Final Report: Group 1 Project

Houseplant Care Web App: PlantPal

Team members:

Jia Chi Leow

Rebecca Hodges

Emina Ergul

Oluwaferanmi Olatunji (Rinsola)

Ying Ting Liu

Introduction:

Aim:

Group 1 has created the backend of a personalised houseplant care web app, called PlantPal. With a few houseplant enthusiasts on the team, we understood the importance of providing the correct conditions to plants based on their unique requirements, and how difficult it can be to keep track of each plant's health and care. Thus, our aim was to create a comprehensive and convenient application that aids plant owners by keeping track of their houseplant needs.

Objective:

Group 1's original goal was to create PlantPal with a fully functional backend and frontend, but after some consideration as the project progressed, the team decided to reprioritise and focus only on PlantPal's backend functionality in order to fulfil the project requirements. The team's new objective was to create a backend with five main features: My Plants, My Plant Calendar, Plant Suggestions, My Plant Friend, and a basic user authentication feature.

Report Roadmap:

This report will review the development process of PlantPal and each of its features in-depth (from design and motivations to functionality and the technologies used). Achievements and challenges faced throughout the project will be further discussed also.

Project Motivation:

Background:

User Flow:

1. The user first sees a login page where they could input their email and password in. After successful authentication, they'll gain access to PlantPal's dashboard. If the users do not own an account, they get redirected to the registration page. Here, they enter the necessary details, and they get added to the user's database. Once registration is successful, they proceed to log in.

2. In PlantPal's dashboard, a user can add a plant to their application by inputting a plant name as well as searching for it. The program then shows a result either matching the user's input or if not, finding the closest possible match, showing the search details, i.e. Latin name, ideal temp/light etc. After this, the program asks the user if they want to add it to their My Plants collection. If yes, the user is prompted to enter whether their plant currently has a disease or not. If so, that data is inserted into the DB, if not it is not included. Users can also view a catalogue showing all the houseplants in their current collection, from there they can select a plant to remove.

3. Using the user input and plant data from Rapid API, PlantPal's My Calendar feature generates a personalised 2-in-1 watering and recovery schedule. The calendar shows the user how many days it has been since the plant has last been tended to and gives a recommendation for the next watering date. If a disease is recorded, a detailed treatment plan for each plant is given in the form of a to-do list. Finally, a summary is given with an overview of which plants are overdue watering and provides a schedule for when plants need to be watered next (plant name and date).

4. PlantPal also includes a 'Plant Suggestions' and a 'My Plant Friend' feature. The 'Plant Suggestions' feature recommends users houseplants that cater to their specifications (i.e., air purifying, sunlight requirement, water requirement, etc) as well as allowing the user to search for other plants as well. The 'My Plant Friend' feature contains useful and specialised houseplant YouTube video tutorials. In this feature, the user would be able to see the latest/ current week feature videos (i.e., repotting week's videos) as well as see other video tutorials based on other topics. The user would also be able to see ratings and comments for each video as well as houseplant ads on the side.

Specifications and Design:

Technical:

Feature(s)	Technical Requirements

User Authentication	1. **DB Interaction**: Data is added to the SQL table and fetched from the table for authorization. 2. **bcrypt**: Hashes the user's password to prevent it from being stolen by intruders, also verifies the user password is the same as the one on the DB. 3. **Uuid**: generates a unique Id for each user, used in the manipulation of a user's data. 4. **Login/Register**: function and endpoints to communicate with DB to initiate this process
My Plants	1. **User-API Interaction:** The requests module is used to call RapidAPI House Plants and insert the user's input into the API request as a query parameter. 2. **Personalised Data Input:** A chain of functions to gather user input about the state of their plant to add to the database. 3. **Python-SQL interaction:** Python_sql_connector file to interact with PlantPal database (via mysql. connector) and insert/retrieve/remove data from the 'PlantDetails' table using SQL queries in Python.
My Calendar	1. **Data Retrieval from SQL Database:** Connects to the DB and makes a SQL query to get the user's current plants, user-disclosed diseases and information provided by the API for a plant's watering need. 2. **Datetime Library:** This library was imported to make calculations, turn input into date format, provide days since the last watering and identify the next watering date (using the time delta function). For each of the user's plants, it will provide information on the recommended watering times (calculated by analysing the watering information from the API for key terms and converting it into an integer (for days)). 3. **Summary:** Plants with overdue watering are appended to an 'overdue' list, otherwise watering dates are appended to a dictionary. The user is alerted if their plant is overdue watering or provides the next watering date. If a disease is identified, advice will be displayed. A summary is provided which lists overdue and provides an index for upcoming watering dates.
Plant Suggestions	1. **Data Retrieval from PERENNIAL API**: Upon user input specifying certain criteria (i.e. sunlight requirement, water requirement, type of plant), the function would send a request to the PERENNIAL API. It then filters the returned data to a list of plants fitting the user's specifications. 2. **Air Purifying Plant List Creation**: This list is based on this paper -> **[5]**. It is manually curated + cross-referenced with data available in the PERENNIAL API to ensure validity and accuracy. This filter can be activated based on user requirements and only shows plants which are air purifying. 3. **Recommendation Endpoint**: Only 3 options will be shown, and only essential data is returned (name of plant and image URL of the plant). This makes it easy for users to view and choose from the recommended options.
My Plant Friend	1. **MySQL Table(s) Creation**: 3 tables (videos, ratings & ads) were created for this feature. The 'videos' table contains information about the videos (i.e. YouTube URL, what topic it belongs to, etc) and has these columns: video_id, topic, title, description, URL, week & date. The 'ratings' table should have 3 columns (video_id, rating & comment) and contains information on the users' ratings and comments whilst the 'ads' table has 5 columns (id, title, description, image URL & ad URL).

	2. **Endpoints Creation**: 7 endpoints were created which satisfies what My Plant Friend does. 1st endpoint makes a get request which executes a select query to fetch the current week's feature videos' data from the 'videos' table in the DB, storing them in a list (JSON format). The 2nd endpoint fetches all topic names from the 'videos' table using a select distinct query, the 3rd endpoint fetches a particular topic's videos' data, and the 4th endpoint fetches all the videos and sorts them (newest to oldest). The 5th endpoint fetches the ads' information from the 'ads' table. The 6th endpoint does a POST request instead and prompts the user for a rating and comment for a particular video, once done, the 'ratings' table on MySQL should update. The last endpoint fetches the ratings and comments for the videos from the 'ratings' table. 3. **MySQL DB Connection in Python**: The same method was used as the other features (via MySQL connector).

Note: Flask app framework was used and ran for these features: User Authentication, Plant Suggestions, and My Plant Friend.

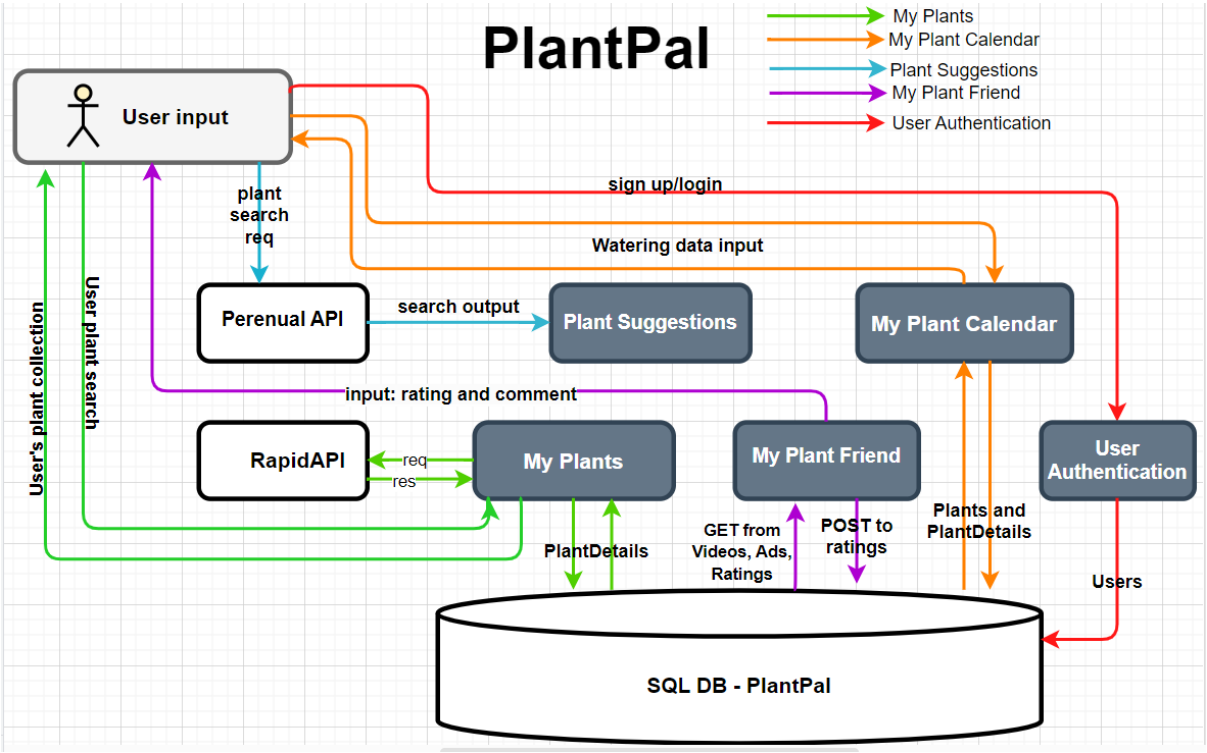
Non-Technical:

