



UNIVERSITATEA DIN
BUCUREŞTI



FACULTATEA DE
MATEMATICA ŞI
INFORMATICA

SPECIALIZAREA INGINERIE SOFTWARE

Lucrare de disertatie

SMART SYSTEM ASSISTANT FOR GROWING PLANTS

Absolvent

Ilie George-Ciprian

Coordonator științific
Conf.dr. Radu Gramatovici

Bucureşti, septembrie 2024

Rezumat

Lucrarea își propune crearea unui asistent "Smart" pentru asistarea utilizatorilor la creșterea și ingrijirea plantelor. Acest sistem smart IoT va fi compus dintr-o componentă fizică, o serie de senzori conectați la o placă Raspberry Pi atașată plantei în cauză, cu scopul obținerii valorilor necesare în timp real, precum și de o aplicație Android cu un UI adecvat, integrată cu chatGPT, aplicație care va efectua diverse operațiuni pe baza valorilor obținute de la senzori. Un alt scop secundar al tezei ar putea fi reprezentat de verificarea gradului în care chatGPT ar putea fi considerat adecvat pentru îngrijirea plantelor.

Ca și tehnologii software, pentru dezvoltarea aplicației se va utiliza Python pentru a scrie scriptul necesar citirii valorii de pe senzori, o bază de date firebase cu datele stocate în format JSON, precum și Java pentru dezvoltarea aplicației în Android Studio.

Funcționalitățile de bază ale aplicației vor fi reprezentate de afișarea în timp real a valorilor mediului în care se află planta (temperatură, umiditate, lumină etc), existența unui mod de oferire a luminii adiționale, interval automat (setat manual de udare la un anumit interval de timp), mod automat de udare și adaptare al luminii conform recomandărilor lui chatGPT, trimitera notificărilor și alertarea utilizatorului în cazul în care planta nu se află în parametrii optimi, incărcarea unui mod de îngrijire recomandat de chatGPT, liveChat cu plantGPT, recomandări de plante asemănătoare (generate cu chatGPT) precum și alte diferite features.

În capitolul final se vor prezenta concluziile, problemele întampinate și rezolvarea acestora, precum și eventuale îmbunatațiri ce ar putea fi implementate ulterior în scopul transformării acestuia într-un produs final pentru un business.

Abstract

The thesis aims at the creation of an "Smart" assistant for caring and growing a plant at home. This smart assistant will be composed of 2 parts, a phisical component and a software component. The physical component will be made of a variety of sensors attached to the plant and connected to a Raspberry Pi motherboard in order to gather real time data about the environment. The software part consists of an Android application with a friendly UI, a chatGPT module integrated and some features tied to the live data feed. Trying to find out how adequate chatGPT's abilities of caring for a plant are in this context could also be considered as a secondary mission for this paper.

From the perspective of the software technologyes used for the development of the project we can consider Python, used for the script that reads the live data from the sensors, a FireBase database that stores data as JSON and Java, for the development of the Android application.

The base features of the application are: display of live-feed data of the environment (Temperature, humidity, light indix etc), a mode for aditional lighting, automated watering interval that can be set manauly at a certain period of time, automated mode for watering and lighting the plant according to chatGPT recommendations, notifications in case of issues regarding the evnvironment, automatic plant care mod generated by chatGPT, liveChat with chatGPT, plant recommendations by chatGPT, based on the selected plant and many more features.

In the final chapter can be found the conclusions, problems and their solution and even future improvements that could be implemented with the purpose of transforming this thesis into a customer ready product.

Cuprins

1 Introducere	6
1.1 Prezentare Generală	6
1.2 Obiectivele și motivația alegerii temei	6
1.3 Starea dispozitivelor smart de îngrijire a plantelor	7
1.4 Structura lucrării	7
2 Tehnologii Utilizate	9
2.1 Partea Hardware	9
2.1.1 Raspberry Pi 4 model B	9
2.1.2 Accesorii Raspberry Pi	10
2.1.3 Senzor de temperatură și umiditate AM2320	11
2.1.4 Senzor pentru presiunea și temperatura atmosferică BMP280	11
2.1.5 Senzor pentru intensitatea luminii BH1750FVI	11
2.1.6 Pompa de apă	12
2.1.7 Sursă adițională de putere	12
2.1.8 Lumina de tip LED	13
2.1.9 Senzor de verificare a umidității solului	13
2.1.10 Senzor detectare Ploaie	13
2.1.11 3VDC relee	14
2.1.12 Modul comparator LM393	14
2.1.13 BreadBoard	15
2.1.14 Diverse fire conectoroare	15
2.2 Partea Software	16
2.2.1 Firebase	16
2.2.2 Python	16
2.2.3 Android	17
2.2.4 Java	17
2.2.5 XML	17
3 Implementarea Lucrarii	19
3.1 Partea Fizica	19

3.2	Partea Software	20
3.2.1	Structura baza de date	21
3.2.2	Script Python pentru Raspberry Pi	21
3.2.3	Aplicatia Android	26
3.2.4	Structura Android Studio	27
3.2.4.1	Login Page Register Page Reset Password Page	27
3.2.4.2	Main Activity	31
3.2.4.3	Navigation Menu	33
3.2.4.4	ChatGPT Class	34
3.2.4.5	Home Page	39
3.2.4.6	Issues Page	43
3.2.4.7	Tips Screen Select	46
3.2.4.8	General Information Page	46
3.2.4.9	Recommendations Page	48
3.2.4.10	Live Chat Page	50
3.3	Probleme Intampinate	52
4	Ghid de utilizare	53
4.0.1	Pregatirea partii Hardware	53
4.0.2	Utilizarea partii Software	54
4.0.3	Utilizarea partii Software	54
5	Concluzii	56
6	Bibliografie	57

Capitolul 1

Introducere

1.1 Prezentare Generală

Lucrarea se încadrează în domeniul aplicațiilor IoT, a sistemelor inteligente de îngrijire a plantelor, aceast ansamblu constând în diferiți senzori legați la o placă Raspberry Pi, care la rândul ei este conectată cu o aplicație Android usor de utilizat datorita interfaței intuitive pentru utilizator.

Aplicația este destinată utilizatorilor care doresc ajutor cu îngrijirea unei plante, fie ea la un nivel mai ridicat sau scăzut. Orice persoană care s-a simtit vreodată compleșită de îngrijirea unei plante poate reprezenta dovada necesității existenței unei aplicații de acest tip.

Pentru utilizarea dispozitivului este necesară atașarea senzorilor pe planta, iar apoi aplicația Android este gata de utilizare, împreună cu toate featurilor sale. Aplicația este alcătuită din mai multe pagini denumite sugestiv. Ca și flow al proiectului, date de la senzori este încărcată din 5 în 5 secunde în baza de date firebase, de unde aplicația android își ia valorile pentru diverse funcționalități.

1.2 Obiectivele și motivația alegerii temei

Motivația alegerii acestei teme a fost determinată de o situație cu care m-am confruntat în viața reală, situație cu care probabil foarte mulți dintre noi putem rezona. Mi-am cumpărat o plantă și din pacate aceasta nu a rezistat deoarece nu am reușit să o îngrijesc corespunzător. Această întâmplare m-a dus la gândul dacă nu cumva, în contextul stadiului inteligenței artificiale, ar putea exista un astfel de asistent care ar putea veni în ajutorul utilizatorilor mai puțin experimentați, care totuși își doresc să aibă grija de o plantă.

Chiar dacă uneori nu pare atât de complicat să întreții o plantă, comparativ cu creșterea unui animal, tot există provocări care pot apărea la orice pas. Putând să monitorizăm

factorii de mediu în orice moment și abilitatea de a avea cunoștiințele inteligenței artificiale de partea noastră poate reprezenta un avantaj substanțial pentru reusirea în realizarea acestui task.

Obiectivele lucrării sunt destul de clare, dezvoltarea unui sistem capabil de a asista utilizatorul cat mai bine în îngrijirea plantei, studierea gradului în care chatGPT, un agent de inteligență artificială integrat în aplicație, poate duce cu succes la final acestă sarcină.

1.3 Starea dispozitivelor smart de îngrijire a plantelor

In prezent există doar câteva dispozitive smart pentru îngrijirea plantelor, dar acestea nu se bazează încă pe inteligența artificială, majoritatea acestora permitând utilizatorului doar setarea manuală a anumitor parametrii precum intervalul de udare și efectuarea operațiilor necesare la intervalul orar stabilit.

Din punctul de vedere al modului de gandire, în industrie nefiind încă folosită inteligența artificială la acest nivel, aplicația noastră ar putea fi considerată drept un "proof of concept" și un studiu de caz asupra întrebării dacă inteligența artificială este destul de avansată la momentul actual încât să poată avea grija de o plantă, și mai ales, în particular, dacă chatGPT este pregătit în acest moment pentru acest task destul de complex.

Pe langă integrarea cu chatGPT, aplicația se deosebește de alte produse software deja existente și prin intermediul altor features care oferă o asistare cat mai bună a utilizatorului, features precum urmărire constantă a valorilor din mediu, prezentarea unei rutine personalizate, adaptată la climatul în care se află planta, precum și asistență de tip live-chat în orice moment al zilei.

1.4 Structura lucrării

Documentația lucrării de disertație va fi structurată în cinci capitole, fiecare dintre acestea prezintă aspecte relevante ale acestei lucrări.

Primul capitol, Introducerea, este descrisă pe scurt aplicația, starea dispozitivelor smart de tip IoT pentru îngrijirea plantelor în piata actuală, precum și motivația alegerii acestei teme, iar la finalul acestui capitol se regăsește structura lucrării.

Cel de-al doilea capitol denumit "Tehnologii utilizate" descrie în detaliu componentele părții fizice, precum și limbajele de programare utilizate pentru dezvoltarea părții software a acestei lucrări, pe parcursul acestui capitol fiind prezentate toate noțiunile tehnice relevante.

Al treilea capitol este alcătuit din trei parti, cale un subcapitol pentru partea fizică,

software și a structurii bazei de date utilizate, în fiecare subcapitol fiind explicate procedee relevante, dar și structura acestora.

În cadrul capitolului al patrulea este prezentat un scurt ghid de utilizare al aplicatiei precum și un "first time setup" necesar.

Ultimul capitol conține concluziile acestei lucrări, problemele întampinate și modul de soluționare al acestora precum și îmbunătățiri ce ar putea fi implementate în viitor.

Capitolul 2

Tehnologii Utilizate

2.1 Partea Hardware

Ansamblul hardware al acestei lucrări este alcătuit dintr-o placă de procesare, diverse senzori, lumini LED, pompă de apă controlată cu ajutorul unui releu și diverse elemente necesare de legatură:

- Raspberry pi 4 model B
- Senzor AM2320 : umiditate and temperatură
- Senzor BH1750FVI: intensitate lumină
- Senzor BMP280: presiune atmosferică si temperatură
- Pompa de apă
- Releu 3V
- Sursa aditională de lumină (LED)
- Senzor sol umed
- Senzor Ploaie
- BreadBoard
- Diverse conectori

2.1.1 Raspberry Pi 4 model B

Raspberry Pi 4 face parte dintr-o serie de plăci de dezvoltare/procesare, în esență aducând toată puterea unui PC într-un format mai compact, sub forma unei mici placute. Este una dintre cele mai populare opțiuni cand vine vorba de dezvoltarea dispozitivelor de tipul IoT, ceea ce avanaje ale acesteia fiind reprezentă de o putere foarte mare pentru mici dimensiuni ale acesteia, ușurința de utilizare și varietatea de module ce pot fi instalate

destul de usor, cu ajutorul sistemului unic de operare Raspbian, o versiune modificată de linux.

Acet model de placă prezintă un procesor quad-core Broadcom BCM 2711 1.5 Ghz, 4 RAM DDR4, o varietate de moduri de a conecta, de la porturi USB pana la wifi.

Pentru pornirea acesteia este nevoie de un încarcator USB type C cu o putere de cel putin 5V/3A.

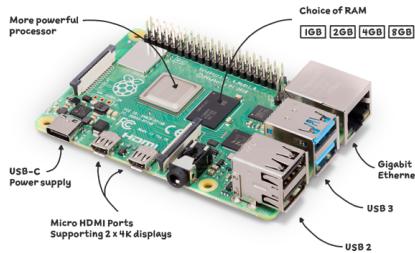


Figura 2.1: Raspberry Pi 4 module B

2.1.2 Accesorii Raspberry Pi

Deoarece placuța vine într-o formă destul fragilă, având toti pinii expuși am ales să ii montez o carcasa de protecție și un sistem foarte basic de racire, deoarece lasată să funcționeze continuu pe o perioadă indelungată de timp placuța poate ajunge până la 80 de grade celsius, o valoare foarte mare, cu mult peste normele recomandate.

Pentru o racire pasivă am instalat 3 heatsinks din aluminiu cu un design special ce încurajează o racire mai bună datorită designului special care creează o mai mare suprafață de contact cu aerul. Între aceste heatsinks și placă se află pastă termo-conductoare.

Adițional pentru o racire activă a fost adăugat și un ventilator de 5V conectat direct pe pinii plăcii.



Figura 2.2: Heatsink și Ventilatoare

2.1.3 Senzor de temperatură și umiditate AM2320

Acesta este un senzor digital care poate măsura umiditatea între 10% -99% cu o acuratețe de +/- 3% și temperatura între -40°C și 80°C cu o acuratețe de 0.5°C.

