Theoretical graduation project

*Application of Interest in Agriculture and Plant Knowledge*

Submitted in partial fulfillment of the requirements for the award of bachelor's degree of the University of Bisha

Student Affairs System for College of Computing & Information Technology Department of Information Systems

**Index:**

**CHAPTER 1:** Introduction to The Project \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1

1.1 Introduction \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2

1.2 Importance of The Project \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2

1.3 Project Objectives\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

1.4 Target Audience \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3

1.5 Application Features \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

1.6 General Structure of The Application \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4

**CHAPTER 2:** Requirements Analysis and Design \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6

2.1 Functional Requirements Analysis \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7

2.2 Non-Functional Requirements \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8

2.3 User Interface Design (UI/UX) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8

2.4 Database Design \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9

2.4.1 Plant Details \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9

2.4.2 Weather Table \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

2.4.3 Users Table \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

2.5 System Design and Overall Architecture \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

2.5.1 General Structure Plan \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

2.5.2 Data Flow Diagram \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

2.5.3 Use Case Diagram \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12

2.6 Schematic diagrams \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

2.6.1 Entity relationships (ER) Diagram of Database\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

2.6.2 Plant Search Sequence Diagram\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_14

2.7 Tools and Techniques \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_14

2.8 Expected Challenges\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_15

**CHAPTER 3:** Application Development\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_16

3.1 Preparing for The Development Environment \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_17

3.2 Building the Database \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_17

3.3 Front-End Development \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_18

3.4 Backend Development \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_19

3.5 Integration with APIs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20

3.6 Adding Additional Features\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20

3.7 Initial Test \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_21

3.8 Code Documentation\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_21

3.9 Release Management\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_21

**CHAPTER 4:** Application Testing and Optimization \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_22

4.1 Types of Tests \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_23

4.2 Functional testing \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_23

4.3 User Interface Testing \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_24

4.4 Performance Testing \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_24

4.5 Security Testing\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_25

4.6 Performance improvement \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26

4.7 Bug fixing \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26

4.8 Final Exam \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_27

4.9 Test Report\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_27

**CHAPTER 5:** Application Launch and Maintenance \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_28

5.1 Launch the application \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_29

5.2 Marketing Campaign\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_30

5.3 Collecting Feedback \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_31

5.4 Periodic Updates \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_31

5.5 Ongoing maintenance \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_32

5.6 Data Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_32

5.7 Future Expansion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_33

5.8 Conclusion\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_33

5.9 Html code\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_34

6.0 References\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_43

**Tables:**

Table.1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9

Table.2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

Table.3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10

**Figures:**

Figure 2.1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

Figure 2.2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_11

Figure 2.3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12

Figure 2.4 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_13

Figure 2.5 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_14

**CHAPTER 1**

Introduction to The Project

**1.1 Introduction**

With the acceleration of technological development and individuals' increasing reliance on smart applications to facilitate their daily lives, the need for innovative technological solutions to support vital sectors such as agriculture has emerged.

The "Interest in Agriculture and Plant Knowledge" application is in response to this trend, it offers an integrated digital platform that helps users, both amateurs and professionals, identify plants, care for them, and provide personalized advice based on geographic location and climatic conditions.

The aim is to combine technology and agricultural knowledge in a simplified manner that contributes to increasing green spaces and improving individual agricultural practices.

**1.2 Importance of The Project**

With rapid technological development, smart applications have begun to play a majorrole in facilitating our daily lives, one area that can benefit from this technology is agriculture.

Whether you are aprofessional farmer or someone interested in home gardening, having accurate information about plants and how to care for them is essential to ensure successful farming.

This application aims to provide a comprehensive platform that provides detailed information about plants, including:

1. Suitable planting time
2. The amount of water required
3. Lighting requirements (sun, shade)
4. The right type of fertilizer
5. Tips for protecting plants based on weather

In addition, the app will help users choose the right plants for their environment, whether in the ground or in pots, making gardening easier and more efficient

**1.3 Project Objectives**

This project aims to achieve the following objectives:

* Provide curate information about plants:
* Provide details on planting time, watering, lighting and fertilizer.
* Providing images of plants when fully grown to help the user plan.
* Link the app to the weather:
* Use weather data to provide personalized plant protection advice.
* Provide notifications about weather conditions and changes that may affect plants.
* Facilitate plant selection:
* Help users choose the right plants for their environment.
* Providing advice on planting location (soil, pots) and appropriate container shape.
* Improving user experience:
* Simple and easy to use interface.
* Notification system to remind user of watering and plant care times.

**1.4 Target Audience**

The application is intended for the following categories:

* Hobby farmers:

People who are interested in home gardening and want to improve their skills, those who want to grow certain plants but lack sufficient experience.

* Professional farmers:

who need accurate information to improve the productivity of their farms, those who wish to expand their knowledge of plant types and ways to care for them.

* Environmentalists:

People who want to increase green spaces in their homes or communities who are looking for sustainable ways to farm.

**1.5 Application Features**

The application has many features that make it a useful tool for users, including:

* Comprehensive information: The application contains a large database containing information about many plants and the information is updated periodically to ensure accuracy.
* Custom tips: Advice provided based on user's location, weather conditions. Reminder notifications for watering and plant care times.
* Ease of use: Simple and intuitive user interface.

Ability to quickly search for plants using plant name or type.

* Weather integration**:** Use weather data to provide personalized advice. Notifications when weather conditions change that may affect plants.
* Multi-language support: The application can be used in different languages to meet the needs of users around the world.

**1.6 General Structure of The Application**

The application consists of the following main sections:

* Search Interface:

1. The user can search for plants using name or species.
2. Display a list of plants with thumbnails.

* Page Plant Details:

1. Information about planting time, watering, lighting and fertilizer.
2. Pictures of the plant when fully grown.

* Weather Section:

1. Personalized tips based on weather conditions.
2. Notifications when weather conditions change.

* Fertilizer and Tips Section:

1. Information on suitable fertilizer types.
2. Tips on where to plant (soil, pots).

* Notification System:

1. Watering reminder Notifications when weather conditions change.

**CHAPTER** **2**

Requirements Analysis and Design

**2.1 Functional Requirements Analysis**

These are the basic tasks that the application must perform to meet the needs of users.

* Find plants and view details.

