Soil Farming Agent

1. Introduction

1.1 Project Overview

This document provides a detailed low-level design for the Soil Farming Agent application. The application helps understand which crops grow best in different soil types. Admin users can update soil and distributor details, and general users can view this information.

1.2 Scope

The scope of this document includes:

- Detailed descriptions of each module
- Database design
- Code snippets
- Integration points
- Security considerations
- Deployment details

2. System Architecture

2.1 Overview

The system follows a client-server architecture. The client side is built with HTML, CSS, and JavaScript, and the server side uses Firebase for database and authentication services.

3. Module Design

3.1 Admin Module

3.1.1 Admin Login:

- **Functionality**: Allows the admin to log in to the system.
- **Input**: Email, Password
- **Output**: Success or failure message
- **Description**: This module uses Firebase Authentication to log in the admin.

- **Code snippet**:

```
    const AdsignInbut = document.getElementById('Adminloginbut');
    AdsignInbut.addEventListener('click', (event) => {
        event.preventDefault();
        const email = document.getElementById('Adminemailsigin').value;
        const password = document.getElementById('Adminpasssigin').value;
        const auth = getAuth();

        signInWithEmailAndPassword(auth, email, password)
            .then((userCredential) => {
                showMessage('Login is successful', 'AdminsignInMessage');
                const user = userCredential.user;
                localStorage.setItem('loggedInUserId', user.uid);
                window.location.href = 'Admin_Homepage.htm';
            })
            .catch((error) => {
                const errorCode = error.code;
                if (errorCode === 'auth/wrong-password') {
                    showMessage('Incorrect Email or Password',
'AdminsignInMessage');
                } else if (errorCode === 'auth/user-not-found') {
                    showMessage('Account does not Exist',
'AdminsignInMessage');
                } else {
                    showMessage('Error: ' + error.message,
'AdminsignInMessage');
                }
            });
    });
```

- If you forgot the password, we can access it through email.

```
const AdforgotPassLabel = document.getElementById('Adminforgotpass');
    AdforgotPassLabel.addEventListener('click', (event) => {
        event.preventDefault();
        const email = document.getElementById('Adminemailsigin').value;
        const auth = getAuth();

        sendPasswordResetEmail(auth, email)
            .then(() => {
                alert("A Password Reset Link has been sent to your email");
            })
            .catch((error) => {
                console.log(error.code);
                console.log(error.message);
            });
```

```
    });
```

- To display messages :

```
function showMessage(message, divId) {
    const messageDiv = document.getElementById(divId);
    messageDiv.style.display = "block";
    messageDiv.innerHTML = message;
    messageDiv.style.opacity = 1;
    setTimeout(() => {
        messageDiv.style.opacity = 0;
    }, 5000);
}
```

3.1.2 Post Soil Details

- **Functionality**: Allows the admin to post and update information about different soil types.
- **Input**: Soil type, Characteristics, Recommended crops
- **Output**: Success or failure message
- **Description**: This module uses Firebase Firestore to store soil details.
- **Code Snippet**:
    - **To add new Soil Type**:

```
insertBtn.addEventListener('click', () => {
                const soilType = document.getElementById('soiltype').value;
                if (soilType.trim() === "") {
                    alert("Please enter soil type.");
                    return;
                }

                set(ref(db, "SoilType/" + soilType), {
                    Soiltype: soilType
                })
                .then(() => {
                    alert("Data stored successfully");
                    window.location.href = 'Admin_Homepage.htm';
                })
                .catch((error) => {
                    alert("Unsuccessful, error: " + error);
                });
            });
```