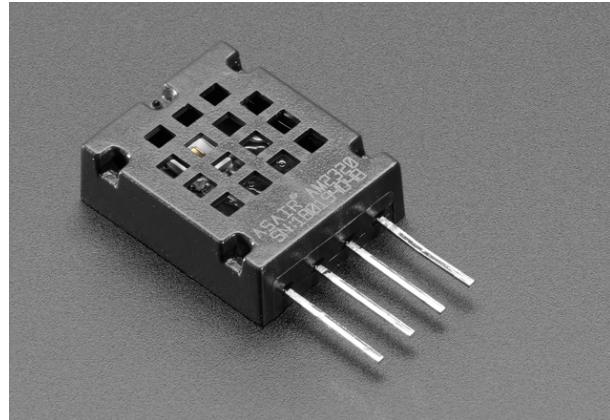


Figura 2.3: AM2320

2.1.4 Senzor pentru presiunea și temperatura atmosferică BMP280

Este un senzor de la Bosch, barometric care măsoară presiunea atmosferică și poate măsura și temperatura. Poate măsura presiunea între 300 hPa și 1100 hPa cu o acuratețe de +/- 1hPa, iar temperatura între -40°C și 85°C cu o acuratețe de +/- 1°C. Este o alegere bună pentru un dispozitiv IoT datorită puterii mari, raportată la dimensiunile sale.

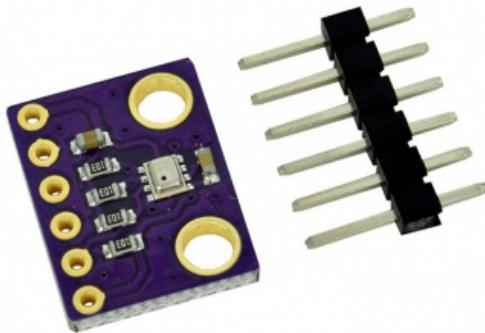


Figura 2.4: BMP280

2.1.5 Senzor pentru intensitatea luminii BH1750FVI

Este un senzor care poate determina intensitatea luminii ambientale, poate detecta valori între 0 - 65535 lx.



Figura 2.5: BH1750FVI

2.1.6 Pompa de apă

Pompa este acționată de un mic motor electric, care atunci cand este activat impinge lichid din rezervor pe tubul special pentru a uda, voltajul optim de operare este intre 5V si 9V.



Figura 2.6: Pompa de apa

2.1.7 Sursă adițională de putere

Deoarece placa Raspberry Pi nu poate oferi destul voltaj prin pinii sai GPIO pentru pompa de apă, este nevoie de această sursă aditională pentru a atinge voltajul de 6V.

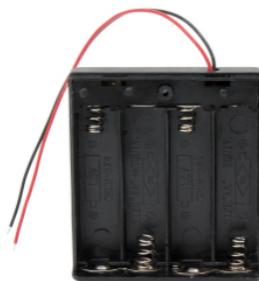


Figura 2.7: Sursa aditionala putere

2.1.8 Lumina de tip LED

Pentru acest proiect, becul de tip LED este folosit pentru a simula existența unei lămpi reale UV, deoarece un astfel de modul este foarte scump și ar avea sens doar în ideea unui produs final. Este un simplu semi conductor care emite lumină atunci când trece curent prin el. În mod normal modulul de lampă ideal ar conține zeci sau chiar sute de beculeți LED cu diverse moduri de iluminare, ca și ledul nostru doar că la scară mai mică.



Figura 2.8: LED

2.1.9 Senzor de verificare a umidității solului

Acest senzor este unul rezistiv, are două prelungiri care sunt infipte în pamant și funcționează pe post de un circuit deschis. Dacă solul este umed, apă va conduce un curent electric de la vârful unuia pană la celălalt vârf inchizând astfel circuitul. Evaluarea este efectuată de modulul LM393, outputul fiind digital.

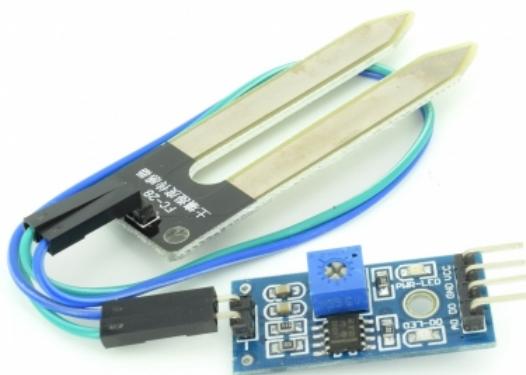


Figura 2.9: Senzor umiditate sol

2.1.10 Senzor detectare Ploaie

Este un senzor rezistiv, în care se află un circuit metalic care este deschis de la naștere. Atunci când picaturile de ploaie se află pe senzor, circuitul se închide. Evaluarea este

data de modulul LM393 iar outputul este digital.

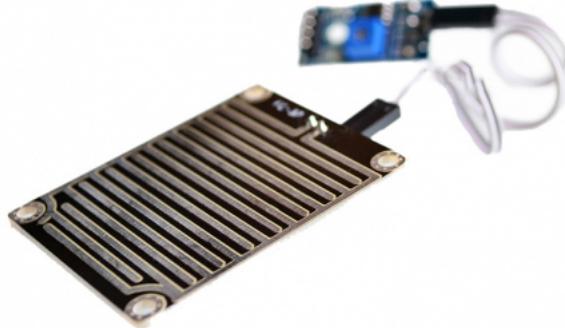


Figura 2.10: Enter Caption

2.1.11 3VDC releu

Acest modul funcționeaza ca un switch care se deschide sau se inchide în funcție de puterea impulsului care este primit. Are la bază un electromagnet care atunci când primește curent, atrage o parte metalică cu ajutorul câmpului magnetic inchizând circuitul, iar atunci când nu mai primește încă dă release pentru a aduce circuitul în starea inițială. Releul este folosit în acest proiect pentru a controla pompa de apă.



Figura 2.11: 3VDC Releu

2.1.12 Modul comparator LM393

Este un modul comparator care transformă inputul primit de către senzorii rezistivi într-unul digital.



Figura 2.12: Modul Comparator LM393

2.1.13 BreadBoard

Un breaboard permite creearea rapidă de circuite temporare, sau de prototipuri cu posibilitatea ca firele să fie aranjate și rearanjate în moduri diferite foarte usor, firele se conectează direct pe aceasta în găurile definite de linii și coloane, acest feature fiind datorat modului intern din care este alcătuită placa.

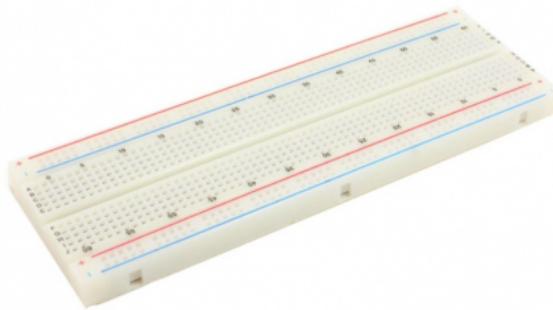


Figura 2.13: BreadBoard

2.1.14 Diverse fire conectoare

Sunt cunoscute sub numele de fire DuPont, după numele companiei care le-a facut populare. Sunt niște simple fire conductoare de curent care au un conector sau un pin la fiecare capăt și sunt folosite pentru a face conexiunile între componentele prototipului.

În exemplul acestei lucrari firele roșii conectează polaritățile negative, cele maro ca să trimită curent de la placă la module, cele mov sunt pentru a recepționa semnale de la senzorii rezistivi, galbene pentru circuitul care corespunde pt SDA pin, iar verde pentru cel de la SCL pin.

Cablurile vin de obicei în trei tipuri, tată la tată, tată la mamă, mamă la mamă, singura diferență între ele fiind la capete, cele cu tată având un pin la capăt iar cele cu mamă având un port.



Figura 2.14: Diverse Fire Conectoare

2.2 Partea Software

Pentru partea de tehnologii software utilizate, pentru a scrie scriptul care ruleaza pe placa Raspberry Pi si citește valorile indicate de senzori a fost folosit python, firebase pentru baza de date iar Java în Android pentru aplicația mobile.

2.2.1 Firebase

Firebase este o platformă de application development de la Google, care prezintă mai multe features ce o fac ideală pentru integrarea acesteia în aplicațiile Android. Firebase oferă utilizatorilor posibilitatea de a integra în aplicațiile software un real-time database care stochează datele sub forma de JSON, transformand-o de fapt într-o bază de date NoSQL salvată în Cloud.

Pe lângă serviciul de real-time database, firebase oferă și analitici detaliate, un sistem de autentificare pe baza de tokens, permitând autentificarea atât cu adrese de email și numere de telefon, cât și cu Google, Facebook sau chiar Twitter, analitici detaliate și multe alte servicii.

În prezent există mai mult de 3 milioane de aplicații care utilizează acest serviciu pentru o mai bună funcționare a backend-ului.

2.2.2 Python

Python este un limbaj de programare interpretat, avansat și destul de modern, pe scurt unul dintre cele mai utilizate limbaje de programare în prezent și special în zona de data science, inteligență artificială sau de scripting.

Ideea de bază a limbajului este să fie ușor de citit și de înțeles, o diferență mare față de restul limbajelor de programare o reprezintă eliminarea acoladelor în favoarea spațiilor și a identarii. Multe keywords în Python folosesc cuvinte din engleză, astfel codul devine de obicei mult mai ușor de citit și de înțeles.

Python este integrat în Raspbianul standard (sistemul de operare linux-based dezvoltat de Raspberry Pi foundation pentru placile lor), astfel python devine alegerea cea mai evidentă pentru dezvoltarea de produse software pe aceste plăci.

2.2.3 Android

Android este un sistem de operare des întâlnit pe dispozitivele mobile fiind gândit în special pentru dispozitivele cu touchscreen, este dezvoltat de către Google și se bazează pe un kernel modificat de Linux, fiind open-source.

În prezent sistemul de operare Android OS, ajuns la versiunea 14, este prezent în peste 70% dintre telefoanele smartphone de pe piață, fiind un lider detasat. Arhitectura sa modulară și usor customizabilă oferă un principal avantaj peste competiție, producătorii de telefoane putând astfel modela experiența utilizatorilor într-un mod foarte vast. Alte avantaje semnificative pot fi reprezentate de existența aplicației "Play Store" care permite utilizatorilor accesul la o gamă largă de aplicații dezvoltate de diversi developeri, google asistant, updateuri de securitate la intervale regulate, precum și integrarea cu toate serviciile Google .

Pentru dezvoltarea aplicației Android am utilizat ca și IDE Android Studio care oferă un mediu de dezvoltare optimizat, un editor de cod inteligent precum și un emulator. Acest IDE permite dezvoltarea aplicațiilor mobile în mai multe limbi de programare precum Java, Kotlin, C++ și multe altele. De asemenea acesta are inclus și un editor pentru fișierele de tip XML, fișiere pentru interfața GUI a aplicației.

2.2.4 Java

Java este un limbaj de programare de tipul multi-platform, bazat pe programarea orientată pe obiecte care poate fi folosit ca o platformă de sine stătătoare. A fost creat de către Microsystems Inc în 1995, unde James Gosling și-a dorit formarea unui limbaj de programare ce ar fi putut permite dispozitivelor electronice să comunice unele cu altele.

Diferența de bază dintre Java și celelalte limbi de programare este faptul că orice bucată de cod scrisă în Java este transformată în bytecodes care sunt apoi interpretate de Java Runtime Environment (JRE).

Câteva avantaje pentru utilizarea limbajului Java ar fi securitatea, faptul că este o platformă independentă, are o comunitate cu suport activ, există tools și IDE foarte calitative și are o varietate de funcții și librării already in-built.

2.2.5 XML

Fișierele XML sunt folosite în Android pentru a crea layouts pentru GUI. Fiecare layout XML este alocat unei anumite activitați, sau unui anumit fragment, în mod

unic. Fiecare layout are elemente din ierarhia View sau Groupview, adaptabile la fiecare rezoluție sau marime de ecran.

Ca și regulă de bază fiecare fisier de layout trebuie să includă doar un element root, iar restul obiectelor trebuie adăugate sub formă de ierarhie. Principalele moduri de layout sunt: Linear Layout (toate elementele sunt așezate într-o linie verticală sau orizontală), Relative Layout (elementele sunt așezate relativ față de alte elemente luate ca reper), Grid Layout (elementele sunt aranjate sub forma unui grid).

Există o multitudine de atribută pentru fiecare element, dar printre cele mai utilizate atribută ale elementelor pot fi considerate: (android:) id, layout_width, layout_height, text, padding, orientation, margin (poate fi right, left, bottom etc)

Capitolul 3

Implementarea Lucrarii

3.1 Partea Fizica

Prototipul constă în placă Raspberry Pi, care este componenta principală a sistemului. Cinci senzori, care iau date-live din mediu sunt conectați la aceasta împreună cu un modul de comparare, un releu o pompă de apă precum și o sursă aditională de putere pentru pompa de apă. Diagrama circuitului ilustrează cum au fost legate componentele pentru realizarea montajului. Pinii de la placă Raspberry Pi sunt numerotati de la colțul stânga sus (3V3) pana la ultimul pin de la dreapta jos (GPIO21).

Modulele sunt conectate în felul următor:

BMP280, doi pini pentru alimentare și aceiasi SCL și SDA pentru transferul de date.

Senzorul BH1750FVI pentru masurarea intensității luminoase are cinci pini, dintre care unul nu este utilizat. Si acesta se conectează prin pinii SCL și SDA pentru transferul de date și prin 3V3 și GND pentru alimentare.

Pompa de apă și sursa de alimentare suplimentară pentru aceasta sunt conectate una la cealaltă, apoi la releu.

Senzorul **BMP280** de temperatură și presiune atmosferică este conectat la placă folosind patru pini. Pinii VCC și GND sunt folosiți pentru oferi alimentare moduluilui și corespund pinilor 3V3 și GROUND de pe Raspberry Pi, iar ceilalți doi pini se ocupă de transferul de date folosind pinii SCL și SDA.

Releul 3DVC este conectat pe placă prin trei pini, 3V3 și GND pentru alimentare și GPIO13 pentru a primii semnalul.