Ability to search for plants using name or species. View details of each plant including:

1. Planting time (suitable seasons).
2. Watering quantity and timing.
3. Lighting requirements: Full sun, partial shade, etc.
4. Recommended fertilizer type.
5. Pictures of plants at different stages of growth.
6. Tips on where to grow (open ground, pots, tubs)

* Integration with the weather service.
* Use OpenWeatherMap as API to fetch weather data based on user's location.
* Provide personalized advice based on the weather:
* Warning of frost or extreme heat.
* Suggest changing watering times on rainy days.
* Notification system.
* Send watering reminders.
* Alerts when weather changes may affect plants.
* Fertilizer and care section.
* Explain the types of organic and chemical fertilizers.
* Tips on how to make homemade compost.
* Plant fertilization schedule.
* Support for choosing the planting site.
* Suggest the appropriate pot or basin shape for the plant.
* Tips on the distance between plants in the open ground.

**2.2 Non-Functional Requirements**

These are the general characteristics that ensure the quality and performance of the application:

* Performance: Response time is less than 2 seconds when searching or loading data Support up to 10,000 daily active users.
* Security: Encrypt user data such as bookmarks.

Use HTTPS to connect to servers and APIs.

* Ease of use: Intuitive user interface design with visual guidance Support multiple languages (Arabic, English, etc.).
* Compatibility: Works on Android and iOS systems.

Support for various smart devices (phones, tablets).

**2.3 User Interface Design (UI/UX)**

1. Search interface
2. Search bar at the top of the page.
3. List of common plants with thumbnails.
4. Filter options by plant type, lighting needs, season).
5. Page Plant details
6. Main image of the plant.
7. Tabs Irrigation Lighting Fertilizer Weather protection.
8. Add to Favorites button. See the "Similar Plants" section below.
9. Weather Section
10. Display current weather conditions (temperature, humidity, precipitation).
11. Warning notices (e.g.: “Tomorrow is hot, increase watering”).
12. Settings section
13. Language options.
14. Manage notifications(on/off) Manually or automatically determine the geolocation.

**2.4 Database Design**

A relational database (such as MySQL or Firebase) was used with the following tables:

**2.4.1 Plant Details:**

**Table.1** Plant Details.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| plant\_id | String (PK) | Unique identifier for the plant |
| name | Map | Plant name (Arabic/English).  Example: { "ar" : "Basil", " "en" : "Basil} |
| watering | String | Watering quantity and timing.  Example: "Every other day in summer. |
| sunlight | String | Lighting requirements.  Example: "full sun" |
| fertilizer | String | Type of fertilizer.  Example: "Organic fertilizer every two weeks |
| season | String | Planting seasons.  Example: "Spring Summer" |
| images | List | Links to plant images.  Example: ["url1", "url2"]. |

**2.4.2 Weather Table:**

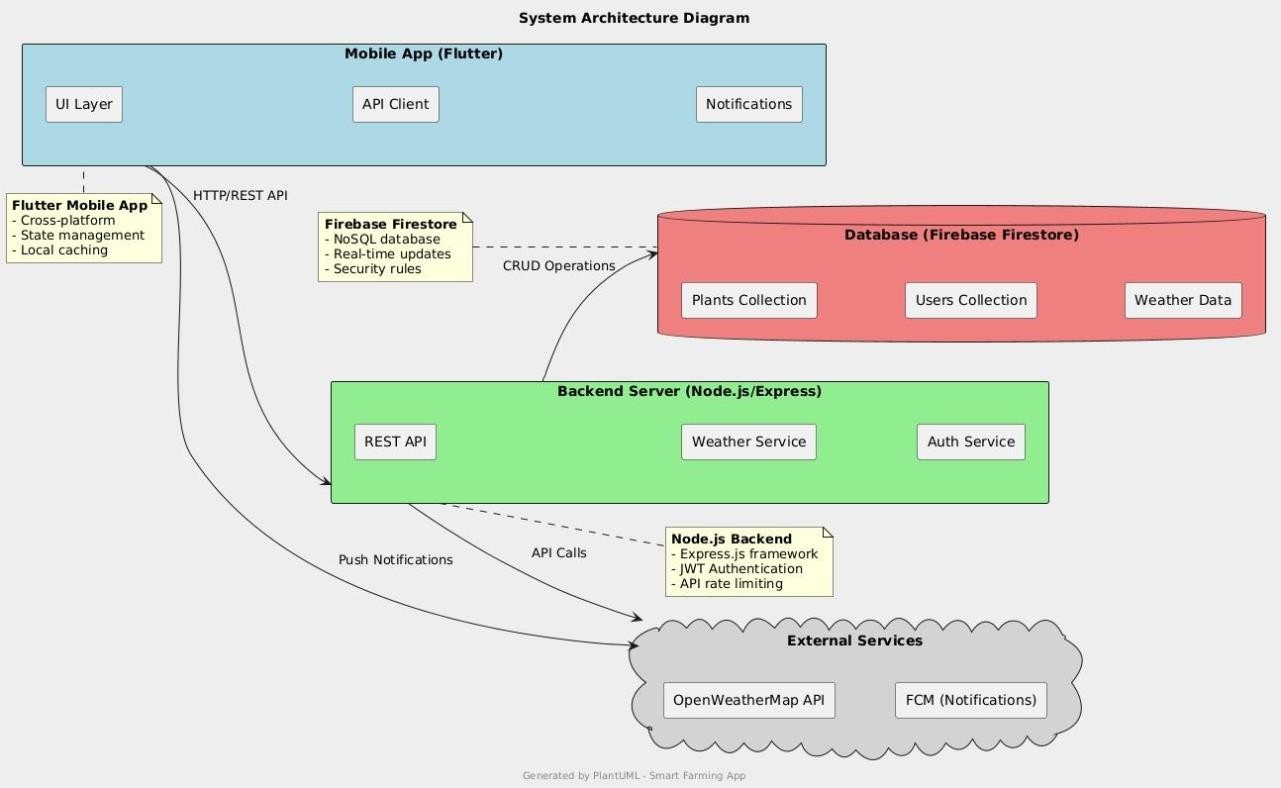
**Table.2** Weather Details.

