Coursework – Group 25

Prince Alexander John	2427675	PXX276@student.bham.ac.uk
Dami Olugbodi	2409165	dxo265@student.bham.ac.uk
Bhuvan Praveen Kumar	2408367	bxp267@student.bham.ac.uk
Amar Purewal	2480763	Axp1144@student.bham.ac.uk
Muhammad Ali Shah	2403348	mas248@student.bham.ac.uk
Zakaria Hussein	2344619	zxh023@student.bham.ac.uk
Mohammed Adnan Hoq	2437604	Mxh1223@student.bham.ac.uk
Awais Rafique	2055367	axr593@student.bham.ac.uk

Contents

Section A – Requirements Engineering (Unit 2)

A1: Proposed System

Problem to solve

Trees play a crucial role in our lives, from providing oxygen to taking CO_2 from our environment which is one of the biggest contributors to climate change. Having interviewed students and community members in Birmingham, it highlighted to us that navigating through environmental initiatives to take part in planting trees and finding effective ways for individuals and corporations to actively contribute to sustainable practice remains a challenge.

The absence of a cohesive platform that allows individuals to not only get involved in planting new trees but to also encourage individuals to reduce their carbon footprint leaves stakeholders without a unified space to foster sustainable practices. Additionally, the connection between environmental sustainability and personal connection feels elusive.

Solution Overview

To achieve our aims, we will develop an app called TreeConnect. An innovative solution designed to bridge this gap. Our app will be a comprehensive platform which will allow users (admin/support workers and customers) to sponsor the planting of a tree with the press of a button. Once a user orders the planting of a tree, we will relay their order to our partnered organisations who will plant the tree, bringing together individuals and organisations. Also, our app will form a bridge between environmental stewardship and personal connection by allowing users to sponsor trees to commemorate somebody's life, celebrate a birth or give a lasting gift to your colleagues, friends or family.

Our app not only makes it easier to sponsor trees, but it also offers detailed information about the environmental impact linked to each tree. This will cause users to develop a strong bond with their environmental acts by learning about the ecological advantages of their efforts. The app will also have a rewards feature. Users will be rewarded for eco-friendly behaviors like walking, cycling (which the app will detect and measure using Google Maps) or reducing their energy usage in their homes (which the app will detect using a smart meter). Users can then sponsor by redeeming their points or by directly paying money. This not only encourages sustainable behavior but also rewards people for their beneficial contributions to the environment. (Note that the app will only display trees that are available for sponsorship).

Furthermore, the sponsorship process is streamlined by our integration of a secure payment source, guaranteeing that stakeholder financial contributions are handled effectively and safely.

Lastly, the app has a data analytics section that uses indicators related to environmental effects and user interaction to offer tailored recommendations to users. These analytics feature analyses user behaviour, preferences, and the effects of their activities using machine learning algorithms (the machine learning algorithm for data analytics responsible for analysing user behaviour will be outsourced and within the system will be called by the server).

Scope

Initially, we are limiting the scope of the project to Birmingham. Upon demonstrating how the app improves the quality of life in Birmingham and contributes towards reducing the environmental impact of climate change. We will use this as proof of concept and demonstrate the effectiveness for broader adoption across multiple cities within the UK.

Assumptions

- Users have a genuine interest in tree sponsorship and are willing to commit time and effort to support tree-related initiatives.
- Users are willing and able to make financial contributions for tree sponsorship.
- Users have access to internet and are comfortable using mobile applications.
- There are available trees for sponsorship in the regions the software covers.
- Environmental impact metrics and calculations are based on reliable scientific data.
- Communities and local organisations are open to collaboration and partnerships.
- Sponsorship fees will provide a sustainable source of funding for tree-related initiatives.
- The user devices have in built **Google Maps** which allow location tracking thus verifying distance cycled and steps walked.
- **Payment Gateways**, external providers are responsible for authentication of payment details.
- Payments providers, Maps, Database and Analytics algorithm are all external and will all be outsourced.
- Server has a built-in clock which is synchronised with the local time

