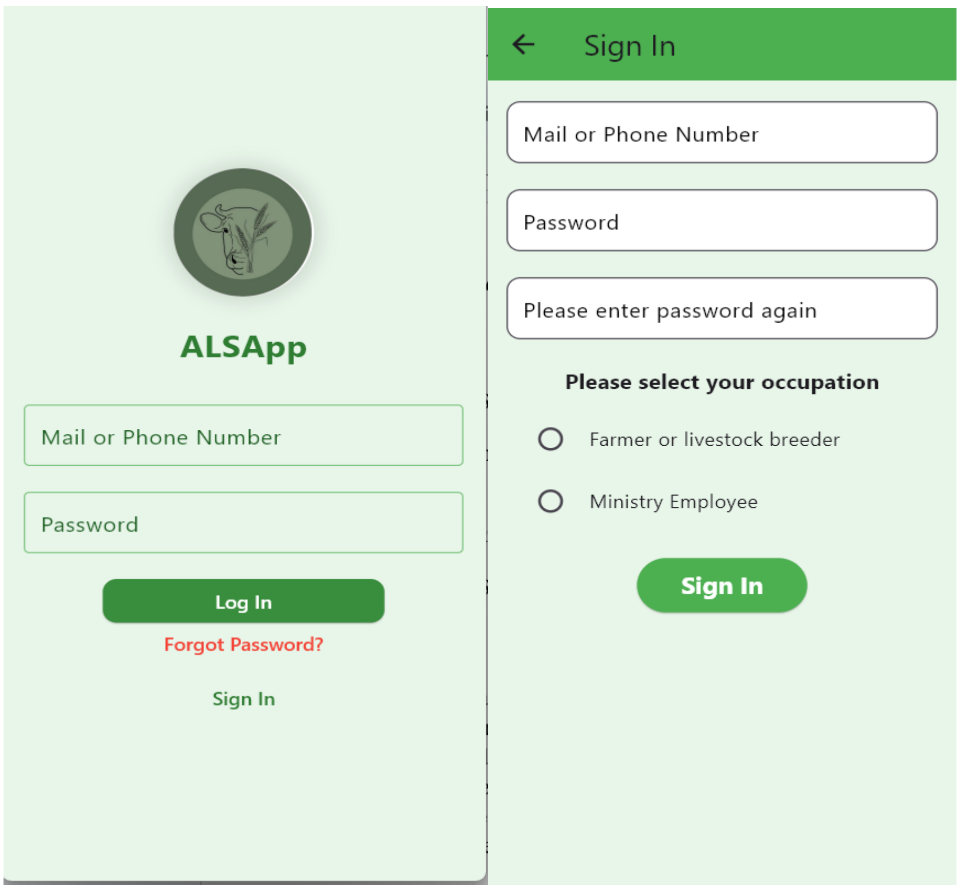


Figure 1: Log-in and Sign-in User interfaces

Figure 1 shows the screens for user authentication, including account creation and login, that ensures secure access to the application.

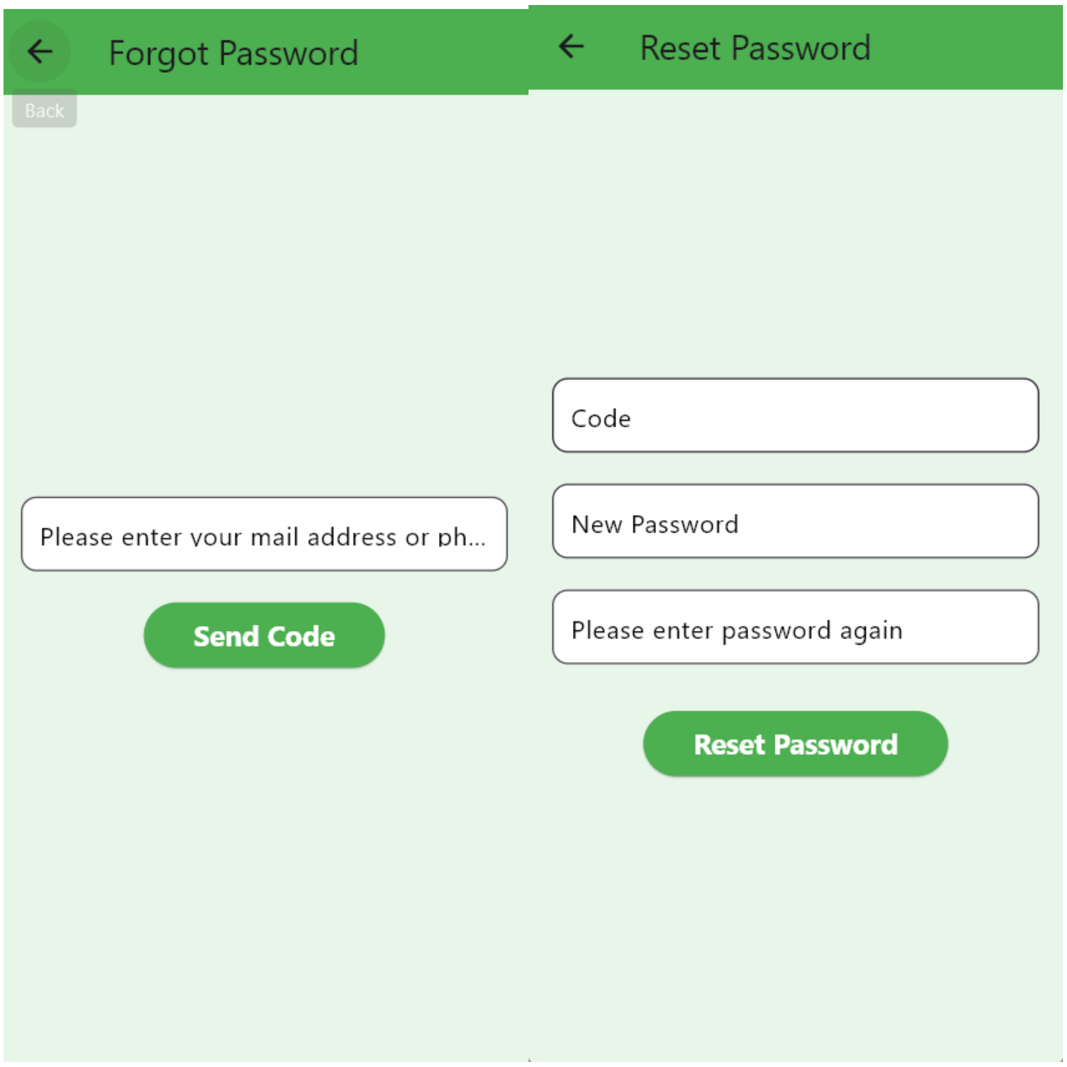
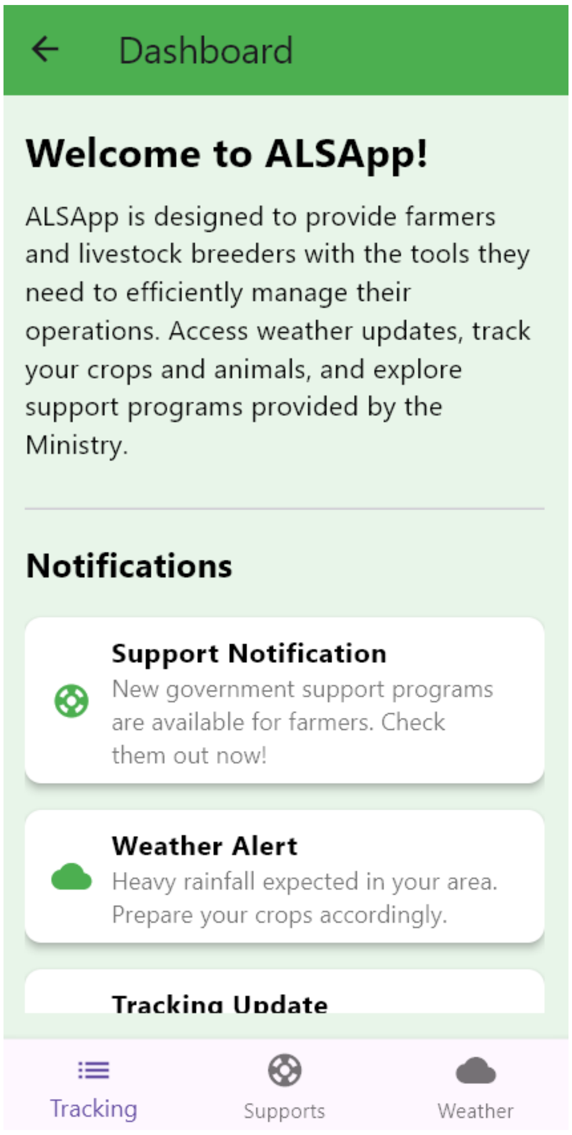