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| location\_id | String(PK) | Location.  Example: "cairo\_eg" |
| temperature | Float | Temperature.  Example: 32.5. |
| humidity | Float | Humidity.  Example: 60. |
| alerts | String | Weather warnings.  Example: "There is no" |

**2.4.3 Users Table:**

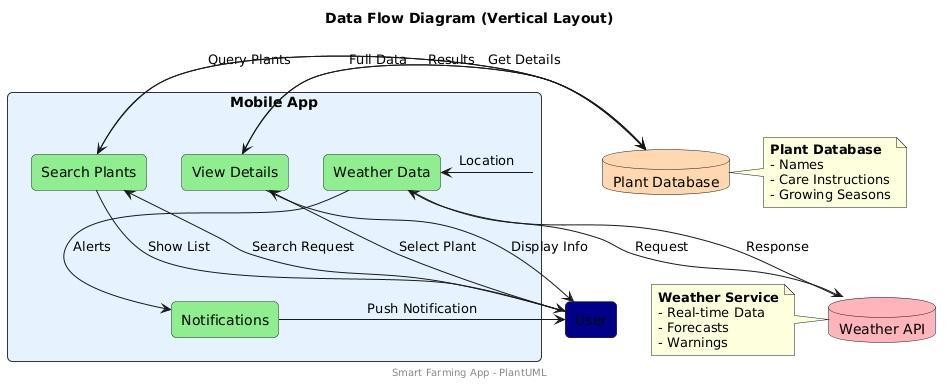
**Table.3** Users Details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Type** |  | **Description** |
| user\_id | String(PK) |  | Unique identifier for the user |
| favorite\_plants | List |  | List of favorite plants.  Example: .["005","001"] |
| location | String |  | the site |

**2.5 System Design and Overall Architecture**

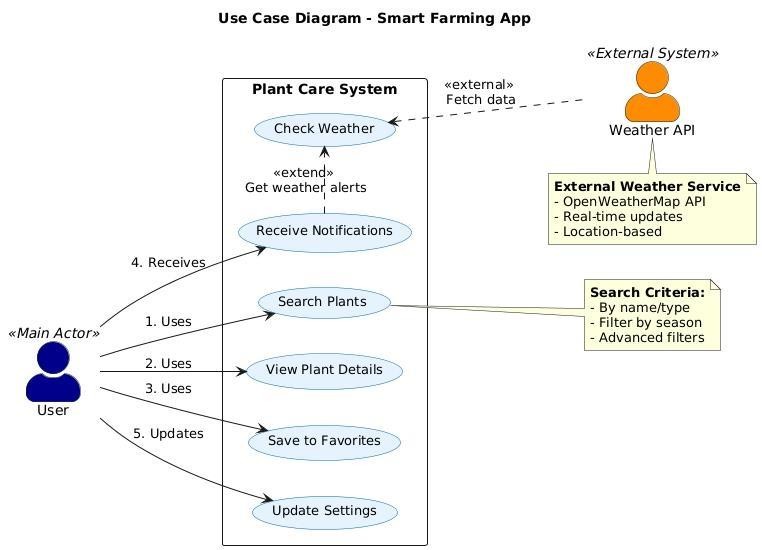
**2.5.1 General Structure Plan**

**Figure 2.1** System Architecture of *IAPK*

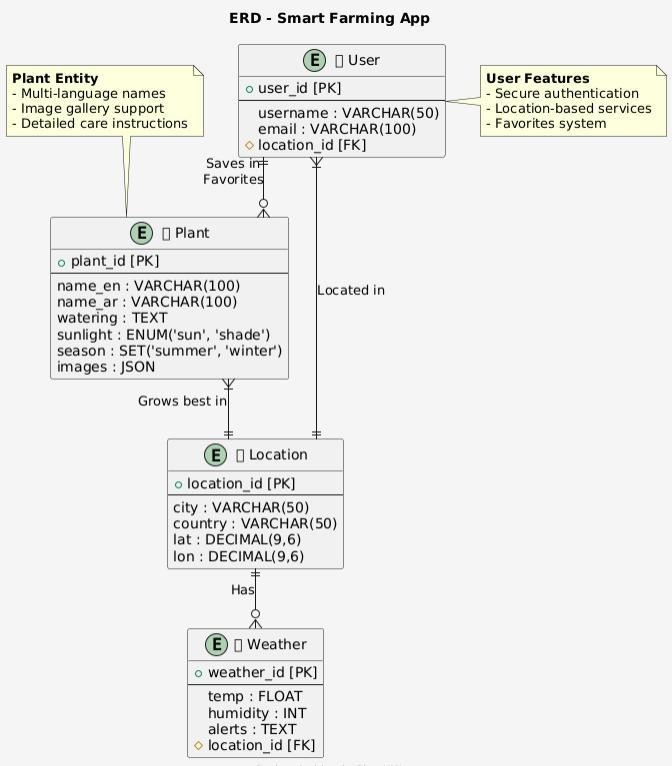
**2.5.2 Data Flow Diagram**

**Figure 2.2** DFD of *IAPK*

**2.5.3 Use Case Diagram**

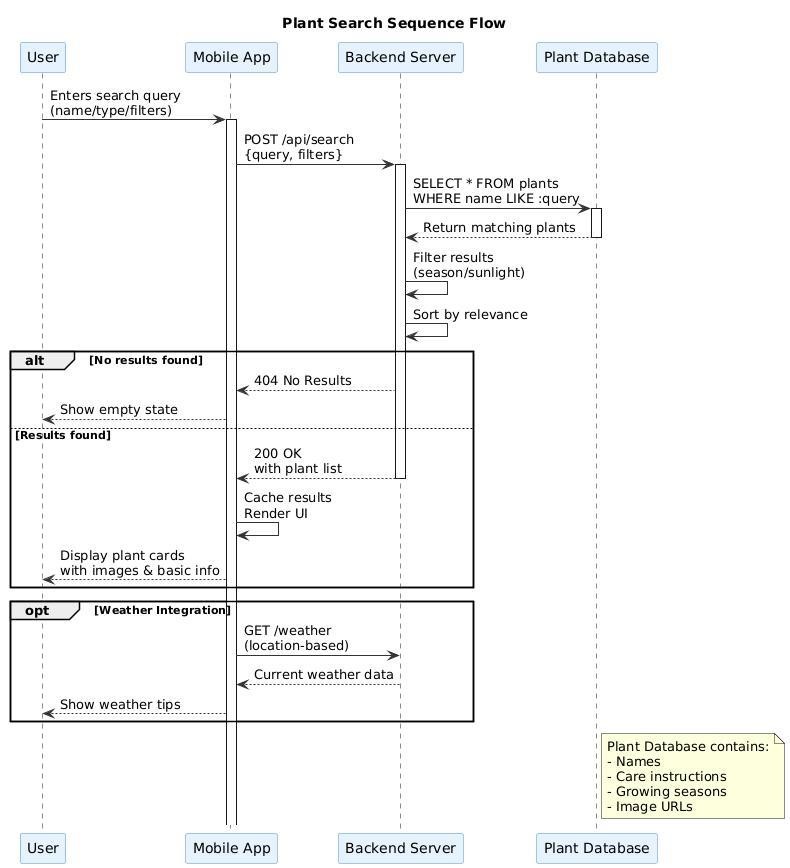
****

**Figure 2.3** Use Case Diagram of *IAPK*.

**2.6 Schematic diagrams**

**2.6.1 Entity relationships (ER) Diagram of Database**

**Figure 2.4** ER Diagram of *IAPK*.

**2.6.2 Plant Search Sequence Diagram**

**Figure 2.5** Sequence Diagram of *IAPK*.

**2.7 Tools and Techniques**

1. Development

Front-end: React Native or Flutter.