Senzorul AM2320 pentru temperatură și umiditate are patru pini și se conectează asemanator cu BMP280. Senzorul de umiditate a solului este folosit împreună cu modului comparator LM393 prin doi pini (+) și (-), iar acest modul este conectat prin pinii VCC și GND pentru alimentare, iar D0 pentru a transmite date către Raspberry Pi (prin GPIO21).

Senzorul folosit pentru verificare stării de ploaie este și el conectat tot prin LM393, tot

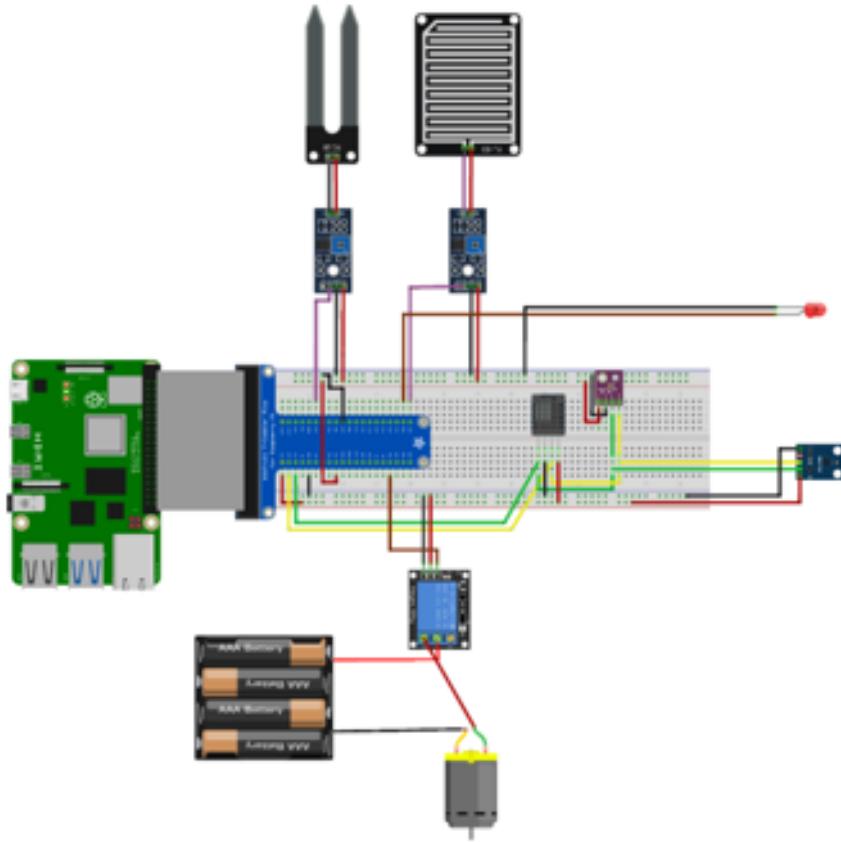


Figura 3.1: Schema in Fritzing

prin doi pini (+) și (-), și este de asemenea conectat la placă prin trei pini. VCC și GND pentru alimentare, tot D0 pentru a transmite date către Raspberry Pi (dar de aceasta data corespunde cu GPIO18).

Becul LED, care simulează lampa cu lumina UV și este conectat la GPIO20 și GND.

Radiatoarele adiționale sunt legate de placă cu pastă termo-conductoare și bandă adezivă termo-conductoare, iar ventilatorul este alimentat cu 5V prin pinii de 5V și GND.

3.2 Partea Software

Pentru partea software, ca și workflow de funcționare al sistemului, la fiecare 5 secunde sunt citite valorile de pe senzori de către scriptul Python atașat placii, apoi placă încarcă datele aferente în baza de date firebase. Singura instanță în care placă citește valori din baza de date este în cazul funcțiilor de udat/lumina adițională. Aplicația Android comunică periodic o dată la 5s cu baza de date pentru obținerea datele de live-feed, apoi, adițional comunică cu baza de date de fiecare dată când este nevoie pentru vreun feature.

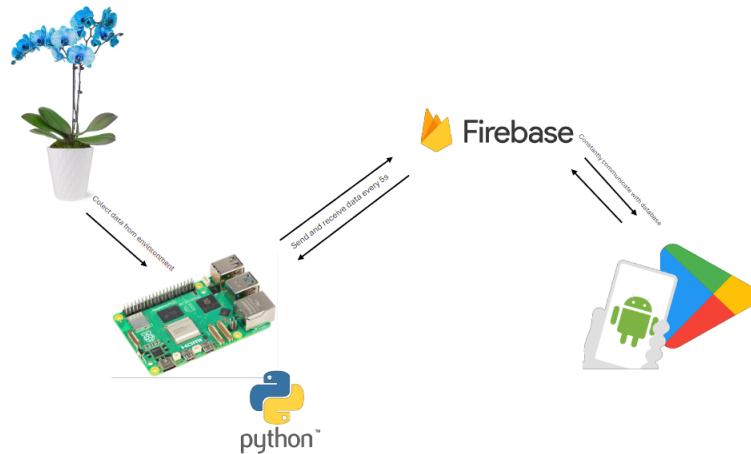


Figura 3.2: Workflow Aplicatie Software

3.2.1 Structura baza de date

Aplicația folosește un real-time database Firebase, de tipul noSQL cu două tabele stocate sub formă de JSON, și o autentificare pe bază de tokene unice implementată prin firebase.

Identifier	Providers	Created ↓	Signed In	User UID
ilieciprian18@yahoo.com	✉	Jun 11, 2024	Jun 11, 2024	nPPKAYmsSLdajTWid7oV4QK...
test@gmail.com	✉	Jun 11, 2024	Jun 11, 2024	DobVMUo000hmbWN2WWvfq...
ilieciprian18@gmail.com	✉	Jun 8, 2024	Aug 28, 2024	kMgfzSKhHPbkCDmoki7lByfV...

Figura 3.3: Authentification Table Firebase

Tabela DataPreferenceApplication conține regimul de îngrijire generat de chatGPT, regim care este regenerat de fiecare dată când se introduce o plantă nouă în aplicație.

Tabela DataPythonRaspberryPi conține informațiile updateate o dată la 5 secunde de către scriptul python de pe placă, adică conținele în principal datele cu care comunică placa.

3.2.2 Script Python pentru Raspberry Pi

Componenta software care se ocupa de citirea, procesarea și trimiterea datelor senzorilor către database este un script în Python atașat placii Raspberry Pi, pe scurt acesta se asigură de comunicarea circuitului cu aplicația android.

Os conține funcții de bază pentru interacționarea cu sistemul de operare, în cazul nostru Raspbian.

Time conține funcții pentru folosirea timpului, de exemplu pentru a rula o funcție cu delay sau pentru a o programa la un anumit interval

```

{
  "DataPreferenceApplication": {
    "City": "Bucharest",
    "Climate": "Temperate Oceanic",
    "Country": "Romania",
    "Description": "To care for an orchid in a continental temperate area, focus on proper watering, feeding, and repotting. Water your orchid early in the day using room-temperature water. Ensure the soil remains moist but not waterlogged. Fertilize every 2 weeks during the growing season (March-September). Repot when roots are crowded, using a well-draining mix. Provide bright, indirect light, avoiding direct sunlight which can scorch the leaves. Temperature should range from 18-26°C (64-79°F) with humidity between 40-70%.",
    "humidityMaxDay": 80,
    "humidityMaxNight": 70,
    "humidityMinDay": 60,
    "humidityMinNight": 50,
    "lightMax": 15000,
    "lightMin": 10000,
    "name": "orchid",
    "pressureMax": 1020,
    "pressureMin": 1000,
    "temperatureMaxDay": 26,
    "temperatureMaxNight": 20,
    "temperatureMinDay": 24,
    "temperatureMinNight": 18,
    "waterIntervalAutomatic": 48
  }
}

```

Figura 3.4: DataPreferenceApplication

```

{
  "DataPythonRaspPi": {
    "boost": 0,
    "humidity": 40.6,
    "lastWetTime": "2024-08-24 00:01:43",
    "latitude": 44.4263,
    "light": 0,
    "longitude": 26.1581,
    "pressure": 1006.65,
    "presure": 1010,
    "rainWetSoil": 0,
    "smartModeActive": 0,
    "temperature": 27.3,
    "waterIntervalSet": 48
  }
}

```

Figura 3.5: DataPythonRaspberryPi

Din librăria **datetime** folosesc funcții pentru citirea orei curente și pentru adăugarea unei anumite perioade de timp la o dată existentă (`deltaTime`)

Pyrebase conține funcții pentru permiterea interacționării cu baza de date Firebase și folosirea de comenzi precum `get`, `set`, `update` etc pe aceasta.

Multiprocessing este o librărie care simulează threadurile

Requests este o librărie ce permite efectuarea de requesturi http catre diferite pagini web pentru a obține diverse date

Restul librăriilor precum **bmp28**, **RPi.GPIO**, **gpiozero** sunt librării special destinate pentru citirea datelor de pe senzori și interacționarea cu acestia

Scriptul Python se conectează la firebase cu ajutorul variabilei config care conține `apiKey`, `authDomain`, `databaseURL` și `storageBucket` pe care le-am preluat din Firebase

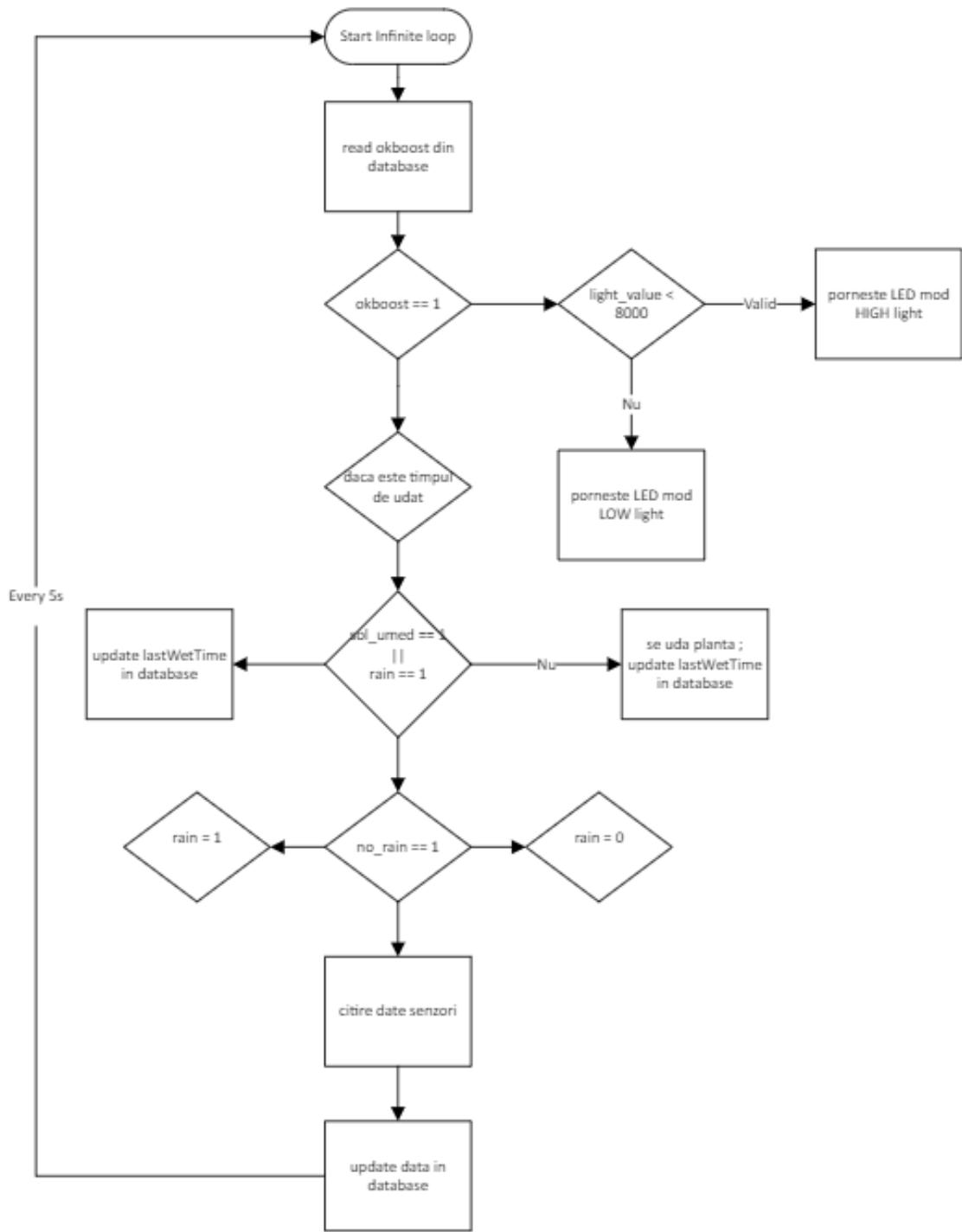


Figura 3.6: Flow Script Python

(Firebase Console > Project Overview > Project Settings > Web Apps > SDK Setup Configuration > npm).

```
#exemplu de initializare si de utilizare
no_rainDevice = InputDevice(18)
sol_umedDevice = InputDevice(21)
```

Pentru a citi date de pe senzorii de tip rezistivi (senzorul pentru umiditate sol și senzorul pentru ploaie) este utilizat modulul comparator LM393 care se ocupă singur de toata munca, trimițând un rezultat binar 0 sau 1, daca este inactiv, respectiv activ.

```
#exemplu citire date senzor BMP280
bmp280 = BMP280(i2c_dev=bus) # detecteaza senzorul bmp280

pressure = round(bmp280.get_pressure(), 2)
temperature1 = bmp280.get_temperature()
```

Al doilea mod de citire a datelor de pe senzori este utilizarea librăriilor specifice pentru fiecare senzor, librarii care au toate funcțiile necesare citirii și interacționării cu acești senzori.