Feature(s)	Non-Technical Requirements
User Authentication	1. **Email & Password:** Users need their email and password to access their PlantPal dashboard. 2. **User-friendly Design:** Interface to take in user's data to initiate login and registration
My Plants	1. **Flexible Plant Search:** Users can search for a plant species by common or Latin name, if the plant is not found, the closest matching plant will be returned. 2. **User Friendly:** Allowing the user to view their current plant collection, and from there decide whether to alter it. Users have the option to add a disease to their data only if their plant currently has any.
My Calendar	1. **User-Friendly Input:** An overview of how often each plant needs watering is provided. Users will be able to easily record when they've watered their plants and receive a tailored schedule. If any disease has been reported, information will be provided to display the steps they can take to treat it. 2. **Takeaway Summary:** Provides a simple summary that alerts users to take action and creates a schedule for them.
Plant Suggestions	1. **User-Friendly Design**: Simplifies the decision-making process by offering up to three plant suggestions based on the user's criteria. 2. **Visual Aid**: Each plant recommendation is accompanied by an image URL, enhancing user understanding and selection ease. 3. **Reliable Research**: The air-purifying plant list is curated from a trusted research paper, ensuring scientifically backed and reliable plant recommendations.

My Plant Friend	1. **Video Tutorials:** Video tutorials are a good visual way to teach PlantPal's users the 'right' way to care for houseplants. Video contents are more popular and preferable for the current society as well. 2. **User-Friendly Video Structure**: The feature was structured in a way that allows the user to see the current week's feature videos as well as pick videos based on the topic. All videos are separated by 8 topics for now (Repotting, Houseplant Care Tips, Watering, Propagation, Houseplant Deco Tips, Pest Care, Terrariums & Greenhouses, Natural Lighting & Growth Lights). 3. **Users' Feedback & Trust**: The users can also view the ratings and comments for any videos shared (if there are any) + post their own ratings and comments. 4. **Visual Ads**: This feature would also feature visual ads on the side which could benefit users (i.e. Ad: B&Q having a clearance sale for gardening products!).

Design & Architecture:

Architecture Diagram of PlantPal:



Implementation & Execution:

Development Approach:

1. **Agile Development:** The group employed continuous development and integration methods, including daily stand-up meetings, code reviews and weekly discussions (ensuring immediate feedback and adjustments).

2. **Modularity:** The group utilised Flask's Blueprint feature to separate different components within the app, enhancing readability and fostering a modular development style.

3. **RESTful Principles**: The group used REST (Representational State Transfer), such as GET, POST and DELETE.

4. **SQL**: The group executed SQL queries directly, without utilizing an Object-Relational Mapping (ORM), to prioritize performance during the relevant actions.

Team Member Roles:

Team Member:	Roles:
Ying Ting Liu	Backend for 'Plant Suggestions' feature; Combining code into one app and making sure it works.
Rebecca Hodges	Backend for 'My Plant Calendar' feature.
Emina Ergul	Create DB; Backend for 'My Plants' feature.
Rinsola	Create DB; Backend for User-Authentication feature; Frontend for PlantPal (working on the user interface of the app).
Jia Chi Leow	Backend for 'My Plant Friend' feature; Performing Scrum Master-like responsibilities (i.e. hosting and leading meetings, sending reminders, communicating between team and instructor, writing up detailed meeting minutes + meeting agenda(s), making sure that everyone knows their responsibilities + what to do moving forward.)