Figure 2: Forgot and Reset Password interfaces

Figure 2 Provides illustrative examples of how users recover and reset their passwords, and improves security with two factor authentication.



Dashboard

Welcome to ALSApp!

ALSApp is designed to provide farmers and livestock breeders with the tools they need to efficiently manage their operations. Access weather updates, track your crops and animals, and explore support programs provided by the Ministry.

Notifications

Support Notification
New government support programs are available for farmers. Check them out now!

Weather Alert
Heavy rainfall expected in your area. Prepare your crops accordingly.

Tracking Update

Figure 3: Dashboard interface

This Figure 3 Displays an overview of key information, such as crop and livestock status, weather updates, and recent notifications for efficient user management.

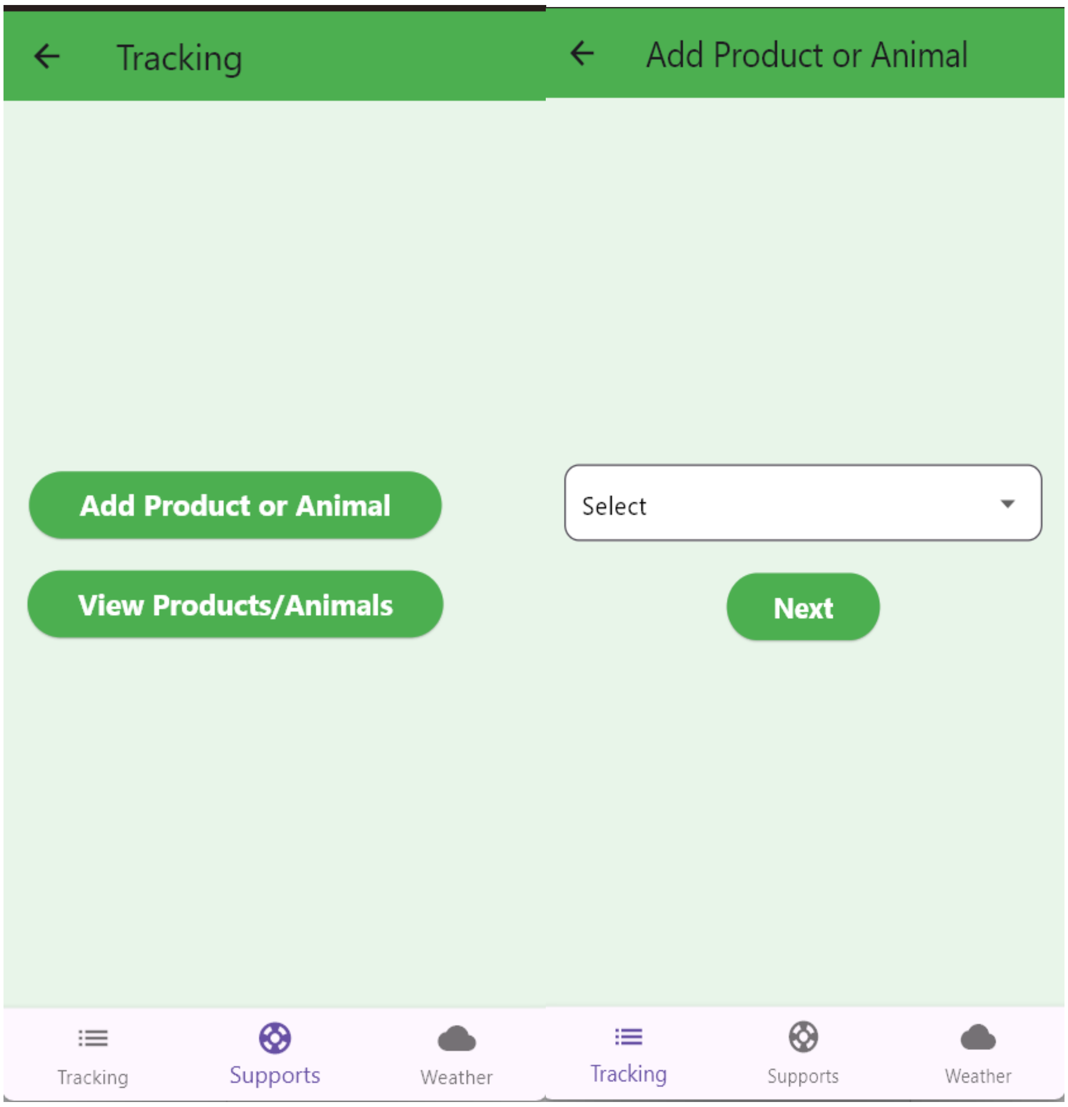


Figure 4: Tracking and Add Product interface

In Figure 4 the first interface is used to add tracking data and observe the added data. The second interface is created for the user to choose the animal or the product to add into the database.