Cel mai dificil mod pentru citirea datelor de pe senzor este în scenariul în care nu există funcții deja existente în librăriile modulelor, datele fiind primite direct sub formă de biți. În acest caz a fost nevoie de definirea câtorva funcții adiționale pentru prelucrarea datelor.

```
#functia convertToNumber care transforma biti in float, apoi functia readLight care citeste date
def convertToNumberFromByte(data):
    # Functie pentru a converti 2 bytes de date
    # intr-un numar zecimal
    resultDataSensor = (data[1] + (256 * data[0])) / 1.2
    return (resultDataSensor)

def readLight(addr=DEVICE):
    dataLightSensor = bus.read_i2c_block_data(addr, ONE_TIME_HIGH_RES_MODE_1)
    return convertToNumberFromByte(dataLightSensor)
```

Pentru funcția de udare am definit o funcție watering() care mai întai verifică dacă solul este umed, în caz pozitiv atunci consideră ca planta a fost udată și se face update la campul "lastWetTime" din database, iar în caz contrar se verifică dacă este timpul să se ude planta și efectuează aceasta operațiune dacă este cazul.

```
#functia watering
def watering():

    nowTimeApplication = datetime.now() #timpul curent în format YYYY-MM-DD H:M:S
    #print(now)
    #print(sol_umed.is_active)
    interval = db.child("waterIntervalSet").get().val() #intervalul de udare
    print(interval)
    waterNeedNow = db.child("lastWetTime").get().val() #valoarea lastWetTime database
    date_format = '%Y-%m-%d %H:%M:%S'
```

```

wateredLastTime = datetime.strptime(waterNeedNow, date_format)
#print(wateredLastTime)
waterMustTime = wateredLastTime + timedelta(hours=interval)
#print(waterNeedNow)
#print(waterMustTime)
if sol_umed.is_active or no_rain.is_active:
    #solulu nu e umed, merge senzoru pe dos
    print("solul nu e umed")
    if now < waterMustTime:
        print("nu udam")
    else:
        print("udam")
        db.update({"lastWetTime" : now.strftime(date_format)})
        GPIO.output(WATERPUMP_PIN, GPIO.LOW)
        time.sleep(10)
        GPIO.output(WATERPUMP_PIN, GPIO.HIGH)
else:
    #check rain
    print("solul este umed")
    db.update({"lastWetTime" : now.strftime(date_format) })

```

Pentru funcția de lightSourceBoost() verific dacă variabila boost din database este 1 sau 0, în caz ca este 1 se activează ledul, iar în caz contrar se oprește/rămâne oprit.

```

#functia lightBoost
def lightSourceBoost():

    okboost = db.child("boost").get().val();

    light_pure = readLight()
    light = round(light_pure, 2)

    if okboost == 1 and light < 5000:
        GPIO.output(LIGHTBOOST_PIN, GPIO.HIGH)
    else:
        GPIO.output(LIGHTBOOST_PIN, GPIO.LOW)

```

In main am doar un loop infinit care o data la 5 secunde citește valorile de pe senzori, apelează watering() și lightSourceBoost(), apoi modifică valorile din baza de date (le da update).

```

while True:

    watering()
    lightSourceBoost()

    light_pure = readLight()

```

```

light = round(light_pure, 2)

pressure = round(bmp280.get_pressure(), 2)
temperature1 = bmp280.get_temperature()

try:
    (temperature2, humidity) = AM2320(1).readSensor()
    temperature = round((temperature1 + temperature2) / 2, 1)
    print(temperature)
except:
    humidity = 0
    temperature = temperature1
    print("eroare senzor am2320")

temperature = round(temperature, 2)

okboost = db.child("boost").get().val()

data = {
    "boost": okb,
    "temperature": temperature,
    "light": light,
    "humidity": humidity,
    "pressure": pressure,
    "latitude": latitude,
    "longitude": longitude
}
db.update(data)

time.sleep(5)

```

3.2.3 Aplicatia Android

Aplicația este ușor de utilizat, user friendly și intuitivă ca și design. Aceasta poate fi instalată și rula pe orice device cu Android care are acces la internet pentru a funcționa și interactionarea cu chatGPT, care este integrat în aplicație. Cateva funcționalități de bază ale aplicației sunt:

Posibilitatea de logare/register/reset password

Vizionarea datelor despre planta în real-time

Monitorizarea permanentă a situației plantei și trimiterea notificărilor în caz de o problemă

Încarcarea unui regim personalizat de îngrijire a plantei selectate pentru zi/noapte

Mod automat pentru udat/lumina adițională

Mod manual pentru setat interval de udare

Pornirea/oprirea manuală pentru lumina adițională a plantei

Pagina pentru regim adițional de îngrijire (schimbare pamant, înmulțire, alte informații)

Pagina special dedicată pentru descrierea incidentelor prezente.

Pagina cu recomandări de plante, generate de chatGPT pe baza plantei curente selectate.

Opțiunea de live chat cu un AI chatbot 24/7 pentru orice întrebare de botanică.

GUI prietenos și intuitiv

Aplicația are un singur Main Activity care rulează în background constant și rezolvă partea de backend a updatarii datelor din baza de date în permanență, iar partea de UI este rezolvată de fragmente care sunt încarcate în activity.

3.2.4 Structura Android Studio

Pachetul com.example.disertatiesmartplantsystemai conține clasele aplicației.

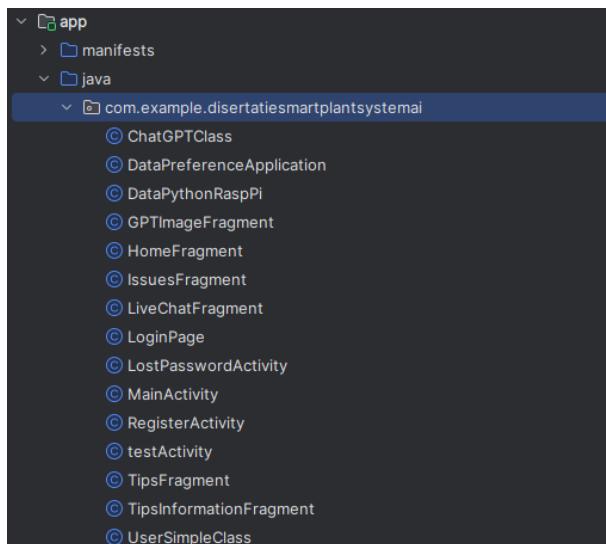


Figura 3.7: Structura clase aplicatie

Partea de UI se află în folderul res, care conține mai multe foldere numite sugestiv:

În folderul color se regăsesc culorile definite de mine precum și codul acestora, în drawable se află imaginile precum și vector asset-urile create de mine, în layouts se află fișierele xml care reprezintă UI paginilor.

3.2.4.1 Login Page | Register Page | Reset Password Page

Pentru design-ul de la login page am ales să folosesc un linear Layout in XML, orientat vertical în care am adăugat un cardview (pe centru), și două texte cu linkuri către paginile de register și reset password. În interiorul cardview-ului se află câmpuri pentru Username, Password și un buton pentru Login.

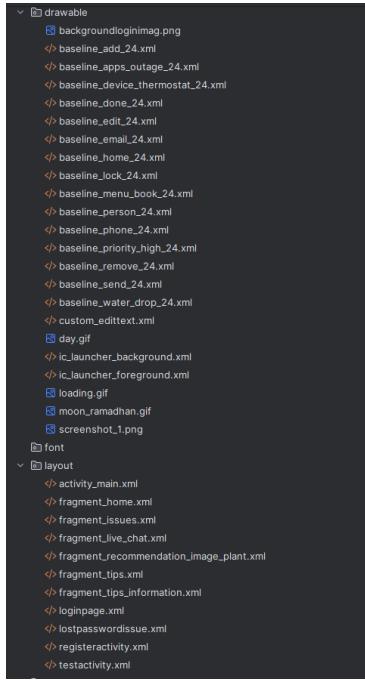


Figura 3.8: Folder Drawable Structura

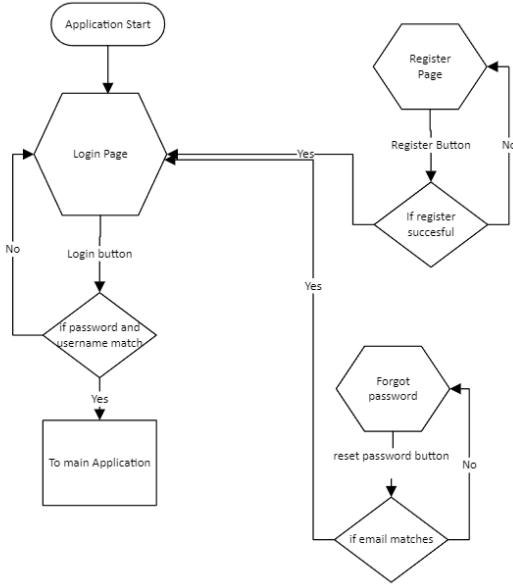


Figura 3.9: Flow Login Register Lost Password

Pentru partea de backend, inițializez câmpurile cu variabilele echivalente din XML, apoi atunci când butonul este apăsat apelez metoda de login în cadrul careia preiau informația din câmpuri și verific cu ajutorul funcției signInWithEmailAndPassword(loginUsername, loginPassword) dacă există un utilizator cu combinația respectivă.

Textele pentru signUp și lost password au și ele functii OnClick care trimit utilizatorul către paginiile respective.

Layoutul pentru SignUp este aproape identic cu cel de la login, singura diferență fiind textul de la linkuri și diferența câmpurilor din cardview. În această pagină regăsim

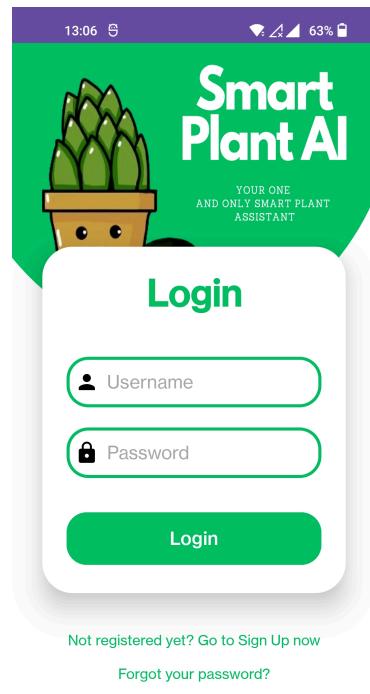


Figura 3.10: Login Page

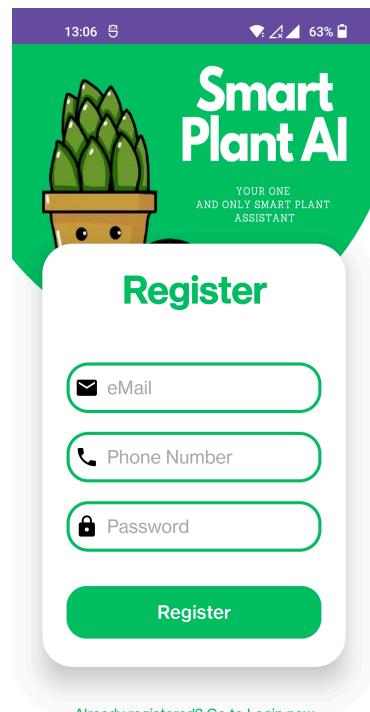


Figura 3.11: Register Page

câmpurile email, phone number, password și un buton de register, câmpuri specifice unei pagini de login. De asemenea fiecare camp are implementat și câte o funcție care obligă respectarea unui anumit format de date, de exemplu pentru câmpul password acesta trebuie să aibe minim 8 caractere, să conțină o literă mare, o literă mică, o cifră dar și un simbol.

Pentru backend se preiau toate datele din câmpuri și se verifică dacă se respectă condițiile (emailul trebuie să fie unic și non-null, numarul de telefon nu trebuie să fie null, iar parola trebuie să aibe minim 8 caractere, o literă mare, una mică, un număr și un simbol. În caz că aceste condiții se respectă se efectuează un insert în baza de date folosind funcția mAuth.createUserWithEmailAndPassword, funcție inclusă în libraria pyrebase. De asemenea ca și în cazul paginii de login există bucați de text care au atașate onClickListeneri de redirect către activitațile de login și reset password.

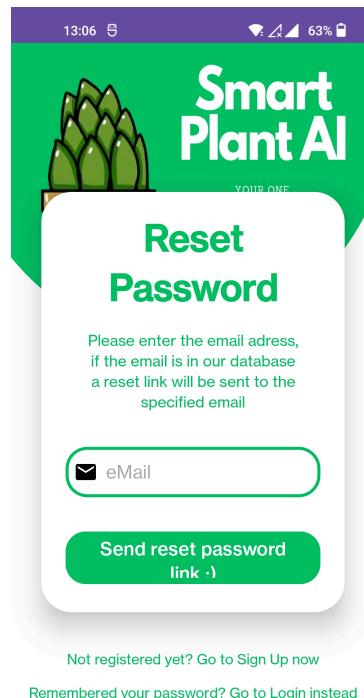


Figura 3.12: Reset Password Page

Layoutul pentru pagina de resetare a parolei este similar celor de login și register, având la bază un Linear Layout. Acest Layout principal conține un cardview care are în interiorul său câmpul introducerii adresei de email atașat contului, un text informativ și un buton. În layoutul principal se află și cele 2 texte specifice, cu linkuri către login și register.

În Backendul acestei activitați se preia emailul din câmpul corespunzator, iar în cazul în care acesta corespunde cu emailul asociat unui cont, utilizatorul va primii un link

de resetare al parolei. Ca și în cazurile anterioare, bucățile de text au implementate OnClickListener pentru a face tranziția către activitățile de login/register.