Tools and Libraries:

- **Framework**: Flask (back-end), React (front-end)
- **Languages**: Typescript, Python, SQL
- **Project Management and Communication Tools:** Jira (for project management and issue tracking), Slack (for team communication), Zoom (for remote meetings), GeekBot(a daily stand-up app within Slack, to emulate agile stand-up meetings.)

- **Libraries**:
  - ***Backend*** - datetime, unittest, mock, bcrypt, python-dotenv, json, mysql.connector, flask, uuid, os, requests, Werkzeug, sys
  - ***Frontend*** - react-modal, moment, axios, react-toastify, tailwindcss, sass

Implementation process:

Achievements:

- **Group**: Successful application of agile development, API integration, DB integration
- **Ying-Ting Liu**: User-friendly recommendation design, reliable source and recommendations, effective error handling, effective testing.
- **Jia Chi Leow**: Managed to achieve all of the objectives for my feature ('My Plant Friend'); Incorporated and shared video tutorials in a structured way; Allowed ways for users to not just see but also rate and comment on videos themselves; Created tables with data from popular sources; effective error handling (with exception classes); performed testing using unittests and Postman.
- **Rebecca Hodges:** Took the lead on the 'My Calendar' feature (designing and implementing its backend structure).
- **Emina Ergul:** Lead the 'My Plants' feature and successfully implemented a Python MySQL interaction program where users can search for specific plant species, or get the closest match returned, and interact with their plant collection (add/remove/view).

Challenges:

- Ensuring that connections to the API and database are performed consistently for all team members.
- Limitation of the APIs: for some of our initial aims (i.e., incorporating fertilisation dates) this wasn't feasible as the API lacked some data initially thought accessible. Other challenges like inconsistent wording meant more time needed to be spent qualitatively analysing the API's return.
- Working with one less team member (when compared to other groups).
- Communication and Commitment - Team members had various non-CFG-related commitments (i.e., work, studying) so this needed to be worked around to ensure members' personal circumstances were considered. Such as providing regular updates (primarily via meeting minutes) of discussions and progress for members who were unable to attend meetings.
- In the initial project plan, some of the formulated ideas were overly ambitious for the time given, experience, tools, and knowledge available. Project changes and compromises had to be made as a result.
- PlantPal is not asking as many personalised questions as intended, making it a less personalised web app than intended.
- Mocking user input for unit testing 'My Plants' proved to be a challenge since the program flowed continuously based on user input. Multiple attempts were made to mock this, but the outcome was unsuccessful.

- Relative File Imports: Initially, importing files from another directory posed a significant challenge, but after extensive research, the group managed to resolve the issue.
- Decision making : While the group had some differences in opinion on a few matters, everyone reached a substantial amount of agreement, which ultimately contributed to the success of the application.

Project Changes:

- Program no longer includes common vegetable plants that people grow as houseplants (i.e. tomato, spring onion, etc).
- PlantPal no longer include info on soil type, and geographical location (i.e., not taking into account location, sunlight, weather, seasons, and changing the user's schedule automatically) as no free APIs provided this info.
- My Plant Calendar no longer includes fertilising and seasonal watering schedule changes.
- PlantPal is not currently available in other languages.
- PlantPal can't send reminders/ notifications via email to their users just yet.
- The Plant Identification feature originally incorporate the use of AI but after researching AI tech that we could use to identify plants through AI image identification, we found that there were limited options, and all options involved a subscription and cost. The team agreed that this was impractical, hence removing it from the group's project.
- Users would not have an option to manually input any plant updates for now.
- Users cannot give their plants pet names (i.e., Bob the cactus) for now.
- Group not using JIRA as a project code management tool.

Agile Development:

Several stages of agile development were employed in the creation of PlantPal:

1. **Ideation**: Members collaborated in the initial stages to creatively envision PlantPal's functionality, features, and the technologies to be used.
2. **Communication:** Members cooperated and communicated well throughout the development of PlantPal. Real-time meetings were regularly held on Google Meets or Zoom. Slack served as the main method of communication. Extensions such as GeekBot were used to allow a 'daily stand-up' that would be completed around the same time each day where members shared what they worked on yesterday, their plan for the day and any challenges.
3. **Collaboration and continuous improvement** - Members provided assistance if they encountered any difficulties and collaborated on more complex tasks (such as setting up the database and connecting it). Regular communication and testing ensured continuous improvement and clear advancement of the software's development.
4. **Responding to change** - If changes were made, these were communicated, and appropriate action was taken if this impacted other aspects of the software. For example, the decision to not incorporate a front-end. If a team member had external constraints other team members would adapt and be flexible.