A2: Functional and Non-Functional Requirements

Functional Requirements

Key: M = Must, S = Should, C = Could, W = Would

ID	Requirement	Priority
R1	**Login Functionality**	
R1.1	The system must allow users to create accounts/log in to their account by accepting their email address and password.	M
R1.2	The system must allow the user to enter their personal details (name, date of birth, phone number) when creating an account.	M
R1.3	The system must prompt the user to meet the password requirements when creating an account.	M
R1.4	The system must hash password at client end before sending it to the database.	M
R1.5	The system must validate the entered email address during registration (sign up) by sending a 4-digit verification code to their email address.	M
R1.6	The system must be able to verify and accept the 4-digit code entered by the user.	M
R1.7	The system must allow users to reset their password by sending a password reset link to the email the user inputs if the user has forgotten their password.	M
R1.8	The system must check if the email address is already associated with an account when creating a new account.	M
R1.9	When a login from a new device is detected, they system should alert the account holder through their preferred method of contact.	S
R2	**Dashboard Functionality**	
R2.1	The system must allow access to account functionality for user management and points functionality through the dashboard interface.	M
R2.2	The system should use data analytics to provide users with sponsorship recommendations that are in line with their interests based on behavioural data.	S
R2.3	The system should display real-time data on the total number of trees sponsored by the user and by all users globally through our system.	S
R2.4	The system should display real-time data on the total number of points accumulated by the user.	S
R2.5	The system would display information about the climate change and how their efforts are important in combating this.	W
R3	**Account Functionality**	
R3.1	The system must allow users to turn on/off location tracking.	M
R3.2	The system must allow users to log out of their account.	M
R3.3	The system must allow users to delete their account.	M
R3.4	The system must allow users to manage their notification preference.	M
R3.5	The system must allow users to change their preferred method of contact.	M
R3.6	The system must allow user to change/update their email address and must send a 4-digit verification code to the entered email for validation.	M
R3.7	The system should allow users to update their personal details (name, date of birth, phone number).	S
R3.8	The system should allow users to enter/update their payment details.	S
R3.9	The system should allow users to control their privacy settings (collected data i.e. data analytics).	S
R3.10	The system could allow users to choose a preferred theme (dark/light mode).	C
R3.11	The system could allow users to set up additional security measures like two-factor authentication.	C
R4	**Tree Functionality**	

R4.1	The system must display all available trees for sponsorship and their price in GBP and points.	M
R4.2	The system must deduct points from the user's account if points were used to sponsor a tree.	M
R4.3	The system must provide a range of payment services (Apple pay, debit/credit card etc.) to allow the user to pay for sponsoring a tree.	M
R4.4	The system must communicate tree sponsorship orders to the organisation(s) responsible for planting the sponsored trees.	M
R4.5	The system must send immediate confirmation for successful orders to the user's preferred contact method.	M
R4.6	The system should continuously track the total number of trees sponsored by individual users and by all users and update the count respectively in real-time to provide accurate data.	S
R4.7	The system should provide information about different types of trees that are available for sponsorship and inform users about their environmental impact.	S
R4.8	The system could allow users to add a personal touch (such as request a plaque with a name of a beloved one) to sponsored trees at checkout.	C
R4.9	The system could allow users to browse trees for sponsorship through filtering species, cost or location.	C
R4.10	The system could offer a search feature for finding trees or sponsorship details.	C
R5	**Points Functionality**	
R5.1	The system must display how many points each activity is worth.	M
R5.2	The system must contain an option called "How to" which explains how to earn points.	M
R5.3	The system must be able to use GPS location to calculate steps taken or distance cycled and use this information to award points.	M
R5.4	The system must be able to use the day energy usage and average daily energy usage readings from the smart meter to calculate the number of points to award the user with.	M
R5.5	The system must accurately keep track of accumulated points based on the user activity.	M
R5.6	The system must send a notification when the user earns points by either cycling, walking or by being energy efficient.	M
R5.7	The system must display real-time updates on points earned and a chart of what activities those points came from and a break of the user's carbon offset.	M
R5.8	The system would display a map of the cycled/walked journey with the number of points earned after the user clicks on the notification sent.	W
R6	**Support Functionality**	
R6.1	The system should lock the admin account after three failed login attempts. To access their account, they should contact the super admin to gain access again.	S
R6.2	The system would require admin and support account creation to be done only through an invite link from an existing admin account.	W
R6.3	The system would allow admin privileges to be expanded/limited or revoked by other admins.	W
R6.4	The system would allow multiple admin and support accounts to be created but only one super admin must exist.	W
R6.5	The system would allow users to contact support accounts so they can get help with account access, sponsorships or close their accounts if there is a security breach.	W