Express.js with Node.js server. Firebase database (for beginners) or MySQL for larger projects).

1. Design

Design interfaces Adobe XD or Figma Lucid chart drawing diagrams.

1. Project management

To track tasks, use Jira or Trello Code Management GitHub.

**2.8 Expected Challenges**

1. Collect accurate data

The need for reliable sources of plant information such as scientific books or experts).

1. Integration with APIs

Ensure stable weather API connection and real-time data update.

1. User experience

Design an interface that suits all age groups and technical levels.

**CHAPTER****3**

Application Development

**3.1 Preparing for The Development Environment**

Before starting development, the necessary tools and software are prepared:

* Frontend

1. Framework: Flutter for cross-platform development.
2. With VS Code or Android Studio tools.
3. Dart Flutter add-on Additional packages: http to connect to the server provider management status.

* Server (Backend)

1. Express.js with Node.js language framework.
2. Tools: Postman for testing, Firebase Fire store or MongoDB APIs.

* Database

1. System: Firebase Fire store for cloud database).
2. Setup: Create a new Firebase project and configure security rules.

**3.2 Building the Database**

The main tables in Firebase are designed as follows Fire store

* Plants Collection

{

}

"plant\_id": "001",

"name": {

"Basil": "ar"

},"en": "Basil"

"Every other day in summer": "watering"

"sunlight": "full sun",

"fertilizer": "biweekly organic fertilizer"

"images": ["url1", "url2"]

* Users Collection

{

"user\_id": "user\_123",

"favorite\_plants": ["001", "005"], "location":

" الرياض , السعودية "

}

* Weather Collection

{

"location\_id": "cairo\_eg",

"temperature": 32.5, "humidity": 60, "alerts": " "لا توجد تحذيرات

}

**3.3 Front-End Development**

The interface is divided into four main units.

* Plant Search Interface. the components:

Input field to search by name or: SearchBar Type Mini plant display card with: PlantCard image and name.

* Jobs:

Connect to Firebase server to fetch plant data Filter results by season or lighting.

//Flutter لعرض قائمة النباتا ت

ListView.builder(itemCount:

plants.length, itemBuilder:

(context, index) {return

PlantCard(name:

plants[index].name, image:

plants[index]. images[0],

);

},

)

* Page Plant details

the components: To view plant images: Carousel Slider Timberview:

Lighting, fertilizer and weather Jobs Fetch plant data from Firebase using plant\_id Add to Favorites button to save the plant to the user's list.

* Weather section

OpenWeatherMap API integration with. Use the http package to send a GET request to API Processing data and display it in the user interface.

مثال لجلب بيانات الطقس // final

response = await http.get(

Uri.parse('https://api.openweathermap.org/data/2.5/weather?q=$location&appid=$ apiKey'),

);

* Notification system

Firebase Cloud Messaging integration with.

(FCM):

Setting up FCM in the project Send automatic notifications when the weather or watering schedule changes

**3.4 Backend Development**

The server is built using Node.js and Express.js. To handle requests

* Endpoints

1. GET plants.
2. Get a list of all the plants.
3. GET/plants/ id: Get plant details specific.
4. Send weather data based on: POST / weather on the site.

مثال بسيط لـ //endpoint فيNode.js app.get('/plants',

async (req, res) => { const plants = await

db.collection('plants'). get(

);

res.json(plants.docs.map(doc => doc.data()));

});

* Error management

Hiring middleware like To handle express-async-errors unexpected errors. Send custom error messages (eg 404 if plant not found).

**3.5 Integration with APIs**

* Weather integration Steps

1. Get an API key from OpenWeatherMap.
2. Send an HTTP request with the location coordinates.
3. Convert the data into a format suitable for display in the application.

* Notifications integration Steps

1. Configure FCM in the application.
2. Send notifications using Firebase functions, Cloud Functions

**3.6 Adding Additional Features**

* Recommendation system

Suggest similar plants based on:

* User's favorite plants.
* Environmental conditions (weather location).
* Frequently asked questions section. Create a page that contains questions such as:

How do you choose the right container? "What do I do if my plant is exposed to frost?"

**3.7 Initial Test**

Before moving to the final testing stage, preliminary tests are conducted.

* Unit Testing

Testing data fetching functions from Firebase. Test converting weather data into a viewable format.

* Integration testing

Ensure that the front-end communicates with the server APIs without errors. Test sending notifications in a timely manner

**3.8 Code Documentation**

* Internal comments

plantId]باستخدام] Firestore /// جلب بيانات النبات من

Future<Plant> fetchPlant(String plantId) async {

// ... كود الجل ب

}

* API document

Fetch code Create an API .md file that explains the connection points and how to use them.

**3.9 Release Management**

1. Storage on GitHub Create and manage a repository.
2. Using Git Use branches to develop features separately.

**CHAPTER 4**

Application Testing and Optimization

**4.1 Types of Tests**

The tests are divided into four main types.

* Functional Testing

Ensure that all features work as expected. Covering basic and extreme cases.

* UI/UX Testing

\*Ensure that the interface is user-friendly and consistent with the planned design.

\*Testing the user experience on different devices (phones, tablets).

* Performance Testing

\*Measure application speed and resource consumption (memory, battery).

\*Testing the application under different loads (large number of users).

* Security Testing

\*Ensure that users' data is protected.

\*Testing the application's resistance to common attacks (such as SQL injection).