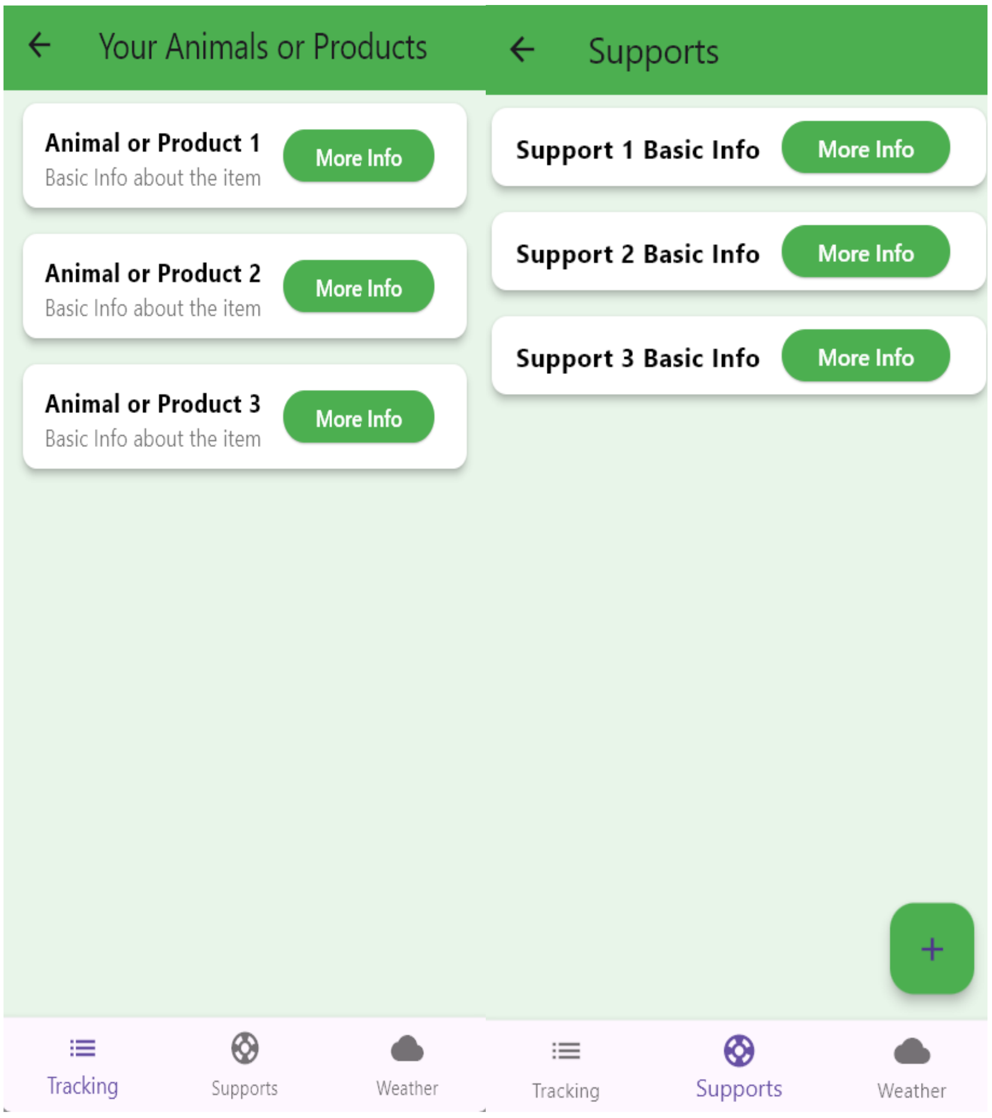
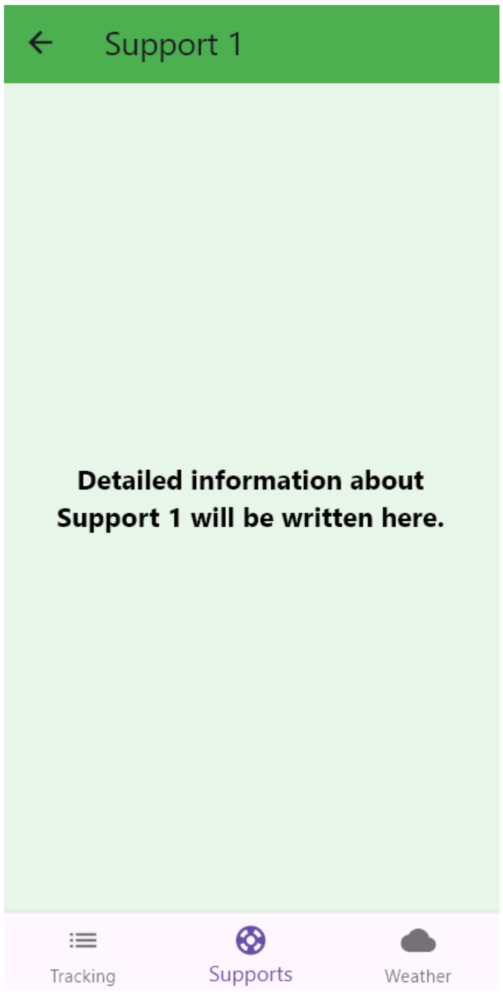


Figure 5: Animals and Products List and Supports interface

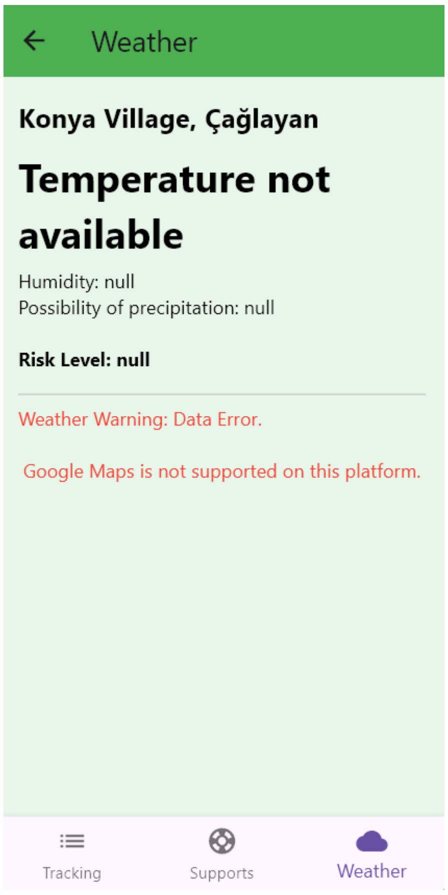
In Figure 5 the first interface gives a look at the saved animals and products. Also, the second one shows specific government support programs that allow the user to see the terms, status and eligibility.



Detailed information about Support 1 will be written here.

Figure 6: Support details interface

Figure 6 Provides a closer look at specific government support programs, enabling users to view eligibility, terms, and status



Konya Village, Çağlayan

Temperature not available

Humidity: null
Possibility of precipitation: null

Risk Level: null

Weather Warning: Data Error.

Google Maps is not supported on this platform.

Figure 7: Weather Interface

Figure 7 offers real-time weather information and alerts tailored to the user's location for proactive decision-making. However, since API's were not activated it is not working completely right now.

3. Proposed software architecture

3.1 Overview

The ALSApp will be created for the benefit of farmers and livestock breeders. It also aims to enable the Ministry of Agriculture and Forestry and its employees to track existing aid. With this application, the main problems that farmers and livestock breeders generally experience will be taken into consideration, as will the Ministry of Agriculture and Forestry and its employees tracking their assistance. This will be explained in detail below in this section.

Farmers and livestock breeders can track their products and resources: This application aims to create a portal where farmers and livestock breeders can record their product and resource data in the application and track them in detail. With this facility, as products and resources are tracked, possible incomplete harvests or low livestock yields can be detected, and early intervention can be made to prevent more significant problems. Moreover, this will increase productivity and make it easier for farmers to achieve more sustainable and economical agriculture and livestock farming since problems can be overcome early.

Facilitating access to Ministry supports: The Ministry of Agriculture and Forestry currently offers various support for farmers and livestock breeders. However, since these supports are available on their own website, and it is tough for farmers to research them due to low digital literacy, this application offers farmers a more accessible option to access these supports.

Weather monitoring: Weather monitoring is very critical for farmers and livestock breeders. A possible weather disaster or excessive rainfall can cause severe problems. This application aims to ensure that farmers are informed about the weather much earlier.