Non-Functional Requirements

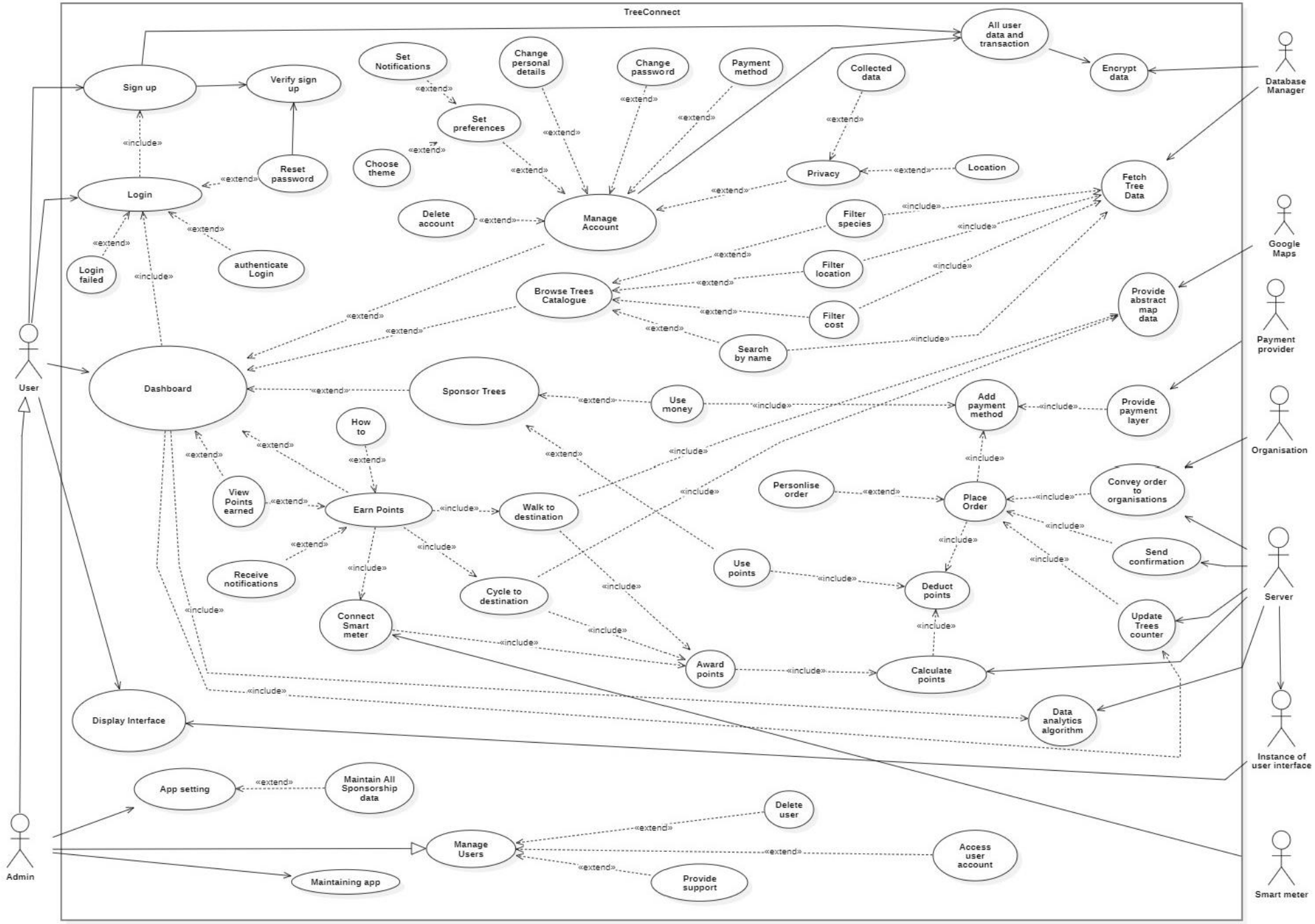
ID	Requirement	Priority	Category
N1	**General**		
N1.1	The system must be able to run on iOS 10 and above and Android 10 and above.	M	Performance, Reliability, Usability
N1.2	Payment gateways and financial transactions must work without major technical issues.	M	Reliability
N1.3	User's name, email, points, payment details and trees sponsored must be added to the system database**.**	M	Reliability
N1.4	The system must allow separate features to function properly and interact with one another accordingly and run smoothly.	M	Performance
N1.5	The system must ensure that each page of the application is scaled correctly according to the device dimensions.	M	Usability
N1.6	Outside of maintenance hours, the system must always function effectively.	M	Reliability
N1.7	Mean time to restore the system following a systems failure must be no longer than 10 minutes.	M	Reliability
N1.8	They system must encrypt and store all data collected in a secure and protected database.	M	Reliability, Security
N1.9	The system must be active 24/7.	M	Reliability
N1.10	The system must inform users how their data will be utilised and allow them to give/revoke permission at any time they wish.	M	Ethical and Legal
N1.11	The system must follow GDPR regulations regarding the user's information.	M	Ethical and Legal
N1.12	The system must defend against network attacks such as DoS, DDoS and ARP.	M	Security
N1.13	The system must defend against software hacks such as SQL injection, brute-force cracking, phishing and XSS.	M	Security
N1. 14	Maintenance and support tasks should be efficiently managed as the user base grows.	S	Reliability, Scalability
N1.15	The app should take under 3 seconds to open.	S	Efficiency, performance
N1.16	The system should not take longer than a second to respond after a button is being pressed.	S	Performance
N1.17	The system should allow at least 15000 people to utilise the app concurrently without it crashing.	S	Reliability
N1.18	At maximum usage, the system should enable a minimum of 20000 people to concurrently utilise the system without crashing.	S	Scalability, Reliability
N1.19	When there is maintenance scheduled, they system should notify all users 2-weeks in advance through their preferred method of contact.	S	Reliability
N1.20	The system should have a user-friendly interface.	S	Usability
N1.21	The maximum downtime allowed when carrying out updates and maintenance should be no more than 2 hours.	S	Security, Reliability
N1.22	The system should block a user for 24 hours if they input the wrong password 5 times when trying to log in.	S	Security
N1.23	Every 6 months the system could create a backup of all the user data that is has stored in the database to ensure in the event of data loss, user data is not lost permanently.	C	Reliability
N1.24	At maximum usage the system could only allow a maximum of 0.5 sec delay to be added to loading time.	C	Scalability, Reliability