- To view ,update and delete the soil type:

```
searchBtn.addEventListener('click', () => {
                const searchBar =
document.getElementById('searchBar').value;
                const category =
document.getElementById('CategorySelected').value;
                const dbref = ref(db);
                debugger;
                if (category == "1") {
                    get(child(dbref, "SoilType/" +
searchBar)).then((snapshot) => {
                        if (snapshot.exists()) {
                            debugger;
                            document.getElementById('searchResults').innerH
TML = `
                                <p>Soil Type:
${snapshot.val().Soiltype}</p>
                                <button
onclick="updateSoil('${snapshot.key}')">Update</button>
                                <button
onclick="deleteSoil('${snapshot.key}')">Delete</button>
                            `;
                        } else {
                            alert("No data found");
                        }
                    })
                    .catch((error) => {
                        alert("Unsuccessful, error: " + error);
                    });
                }
            });
```

```
window.updateSoil = (soilType) => {
            const newSoilType = prompt("Enter new soil type:", soilType);
            if (newSoilType && newSoilType !== soilType) {
                const dbref = ref(db);
                set(child(dbref, "SoilType/" + newSoilType), { Soiltype:
newSoilType })
                    .then(() => {
                        alert("Soil type updated successfully");
                        document.getElementById('searchResults').innerHTML =
'';
                    })
                    .catch((error) => {
                        alert("Unsuccessful, error: " + error);
                    });
                remove(child(dbref, "SoilType/" + soilType));
            }
```

```
        };
window.deleteSoil = (soilType) => {
    if (confirm("Are you sure you want to delete this soil type?")) {
        debugger;
        const dbref = ref(db, "SoilType/" + soilType);
        console.log("Attempting to delete soil type:", soilType, "at path:",
dbref);

        remove(dbref)
            .then(() => {
                alert("Soil type deleted successfully");
                console.log("Soil type deleted successfully:", soilType);
                document.getElementById('searchResults').innerHTML = '';
                window.location.href = 'Admin_Homepage.htm';
            })
            .catch((error) => {
                alert("Unsuccessful, error: " + error.message);
                console.error("Error deleting soil type:", error);
            });
    }
};
```

- To add crop information: We have to select soil type to which you are adding crop information.

```
 const soilSelect = document.getElementById('soiltypebtn');

            async function GetSoiltype() {
                const dbRef = ref(db);
                try {
                    const snapshot = await get(child(dbRef, 'SoilType/'));
                    if (snapshot.exists()) {
                        soilSelect.innerHTML = '<option value="select">Soil
type</option>'; // Reset options
                        const soilTypes = snapshot.val();
                        for (const key in soilTypes) {
                            const option =
document.createElement('option');
                            option.value = soilTypes[key].Soiltype;
                            option.textContent = soilTypes[key].Soiltype;
                            soilSelect.appendChild(option);
                        }
                        soilSelect.disabled = false;
                    } else {
                        alert("No soil types found in the database.");
                    }
```

```
                } catch (error) {
                    console.error("Error fetching soil types:", error);
                }
            }

    window.addEventListener('load',GetSoiltype);

    var soilbtntype=document.getElementById('soiltypebtn');
    var waterbtntype=document.getElementById('waterbtn');
    var cropname=document.getElementById('crop');
    var cropdes=document.getElementById('cropdes');
    var process1=document.getElementById('croppro1');
    var process2=document.getElementById('croppro2');
    var process3=document.getElementById('croppro3');
    var process4=document.getElementById('croppro4');
    var process5=document.getElementById('croppro5');
    var process6=document.getElementById('croppro6');
    var subcrop=document.getElementById('subcrop');
    var subcropdes=document.getElementById('subcropdes');
    var submitbtn=document.getElementById('submitform');
```

```
function  Insertdataful(){
        set(ref(db,"Soil Information/"+soilbtntype.value),{
            Typeofsoil:soilbtntype.value,
            Watersupply:waterbtntype.value,
            Nameofcrop:cropname.value,
            descriptionofcrop:cropdes.value,
            process1:process1.value,
            process2:process2.value,
            process3:process3.value,
            process4:process4.value,
            process5:process5.value,
            process6:process6.value,
            subcrop:subcrop.value,
            descriptionofsubcrop:subcropdes.value
        })
        .then(()=>{
            alert("Data stored successfully");
            window.location.href = 'Admin_Homepage.htm';
        })
        .catch((error)=>{
            alert("unsuccessful,error"+error);
        });
    }
    submitbtn.addEventListener('click', (event) => {
    event.preventDefault(); // Prevent default form submission
    Insertdataful();
    });
```