5. **Short 'sprint' like activities** - Consist of a sprint iteration per week, which includes daily stand-up meetings, weekly reflection, and progress tracking, as well as backlog management. The workload and release plan were flexibly adjusted based on the agile methodology. Deadlines were set to ensure progress. Such as a 50% code completion deadline and the date for final code compilation. Mini 'sprint reviews' were conducted to check on progress.
6. **Scrum Team Roles**: Each team member takes one or more specific scrum roles, including the product owner, scrum master, developer, and tester roles, to collaborate effectively. Since the teams consist of only 5 members, there was some overlap in roles.

Overall incorporating agile principles facilitated communication, collaboration, continuous development, and adaptability throughout the development of PlantPal.

Testing & Evaluation:

Testing Strategy:

Feature(s)	Testing Strategy
User Authentication	Manual and unit tests - Simulated when a user logs in, registers, the DB connection with mock.patch a module in unittest.
My Plants	Manual testing to check that API request was retrieving data based on user input as expected, and that SQL queries in Python were inserting/deleting/displaying entries as expected. Unit testing was implemented for functions that did not require mocking. Mocking user input was tried but unsuccessful but manually tested the program flow based on different user input.
My Plant Calendar	Manual and unit testing for functions and class methods. Manual testing included checking all connections with the DB and ensuring output displayed correctly in the terminal and as expected. Manually checked the function that detected keywords in the API's watering description. Some test cases used mocking for simulating input (such as today's date). The majority of tests used assert equal statements - primarily to ensure datetime and time delta functions performed correctly.
Plant Suggestions	Development stage focus on feature and method-based testing, using unit testing and endpoint Testing. After Integrated the project testing endpoint and checking the test coverage.
My Plant Friend	Manual, unit testing as well as using Postman testing for each endpoint. Manual testing includes running the Flask app and making sure that the endpoints are fetching/ posting the right data as intended (via posting URLs on a web browser/ Postman). Unit tests were used in combination with exception classes to test if the app would act and react as intended.
Final PlantPal App	Successfully tested project endpoints and functions in the final stage and successfully pass all the test cases for the whole project.

System Limitations:

1. **Dependency on External APIs**: Our app relies on the PERENNIAL and Rapid APIs for data. Thus, any limitations, inaccuracies, or availability issues with these APIs could directly impact our app's functionality.
2. **Data Coverage**: While our app endeavours to provide comprehensive plant data, it may not cover certain types of plants or specific plant-related data.
3. **Database Connection**: The performance of our application is contingent on the speed and stability of the database connection.

Conclusion:

To conclude, the result of PlantPal can be considered a success, because the main functionalities (My Plants, My Plant Calendar, etc) that we aimed to achieve from the start were successfully implemented.

There were however some changes and challenges we faced. In the early stages of the design process, some practicalities were overlooked, causing us to remove some of the planned functionalities we had. For example, there were limited API options for the plant AI identification feature, which were also costly. Although this feature would have enhanced our app's ability to dynamically help users identify their plants, we understood the tech needed to build this feature was specific.

In addition, time management was also overlooked when initially deciding to build a frontend. As we began developing our features and reflecting on the project requirements, the team mutually recognised that the backend was our priority, and the integration of a frontend was to be an addition if we had time left over to do so, which we did not.

However, the team kept in regular contact, with weekly scheduled meetings, and many spontaneous meetings when needed, whilst also posting updates on Slack. For this reason, we were able to understand how one of our features might affect the other, and as a result, faced no major issues when our code was combined. This communication was especially important when designing the PlantPal database.

Overall, each team member contributed significant effort to each of their agreed roles, and the outcome is the PlantPal backend that provides the most beneficial functionalities that we originally intended to be useful for plant owners. Moving forward, we could expand on our app by integrating the frontend like initially planned, adding a filter system to the API fetch result, and even a social feature that allows users to share posts about their plant collections, similar to a forum. PlantPal could also have user roles such as admin and customer, where an admin is an internal user and can view insights on most active users, set plant of the week/month (e.g. Most popular plant among users or most suited plant for the season), and has permission to restrict/remove users. The customer would be general external users who use the standard features of PlantPal, like Plants Suggestions, My plants, etc.