N2	**Login functionality**		
N2.1	The system must authenticate users' credentials (email and password) to grant access to the platform.	M	Security
N2.2	When resetting the password, the system must censor the first 4 characters of the email address with * for security purposes.	M	Security
N2.3	The system must require a password to be a minimum of 8 characters long and contain at least 1 upper case, 1 lower case and 1 special character.	M	Security
N2.4	Each email must only be associated with one account.	M	Security
N2.5	The system must ensure that the name of the user only contain letters.	M	Usability, Reliability
N2.6	The system must ensure that the date of birth is in the DD-MM-YYYY format.	M	Usability, Reliability
N2.7	The system must ensure that the phone number only consists of numbers.	M	Usability, Reliability
N2.8	The system should only allow users to enter the next filed in account creation when previous field is valid.	S	Security, Usability
N3	**Account Functionality**		
N3.1	The system must ensure that privacy settings management such as location tracking and analytical data collected must be easily accessible to users.	M	Security, Ethical and Legal
N3.2	If a payment detail is updated, the system must remove completely any previous payment detail saved.	M	Security
N3.3	When the user updates their method of contact, the system must implement the change effect immediately.	M	Reliability, Performance
N3.4	The system must support an error free notification preference choice to ensure the user in not disturbed or annoyed.	M	Performance
N3.5	The system must adhere to data privacy and data retention policies upon account deletion.	M	Security
N3.6	The system should ensure process of updating personal information should be straightforward and intuitive. Name text-fields cannot be left empty and must contain only letters.	S	Usability, Reliability
N3.7	The system should only log out the user when the logout button is pressed and not when the app is closed.	S	Usability
N3.8	The system could support text-to-speech and increased font size to allow accessibility to users with disability.	C	Usability
N4	**Tree Functionality**		
N4.1	The system must keep up to date data of sponsorship details and display this in real-time to users so they can make a well-informed choice.	M	Usability
N4.2	The tree data in the database must be accurate and always up to date.	M	Reliability
N4.3	The system must be able to handle multiple tree sponsorship orders from multiple users at the same time without performance degradation or errors.	M	Reliability, Performance
N4.4	The system should provide clear, concise and accurate data about each species of trees available for sponsorship.	S	Usability
N4.5	The system should comply with relevant financial regulations and laws when processing payments. For example • Anti-money laundering. • Risk management. • Data protection.	S	Security