3.2.4.2 Main Activity

Main activity reprezintă principală și singura activitate a aplicației, în principal această activitate rulează în permanentă în background, indiferent de fragmentele care sunt active la anumite momente ale execuției aplicației. Aceasta asigură încărcarea datelor live în aplicație de fiecare dată când se întâmplă o modificare în baza de date, încărcarea datelor din preferință la începutul aplicatiei, serviciul de notificări, tranziția dintre ciclul de zi și noapte al aplicatiei.

```
public Float temperatureCurrent, lightCurrent, pressureCurrent, humidityCurrent;
public Float longitude, latitude ;
public Integer boostLight, userWaterInterval, wetSoilCheck, waterIntervalPreference,
smartModeSet;
public Float temperatureMinDayPreference, temperatureMinNightPreference;
public Float temperatureMaxDayPreference, temperatureMaxNightPreference;
public Float lightMinPreference, lightMaxPreference;
public Float pressureMinPreference, pressureMaxPreference;
public Float humidityMinDayPreference, humidityMaxDayPreference;
public Float humidityMinNightPreference, humidityMaxNightPreference;
public Float temperatureMinPreference, temperatureMaxPreference, humidityMinPreference;
public Float humidityMaxPreference, lightMaxDayPreference, lightMinDayPreference;
public String City, Country, Climate, Cycle;
public String name;
public String testX;
private FirebaseAuth mAuth;
private FirebaseUser user;
private DatabaseReference reference, referencePreferenceSettings, newReference;
private String userID;
public String x;

ActivityMainBinding binding;
boolean firstRun = true;
Handler handler=new Handler();
boolean needNotif = false;
private Button experience;
private String emailUserApplication;
```

Pentru a reincarca de fiecare data cand exista o modificare in baza de date am utilizat un ValueEventListener care asculta pe referinta ”/DataPythonRaspPi” din baza de date. Astfel de fiecare data cand o valoare se modifica date din aplicatie sunt reincarcate.

Am implementat si un smart mode care seteaza intervalul de udare cu cel recomandat de chatGPT, iar in cazul in care lumina plantei este insuficienta se activeaza modul suplimentar de lumina.

```

//exemplu logica smartMode

smartModeSet = dataSnapshot.child("smartModeActive").getValue(Integer.class);

if((smartModeSet == 1) && (lightCurrent < lightMinPreference))
{
    newRefference = FirebaseDatabase.getInstance().getReference("/DataPythonRaspPi/boost");
    newRefference.setValue(1);
    boostLight = 1;
}
if(smartModeSet == 1 && lightCurrent > lightMaxPreference && boostLight == 1)
{
    newRefference = FirebaseDatabase.getInstance().getReference("/DataPythonRaspPi/boost");
    newRefference.setValue(0);
    boostLight = 0;
}
}

```

Valorile din preferință (algoritmul recomandat de chatGPT) este stocat în tabela ”DataPreferenceApplication”, datele din această tabela fiind încarcate doar o dată din baza de date. De asemenea în timpul încărcării datelor se face și o verificare a stării ciclului de zi/noapte astfel încât să se seteze ca valori maxime/minime curente, valorile corect aferente ciclului temporal în care se află aplicația.

Pentru a updata în permanentă ciclul de zi/noapte am ales să utilizez un Runnable care o dată la 60s verifică dacă timpul curent se incadrează pentru zi sau noapte (am considerat noapte în intervalul 22:00 - 06:00, iar zi în rest).

```

Runnable updateTextRunnable=new Runnable(){
    public void run() {

        if( firstRun == true)
        {
            firstRun = false;
        }
        else {

            Integer currentTimeApplication = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

            if(currentTimeApplication > 22 || currentTimeApplication < 6)
            {
                Cycle = "night";
                humidityMinPreference = humidityMinNightPreference;
                humidityMaxPreference = humidityMaxNightPreference;
                temperatureMinPreference = temperatureMinNightPreference;
            }
        }
    }
}

```

```

        temperatureMaxPreference = temperatureMaxNightPreference;
    }
    else {
        Cycle = "day";
        humidityMinPreference = humidityMinDayPreference;
        humidityMaxPreference = humidityMaxDayPreference;
        temperatureMinPreference = temperatureMinDayPreference;
        temperatureMaxPreference = temperatureMaxDayPreference;
    }
    Log.d("CREATION", Cycle);

    if(temperatureCurrent < temperatureMinPreference || temperatureCurrent > temperature
    {
        //send notif
        if( needNotif == false)
        {
            needNotif = true;
            //send notif
            Log.d("CREATION", "se trimite notif");
            sendNotification("Issue", "Your plant needs your attention");
        }
    }
    else {
        needNotif = false;
    }

}
handler.postDelayed(this, 60000);
}
};


```

Pentru serviciul de notificări verific periodic dacă un element din mediul înconjurător nu este în parametrii optimi, iar în caz afirmativ suplimentat de inexistentă unei notificări deja trimise, aplicația va trimite o notificare către utilizator.

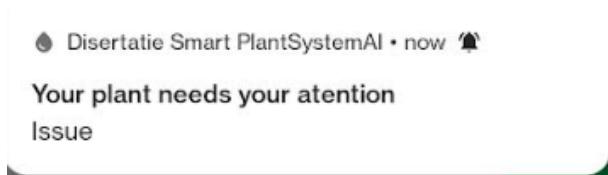


Figura 3.13: Notifications

3.2.4.3 Navigation Menu

Pentru navigation menu am ales să folosesc un bottom navigation care este atașat pe Main Activity. Datorită atașării sale direct pe main activity fiecare fragment care este

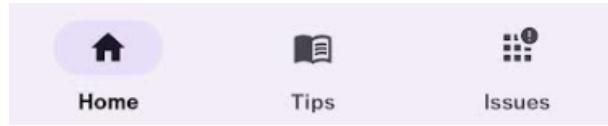


Figura 3.14: Bottom Navigation Bar

atașat acesteia o conține în partea de jos a paginii. Pe navigation bar se află trei butoane, pentru Home Screen, Issues Screen și Tips Screen.

3.2.4.4 ChatGPT Class

Pentru aplicație am folosit 2 versiuni de chatGPT, chatGPT 4o și Dall-E 3 pentru generarea de imagini. În termeni de inteligență a acestui AI, există o diferență foarte mare de la o versiune la cealaltă, ca și comparație voi arăta diferența dintre regimul de îngrijire recomandat de versiunile de chatGPT 3.5 turbo, 4o, 4o-mini pe aceași plantă.

```
ASSISTANT
temperature (day): 28-32
temperature (night): 18-22
humidity (day): 40-50
humidity (night): 60-70
light index (lx) (day): 20000-25000
pressure: 800-900
watering interval (h): 72
```

Figura 3.15: chatGPT 3.5-turbo

```
ASSISTANT
temperature (day): 25-35
temperature (night): 10-20
humidity (day): 20-30
humidity (night): 30-50
light index (lx) (day): 15000-30000
pressure: 950-1050
watering interval (h): 72
```

Figura 3.16: chatGPT 4o-mini

```
ASSISTANT
temperature (day): 24-29
temperature (night): 16-20
humidity (day): 10-30
humidity (night): 30-40
light index (lx) (day): 20000-40000
pressure: 950-1050
watering interval (h): 168
```

Figura 3.17: chatGPT 4o

După cum se poate observa, cel mai bun regim de îngrijire a fost efectuat de chatGPT 4o, care este și cel mai modern și performant. De asemenea un alt argument legat de

faptul ca AI nu poate încă să fie încredințat cu 100% îngrijirea unei plante sunt erorile din răspunsuri, erori ce sunt din ce în ce mai rare cu cât este o utilizată o verisune mai bună.

În timpul dezvoltării am observat că problemele contextuale care susțin faptul că chatGPT nu este încă pregătit pentru a avea singur grija de o plantă, de exemplu dacă pui orice cuvânt urmat de cuvântul "tree", chatGPT o să credă că este vorba de o plantă.



Figura 3.18: Bad Quote Answer

Principala dificultate în utilizarea acestui API a fost găsirea unui mod în care chatGPT să raspundă cât de căt într-un format pe care să îl pot prelucra. De obicei, cea mai de succes metodă pe care am găsit-o a fost să folosesc 2-3 metode de prelucrare pe 2-3 formate, iar dacă chatGPT nu a generat date într-unul dintre formatele aceleia, să îl fac să regenereze răspunsul.

Funcțiile specifice pentru accesare lui chatGPT API au ca și parametrii un quote. Pentru a face requesturi https am utilizat librăria Volley. Există două tipuri de funcții cu care accesez chatGPT, funcțiile în care aștept să primesc un răspuns de tip String de la chatGPT, iar celălalt tip în care mă aștept să primesc o imagine generată de acesta.

Ca și exemplu voi arăta structura pentru generarea de regim de îngrijire al plantei, este asemănatoare cu cea pentru generarea de imagini, ca și regulă generală fac un apel către API, apoi prelucrez răspunsul după cum am nevoie.

```

RequestQueue queue = Volley.newRequestQueue(getActivity().getApplicationContext());
// sau "https://api.openai.com/v1/images/generations" pentru imagini
String url = "https://api.openai.com/v1/chat/completions";
// sau poate avea parametru luat din funcție, în acest caz nu este nevoie
String prompt = "Answer in only one word (Integer) for each of the questions, format the answer as
+ "what are the temperature for " + plantVerifyName + " at day and at night ( interval ) \n"
+ "what is the humidity for " + plantVerifyName + " at day and at night ( interval ) \n"
+ "what is the light index in lx for " + plantVerifyName + " at day (interval) \n"
+ "what is the pressure for " + plantVerifyName + " (interval ) \n"
+ "what is the watering interval in h for " + plantVerifyName + " (single value ) \n";

JSONObject jsonObject = new JSONObject();

try {

```



```

mainApplicationData.temperatureMinNightPreference = Float.valueOf(tempResultVector.elementAt(0));
mainApplicationData.temperatureMaxNightPreference = Float.valueOf(tempResultVector.elementAt(1));

}

if (i == 2) {
referenceInsert.child("humidityMinDay").setValue(tempResultVector.elementAt(0));
referenceInsert.child("humidityMaxDay").setValue(tempResultVector.elementAt(1));

mainApplicationData.humidityMinDayPreference = Float.valueOf(tempResultVector.elementAt(0));
mainApplicationData.humidityMaxDayPreference = Float.valueOf(tempResultVector.elementAt(1));
}

if (i == 3) {
referenceInsert.child("humidityMinNight").setValue(tempResultVector.elementAt(0));
referenceInsert.child("humidityMaxNight").setValue(tempResultVector.elementAt(1));

mainApplicationData.humidityMinNightPreference = Float.valueOf(tempResultVector.elementAt(0));
mainApplicationData.humidityMaxNightPreference = Float.valueOf(tempResultVector.elementAt(1));
}

if (i == 4) {
referenceInsert.child("lightMin").setValue(tempResultVector.elementAt(0));
referenceInsert.child("lightMax").setValue(tempResultVector.elementAt(1));

mainApplicationData.lightMinDayPreference = Float.valueOf(tempResultVector.elementAt(0));
mainApplicationData.lightMaxDayPreference = Float.valueOf(tempResultVector.elementAt(1));
}

if (i == 5) {
referenceInsert.child("pressureMin").setValue(tempResultVector.elementAt(0));
referenceInsert.child("pressureMax").setValue(tempResultVector.elementAt(1));

mainApplicationData.pressureMinPreference = Float.valueOf(tempResultVector.elementAt(0));
mainApplicationData.pressureMaxPreference = Float.valueOf(tempResultVector.elementAt(1));
}

if (i == 6) {
referenceInsert.child("waterIntervalAutomatic").setValue(tempResultVector.elementAt(0));

mainApplicationData.waterIntervalPreference = tempResultVector.elementAt(0);
}

}

} catch (Exception e) {
// Toast.makeText(MainActivity.this, "ChatGPT gave bad answer, please try again", Toast.LENGTH_LONG);
Toast.makeText(getContext(), "ChatGPT gave bad answer, pleaseeee try again", Toast.LENGTH_LONG);
}

```

```

    }

    plantName.setText(plantVerifyName);
    referenceInsert.child("name").setValue(plantVerifyName);
    mainApplicationData.name = plantVerifyName;
} else {
    Toast.makeText(getContext(), "ChatGPT gave bad answer, please try again", Toast.LENGTH_LONG).show();
}

}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        ...
    }
};

@Override
public Map<String, String> getHeaders() throws AuthFailureError {
    HashMap<String, String> headerChatGPT = new HashMap<>();
    headerChatGPT.put("Content-Type", "application/json");
    headerChatGPT.put("Authorization", "Bearer sk-I2kfVHadYwzohaAkR6DT3B1bkFJ9ZNygmvd0uM8Z7bfy839");
    return headerChatGPT;
}

};

queue.add(stringRequest);
} catch (JSONException e){
    throw new RuntimeException(e);
}

public static String extractValuesGPT(String n){

    String result = n.replaceAll("[^\\d-]", "");
    return result;
}

public static Vector<Integer> extractIntegerValues(String n){

    String valuesNoSpace = n.replace("\\s", "");
    String[] result = valuesNoSpace.split("-");
    Vector<Integer> resultVectorGPT = new Vector<>();
    if( result.length == 1)
    {
        resultVectorGPT.add(Integer.parseInt(result[0]));
    }
    else {
        resultVectorGPT.add(Integer.parseInt(result[0]));
        resultVectorGPT.add(Integer.parseInt(result[1]));
    }
}