- To view,update and delete the crop information:

```
 document.getElementById('searchBtn1').addEventListener('click', async () => {
                const searchBar = document.getElementById('myInput').value;
                const category =
document.getElementById('CategorySelected').value;
                const dbref = ref(db);

                if (category == "1") {
                    debugger;
                    try {
                        const snapshot = await get(child(dbref, "Soil
Information/" + searchBar));

                        if (snapshot.exists()) {
                            const soilData = snapshot.val();
                            document.getElementById('viewSearchResults').inner
HTML = `
                                <form id="updateSoilForm">
                                    <label for="soilType">Soil Type:</label>
                                    <input type="text" id="soilType"
value="${soilData.Typeofsoil}" required>
                                    <label for="waterSupply">Water
Supply:</label>
                                    <input type="text" id="waterSupply"
value="${soilData.Watersupply}" required>
                                    <label for="cropName">Crop Name:</label>
                                    <input type="text" id="cropName"
value="${soilData.Nameofcrop}" required>
                                    <label for="cropDes">Description of
Crop:</label>
                                    <input type="text" id="cropDes"
value="${soilData.descriptionofcrop}" required>
                                    <label for="process1">Process 1:</label>
                                    <input type="text" id="process1"
value="${soilData.process1}" required>
                                    <label for="process2">Process 2:</label>
                                    <input type="text" id="process2"
value="${soilData.process2}" required>
                                    <label for="process3">Process 3:</label>
                                    <input type="text" id="process3"
value="${soilData.process3}" required>
                                    <label for="process4">Process 4:</label>
                                    <input type="text" id="process4"
value="${soilData.process4}" required>
                                    <label for="process5">Process 5:</label>
```

```
                                    <input type="text" id="process5"
value="${soilData.process5}" required>
                                    <label for="process6">Process 6:</label>
                                    <input type="text" id="process6"
value="${soilData.process6}" required>
                                    <label for="subCrop">Sub Crop:</label>
                                    <input type="text" id="subCrop"
value="${soilData.subcrop}" required>
                                    <label for="subCropDes">Description of Sub
Crop:</label>
                                    <input type="text" id="subCropDes"
value="${soilData.descriptionofsubcrop}" required>
                                    <button
type="submit"id="update">Update</button>
                                </form>
                                <button
onclick="deleteSoilInfo('${snapshot.key}')" id="delete">Delete</button>
                            `;

            document.getElementById('updateSoilForm').addEventListener('submit
', async (e) => {
                                e.preventDefault();

                                const updatedData = {
                                    Typeofsoil:
document.getElementById('soilType').value,
                                    Watersupply:
document.getElementById('waterSupply').value,
                                    Nameofcrop:
document.getElementById('cropName').value,
                                    descriptionofcrop:
document.getElementById('cropDes').value,
                                    process1:
document.getElementById('process1').value,
                                    process2:
document.getElementById('process2').value,
                                    process3:
document.getElementById('process3').value,
                                    process4:
document.getElementById('process4').value,
                                    process5:
document.getElementById('process5').value,
                                    process6:
document.getElementById('process6').value,
                                    subcrop:
document.getElementById('subCrop').value,
                                    descriptionofsubcrop:
document.getElementById('subCropDes').value,
```

```
                                };

                                try {
                                    await update(ref(db, "Soil Information/" +
searchBar), updatedData);
                                    alert('Soil information updated
successfully!');
                                } catch (error) {
                                    alert('Error updating soil information: '
+ error.message);
                                }
                            });

                        } else {
                            alert("No data found");
                        }
                    } catch (error) {
                        alert("Unsuccessful, error: " + error.message);
                    }
                }
            });

    window.deleteSoilInfo = (soilType) => {
    if (confirm("Are you sure you want to delete this soil type?")) {
        const dbref = ref(db);
        remove(child(dbref, "Soil Information/" + soilType))
            .then(() => {
                alert("Soil type deleted successfully");
                document.getElementById('viewSearchResults').innerHTML = '';
// Clear search results
                window.location.href = 'Admin_Homepage.htm'; // Replace with
your homepage URL
            })
            .catch((error) => {
                alert("Unsuccessful, error: " + error);
            });
    }
};
```