Access for the Ministry and its employees to check the support: This application aims to facilitate the checking and detection of errors and deficiencies for the employees working in the units related to support in the Ministry. In this way, the Ministry employees will be able to notice and solve possible deficiencies and errors in the supports in a shorter time.

The application will be created with these purposes in mind, and it aims to minimize the existing deficiencies of both parties. Thanks to the application, farmers, livestock breeders, and the Ministry will benefit from increased benefits and digital convenience. For the proposed software architecture, the future implementation was explained with the subheadings below:

Subsystem Decomposition

The purpose of dividing the system into smaller independent subsystems for the purposes of development, maintainability and scalability, are described in this section. It allows us to focus on certain functionalities like user registration or weather alerts, but at the same time making these two interact seamlessly.

Hardware/Software Mapping

The hardware and software components are joined in a way to support the operations of the application using this part. This complete configuration guarantees, choosing the proper tools and platforms for data storage, processing as well as user interaction, so that the system has the best performance and reliability.

Persistent Data Management

As with previous sections, this section addresses strategies for storing and managing the system's data reliably over time. On the other hand, it ensures all important information like user inputs, weather inputs, and system generated insights is being saved securely, easy to retrieve and safe from loss.

Access Control and Security

In this topic, we focus on guarding the system against the access and safeguarding the user's data safety. Being roles, permissions, encryption methods and monitoring protocols that help maintain system integrity and keep the users trust.

Boundary Conditions

Describes the events when interacting with external systems or what to do when the network fails, and for example if the input data is invalid. It guarantees the robustness, if possible defined fallback mechanisms, compliance with standards and resilience to anomalous conditions.

3.2 Subsystem Decomposition

System breaks down into several modular subsystems to focus each separately, so that maintainability and scalability is preserved. Each subsystem has a component that can work both independently and collaboratively within the system. In general, communication between the components is usually done through API calls or with using other inter process communication.

3.2.1 User **Management Subsystem (Frontend & Backend)**

This subsystem is responsible for all forms of user interaction process ranging from registration, login and logout and profile management including access to data.

Registration Component: It manages new user registration, collects and validates information about the user in order to enroll it in our database.

Password Recovery Component: It grants a means for users to forget their password when they forget it.

Login Component: It validates user's credentials by comparing it with the stored information and authenticates considering it if it is ready to be used.

Profile Management Component: Under this component's authorization users can view, update and manage information.

User Database: It stores user information securely.

3.2.2 Interactions

With Database: Database is a place where data about user information is stored and the backend interacts with the database to fetch data.

Session Management: Database also keeps track of the time spent on the system and the users activity-inactivity status.

3.2.3 Data Entry Subsystem (Frontend & Backend)

This subsystem comprises crop and livestock data management and processes user input related to them.

Product Data Entry Component: Interfaces for user input and management of data for a user's crop and livestock.

Data Storage Component: It takes progress data and puts it in the database.

Data Access Component: It manages to retrieve data and update data on the database.

Database: Database System to store all data of the application.

3.2.4 Crop Tracking Module:

Constantly stores some information users input regarding their crops, including type, expected harvest date and health conditions.Works by integrating with the notification system and crop growth tracking to notify a user when it's time to plant, fertilize and harvest.

3.2.5 Livestock Tracking Module:

Keeps information of livestock like type of species, breed, age, health conditions and immunization records. Livestock productivity will be tracked utilizing certain criterias such as milk production of a cow, feeding schedules and changeable traits (their weight or age). Moreover, it integrates with the notification subsystem to notify the user of to remind them immunization, health problems etc.

Interactions: It is integrated with the notification subsystem to send alerts for planned activities on farms or for possible problems.

3.2.6 Weather Subsystem (Backend)

This subsystem takes weather information from an external weather API and presents it to the users.

Weather Data Fetcher: Receives current weather data utilizing the OpenWeatherMap API. **Weather Notification Module:** Informs the users of severe changes in weather conditions. Uses external climatic web services to obtain the data.

Weather API Integration Component: Is responsible for interfacing with the external weather API's e.g. OpenWeatherMap. It comprises request making, response management, as well as the management of API keys.

Weather Data Processing Component: Processes the data which is obtained from the API then converts the data into format that can be of benefit to the application.

Weather Display Component: Displays the weather information to the user by use of the user interface.

Interactions: Users will be notified by using the Notification Subsystem.

3.2.7 Support Management Subsystem (Frontend & Backend)

This subsystem concerns the records of support that the Ministry offered and to which farmer applicants responded.

3.2.8 Components:

Support Display Component: Shares information on the available support programs for the farmers.

Issue Flagging Component: Enables ministry employees to raise concern or make complaints concerning particular support programmes.

Issue Management Component: Manages the reported problems part of it, this lets you track these issues in a certain workflow that leads to their resolution.

The Support Data Fetcher: Component that opens support information from the Ministry's Application Programming Interface in real-time.

3.2.10 Notification Subsystem (Backend)

Responsible for managing alerts as, support alerts, weather updates and extreme weather alerts, situations on, crops and livestock.

Notification Generation Component: Builds text messages to match particular occurrences or signals for safe keeping.

Notification Sending Component: Responsible for the delivery of notifications to users every time using phone, in app, or email notification.

Weather Notification Module: It offers the necessary precautions in cases of weather disturbances.

Support Notification Module: It also notifies users in case of any update or modification of the support programs.

Interactions: It gives the weather based alert and also backs up data alteration by interfacing with the back end.

3.2.11 Data Management Subsystem (Backend)

This Subsystem manages raw and processed data storage and retrieval.

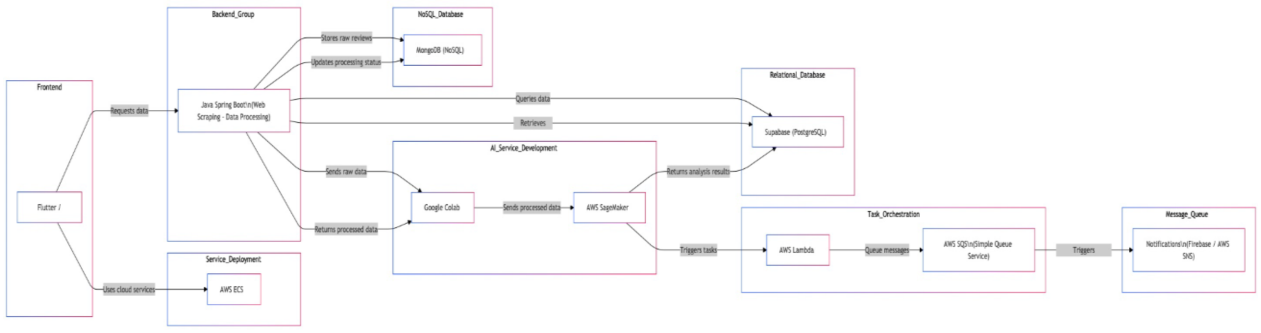
Apart from that, another component is a NoSQL database, which will store raw input data provided by the user or coming from other APIs.

Also, the refined data and analysis outcome can be stored in a relational database.

The data access layer is responsible for database queries and transactions.

Interaction: This ensures data coherence by interacting with every other subsystem.

3.3 Hardware/software mapping



3.3.1 Data Storage

Relational Database: Supabase (PostgreSQL)

It is being used to store processed review data, analysis findings, client information, product information, and other relational data.