Appendix:

Why make a plant care program?

The initial motivation behind PlantPal includes several team members' own struggles with their own house plants. The range of houseplants across each team member varies from 0 to 30 houseplants. Many studies and research

For example, a study was conducted by launching a survey to 2000 millennials (between the ages of 25 to 39) [1]. The aim of the study was to understand better the relationship and experience between the general public and houseplants [1].

The **results and statistics** from this study showed that: (**Points 1 to 6**: [1])

1. **7 in 10** millennials call themselves plant parents.
2. **81%** agreed that plants have positively affected their mental and physical health.
3. **40%** of millennials plan to buy more houseplants this year.
4. **48%** are not as confident in keeping their plants alive.
5. **47%** don't own plants as they don't know how to take care of them.
6. **67%** called themselves plant murderers.

Other statistics include:

a. There is a growing interest in houseplant apps and accessories [2].

b. House plants are the fastest-growing part of the UK plant market [3].

c. A person in the UK spends over **£300** (on average) over a year on houseplants (with **Gen Z** spending a total of **£414.84** annually) [3].

From these statistics, It is clear that millennials and GenZ plant owners felt that having plants had an overall positive effect on their mental and physical health [1]. Thus, some of them would be inclined to buy more houseplants but would also love to learn more about growing houseplants, as opposed to killing them [1][2][3]. So, a houseplant care web app, such as PlantPal could greatly benefit houseplants owners (especially since there is a growing demand for houseplant apps right now [2]). A personalised web app like PlantPal could allow houseplant owners to feel less anxious and encourage non-houseplant owners to start growing some.

Other than that, a study also highlighted the various challenges millennials had while growing houseplants: (**all points below:** [1])

- 50% – The amount of sunlight needed for each houseplant.
- 46% – The amount of water needed for each houseplant.
- 43% – Whether the houseplant is an indoor or outdoor one.
- 48% – Keeping their houseplants alive.

Above that, another study in the UK showed that factors, such as overwatering, insufficient sunlight, incorrect potting/repotting, etc, are the few main reasons that people kill their plants [3].

These statistics, again, show the need for a simple, user-friendly and personalised web app like PlantPal which could better help plant owners understand and grow their houseplants.

How is PlantPal different from other plant care apps in the market?

Currently, in the market, there are many plant care apps which offer different types of services/ help. Most are relatively user-friendly and include plant identification, plant APIs, database, reminders, recommendations, etc.…[4].

One of the most popular plant care apps at the moment, is the Planta app. It is also the app that is most similar to PlantPal, in that it reminds its users when to water their plants [4].

How is PlantPal different from Planta?

1. On top of generating a watering schedule for each houseplant. The schedule would also include treatment plans (if a particular houseplant is struggling), making it a 2-in-1 plant care schedule.
2. When tapping into a houseplant on the catalogue page, the user would then enter a new page which shows general information regarding that particular plant (i.e., how much sunlight it requires).
3. The web app will also have a My Plant Friend feature which posts videos weekly depending on the current week's feature topic (i.e., Repotting week: see video tutorials on knowing the right soil type, how to repot, what pot size to choose, etc.).

References:

[1]

Fritz, K. (2020) 'SURVEY: DECORATING WITH HOUSEPLANTS', *Articulate*. 8 April. Available at: https://www.article.com/blog/survey-decorating-with-houseplants/ (Accessed: 12 May 2023).

[2]

'The Most Surprising Houseplant Industry Statistics And Trends in 2023' (2023) *GITNUX*, 13 April. Available at: https://blog.gitnux.com/houseplant-industry-statistics/ (Accessed: 12 May 2023).

[3]

Harris, C. (2022) *25 latest Houseplant Statistics & Facts (2023 UK), DIY Garden*. Edited by S. Franks. Available at: https://diygarden.co.uk/statistics/houseplant-statistics (Accessed: 12 May 2023).

[4]

'Best Free Plant Care App in 2023' (2023) *GITNUX*, 13 March. Available at: https://blog.gitnux.com/best-free-plant-care-app/ (Accessed: 12 May 2023).

[5]

BC Wolverton; WL Douglas; K Bounds (September 1989). Interior landscape plants for indoor air pollution abatement (Report). NASA. NASA-TM-101766. Available at: https://ntrs.nasa.gov/citations/19930073077 (Accessed: 27 May 2023).