3.1.3 Post Distributor Details

- **Functionality**: Allows the admin to post and update details of crop/seed distributors.
- **Input**: Distributor name, Location, Contact information, Crop type
- **Output**: Success or failure message

- **Description**: This module uses Firebase Firestore to store distributor details.
- **Code Snippet**:
    - Creating distributor information:

```
var Compname=document.getElementById('Compname');
   var name=document.getElementById('name');
   var Cropbuy=document.getElementById('Cropbuy');
   var locationname=document.getElementById('locationname');
   var address=document.getElementById('address');
   var phoneno=document.getElementById('phoneno');
   var submitform2=document.getElementById('submitform2');

  function  Insertdataful2(){
       set(ref(db,"Distributor/"+Cropbuy.value),{
            Companyname:Compname.value,
            Ownername:name.value,
            Nameofcrop:Cropbuy.value,
            locationofcenter:locationname.value,
            Address:address.value,
            Phonenumber:phoneno.value
       })
       .then(()=>{
           alert("Data stored successfully");
           window.location.href = 'Admin_Homepage.htm';
       })
       .catch((error)=>{
           alert("unsuccessful,error"+error);
       });
   }

   submitform2.addEventListener('click', (event) => {
   event.preventDefault(); // Prevent default form submission
   Insertdataful2();
   });
```

    - To view, update and delete the distributor details:

```
document.getElementById('searchBtn2').addEventListener('click', async () =>
{
                const searchBar = document.getElementById('mycrop').value;
                const category =
document.getElementById('CategorySelected').value;
                const dbref = ref(db);

                if (category == "1") {
```

```
                    debugger;
                    try {
                        const snapshot = await get(child(dbref,
"Distributor/" + searchBar));

                        if (snapshot.exists()) {
                            const cropData = snapshot.val();
                            document.getElementById('searchCenter').innerHT
ML = `
                                <form id="updateCropForm">
                                    <label for="Compname">Name of
Distributor center:</label>
                                    <input type="text" id="Compname"
value="${cropData.Companyname}" required>
                                    <label for="name">Owner Name:</label>
                                    <input type="text" id="name"
value="${cropData.Ownername}" required>
                                    <label for="Cropbuy">Crop Name:</label>
                                    <input type="text" id="Cropbuy"
value="${cropData.Nameofcrop}" required>
                                    <label for="locationname">Location of
center:</label>
                                    <input type="text" id="locationname"
value="${cropData.locationofcenter}" required>
                                    <label for="address">Adress of
center:</label>
                                    <input type="text" id="address"
value="${cropData.Address}" required>
                                    <label for="phoneno">Phone
number:</label>
                                    <input type="tel" id="phoneno"
value="${cropData.Phonenumber}" required>
                                    <button
type="submit"id="update2">Update</button>
                                </form>
                                <button
onclick="deleteCrop('${snapshot.key}')" id="delete2">Delete</button>
                            `;

            document.getElementById('updateCropForm').addEventListener('sub
mit', async (e) => {
                                e.preventDefault();

                                const updatedData = {
                                    Companyname:
document.getElementById('Compname').value,
                                    Ownername:
document.getElementById('name').value,
```

```
                                    Nameofcrop:
document.getElementById('Cropbuy').value,
                                    locationofcenter:
document.getElementById('locationname').value,
                                    Address:
document.getElementById('address').value,
                                    Phonenumber:
document.getElementById('phoneno').value
                                };

                                try {
                                    await update(ref(db, "Distributor/" +
searchBar), updatedData);
                                    alert('Details updated successfully!');
                                } catch (error) {
                                    alert('Error updating distributor
information: ' + error.message);
                                }
                            });

                        } else {
                            alert("No data found");
                        }
                    } catch (error) {
                        alert("Unsuccessful, error: " + error.message);
                    }
                }
            });

            window.deleteCrop = (Cropbuy) => {
            if (confirm("Are you sure you want to delete this crop type?"))
{
        const dbref = ref(db);
        remove(child(dbref, "Distributor/" + Cropbuy))
            .then(() => {
                alert("crop type deleted successfully");
                // document.getElementById('viewcenterdetails').innerHTML =
'''; // Clear search results
                window.location.href = 'Admin_Homepage.htm'; // Replace
with your homepage URL
            })
            .catch((error) => {
                alert("Unsuccessful, error: " + error);
            });
    }
};
```