Deployment Details:

Supabase is a fully managed PostgreSQL database service that offers simple backend integration and real-time functionality. For structured data which includes processed data and relational models, it guarantees dependable storage and also quick queries.

3.3.2 NoSQL Database: MongoDB

It is being used to archive semi structured data primarily obtained through web scraping and untainted and unprocessed review data.

Deployment Details:

MongoDB is a NoSQL document database which is incredibly versatile and suitable for semi structured data. This is particularly excellent for pre-processed raw reviews since it has the advantage of scalability.

3.3.3 Backend (Java Spring Boot)

It is being used to manage data processing, web scraping, and backend services and APIs.

Deployment Details:

The Java powered Spring Boot framework is used to create the backend as it provides for scalability, dependability, as well as multi-layered architecture.The backend also uses AI services for more data analysis, for extraction of raw data from MongoDB, and for uploading of processed data into PostgreSQL.

3.3.4 AI Service Development (AWS SageMaker, Google Colab)

It is being used to develop, train, and distribute AI models for data analysis and the Natural Language Processing (NLP).

Deployment Details:

Google Colab: For quick model development and prototyping. It offers a productive and cooperative setting for testing the AI models.

AWS SageMaker: It is ideal for model deployment at high levels and for large scale analysis. Processed data returns to PostgreSQL with the help of SageMaker, AI-generated decision-making processes are being supported.

3.3.5 AWS Lambda Functions (Scheduler Subsystem)

It is being used to orchestrate data flow and schedule tasks using queue-triggered serverless functions. It uses AWS Lambda, AWS SQS (Simple Queue Service)

Deployment Details:

AWS Lambda and AWS SQS: AWS SQS is a safe and reliable queues that assist in invoking AWS Lambda serverless functions. For precise functions like triggering web scraping, data processing or processing of natural language, Lambda can be used For businesses where there are time sensitive actions, AWS is good since it offers automatic scaling and efficiency.For message coordination, AWS uses SQS to send Lambda functions initiating subsequent processes properly including the schedule of the processes and how they will be executed.

3.3.6 Frontend

Technologies: Flutter (mobile app)

Purpose: It is being used to deliver the user interface and enable interactions with the backend.

Deployment Details:

Flutter: It is a cross-platform framework for creating iOS and Android mobile apps that function like native apps.

3.3.7 Service Deployment (AWS ECS (Elastic Container Service))

It is being used to deploy backend services and other microservices using a container-based solution.

Deployment Details:

The ability of the ECS to have a portable and versatile deployment setup allows the backend to run in Docker-contracted containers. Get as a result leverage AWS infrastructure to ensure scalability and reliability are not compromised.

3.3.8 Message Queue (Firebase Cloud Messaging, AWS SNS (Simple Notification Service))

It is being used to send notifications to users and manage system messages effectively.

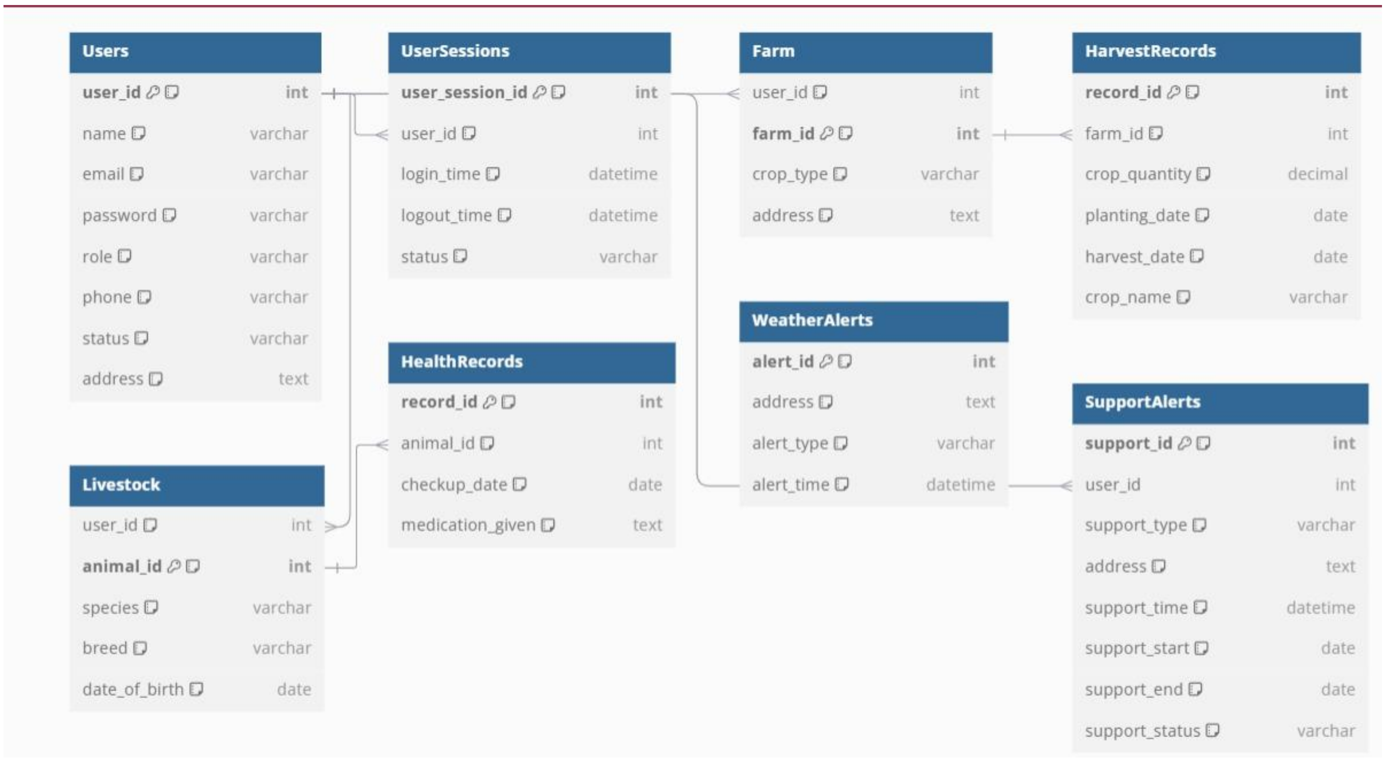
Deployment Details:

Firebase Cloud Messaging: Enhances user engagement by delivering real-time push alerts to mobile devices. Here, AWS SNS offers a dependable system-level messaging and alerting architecture, guaranteeing prompt transmission of important data.

3.4 Persistent Data Management

Persistent data management shows that data is protected and stored, and what techniques are being employed, even after apps have terminated. Files, databases, and other permanent storage systems are used to accomplish this**.**

1. **Database Schema:**



This table has been used to explain the database schema and make it more understandable to the readers of this report. Moreover, to make everything more visible the sense of the table is explained below:

Table: Users

• **user_id:** It is an individual user’s unique identification**.**

• **name:** The user's complete name.

• **email:** Email used for authentication and communication with the user.

• **password:** An encrypted password for securing authentication.

• **role:** It defines who is the user in this system such as Ministry Employee or Farmer.

• **phone:** In the case of multi factor authentication or contact, the user's phone number.

• **status:** It indicates whether or not a user account is active or inactive at the time.

• **address:** It tracks the user's address.