**4.2 Functional testing**

Functional tests were performed using tools such as: Jest Flutter Driver

* Plant Search Test

Ensure that the search returns the correct results. Testing for edge cases such as searching for non-existent text) مثال باختبار //Flutter Driver test('', () البحث عن نبات الريحانasync { await driver.tap(find.byType('TextField'));

expect(await driver.getText(find.text('');الريحان'((, 'الريحان });

Test View plant details await driver.enterText(''); ريحان await driver.tap(find.text('')); بحث

- Ensure that the page displays all the required information.

- Test the "Add to Favorites" button.

* Test the notification system

Ensure that notifications arrive in a timely manner Test weather notifications (such as frost warning).

**4.3 User Interface Testing**

Tools like Widget Testing have been used in Figma Mirror وFlutter

* design test

Ensure that the interface matches the design in Figma Test colors, fonts, and spacing

مثال باختبار //Widget testWidgets('', (عرض بطاقة النباتWidgetTester tester) async {

await tester.pumpWidget(MaterialApp(home: PlantCard(name: ' ',الريحان image:

'url'))); expect(find.text(''), الريحانfindsOneWidget);

});

* Interaction test

Make sure the buttons are working properly. Scrolling test on long pages.

* Compatibility testing

1. Test the app on different devices (Android).
2. (iOS) Ensure that the interface adapts to different screen sizes

**4.4 Performance Testing**

Tools like Firebase Performance have been used.

Flutter Dev Tools and Monitoring

* Test the speed of the application
* Measure the loading time of pages such as the plant details page.
* Ensure that the response time is less than 2 seconds.
* Resource consumption test
* Measure memory and battery usage.
* Ensure that the application does not consume excessive resources.
* Pregnancy test
* Simulate many users (10,000 active users).
* Ensure that the server processes requests without delay

**4.5 Security Testing**

Tools like OWASP ZAP and Firebase are used in Security Rules:

* Data protection test

1. Ensure that user data is encrypted (such as preferred location).
2. Testing application resistance to SQL injection attacks.

* Test APIs

1. Ensure that the connection points are secured such as using (JWT for verification).
2. Testing APIs resistance to common attacks (such as DDoS)

* Test Firebase Rules
* Ensure that security rules prevent unauthorized access service cloud.

firestore {match

/databases/{database}/documents {match

/plants/{plant} {allow read: if true; allow

write: if request.auth != null;

}

}

}

**4.6 Performance improvement**

Based on the test results, the following improvements have been made:

* Improve data fetching

Use caching to reduce the number of requests to the server

final response = await http.get(

Uri.parse('https://api.example.com/plants'), headers:

{'Cache-Control': 'max-age=3600'

},

);

* Improve images

Use WebP format to reduce image size, Lazy Loading.

* Server optimization
* Use Load Balancing technology to distribute the load.
* Optimize database queries to be faster

**4.7 Bug fixing**

The following errors were detected and fixed:

* Interface errors:
* Fixed an issue where the “Add to Favorites” button was not working on iOS.
* Improved display of long texts in plant cards.
* Performance errors:
* Fix Memory Leak in Plant Details Page.
* Improve loading time of large images.
* Security errors
* Fixed a security vulnerability in the Weather API that allowed unauthorized access.
* Updated Firebase rules to prevent unauthorized access to user data

**4.8. Final Exam**

After making improvements and fixing bugs, testing was done, Final includes:

* Comprehensive Functional Test
* Ensure that all features are working properly.
* Compatibility test
* Test the app on 10 different devices. (Android, iOS)
* Performance test under load
* Simulate 10,000 active users to ensure app stability

**4.9 Test Report**

Test results are documented in a report that includes:

* Job test results

Success rate: 98%.

* Performance test results:

Average response time: 1.5 seconds.

* Security test results

No critical security vulnerabilities.

**CHAPTER 5**

Application Launch and Maintenance

**5.1 Launch the application**

The application will be launched on the Google Play and App Store platforms after all stages of development and testing have been completed.

* Preparing the application for launch

1. Create accounts Google Play Console Developer Account on.
2. Apple Developer Account on Program
3. Asset preparation
4. App Icon Design.
5. Prepare screenshots and an introductory video.
6. Write the description
7. An attractive description of the application with mention of the main features.

Add keywords to improve search. (THIS)

* Upload the application

1. Google Play: Download APK or App Bundle file.

* Fill in the store information Category Age

1. App Store: Download the app via Xcode or App Store.

* Connect Submit the application for review by Apple
* Gradual release

1. Launch with 10% of users monitoring performance.

* Increase the percentage to 100% after ensuring the stability of the application

**5.2 Marketing Campaign**

The app was promoted using various marketing strategies.

* Digital marketing
* Create pages on social media (Facebook, Instagram, Twitter).
* Paid advertising on Google Ads.
* Facebook Ads Sending emails to those interested in agriculture.
* Content marketing
* Publish articles about the benefits of home gardening.
* Share educational videos on how to use the app.
* Collaborate with influencers
* Collaborate with agricultural and environmental bloggers to promote the application Offer free versions of the app to influencers to test

**5.3 Collecting Feedback**

Channels have been provided to communicate with users to collect feedback.

* ratings and reviews

1. Encourage users to rate the app in stores.
2. Monitor reviews and respond to inquiries.

* Opinion polls

1. Send in-app surveys to measure user satisfaction.

Example: How would you rate your experience with the app?

* communication channels

1. Regular updates are released to improve the app and add new features.
2. Provide an email for technical support.
3. Create a “Contact Us” section within the app.

**5.4 Periodic Updates**

* Adding new plants
* Database updated monthly by adding new plants.

Example: adding seasonal plants such as lettuce in the winter)

* Improved functionality
* Added "fertilization reminder" feature.
* Improved recommendation algorithm based on weather.
* Bug fixes
* Release quick updates to fix critical bugs

Example: Fixed an issue with displaying images on some devices

**5.5 Ongoing maintenance**

An ongoing maintenance plan has been put in place to ensure the stability of the application.

* Performance monitoring:

Use tools like Firebase To monitor the speed of the application Performance Monitoring Fix any performance issues as soon as they are discovered.

* Update libraries

Regularly update Flutter and Node.js libraries Ensure that the application is compatible with the latest versions of operating systems.

* backup copies

Perform daily database backups Store backups on Google Cloud services such as

**5.6 Data Analysis**

Data analysis tools were used to understand user behavior.

1. Google Analytics:

Track the number of active users Analysis of the most used features**.**

1. Firebase Analytics:

Measuring user retention rate (Retention Rate) Identify points where users lose interest.

1. custom reports

Create monthly reports on application performance Example: “20% increase in active users.