3.2 User Module

3.2.1 Register:

- **Functionality**: Allows users to register and create an account.
- **Input**: Email, Password
- **Output**: Success or failure message
- **Description**: This module uses Firebase Authentication to register the user.
- **Code snippet**:

```
const signUp = document.getElementById('upbutton');
signUp.addEventListener('click', (event) => {
    event.preventDefault();
    const email = document.getElementById('emailsignup').value;
    const password = document.getElementById('passsignup').value;
    const firstName = document.getElementById('fname').value;
    const lastName = document.getElementById('lname').value;

    // const auth = getAuth();
    // const db = getFirestore(app);
    const authUser = getAuthUser(userApp);
    const dbUser =getFirestore(userApp);

    createUserWithEmailAndPassword(authUser, email, password)
        .then((userCredential) => {
            const user = userCredential.user;
            const userData = {
                email: email,
                firstName: firstName,
                lastName: lastName
            };
            showMessage('Account Created Successfully', 'signUpMessage');
            const docRef = doc(dbUser, "users", user.uid);
            setDoc(docRef, userData)
                .then(() => {
                    window.location.href = 'login.htm';
                })
                .catch((error) => {
                    console.error("Error writing document", error);
                });
        })
        .catch((error) => {
            const errorCode = error.code;
            if (errorCode === 'auth/email-already-in-use') {
                showMessage('Email Address Already Exists !!!',
'signUpMessage');
```

```
            } else {
                showMessage('Unable to create User', 'signUpMessage');
            }
        });
});
```

3.2.2 Login

- **Functionality**: Allows users to log in to the system.
- **Input**: Email, Password
- **Output**: Success or failure message
- **Description**: This module uses Firebase Authentication to log in the user.
- **Code Snippet**:

```
const signInbut=document.getElementById('loginbut');
signInbut.addEventListener('click',(event)=>{
    event.preventDefault();
    const email=document.getElementById('emailsigin').value;
    const password=document.getElementById('passsigin').value;
    const authUser = getAuthUser(userApp);


    signInWithEmailAndPassword(authUser,email,password)
    .then((userCredential)=>{
        showMessage('Login is successful','signInMessage');
        const user=userCredential.user;
        localStorage.setItem('loggedInUserId',user.uid);
        window.location.href='Homepage.htm';
    })
    .catch((error)=>{
       const errorCode=error.code;
       if(errorCode==='auth/invalid-credential'){
        showMessage('Incorrect Email or Password','signInMessage');
       }
       else{
        showMessage('Account does not Exist','signInMessage');
       }
    });
});
```

As mentioned in admin same functions to show messages and forgetpassword.