Table: UserSessions

user_session_id: It uniquely identifies every session.(Primary Key)

user_id: Assigns a unique user to the session. (Foreign Key)

login_time: It mainly records the time at which a certain session for the login time begins.

logout_time: Time at which the session was logged out

status: Gives the indication if a session is in progress or it is not.

Table:

Farm user_id: links the farm to its owner. (Foreign Key)

farm_id: For each farm, a unique identifier will be used. (Primary Key)

crop_type: It specifies what type of crop it is growing.

address: It is the farm’s physical address or location.

Table: HarvestRecords

record_id: Unique identifier number for each harvest record. (Primary Key)

farm_id: Links the record to a specific farm. (Foreign Key)

crop_quantity: The quantity of crop harvested.

planting_date: The date the crop was planted.

harvest_date: The date the crop was harvested.

crop_name: The name or type of crop planted.

Table: Livestock

user_id: It connects livestock data and owner. (Foreign Key)

animal_id: A unique animal identifier for every animal. (Primary Key)

species: The number of the animal (e.g., one, two, three, etc).

breed: The breed of the animal. date_of_birth: The birthdate of the animal.

Table: HealthRecords

record_id: This is a unique indicator of the health record. (Primary Key)

animal_id: Buys the health record to a specific animal. (Foreign Key)

checkup_date: Date of the health checkup.

medication_given: Any medicine given noted.

Table: WeatherAlerts

Alert_id: Unique identifier for each of alerts. (Primary Key)

Address: It specifies where the alert location is relevant to.

Alert_type: Tells the type of the weather alert (e.g. storm, frost).

Alert_time: When the alert was issued.

Table: SupportAlerts

Support_id: A unique identifier for every support entry (Primary Key)

Support_type: Represents the type of support that the support provider gave, e.g., money, or material.

Support_time: The time when the support entry was written.

Support_start: The start date of the period for which the support was given.

Support_end: The date when the support was stopped.

Support_status: Indicates whether there is ongoing or completed help.

Data Storage Solutions

Robust and scalable solutions are required in data storage for the system. To keep and manage several types of data (such as structured, semi-structured, and potentially unstructured data) the following approach is taken to ensure flexibility, efficiency, and security:

Primary Storage: A relational database will store structured data such as user accounts, crop records, animal information. Given this, PostgreSQL is good for this relational database as its ACID properties (Atomicity, Consistency, Isolation, Durability), data integrity, reliability, and plain query strong capabilities.

Secondary Storage: Relational database will serve as the core of the application; NoSQL database will also act as a choice of expansion.

Basic system: If a system integrates with IoT devices or semi structured or unstructured data No SQL Database (MongoDb) can be applied. Data Backup and Recovery Plans are very common practices nowadays. System requires that data be consistent. Data loss risk will be minimized using a comprehensive backup and recovery plan.

Backup Strategy: As part of the data integrity measures, both the relational(PostgreSQL) as well as the NoSQL database are daily auto backed up to avoid loss of data.

Recovery: Data restoration from backups will be automatic with scripts and recovery tests will run periodically to validate the process.

Data Processing and Access:

Storage Workflow First it stores raw data that is available from external APIs or from user input in a NoSQL database, which must be easily reached. The data is then transferred from the compiled and verified data to a relational database for storage and querying for a long term.

Data Access Methods

Data will be securely retrieved by APIs, and system modules and user dashboards will use these APIs. The optimized queries and indexing will be able to simplify easy and fast access to constantly utilized datasets.

Data Processing Workflow

It will be converted and verified data input so that it satisfies schema requirements. Processed data is analyzed by analytics modules and AI algorithms, to produce actionable insights.

3.5 Access Control and Security

To avoid unauthorized users to access the system we can adopt Access Control and Security procedures, Data Protection in Transit and at Rest.

Authentication: Authentication proves the identity of a user.

• **Username and Password:** To log in, users can use both email or their phone number with any chosen password.

• **Two-Factor Authentication (2FA):** There is a layer of security included that involves a user being required to confirm they are a person via email or phone, and 2FA login during authentication.

Authorization: It aims to refer to what actions a user has permission to carry out within the system**.**

• Role-Based Access Control (RBAC): This is an accessibility procedure which is not policy based but role and responsibilities specific, especially in the organization.

1. Farmer: Gives input data like weather, crop and livestock, and view and for government help**.**

2. Ministry Employee: It helps manage users and generates reports on government assistance related issues and reports.

3. Administrator: It has a unique responsibility to manage the system, including management of user roles, settings and permissions of individual users.

Data Protection:

• Encryption:

At Rest: Encryption will be utilized in the database using AES-256 to dissuade unauthorized access.

In Transit:

· Data that has been transferred by the server to the clients happens in the form of SSL (Secure Socket Layer) and TLS (Transport Layer Security).

· All API connections will be through HTTPS.

Monitoring and Auditing:

• Activity Logs: It is used in order to call people into responsibility and to solve problems major activities take place like trying to log in or change something.
• Automated Alerts: Appears in response to suspicious action for instance multiple login attempts, unauthorized access, etc.

System Resilience:

Mechanisms for Failures: If there is an attack or the system has shut down, a backup system ensures that the system will be able to run.

Periodic Security Reviews: In individual audits and vulnerability tests, which are carried out systematically, will help to detect weaknesses.

3.6 Global Software Control

The ALSApp global control ensures all subsystems, data flows, sync, and performance

optimization. Here’s how global control works:

3.6.1 Workflow and Request Management

User interfaces (farmers, livestock breeders, ministry officials) send the requests to system and the system goes through a workflow to process the request.

Request Sources: The mobile app allows farmers to input crop and livestock data and getting help or weather info. Farmers can submit help or check system status from their portal.

API Gateway: Requests are routed through a central API gateway that dispatches requests to the appropriate backing service (weather, livestock tracking, support management). Used for secure communication between frontend & backend using token-based authentication.

3.6.2 Subsystem Synchronization

The interface of ALSApp consists of multiple subsystems that perform different functions, but they are linked through API calls and message queues for fine grained synchronization.

Weather Monitoring Subsystem: It receives weather data regularly from external APIs and this subsystem synchronizes it with crop and livestock records. Moreover, this subsystem is totally asynchronous so that weather updates can keep running in the background without blocking the user's work.

Crop and Livestock Management: Input from users (i.e health analytics, crop yield forecasting) is processed and stored in MongoDB NoSQL database; migrated to PostgreSQL for storage and reporting.

Support Services: A queue-based architecture (AWS SQS) is used for tracking support requests in ministry programs, so that the system processes high priority support requests without over-complicating the system.

3.6.3 Concurrency and Data Consistency

Concurrency is key to sustain performance of ALSApp.

Concurrency Management: Data entry, weather updates and livestock health tracking are done by task queues and worker threads and rate limiting is used to ensure API isn't overloaded during peak hours.

Data Consistency: The system is using transactional protocols for data consistency across the databases, we have locking in place when the critical data updates are being written by multiple database servers concurrently to prevent concurrent data update conflicts.

3.6.4 Synchronization Mechanisms

Smooth data flow and synchronize with users when needed.

Message Queues: Weather, livestock, and support subsystems are connected by a distributed message queue (AWS SNS) and notifications (i.e. weather alerts, livestock health updates) shared between the two synced via Firebase Cloud Messaging for real time delivery.