```

```
        }  
  
    return resultVectorGPT;  
  
}
```

3.2.4.5 Home Page

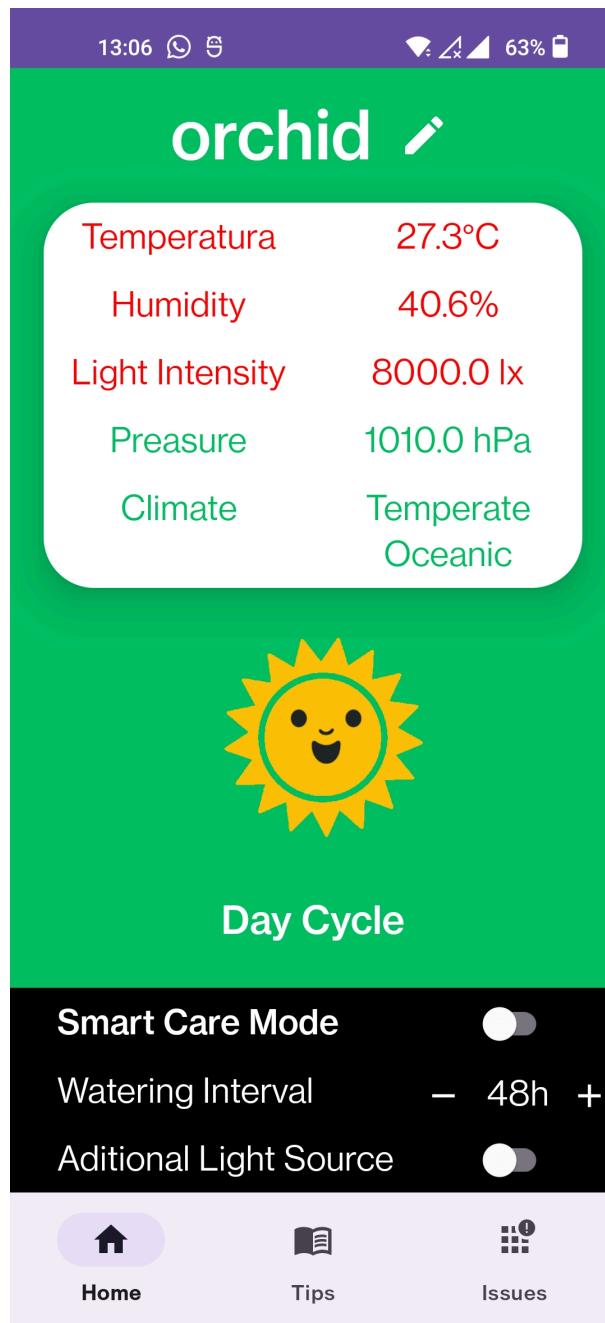


Figura 3.19: Home Page

Layout

Designul se află intr-un Relative Layout ca și root element. Interiorul acestuia poate fi separat în trei regiuni.

Prima dintre ele conține un linear layout orientat orizontal în care se află numele planetei curente precum și un buton de edit.

A doua parte este reprezentată de un cardview în care se află textview pentru temperatură, umiditate, intensitatea luminii, presiunea atmosferică și climatul curent în care se află planta. Culoarea elementelor de tip textView este de culoare verde dacă valorile lor sunt în normele optime, și rosii în caz contrar. În aceeași secțiune regăsim și un Gif, plasat sub acest cardview, în interiorul căruia se află starea aplicației (Loading... / Mod Day / Mod Night).

In cea de-a treia regiune, aflată într-un layout linear orientat orizontal ce conține la rândul său două layouturi lineare orientate vertical se află sliderele/butoanele pentru setarea modului "Smart Care", Intervalul de udare și de activare a luminii suplimentare.

Backend

In partea de inițializare în funcția onCreate() preiau valorile din aplicație, apoi modific textul din textView-urile aferente valorilor, funcție de valoarea acestora în raport cu parametrii de îngrijire culoarea textului este verde sau rosie.

```
//exemplu initializare temperature
mainApplicationData = (MainActivity) getActivity();
Log.d("CREATION", mainApplicationData.temperatureCurrent.toString());

String tempStringShow = mainApplicationData.temperatureCurrent.toString() + "°C";
temperatureHomeShow.setText(tempStringShow);

if(mainApplicationData.temperatureCurrent < mainApplicationData.temperatureMinPreference
|| mainApplicationData.temperatureCurrent > mainApplicationData.temperatureMaxPreference)
{
    temperatureHomeShow.setTextColor(Color.RED);
    temperatureShowName.setTextColor(Color.RED);
}
else {
    temperatureHomeShow.setTextColor(Color.parseColor("#00bf63"));
    temperatureShowName.setTextColor(Color.parseColor("#00bf63"));
}

//update imagine Gif pentru dai/night

ShowImageGif = view.findViewById(R.id.cycleGifShow);
if (mainApplicationData.Cycle.equals("day"))
{
    ShowImageGif.setImageResource(R.drawable.day);
    textGifUI.setText("Day Cycle");
}
```

```

    }

    else {
        // night
        ShowImageGif.setImageResource(R.drawable.moon_ramadhan);
        textGifUI.setText("Night Cycle");
    }

    if(mainApplicationData.smartModeSet == 1)
    {
        smartModeCheck.setChecked(true);
    }
    else {
        smartModeCheck.setChecked(false);
    }

    if(smartModeCheck.isChecked())
    {

        // update de UI
        // addWaterInterval, minusWaterInterval, waterIntervalShow, lightModeCheck gone
        // autoTextWater,autoTextLight visible

    }
    else {
        // update de UI not checked
        // addWaterInterval, minusWaterInterval, waterIntervalShow, lightModeCheck visible
        // autoTextWater,autoTextLight gone
    }
}

```

Functiile de addWater() si minusWater() modifica intervalul de udare al plantei si se ocupa si de updateUI() pentru acestea.

```

private void setAddWater(){

    Integer auxAddWater = mainApplicationData.userWaterInterval;
    if(mainApplicationData.userWaterInterval < 12)
    {
        referenceAdd = FirebaseDatabase.getInstance()
            .getReference("/DataPythonRaspPi/waterIntervalSet");
        referenceAdd.setValue(auxAddWater + 1);
    }
    else {
        referenceAdd = FirebaseDatabase.getInstance()
            .getReference("/DataPythonRaspPi/waterIntervalSet");
    }
}

```

```

        referenceAdd.setValue(auxAddWater + 12);
    }
}

private void setMinusWater(){
    Integer auxMinusWater = mainApplicationData.userWaterInterval;

    if(mainApplicationData.userWaterInterval > 12)
    {
        referenceMinus = FirebaseDatabase.getInstance()
            .getReference("/DataPythonRaspPi/waterIntervalSet");
        referenceMinus.setValue(auxMinusWater - 12);
    }
    else {
        referenceMinus = FirebaseDatabase.getInstance()
            .getReference("/DataPythonRaspPi/waterIntervalSet");
        referenceMinus.setValue(auxMinusWater - 1);
    }
}

```

Functia pentru editPlantName trece aplicatia in modul de modificare al plantei selectate, modificand textul cu numele plantei intr-un editText, iar butonul de edit intr-unul de confirm.

```

private void editPlantNameMethod(){

    plantName.setVisibility(View.GONE);
    editNameButton.setVisibility(View.GONE);

    tempPlantName.setVisibility(View.VISIBLE);
    confirmNameButton.setVisibility(View.VISIBLE);

    tempPlantName.setHint(mainApplicationData.name);
}

```

Functia pentru confirmPlantNameMethod() face un apel catre chatGPT API pentru a verifica dacă presupusul nume de plantă introdus este o plantă validă. În caz afirmativ se incarcă în memorie un regim de îngrijire adekvat plantei, iar în caz contrar se afisează un mesaj de eroare. În timpul încarcării raspunsului Gif-ul pentru starea aplicației se află în modul Loading..., urmând ca la finalul operatiunii să se treacă pe modul day/night.

```
String prompt = "Is " + " " + plantVerifyName + " " + "a plant? Answer with yes or no only";
```

```

ShowImageGif.setImageResource(R.drawable.loading);

if(answerCheckGPT.equals("Yes"))
{
    //load plant care request to chatGPT

    if (mainApplicationData.Cycle.equals("day"))
    {
        ShowImageGif.setImageResource(R.drawable.day);
        textGifUI.setText("Day Cycle");
    }
    else {
        // night
        ShowImageGif.setImageResource(R.drawable.moon_ramadhan);
        textGifUI.setText("Night Cycle");
    }
}
else {
    Toast.makeText(getContext(), "That... is not... a plant...", Toast.LENGTH_LONG).show();

    if (mainApplicationData.Cycle.equals("day"))
    {
        ShowImageGif.setImageResource(R.drawable.day);
        textGifUI.setText("Day Cycle");
    }
    else {
        // night
        ShowImageGif.setImageResource(R.drawable.moon_ramadhan);
        textGifUI.setText("Night Cycle");
    }
}
}

```

3.2.4.6 Issues Page

Layout

Layout-ul de la issues se află într-un RelativeLayout principal în cadrul căruia se află un Linear layout ce conține erorile active (dacă există). În caz că nu există erori este afișat un textView ”Everything seems to be fine for the time beeing...”. În josul paginii se află un buton pentru logout.

Backend

În interiorul funcției de onCreate() se verifică dacă există vreo problemă cu planta, iar în caz afirmativ se afișează eroarea sub forma unui design stabilit din cod.

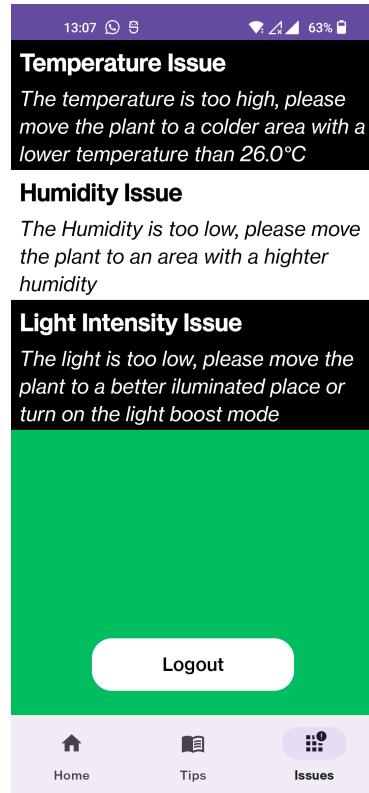


Figura 3.20: Issues Page

```
//exemplu verificare de ceorii si adaugarea lor in vectorul de erori

Vector<String> activeIssuesTitle = new Vector<>();
Vector<String> activeIssueDescription = new Vector<>();

if(mainDataIssues.temperatureCurrent < mainDataIssues.temperatureMinPreference || mainDataIssues
{
    activeIssuesTitle.add("Temperature Issue");

    if(mainDataIssues.temperatureCurrent < mainDataIssues.temperatureMinPreference)
    {
        activeIssueDescription.add("The temperature is too low, please move the plant to a hotter
        area with higher temperature");
    }
    else {
        activeIssueDescription.add("The temperature is too high, please move the plant to a colder
        area with a lower temperature than " + mainDataIssues.temperatureMaxPreference + "°C");
    }
}

//exemplu afisare erori in caz ca exista

Integer count = 0;
```

```

if(activeIssuesTitle.size() >= 1)
{
    Log.d("CREATION", "Some error logs");
    for(int i = 0; i < activeIssuesTitle.size();i++)
    {

        TextView textViewX = new TextView(getActivity());
        textViewX.setText(activeIssuesTitle.elementAt(i));
        RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT, RelativeLayout.LayoutParams.WRAP_CONTENT);
        textViewX.setLayoutParams(params);
        textViewX.setTextSize(23);
        textViewX.setBackgroundColor(Color.BLACK);
        textViewX.setTextColor(Color.WHITE);
        textViewX.setPadding(25,25,25,25);
        textViewX.setTypeface(Typeface.DEFAULT_BOLD);

        TextView issueDescriptionText = new TextView(getActivity());
        issueDescriptionText.setText(activeIssueDescription.elementAt(i));
        RelativeLayout.LayoutParams paramsDescriptionIssue = new RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT, RelativeLayout.LayoutParams.WRAP_CONTENT);
        issueDescriptionText.setLayoutParams(paramsDescriptionIssue);
        issueDescriptionText.setTextSize(20);
        issueDescriptionText.setBackgroundColor(Color.BLACK);
        issueDescriptionText.setTextColor(Color.WHITE);
        issueDescriptionText.setPadding(25,0,25,10);
        issueDescriptionText.setTypeface(null, Typeface.ITALIC);
        if( count% 2 == 0) {

            textView.setBackgroundColor(Color.BLACK);
            issueDescriptionText.setBackgroundColor(Color.BLACK);
            textView.setTextColor(Color.WHITE);
            issueDescriptionText.setTextColor(Color.WHITE);

        }
        else {
            textView.setBackgroundColor(Color.WHITE);
            issueDescriptionText.setBackgroundColor(Color.WHITE);
            textView.setTextColor(Color.BLACK);
            issueDescriptionText.setTextColor(Color.BLACK);
        }

        thisLinearIssues.addView(textView);
        thisLinearIssues.addView(issueDescriptionText);
        count++;
    }
}
else{

```

```

goodStatusShow.setVisibility(View.VISIBLE);
}
//functie logout()
private void logout(){
    FirebaseAuth.getInstance().signOut();
}

```

3.2.4.7 Tips Screen Select

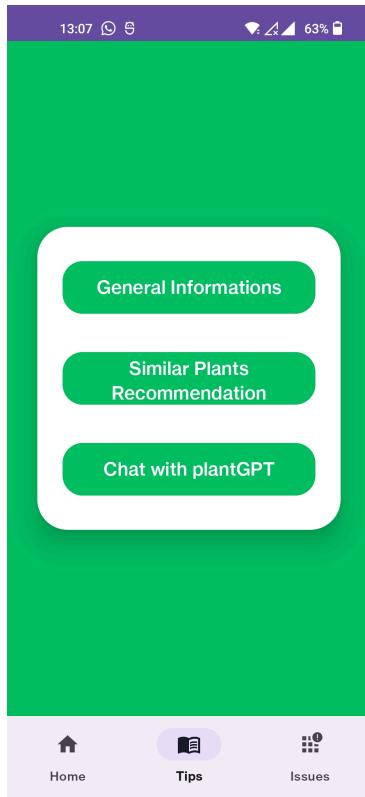


Figura 3.21: Tips Select Screen

Layout

În acest fragment există doar un cardview central în interiorul căruia se află trei butoane care redirecționează utilizatorul către paginile specifice.