3.2.3 View Crop Details:

- **Functionality**: Users can view crop information in different soil types.
- **Input**: Crop type and water supply in your area.
- **Output**: List of crop types which is suitable for those inputs
- **Description**: This module retrieves crop details from Firebase Firestore(Realtime database).
- **Code snippet**:

```
  const soilSelectuser = document.getElementById('soiltypebtnuser');

    async function GetSoiltypeuser() {
    const dbRef = ref(db);
    try {
        const snapshot = await get(child(dbRef, 'SoilType/'));
        if (snapshot.exists()) {
            soilSelectuser.innerHTML = '<option value="select">Soil
type</option>'; // Reset options
            const soilTypes = snapshot.val();
            for (const key in soilTypes) {
                const option = document.createElement('option');
                option.value = soilTypes[key].Soiltype;
                option.textContent = soilTypes[key].Soiltype;
                soilSelectuser.appendChild(option);
            }
            soilSelectuser.disabled = false;
        } else {
            alert("No soil types found in the database.");
        }
    } catch (error) {
        console.error("Error fetching soil types:", error);
    }
}

window.addEventListener('load', GetSoiltypeuser);

function searchSoilInfo() {
    const soilType = document.getElementById('soiltypebtnuser').value;
    const waterSupply = document.getElementById('waterbtnuser').value;
    const resultsDiv = document.getElementById('sandDetails');

    if (soilType === 'select' || waterSupply === 'select') {
        alert('Please select both soil type and water supply');
        return;
    }
```

```
    const dbRef = ref(db);
    get(child(dbRef, `Soil Information/${soilType}`)).then((snapshot) => {
        if (snapshot.exists()) {
            const data = snapshot.val();
            resultsDiv.innerHTML = `
                <h3 style="color:rgb(0, 66, 0);">Crop Information</h3>
                <p><strong style="color:rgb(0, 66, 0);">Soil Type:</strong>
${data.Typeofsoil}</p>
                <p><strong style="color:rgb(0, 66, 0);">Water
Supply:</strong> ${data.Watersupply}</p>
                <p><strong style="color:rgb(0, 66, 0);">Crop Name:</strong>
${data.Nameofcrop}</p>
                <p><strong style="color:rgb(0, 66,
0);">Description:</strong> ${data.descriptionofcrop}</p>
                <p><strong style="color:rgb(0, 66, 0);">Process 1:</strong>
${data.process1}</p>
                <p><strong style="color:rgb(0, 66, 0);">Process 2:</strong>
${data.process2}</p>
                <p><strong style="color:rgb(0, 66, 0);">Process 3:</strong>
${data.process3}</p>
                <p><strong style="color:rgb(0, 66, 0);">Process 4:</strong>
${data.process4}</p>
                <p><strong style="color:rgb(0, 66, 0);">Process 5:</strong>
${data.process5}</p>
                <p><strong style="color:rgb(0, 66, 0);">Process 6:</strong>
${data.process6}</p>
                <p><strong style="color:rgb(0, 66, 0);">Subcrop:</strong>
${data.subcrop}</p>
                <p><strong style="color:rgb(0, 66, 0);">Subcrop
Description:</strong> ${data.descriptionofsubcrop}</p>
            `;
        } else {
            resultsDiv.innerHTML = "No data available for the selected
options.";
        }
    }).catch((error) => {
        console.error(error);
        resultsDiv.innerHTML = "Error retrieving data.";
    });
}
```