Scheduler Subsystem: AWS Lambda functions can be used to schedule a periodic task, for example automated health check of livestock but also crop growth tracking. Such tasks are triggered either according to the weather forecast or predefined intervals for proactive monitoring.

3.6.5 Error Handling and Recovery

The service will be kept up with the error handling mechanisms for system anomalies.

API Error Handling: For 500 Internal Server Errors the system retries failed requests with exponential backoff. For audit purposes error logs are kept.

Failover and Redundancy: If some servers fail, then services will fall back to backup servers. On the other hand, data is always backed up to prevent lost data and in case of unexpected downtime recovery.

ALSApp covers all critical components, including the productivity, data integrity and user experience for farmers, livestock breeders and ministry officials.

3.7 Boundary Conditions

It explains how the system interfaces with the outside world and how best to handle conditions when things go wrong to ensure that systems stay optimised and reliable.

External System Interactions

Weather API: Meteorological forecasts and current information on the preparation of farms are obtained out of derived predictions by using it. To get the live weather information it incorporates the Accu-Weather API. It is done authentically through secure API keys safely built in the system They include the following; Ministry Support API: It also analyzes and retrieves information about governmental support programs and facilities. Validates flagged issues. Some of them keep around those monitors which provide aid to farmers. Secure access with HyperText Transport Protocol secured using tokens as means of authenticating users.

MapsSDK for Android API: To explain its use with this API it is used to get a map off of google maps, which we will integrate later on, a map of whatever current location we happen to be in. Moreover, it shows specific markers near Farm/User village only.

Places API (Google Places): This makes it possible for the users to search for a specific place which is a farm, field or nearest weather station. Moreover, for users typing the search queries (for example, ‘Konya’) display the recently popular places nearby. Provide you this quite uncomplicated scope like latitude, longitude and administrative division based on the users input or properties of chosen places.

Edge Cases and Error Handling

Network Failures: It combines cached data with notifications which inform the users of the delay if the external API is not working because of network breakdowns. This extension implements retry logic with exponential backoff API calls.

Data anomalies: data that has been obtained from external APIs which will undergo anomaly validation meant for detecting various errors such as whether the timestamp is either invalid or the values are out of range. Those anomalies will be logged to make administrations notify and fix, before they are fixed an appropriate error message will be displayed.

API Errors: It will also manage HTTP status codes (401, 403, 404, 500 etc) and return appropriate error messages to the client. They will serve as error logs for checking and troubleshooting in case of problems.

Hardware Failures (Server-Side): In order to solve the problem of frequent cases of server hardware failure, failover to backup servers /redundant systems will be instated. It should be mentioned here in more detail in the Deployment section.

Client-Side Errors: It should detect and manage client-side anomalies (crashes, unanticipated input) without data loss and give clear messages to the client.

Compliance and Standards

• **Data Format Standards:** For the communication with other APIs the system will be able to work with JSON, XML, etc as other formats that are used in most applications. This makes it possible to interact with different systems.

• **Security Standards:** All data will be transmitted in encrypted form using SSL/TLS – HyperText Transfer Protocol (HTTP) Secure). To ensure users' data privacy is protected the application will be designed to be GDPR compliant.

4. Subsystem Services

Subsystems in ALSApp are structured applications made up of services meant to deal with functions within the application. Naturally, these services work standalone for better scalability and maintainability and interacting with the external services and internal applications. The following are the major ALSApp architecture subsystem services that have been put in place.

4.1 Authentication Service

This subsystem manages user authentication and authorization

Features:

4.1.1 User login, registration, and password recovery

The user will be able to log in to the application with the information stored in the database; if the user is not registered in the application, the user will be directed to the interface created for registration by pressing the ‘Sign In’ button. Moreover, if the user has forgotten his password, he will be directed to the password renewal interface by pressing the ‘Forgot password’ button.

After filling in the required personal information in the form, the user will be able to register the application by pressing the ‘Sign In’ button.
They will be able to recover their password by entering their email or telephone address because they have forgotten their password, and they will receive the verification code required when they press the “Send Code” button for two-factor authentication.

4.1.2 Token-based authentication for secure API access

This mechanism helps to provide safe and effective communication with a client and a backend server. Upon successful authentication, the server provides a token that will be used in other requests. It is forwarded to the client application and securely stored away. In each of the HTTP requests, the client sends the token in the Authorization HTTP header for every API call. The server examines the token for its genuineness and then checks. If so, the request goes on and if not, the server returns a 401 Unauthorized error.

4.2 Weather Service

This subsystem fetches real-time weather data for farmers and livestock breeders.

Features:

4.2.1 Retrieves localized weather forecasts and alerts via APIs

The weather monitoring for farmers and livestock breeders aims to provide instant, precise, and reliable weather monitoring. For this, weather information will be updated using instant information obtained from the internet. Users will be notified of possible hazards or weather events according to their importance.

4.2.2 Displays data in a user-friendly interface

The system will allow the user to check the weather conditions by touching the “Weather” button on the interface. Instead of showing the weather window at first, the “Weather” button will navigate the user to an easy-to-use interface where the current weather conditions can be seen.

4.3 Crop and Livestock Management Service

This subsystem allows users to track and manage their crops and livestock.

Features:

4.3.1 Allow data entrance and record details about crops and livestock

The system should enable the user to enter and follow the data of crops and animals. If there is any change, such as a decrease in wheat production, the user should be warned about this change. In the app, the user will see a window where they can input the product name and characteristics by clicking the "Enter Product" button. The system will check the data entered by the user in the product form; thus, all necessary fields must be correctly filled out. If data is missing or contains factual errors, the system shall send a plain text message describing what should be filled in or corrected. After the entrance the system will save product data to the database and store the usual data entries in the database, explaining and covering the animal tracking and routine status data.

4.3.2 Provide analytics and historical data for better decision-making

The system shall then perform statistical data analysis in order to foresee situations that are likely to be discovered in the future or unfavorable trends. It will evaluate data changes that occur over time to determine whether they reveal a potential hazard or have a negative impact. If the information analysis determines a dangerous scenario or a negative impact, the system shall notify the user and advise them on possible risk management.

4.4 Notification Service

This subsystem notifies users about critical updates and reminders.

Features:

4.4.1 Weather alerts, crop management tasks, and livestock health updates.

If the weather is unsafe or inappropriate, the system will alert the user as a preventative measure by sending notifications. It will provide the user with information on what environmental conditions, such as humidity and temperature, are suitable for activities like planting or other sensitive atmospheric phenomena and even diseases.

4.5 Support Service

This subsystem facilitates communication between farmers and agricultural/livestock support agencies.

Features:

4.5.1 Submit support requests to the Ministry of Agriculture or other agencies.

The application aims to use a convenient system for displaying support to facilitate access to the Ministry’s support. The data will be taken directly from the Ministry of Agriculture and Forestry and will be added to the database of ALSApp immediately; throughout this platform, livestock breeders and ministry employees will be able to be aware of the possible supports by using this feature to access support more easily since it’s even hard for the ministry employees to be aware of the support that they are giving because of the current complicated system.

4.5.2 Track the status of requests and receive responses.

This feature ensures that farmers or livestock breeders can track their submitted requests and also get updates on the progress or status of these requests, such as whether a government support application is under review, approved, or denied. This feature aims to make the application procedure easy for the users with an increased transparency and accountability. And also keeps the user informed without needing to check the requests manually multiple times.