Backend

În clasa aceasta se află doar implementări de onClick() către paginile specifice fiecărui buton

3.2.4.8 General Information Page

Layout

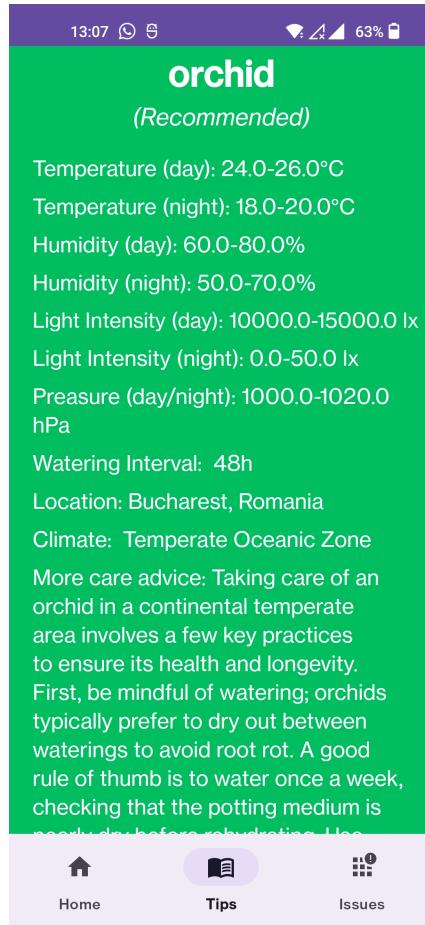


Figura 3.22: General Information Page

Ca și design această pagină conține mai multe textView de diferite mărimi așezate într-un linear layout, cu orientare verticală. Se regăsesc câmpuri precum temperatura preferată pentru zi și noapte, umiditatea preferată pentru zi și noapte, intensitatea luminoasă pentru zi și noapte, presiune atmosferică recomandată, interval de udare și multe altele. De asemenea campurile location, climate, more caring advice au ca valoare afisata Loading... până se încarcă recomandările generate pe loc de către chatGPT

Backend

Pentru partea de backend, se preiau datele din DataPreferenceApplication pentru toate textView urile în afară de Location, Climate, More care advice unde este necesară o generare de text pe loc.

```
//exemplu requests pentru generarea datelor
```

```
mainDataGeneral = (MainActivity) getActivity();

longitudeGeneralInformation = mainDataGeneral.longitude;
latitudeGeneralInformation = mainDataGeneral.latitude;
```

```

chatGPT("What is the climate at" + " " + latitudeGeneralInformation + " and " +
longitudeGeneralInformation + " ( maximum 3 words each word starts with capital letter, don't put
,), generalShowClimate);
chatGPT("What city and from which country is at this coordinates" + " " +
latitudeGeneralInformation + " and " + longitudeGeneralInformation + ", answer in 2 words
separated by ,", generalShowCityFromCountry);
chatGPT("Tell me in one paragraph made of 300 words or less how to take care of an" + " " +
mainDataGeneral.name + " " + "( i live in a continental temperate area ), do not tell me about
temperature, light or humidity", generalShowCare);

```

3.2.4.9 Recommendations Page

Layout

Pentru partea de recomandări, aceasta este în interiorul unui Linear Layout care conține mai multe layout-uri relative pentru a putea aranja elementele unul față de celălalt. Pagina conține 3 poze, una pe stânga, a doua pe dreapta și a treia tot pe stânga, respectiv textViews pentru numele și descrierea plantelor din aceste imagini, pe dreapta sau pe stânga, în funcție de unde se află imaginea. Imaginele sunt generate și încarcate dinamic de către DALL-3



Figura 3.23: Generate Recommendations chatGPT

Backend

Pentru partea de backend generez descrieri și nume de plante care s-ar potrivii cu planta selectată curent, descrieri și imagini generate cu chatGPT pentru text și DALL-e3 pentru imagini. În cazul în care chatGPT nu generează text conform unui format pe care

să îl pot prelucra, regenerez raspunsul primit. Clasa lui chatGPT pentru imagini este asemănătoare cu cea pentru text, singura diferență fiind parametrii diferiți, iar în aceasta situație a fost nevoie de marirea timpului de timeout la 60s folosind .setRetryPolicy()

```

topImageGPT = view.findViewById(R.id.imageGeneratedTop);
middleImageGPT = view.findViewById(R.id.imageGeneratedMiddle);
bottomImageGPT = view.findViewById(R.id.generatedImageBottom);

generateImagedata = (MainActivity) getActivity();

//request chatGPT

String prompt = "I have an" + " " + generateImagedata.name + ", recommend me 3 more plants
that you think that I would like. Format the answer with plant_name: and a new paragraph for
each answer, no more than 5 rows of text for each answer";

//apoi prelucrez raspunsul care este in answer

String str[] = answer.split("\n");
Vector<String> answerVectorString = new Vector<>();
Log.d("CREATION", String.valueOf(str.length));
if(str.length == 8)
{
    for( int i = 0; i < str.length;i++)
    {
        Log.d("CREATION", str[i].replaceAll("[*:]", "").replace("\n", ""));
        String check = str[i].replaceAll("[*:]", "").replace("\n", "");
        if(!check.equals(""))
        {
            Log.d("CREATION", "valid");
            answerVectorString.add(check);
        }
    }
    topTitleGPT.setText(answerVectorString.elementAt(0));
    topSpecsGPT.setText(answerVectorString.elementAt(1));
    topSpecsGPT.setMovementMethod(new ScrollingMovementMethod());
}

topSpecsGPT.setVisibility(View.VISIBLE);

generateImage("Generate me an image of " + answerVectorString.elementAt(0), topImageGPT);

middleTitleGPT.setText(answerVectorString.elementAt(2));
middleSpecsGPT.setText(answerVectorString.elementAt(3));
middleSpecsGPT.setMovementMethod(new ScrollingMovementMethod());

middleSpecsGPT.setVisibility(View.VISIBLE);

```

```

        generateImage("Generate me an image of " + answerVectorString.elementAt(2), middleImageGPT);

        bottomTitleGPT.setText(answerVectorString.elementAt(4));
        bottomSpecsGPT.setText(answerVectorString.elementAt(5));
        bottomSpecsGPT.setMovementMethod(new ScrollingMovementMethod());

        bottomSpecsGPT.setVisibility(View.VISIBLE);

        generateImage("Generate me an image of " + answerVectorString.elementAt(4), bottomImageGPT);
    }
    else{
        Toast.makeText(getContext(), "Regenerating...", Toast.LENGTH_LONG).show();
    }

//aditional a trebuit sa maresc timpul dupa care se da timeout deoarece dureaza mult

```

3.2.4.10 Live Chat Page

Layout

Pentru această pagina, elementele se află într-un relative layout. În partea de jos a paginii se găsește un editText în care utilizatorul poate să scrie întrebarea pentru chatGPT și un buton de send. Pentru a simula un chat, mesajele sunt scrise sub forma de cardview cu un text în interior, și o data langa, pe stânga sau dreapta în funcție dacă sunt trimise de utilizator sau de AI. În susul paginii se găsește un textView cu textul de "Live Chat".

Backend

O data ce utilizatorul apasă butonul de send, se preiau datele din textView și se efectuează un request către chatGPT pentru a verifica dacă întrebare tine de botanică, în caz afirmativ chatGPT răspunde la întrebare, iar în caz contrar se afisează un răspuns "I am sorry, I only know to answer to question related to botany".

```

String promptForGPT = getQuoteGPT.getText().toString();

getQuoteGPT.getText().clear();

String prompt = " Answer with Yes or No only, is question " + "!" + promptForGPT + "!" +
"related to botany";

String answer = requestChatGPT(prompdff)

```

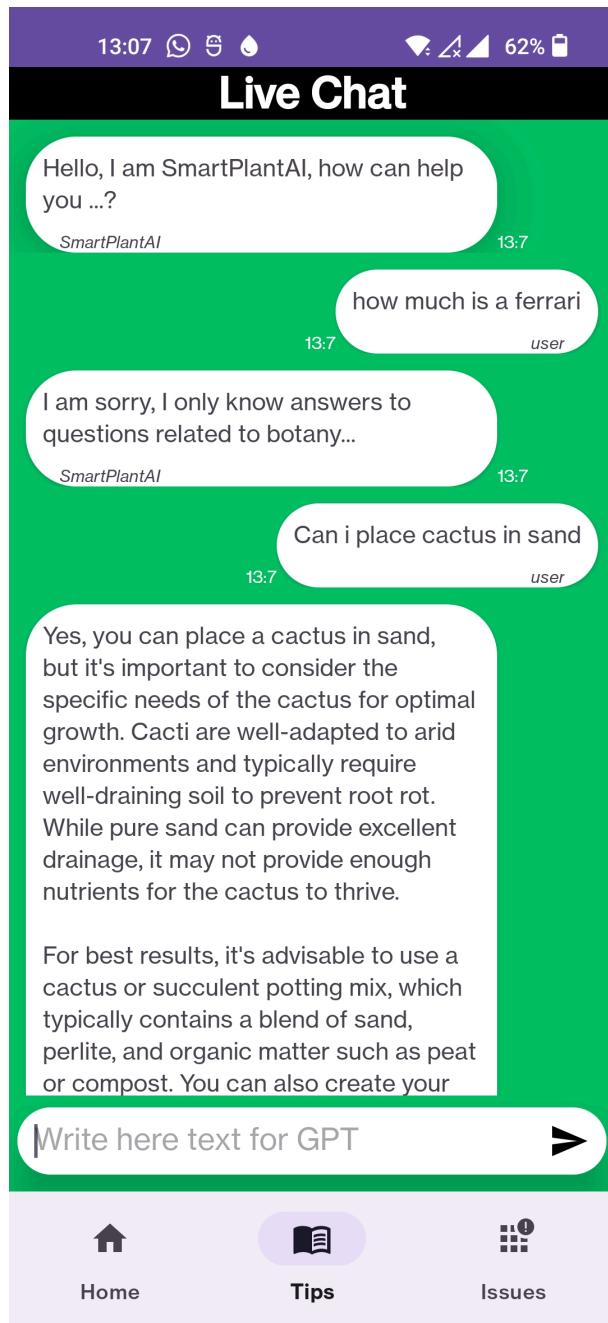


Figura 3.24: Live Chat Page

```
String str = answer.replace(".", "");  
  
if (str.equals("Yes"))  
{  
    //request catre chatGPT  
    // generez un cardview pe dreapta in care se afla raspunsul  
    // il adaug in relative layout apoi in root layout  
}  
}  
else {
```

```

    // generez un cardview pe dreapta in care se afla textul
    // "I am sorry, I only know to answer to question related to botany".
    // il adaug in relative layout apoi in root layout
}

```

3.3 Probleme Intampinate

Pentru partea de hardware au existat multe probleme legate de nefuncționarea senzorilor și chiar și a plăcii Raspberry Pi, fiind nevoie chiar de schimbarea primei plăci primite deoarece nu era funcțională. O parte din senzori au fost cumpărați din China, iar și aceștia au avut probleme unii chiar venind deja arși. O altă problemă pe partea fizică a fost dată de faptul că a fost foarte greu să găsesc un releu de 3V, majoritatea venind sub formă de 5V iar placa nu dădea destul curent pentru a activa releul de 5V folosit pentru pompa de apa.

Pe partea de software, cele mai mari probleme întâlnite au fost date de răspunsurile venite de la chatGPT care nu erau într-un format constant. Alte probleme legate de API au fost din cauza documentației outdated de la ei de pe site, pentru unele modele nu mai erau scrisi parametrii și a trebuit să incerc pană am reusit să îi fac să meargă.

O altă problema majoră a fost găsirea unui sistem de optimizări astfel încât aplicația să nu se încarce prea greu, astfel am ajuns la soluția de încarcare a regimului de îngrijire într-o tabelă în baza de date, apoi încarcarea lui din aceasta, în loc de regenerarea acestuia de fiecare dată când aplicația era deschisă.

Sistemul de day/night a venit și el cu problemele sale deoarece nu îmi funcționau cum trebuie threads sau synchronise și a trebuit să gasesc o cale inventivă, prin intermediul librăriei Runnable pentru a verifica o data la 60s dacă aplicația trebuie trecută pe alt ciclu.