**5.7 Future Expansion**

A plan has been put in place for future expansion that includes:

1. Add new languages

Support for additional languages such as French and Spanish Localize the application for new markets.

1. new features

Add a “Community” feature to share tips between users Integrate with smart devices to water plants automatically

1. Geographical expansion

Promote the application in new countries Adding native plants to each area

**5.8 Conclusion**

This app comes to make the planting process easier and more accessible to everyone, whether they are amateurs or professionals.

By providing accurate information and personalized tips, the app will help users improve the productivity of their plants and keep them in the best condition.

Considering what was presented and developed during this project, it can be said that the application represents an important step toward using technology to improve and simplify agricultural practices for all segments of the population.

By providing accurate information, an easy-to-use experience, and integration with weather data, the project was able to meet users' needs and enhance their ability to care for plants efficiently.

The application has the elements of sustainability and continuous development to serve the digital agricultural community.

**5.9 Html code**

<!DOCTYPE html>

<html lang="ar" dir="rtl">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1" />

<title>المزارع التقني - Tech Farmer</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

display: flex;

flex-direction: column;

height: 100vh;

background: #f0f0f0;

}

header, footer {

background: #009578;

color: #fff;

padding: 10px;

text-align: center;

}

#chat-container {

flex: 1;

overflow-y: auto;

padding: 10px;

background: #fff;

}

.message {

margin: 5px 0;

padding: 8px 12px;

border-radius: 20px;

max-width: 80%;

word-wrap: break-word;

}

.user-msg {

background: #ddf;

align-self: flex-end;

margin-left: auto;

}

.ai-msg {

background: #ffd;

align-self: flex-start;

margin-right: auto;

}

footer {

display: flex;

align-items: center;

gap: 5px;

flex-wrap: wrap;

}

#message-input {

flex: 1;

padding: 8px;

border: none;

border-radius: 4px;

}

button {

padding: 8px 12px;

border: none;

background: #007d60;

color: #fff;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background: #00604d;

}

</style>

</head>

<body>

<header>

<div id="site-name">المزارع التقني</div>

<button id="lang-toggle">English</button>

<button id="login-btn-header">تسجيل دخول</button>

</header>

<div id="chat-container"></div>

<footer>

<button class="media-btn" id="camera-btn">📷</button>

<button class="media-btn" id="file-btn">📁</button>

<input type="text" id="message-input" placeholder="اكتب رسالتك هنا..." />

<button id="send-btn">إرسال</button>

<input type="file" id="camera-input" accept="image/\*,video/\*" capture="user" style="display:none;">

<input type="file" id="file-input" accept="image/\*,video/\*" style="display:none;">

</footer>

<!-- نموذج تسجيل مبسط (يمكن تطويره لاحقاً) -->

<div id="register-modal" class="modal" style="display:none;">

<!-- محتوى التسجيل -->

</div>

<!-- تضمين مكتبة Socket.IO من CDN -->

<script src="https://cdn.socket.io/4.5.4/socket.io.min.js"></script>

<script>

document.addEventListener('DOMContentLoaded', function() {

const socket = io();

socket.on('connect', () => {

console.log('متصل بالخادم');

});

// استقبال رد الذكاء الاصطناعي وعرضه في واجهة الدردشة

socket.on('bot-response', (data) => {

const chatContainer = document.getElementById('chat-container');

const aiMessage = document.createElement('div');

aiMessage.className = 'message ai-msg';

aiMessage.textContent = data;

chatContainer.appendChild(aiMessage);

chatContainer.scrollTop = chatContainer.scrollHeight;

});

// إرسال الرسالة عند النقر على زر "إرسال"

document.getElementById('send-btn').addEventListener('click', function() {

const input = document.getElementById('message-input');

const message = input.value.trim();

if (message) {

const chatContainer = document.getElementById('chat-container');

const userMessage = document.createElement('div');

userMessage.className = 'message user-msg';

userMessage.textContent = message;

chatContainer.appendChild(userMessage);

chatContainer.scrollTop = chatContainer.scrollHeight;

// إرسال الرسالة إلى الخادم

socket.emit('user-message', message);

input.value = '';

}

});

// دعم إرسال الرسالة عبر مفتاح Enter

document.getElementById('message-input').addEventListener('keydown', function(e) {

if (e.key === 'Enter') {

document.getElementById('send-btn').click();

}

});

});