3.2.4 View Distributor Details

- **Functionality**: Users can view details about crop/seed distributors.
- **Input**: Crop they want to sell/buy and location which is nearby to them these inputs will be from database(firebase).
- **Output**: List of distributor details
- **Description**: This module retrieves distributor details from Firebase Firestore(Realtime database0.
- **Code snippet:**

```
const cropSelectuser = document.getElementById('croptypebtnuser');

async function GetCroptypeuser() {
const dbRef = ref(db);
try {
    const snapshot = await get(child(dbRef, 'Distributor/'));
    if (snapshot.exists()) {
        cropSelectuser.innerHTML = '<option value="select">Crop
type</option>'; // Reset options
        const cropTypes = snapshot.val();
        for (const key in cropTypes) {
            const option = document.createElement('option');
            option.value = cropTypes[key].Nameofcrop;
            option.textContent = cropTypes[key].Nameofcrop;
            cropSelectuser.appendChild(option);
        }

        cropSelectuser.disabled = false;
    } else {
        alert("No crop types found in the database.");
    }
} catch (error) {
    console.error("Error fetching crop types:", error);
}
}

window.addEventListener('load', GetCroptypeuser);


const locSelectuser = document.getElementById('locationbtnuser');

async function GetLoctypeuser() {
const dbRef = ref(db);
try {
    const snapshot = await get(child(dbRef, 'Distributor/'));
    if (snapshot.exists()) {
        locSelectuser.innerHTML = '<option
value="select">Location</option>'; // Reset options
```

```
        const locTypes = snapshot.val();
        for (const key in locTypes) {
            const option = document.createElement('option');
            option.value = locTypes[key].locationofcenter;
            option.textContent = locTypes[key].locationofcenter;
            locSelectuser.appendChild(option);
        }

        locSelectuser.disabled = false;
    } else {
        alert("No  location found in the database.");
    }
} catch (error) {
    console.error("Error fetching location:", error);
}
}

window.addEventListener('load', GetLoctypeuser);

function searchCenterInfo() {
    const results = document.getElementById('distributorDetails');
    const cropType = document.getElementById('croptypebtnuser').value;
    const locSelectuser = document.getElementById('locationbtnuser');
    if ( cropType=== 'select' ||  locSelectuser=== 'select') {
        alert('Please select both soil type and water supply');
        return;
    }

    const dbRef = ref(db);
    get(child(dbRef, `Distributor/${cropType}`)).then((snapshot) => {
        if (snapshot.exists()) {
            const data = snapshot.val();
            results.innerHTML = `
                <h3 style="color:rgb(0, 66, 0);">Distributor
Information</h3>
                <p><strong style="color:rgb(0, 66, 0);">Name of
Distributor center:</strong> ${data.Companyname}</p>
                <p><strong style="color:rgb(0, 66, 0);">Owner
Name:</strong> ${data.Ownername}</p>
                <p><strong style="color:rgb(0, 66, 0);">Crop
Name:</strong> ${data.Nameofcrop}</p>
                <p><strong style="color:rgb(0, 66, 0);>Location of
center:</strong> ${data.locationofcenter}</p>
                <p><strong style="color:rgb(0, 66, 0);>Adress of
center:</strong> ${data.Address}</p>
                <p><strong style="color:rgb(0, 66, 0);>Phone
number:</strong> ${data.Phonenumber}</p>
            `;
```

```
        } else {
            results.innerHTML = "No data available for the selected
options.";
        }
    }).catch((error) => {
        console.error(error);
        results.innerHTML = "Error retrieving data.";
    });
}

document.getElementById('searchBtndis').addEventListener('click',
searchCenterInfo);
```

4. Database Design

4.1 Schema

Crop Details

- Typeofsoil:soil type
- Watersupply:Continous Water Supply or Seasonal Rainfall
- Descriptionofcrop:About crop
- process1: Steps to be taken for that crop production
- process2: Steps to be taken for that crop production
- process3: Steps to be taken for that crop production
- process4: Steps to be taken for that crop production
- process5: Steps to be taken for that crop production
- process6: Steps to be taken for that crop production
- subcrop:suggestions of subcrop to avoid weed problem
- descriptionofsubcrop:About subcrop

Distributor Details

- Address : Address of Distributor company
- Companyname : Distributor company name.
- Nameofcrop : Crop name they will buy
- Ownername : Name of the distributor
- Phonenumber : Contact of Distributor
- locationofcenter : Location of distributor company

5. Security Considerations

5.1 Authentication

- Use Firebase Authentication for secure login and registration processes.

5.2 Authorization

- Ensure that only admins can post and update soil and distributor details.
- Users can only view details.

5.3 Data Validation

- Validate all input data to prevent injection attacks and ensure data integrity.

6. Integration Points

6.1 Firebase

- Firestore for storing soil and distributor details.
- Firebase Authentication for managing user and admin logins.

6.2 Deployment

- Use Firebase Hosting for deploying and hosting the web application.

7. Deployment Details

7.1 Environment Setup

- Configure Firebase project and add Firebase configuration to the project.
- Set up Firebase Authentication and Firestore.

7.2 Deployment Steps

- Deploy the application to Firebase Hosting using the Firebase CLI.

8. Testing

8.1 Unit Testing

- Write unit tests for individual functions using a JavaScript testing framework like Jest.

8.2 Integration Testing

- Test the integration between the client application and Firebase services.

8.3 End-to-End Testing

- Use a framework like Cypress to write end-to-end tests that simulate user interactions.

9. Conclusion

This LLD document provides a detailed design for the Soil Farming Agent application. By following this document, the development team can ensure that the application is modular, safe, testable, maintainable, and portable.