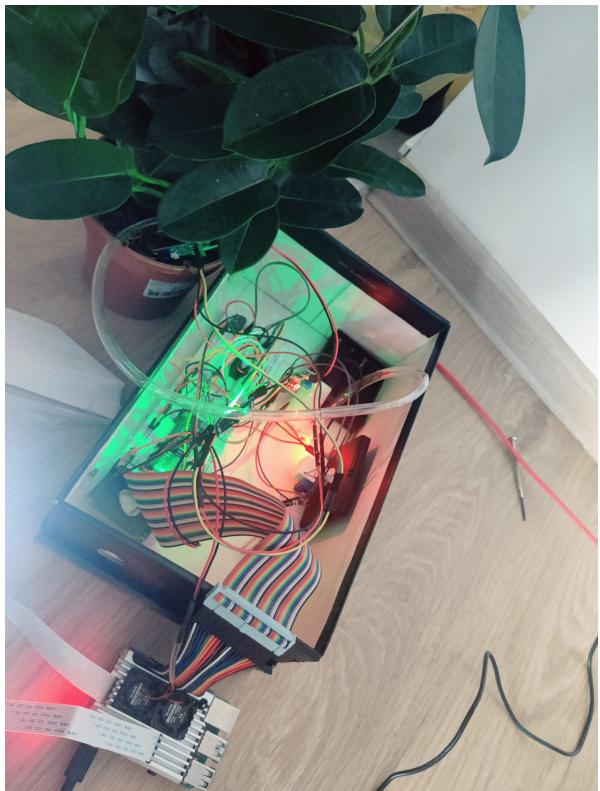
Tot în Main Activity am întampinat probleme în care unele variabile care trebuiau inițializate cu variabile ce depindeau de încărcarea din baza de date, nu asteptau să fie încărcate. În acest caz acestea se inițializau cu null și aplicația dadea crash. Ca și soluție am implementat o metodă care obligă variabilele să astepte mai întâi inițializarea din baza de date, acolo unde este necesar.

Capitolul 4

Ghid de utilizare

4.0.1 Pregatirea partii Hardware

Primul pas pentru utilizarea aplicației constă atașarea senzorilor pe plantă ca în imaginea urmatoare. Următorul pas din setup-ul fizic reprezintă conectarea placii Raspberry Pi la o sursă de curent folosind incarcatorul acesteia. O data conectată placa va porni, următorul pas fiind deschiderea ei, apoi pornirea scriptului "script.py" de desktop.



(a) Atasare senzori pe placa



(b) Conectare RaspberryPi si rulare script

Figura 4.1: Hardware Setup Device

4.0.2 Utilizarea partii Software

Aplicația va porni pe pagina de login. Pentru prima utilizare a aplicației va fi nevoie de creearea unui cont.

Se completează câmpurile de email, număr de telefon și o parola, apoi se apasă butonul de register. În cazul în care înregistrarea a fost realizată cu succes, utilizatorul va fi redirecționat către pagina de Login, în caz contrar vor fi afișate mesaje de eroare specifice.

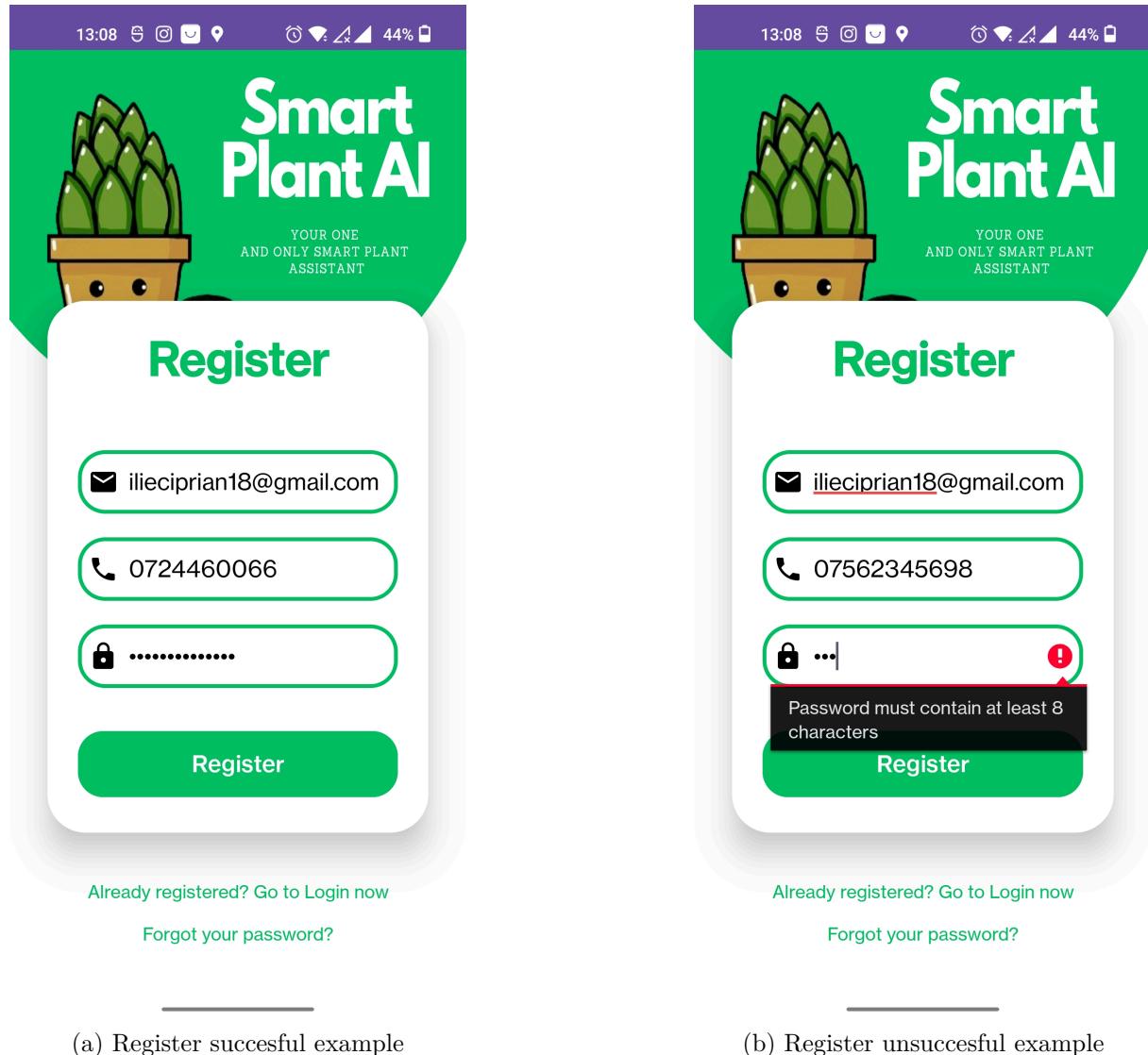


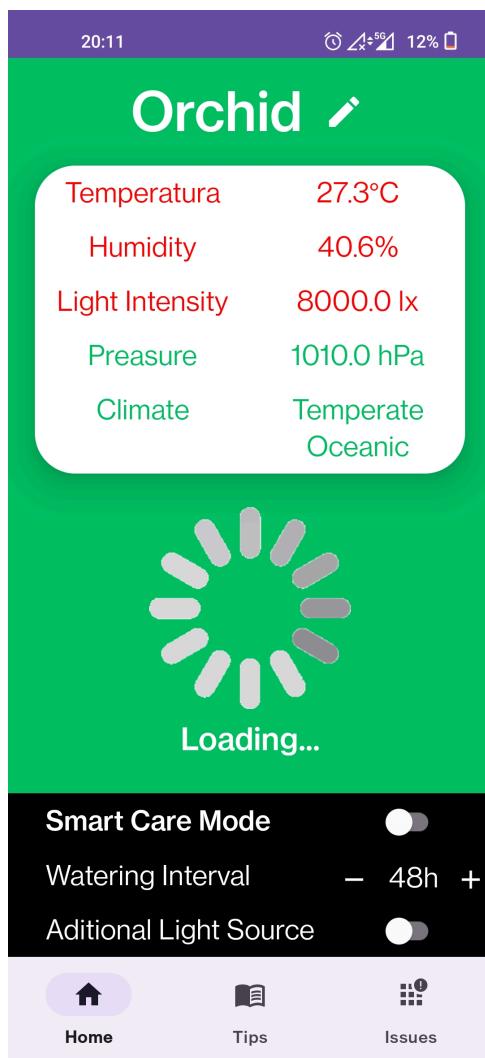
Figura 4.2: Register Page Application

4.0.3 Utilizarea partii Software

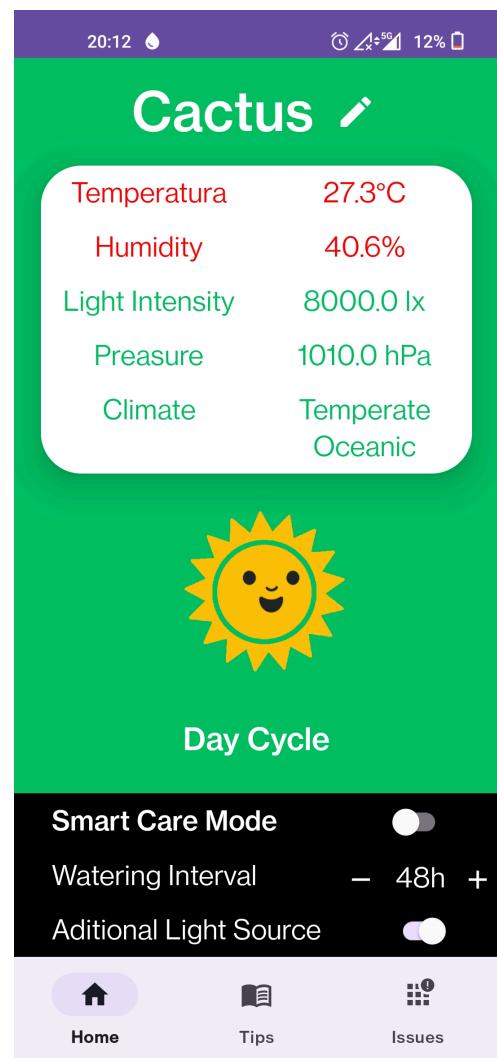
Se introduc datele contului creat în campurile de login

Se navighează către pagina Home din meniu, apoi se apasă pe butonul de editat numele plantei. Se introduce un nume valid de plantă, apoi se așteaptă încarcarea unui regim

de îngrijire de către chatGPT. Odata ce incarcarea a fost realizata cu succes statusul de Loading... se va transforma in Day/Night Cycle.



(a) Load Plant into Android Application



(b) Plant loaded succesfully

Figura 4.3: Load new plant into Application

Aplicația este gata de utilizare.

Capitolul 5

Concluzii

In urma dezvoltării acestei lucrări s-a obținut o aplicație care poate, într-un grad mai mare sau mai mic, ajuta un utilizator să aibă grijă de o plantă. În aplicație au fost implementate cu succes features precum udare automată, activarea unei lumini adiționale, monitorizarea factorilor de mediu în care se află plantă, chat live 24/7, recomandări pe baza plantei selectate, notificări, informații adiționale pentru îngrijirea plantei în climatul în care se află.

Pe partea de utilizare a lui chatGPT în dezvoltarea unei astfel de aplicații, s-a observat faptul că momentan inteligența artificială nu este chiar 100% potrivită pentru astfel de taskuri, dar cu timpul aceasta ar putea să devină din ce în ce mai performantă și să poată duce acest task până la capăt, într-un mod mai bun.

Pentru viitor ca și îmbunătățiri pentru aplicație, ar fi actualizarea în aplicație a unei versiuni de GPT mai performantă care să permită generarea unui mod de îngrijire mai bun. În prezent în funcție de versiunea de GPT folosită, precum am descris și pe parcursul lucrării, mai pot apărea erori și greseli de context.

Pentru partea de aplicație de android, pe viitor se pot obține îmbunătățiri pe partea de customizabilitate, adăugarea unui ecran de setări în care să existe funcții precum modificarea intervalului de notificări, ignorarea anumitor factori de mediu sau existența unui "night mode" pentru GUI, cu scopul obținerii unei aplicații IoT gata de lansare pe piață pe piata.

Pe partea de hardware pot fi făcute îmbunătățiri prin adăugarea mai multor senzori, eventuala adăugare a unei camere pentru un modul de recunoaștere a speciei plantei dintr-o poză precum și trecerea la niște piese hardware mai ieftine deoarece datorită costului lor ridicat această lucrare reprezintă doar un proof of concept, imposibil de reprodus la scară largă din cauza acestei limitări.

Capitolul 6

Bibliografie

- [1] <https://developer.android.com/guide/components/processes-and-threads> [Accesat 01.09.2024]
- [2] <https://firebase.google.com/docs/auth/android/start> documentation [Accesat 10.08.2024]
- [3] <https://developer.android.com/guide> [Accesat 10.08.2024]
- [4] <https://developer.android.com/studio> [Accesat 10.08.2024]
- [5] <https://developer.android.com/develop> [Accesat 09.08.2024]
- [6] https://www.tutorialspoint.com/java_xml/index.htm [Accesat 06.08.2024]
- [7] <https://dev.java/learn/> [Accesat 05.08.2024]
- [8] <https://learn.adafruit.com/adafruit-bmp280-barometric-pressure-plus-temperature-sensor-breakout/circuitpython-test> [24.07.2024]
- [9] lastminuteengineers - <https://lastminuteengineers.com/am2320-temperature-humidity-sensor-arduino-tutorial/> [Accesat 23.07.2024]
- [10] <https://forums.raspberrypi.com/viewtopic.php?t=206154>, forum util cand nu functiona ceva [Accesat 22.07.2024]
- [11] <https://learn.adafruit.com/adafruit-bh1750-ambient-light-sensor/python-circuitpython> [Accesat 22.07.2024]
- [12] <https://learn.adafruit.com/adafruit-stemma-soil-sensor-i2c-capacitive-moisture-sensor/python-circuitpython-test> [Accesat 22.07.2024]
- [13] <https://docs.python.org/3/> [Accesat 21.07.2024]
- [14] <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html> [Accesat 16.07.2024]
- [15] <https://www.tomshardware.com/how-to/set-up-raspberry-pi> [Accesat 15.07.2024]
- [16] <https://www.sciencedirect.com/science/article/pii/S2666154323002831> [Accesat 03.07.2024]
- [17] <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0268-2> [Accesat 03.07.2024]
- [18] https://www.researchgate.net/publication/365450641_Internet_of_things_applications_using_Raspberry-Pi_a_survey [Accesat 03.07.2024]