</script>

</body>

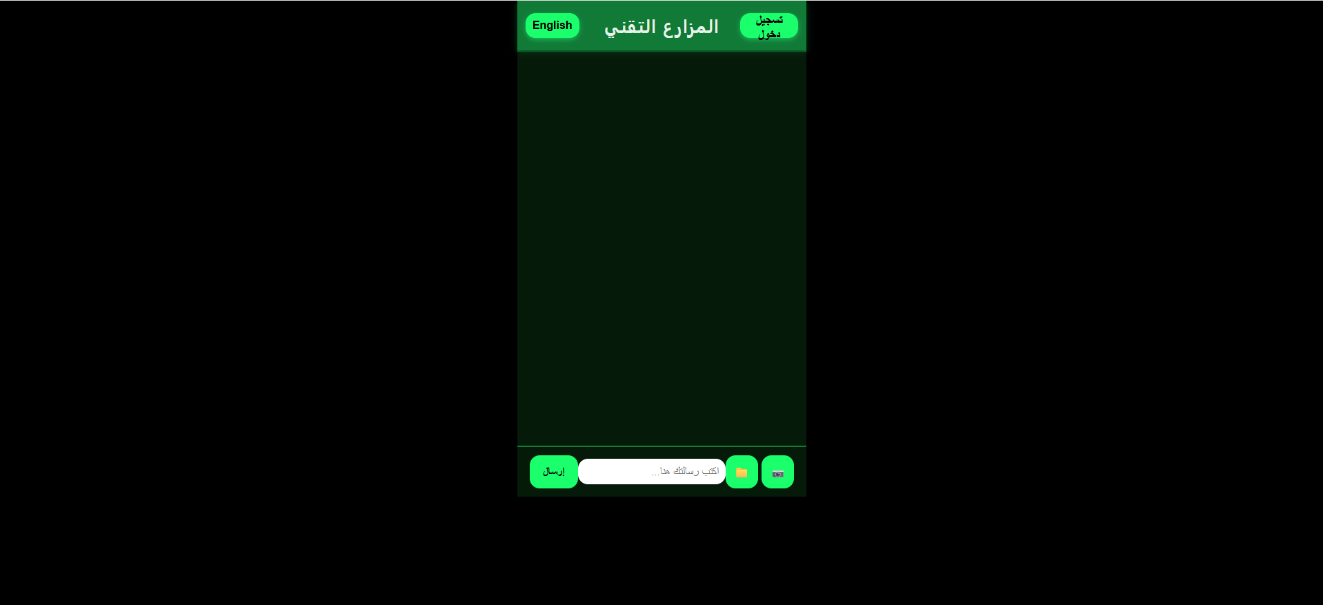
</html>

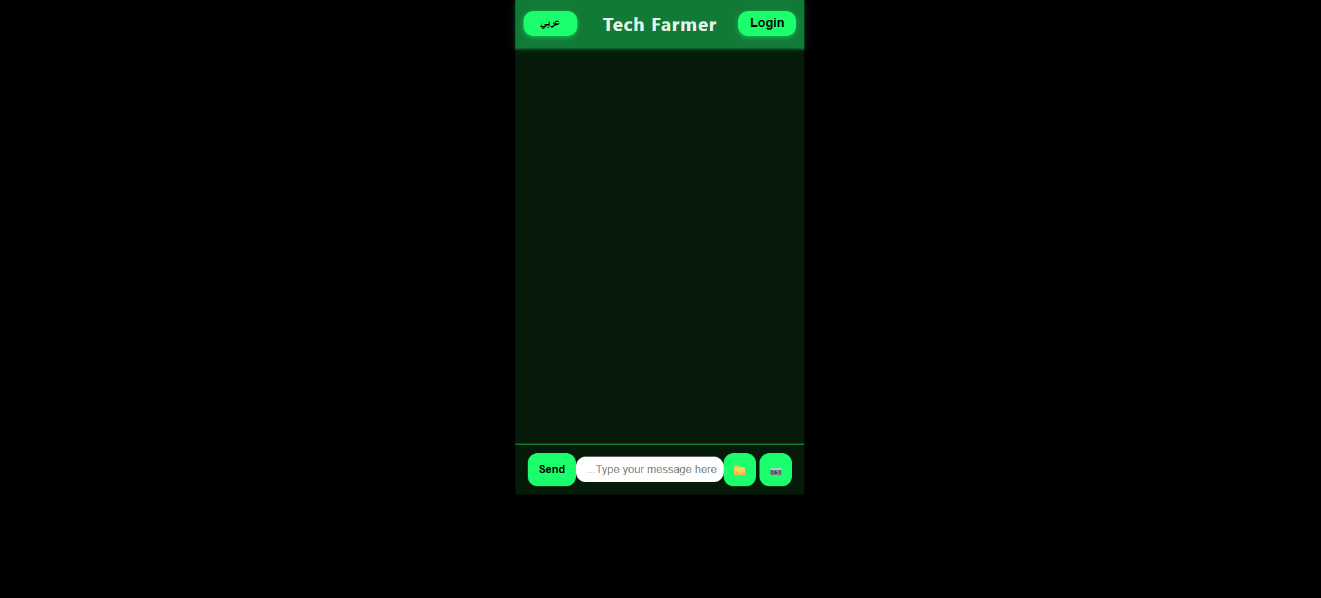
صورة تحتوي على لقطة شاشة, نص, الهاتف النقال, التصميم

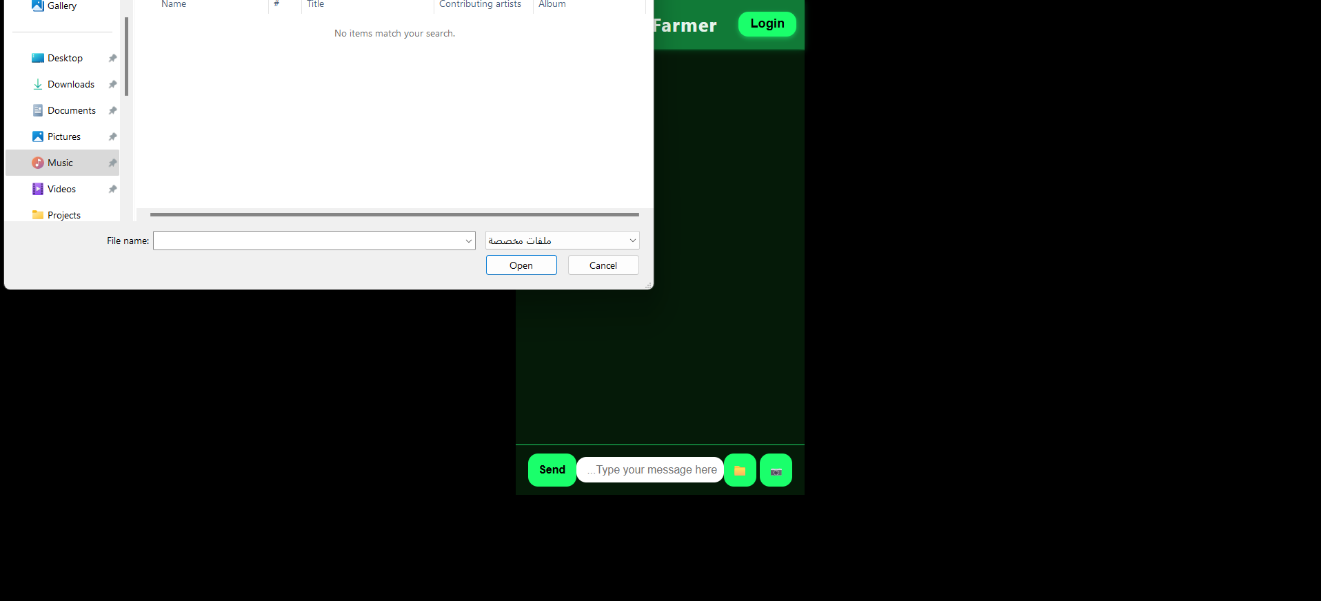
قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