4.6 Analysis Service

This subsystem provides insights using machine learning for predicting the crop yield and also livestock health analysis.

Features:

4.6.1 Predict weather impact on crops and suggest best planting times.

This feature makes use of the weather data to help farmers by making predictions about the weather impacts. This feature aims to improve crop yield by optimizing planting decisions and to reduce the risks that are caused by the severe weather conditions. It will look at conditions such as rainfall, frost or drought and determine how favorable such climate will be for certain particular crops or crop varieties.To this it will make use of the best planting times by use of historical and real-time weather data, especially in relation to factors such as soil moisture, temperature, and precipitation level.

4.6.2 Diagnose livestock diseases from reported symptoms.

Farmers can enter the observed symptoms in their livestock such as fever, lethargy, tuberculosis or even trivial symptoms like appetite loss , into the system. This feature aims to reduce time and cost associated with diagnosing livestock issues and to help prevent the spread of infectious diseases by early diagnosing. To do so, it pairs these symptoms with a list of all known diseases to give potential results for a diagnosis. It will provide recommendations on quick preventative measures that should be taken to eliminate a hazard such as quarantining a sick animal or calling a vet. Suggest preventives depending on the range of diseases occurring in different regions. Such symptoms as coughing and nasal discharge when a farmer presents his/her animals for inspection. The system may provide diagnoses like a possible respiratory infection, check for pneumonia.

5. Glossary

ALSApp: Agriculture and Livestock Support Application, a digital platform for farmers and livestock breeders.

Application: A software program designed to perform specific user tasks, often used on mobile devices or computers.

API (Application Programming Interface): A set of protocols and tools that allow different software systems to communicate and share data.

Alert System: A feature that notifies users of critical conditions, such as extreme weather, pest outbreaks, or equipment malfunctions.

Livestock: Domesticated animals raised in agriculture for food, fiber, or labor. Examples include cattle, sheep, and poultry.

Real-Time: Data or information that is updated and available immediately as events occur.

Interface: The point of interaction between the user and the application, often designed to be user-friendly and intuitive.

Crop: Cultivated plants grown on a large scale for food, fiber, or other commercial purposes. Examples include wheat, corn, and rice.

Crop Tracking: The process of monitoring and managing crop growth, health, and yield using digital tools.

Sequence: The specific order in which tasks, processes, or events occur within a system or workflow.

Stimulus: An external factor or input that triggers a response in the system, such as a user action or environmental change.

Flagging: The act of marking or identifying data, issues, or conditions that require attention or action, such as potential errors or anomalies.

Integration: The ability of the application to connect and work with other systems, such as external weather APIs or financial platforms.

Compatibility: The application's ability to work seamlessly with different devices, platforms, or systems without conflict.

Scalability: The capacity of the system to handle an increasing amount of work or adapt to growing user demands without compromising performance.

Maintainability: The ease with which the system can be updated, repaired, or enhanced to ensure it remains functional and relevant over time.

Factual Errors: Mistakes or inaccuracies in the data or information provided by the system that can impact its reliability or decision-making.

Yield: The amount of produce harvested from a crop or the output from livestock, often measured per area or animal.

Humidity: A quantity representing the amount of water vapor in the atmosphere or in a gas.

OpenWeatherMap: weather data service that provides real-time and historical weather information through a RESTful API.

Front-end: It refers to the user-facing part of the application – the interface that users interact with directly. It is responsible for displaying data, collecting input, and ensuring a smooth user experience.

Back-end: It refers to the server-side part of the application. It is responsible for data processing, storage, and the logic required to handle user requests. The backend is not visible to the user but powers all the frontend operations.

Fetcher: It is a component or module in a system responsible for retrieving data from an external source, such as an API, database, or file system. It acts as the intermediary between the system and the external data source, ensuring that data is fetched in the correct format and transferred securely.

NoSQL: It stands for "Not Only SQL" and refers to a category of databases that do not rely solely on the traditional relational database model. NoSQL databases are designed to handle a wide variety of data models, making them highly flexible and scalable.

Partially structured data: It means something that doesn't follow the tabular structure associated with relational databases or other forms of data tables

Two-factor authentication (2FA): It is an identity and access management security method that requires two forms of identification to access resources and data. 2FA gives businesses the ability to monitor and help safeguard their most vulnerable information and networks.

Role-based access control (RBAC): It is a model for authorizing end-user access to systems, applications and data based on a user's predefined role.

AES-256: The Advanced Encryption Standard (AES) 256 is a virtually impenetrable symmetric encryption algorithm that uses a 256-bit key to convert your plain text or data into a cipher.

SSL/TLS: It stands for secure sockets layer and transport layer security. It is a protocol or communication rule that allows computer systems to talk to each other on the internet safely. SSL/TLS certificates allow web browsers to identify and establish encrypted network connections to web sites using the SSL/TLS protocol.

HTTPS: It is the primary protocol used to send data between a web browser and a website. HTTPS is encrypted in order to increase security of data transfer. This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.

Token-based authentication: It is a protocol that generates encrypted security tokens

MQTT: It is the standards-based messaging protocol, or set of rules, used for machine-to-machine communication.

500 (Internal Server Error) 500 HTTP: It is the status code that means requesting a URL is not fulfilled because the server encounters an unexpected condition. It gives information about the request made if it is successful, and throws an error. When there’s an error during a connection to the server, and the requested page cannot be accessed then this message is displayed.

404 (Not Found) 404 HTTP: It is the status code that appears when you request a URL and then the server has not found anything. This happens when the server doesn’t wish to describe the reason why the request has been refused. Also, the server is not able to find a representation for the target resource.

6. References

[1]: Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2]: Mermaid Chart. (n.d.). *Mermaid Chart.* Retrieved December 22, 2024, from https://www.mermaidchart.com/

[3]: Bruegge, B., & Dutoit, A. (n.d.). System design. Technical University of Munich. Retrieved from https://ase.in.tum.de/tramp.globalse.org/doc/presentations/system_design2.pdf

[4]: Gülyaşar, D. (n.d.). 1. Flutter Hakkında Bilgi - Flutter Dersleri 2023. YouTube. https://www.youtube.com/watch?v=eS9R35nHpfk&list=PLJbRSPP1eDeWJO9qKA7gI6eExO6ta44pc&index=1

[5]: Google. (n.d.). *Flutter: Build apps for any screen.* Retrieved December 27, 2024, from https://flutter.dev/

[6]: GeeksforGeeks. (n.d.). What is web scraping and how to use it? Retrieved from

https://www.geeksforgeeks.org/what-is-web-scraping-and-how-to-use-it/

[7]: AccuWeather. (n.d.). *AccuWeather APIs.* Retrieved December 27, 2024, from https://developer.accuweather.com/

[8]: Google. (n.d.). *Google Cloud Console.* Retrieved December 27, 2024, from https://console.cloud.google.com

[9]: Wikipedia contributors. (n.d.). *Separation of concerns.* Wikipedia, The Free Encyclopedia. Retrieved December 27, 2024, from https://en.wikipedia.org/wiki/Separation_of_concerns

[10]: Dede, E. S., Demirci, S. D., & Ünsal, G. (2024). *ALSApp: Agriculture and Livestock Support Application - Analysis Report*. TED University, Department of Computer Engineering.

[11]: Google. (n.d.). *Flutter API documentation.* Retrieved December 27, 2024, from https://api.flutter.dev/