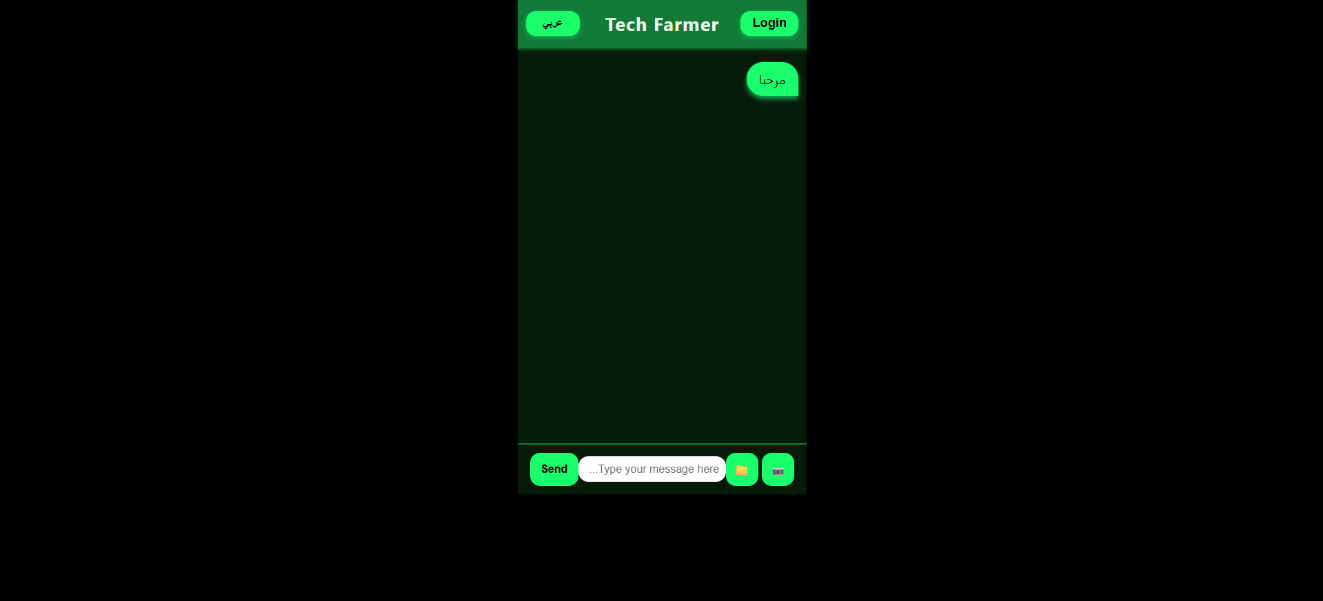
صورة تحتوي على لقطة شاشة, نص, برمجيات, برامج الوسائط المتعددة

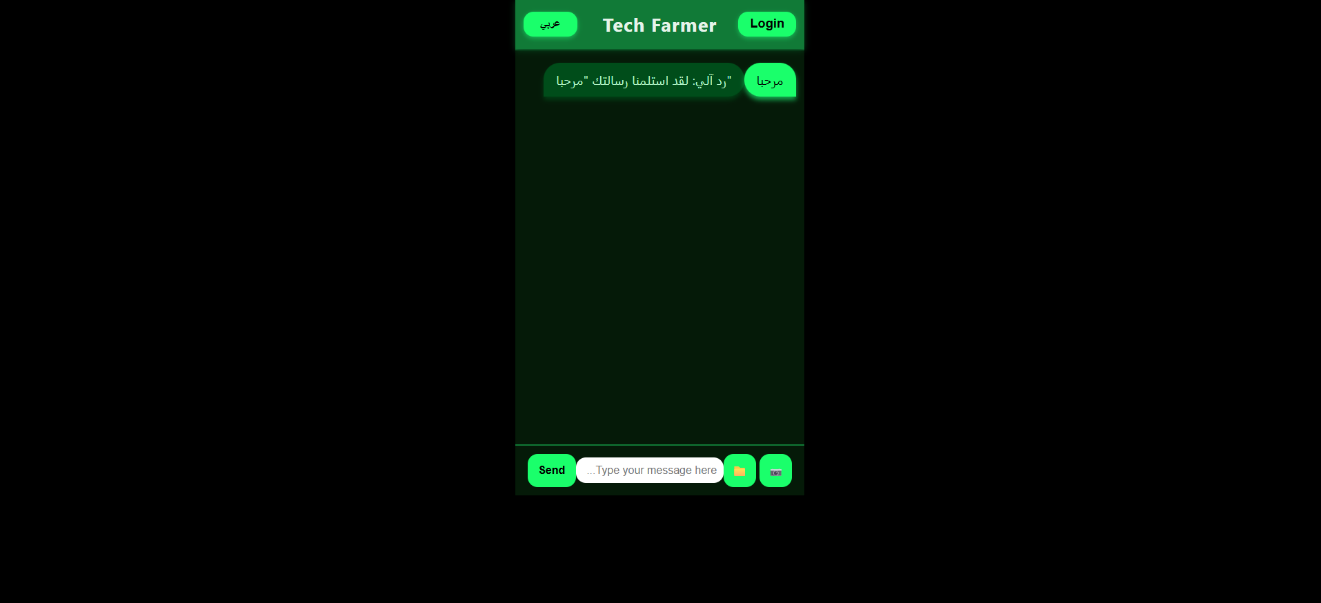
قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.











**6.0 References**

1. Flutter - flutter.dev
2. React Native - reactnative.dev
3. Node.js - nodejs.org
4. Django - djangoproject.com
5. Firebase - firebase.google.com
6. OpenWeatherMap API - openweathermap.org/api
7. Google Maps API - developers.google.com/maps
8. Let's Encrypt - letsencrypt.org
9. OWASP Security - owasp.org
10. Figma - figma.com
11. Adobe XD adobe.com/xd
12. Material Design – material.io
13. Testing and deployment
14. Postman - postman.com
15. It is - jestjs.io
16. AWS aws.amazon.com
17. Docker docker.com
18. <https://futurumcareers.com/fostering-love-and-knowledge-of-agriculture-and-the-environment>
19. <https://fyi.extension.wisc.edu/wateroutreach/changing-public-behavior/target-audience-research/target-audience-research-referenced-audiences/target-audience-research-farmers/>
20. <https://www.mdpi.com/2071-1050/13/9/4883>
21. <https://www.researchgate.net/publication/351698642_A_Study_on_Data_Analysis_and_Electronic_Application_for_the_Growth_of_Smart_Farming>
22. <https://upkeep.com/learning/agricultural-maintenance/>
23. <https://www.sciencedirect.com/science/article/pii/S277323712200020X>
24. <https://www.researchgate.net/publication/322090006_Importance_of_feedback_information_from_farm_accountancy_data_network_of_the_Republic_of_Serbia>
25. <https://esajournals.onlinelibrary.wiley.com/doi/10.1002/eap.2501>