N4.6	The system should display each tree in a visually appealing tile with cost in both points and money displayed on bottom right corner	S	Usability
N5	**Points functionality**		
N5.1	The system must be able to connect to a smart meter.	M	Usability
N5.2	The system must reward users 10 points for each km cycled and 0.1 points for each step taken.	M	Usability
N5.3	The system must collect smart meter reading data at 00:00 am and reward 0.5 points for each unit of energy they are below their daily average use.	M	Usability
N5.4	The system must have a standard conversion rate of 10000 points = £10 and enable users to see how much money their points translate to.	M	Usability
N6	**Support Functionality**		
N6.1	Different users (such as admins and customers) must have different access privileges.	M	Security
N6.2	The system must keep a log of admin login sessions for accountability.	M	Security
N6.3	The system should generate a unique ID for each admin account upon account creation and only allow to be signed in with this ID and password. One unique ID per account.	S	Security
N6.4	The system should require a different password requirement for admins and users. For admins, 10-characters minimum, 2 upper cases, a special character and 2 numbers.	S	Security
N6.5	The system should add an expiration period (30 mins) to generated invite links and limit each link to only one admin account creation.	S	Security
N7	**External Requirements**		
N7.1	The system must comply with the General Data Protection Regulation (GDPR) and the Data Protection Act 2018 in the UK for data protection and privacy.	M	Security, Ethical and Legal

Section B – Software Specification, Analysis and Design with UML

B1: Use Case Diagram



B2/B3: Use Case Specifications/Scenarios

Use case 1: Sponsoring a tree.

Introduction	The system will allow the user to sponsor a tree by either spending their points or using their own money.
Actors	• User • Organizations • Payment provider
Preconditions	• The user must be logged in with a registered account. • The user has enough points to sponsor a tree. • The user has enough GBP to sponsor a tree.
Flow of Events	1. (Start) The user starts the app and navigates to the sponsorship page 2. The user is met with a "Tree catalogue" specifying the trees they can choose to sponsor and plant, a short description and the cost in points and GBP. 3. The system (the tree catalogue) would have options to filter through location, cost and species filtering. 4. The user chooses the specified tree and presses the "sponsor" button. 5. The user is given the option of choosing either to pay using their points or to pay using money. 6. The user chooses to pay using money; they are prompted to input a payment method where they will be asked to input their card details. 7. The system provides a range of payment services (Apple pay, debit/credit card etc.). 8. The system gives the option to the user to input a personalised message. 9. The system asks them to confirm their payment. 10. The system would wait until a confirmation has been sent by the user's bank. 11. The system would inform them the transaction has gone through. 12. The server would update the connected organization about all the details of the tree plantation order immediately after confirmation is received by payment provider. 13. The server sends a digital receipt to the registered mode of contact and would display the sponsored tree on the user interface directly after sponsoring. 14. The system redirects the user to the dashboard. (End)
Alternate Flow	1. The user decides to use their points instead of using money to sponsor a tree, the server will deduct their points. 2. The server sends a digital receipt to the registered mode of contact and would display the sponsored tree on the user interface directly after sponsoring. 3. The server would update the connected organization about all the details of the tree plantation order immediately after confirmation is received by payment provider. 4. The system redirects the user to the dashboard. (End)
Postconditions	• The server would automatically update the tree counter once the order has been placed. • The server would automatically update the tree data in the database once an order has been placed. • The server would automatically update the number of points the user has once the order has been placed. • The server updates the real time data of all trees sponsored by users.

Scenarios	**Scenario 1:** Sarah opens the app and navigates to the Dashboard. Intrigued by the "Sponsor" tab, she clicks on it. The Tree catalogue appears, showcasing various tree options. Being passionate about preserving local ecosystems, she decides to filter the options based on species and location. She chooses the "Red Maple" as it's well-suited to Birmingham's climate. After reading the description and noting the cost (3000 points or 20 pounds), she selects the tree. She decides to use her points for this sponsorship and confirms her choice. The app deducts 3000 points from her account, sends a digital receipt to her email, and displays the sponsored tree on her Dashboard. The server automatically updates the connected organization with details of Sarah's tree plantation order, including the chosen species and location. **Scenario 2:** John, a representative from a local company, opens the app and explores the Dashboard. Intrigued by the idea of corporate social responsibility, he clicks on the "Sponsor" tab. After reviewing the Tree catalogue, he filters options based on cost and decides to sponsor a "Southern Magnolia" for 15 pounds. He opts to pay with money, inputs his card details, and confirms the payment. The system waits for confirmation from the bank, and once received, it informs John that the transaction has gone through. The app sends a digital receipt to John's registered email and displays the sponsored tree on his Dashboard. Simultaneously, the server updates the connected organization with all the details of the tree plantation order, including the chosen species, location, and the corporate sponsorship.

Use case 2: Earning points by connecting to a smart meter. (Once the smart meter is connected to the user's account, there is no need for the user to have the app open to earn points using this method).

Introduction	The system allows users to earn points by saving energy at home. The system can check how much energy the user saved by using the user's smart meter which they can connect to the system. The check is done daily meaning the user may be able to earn points daily.
Actors	Server (system), Database manager, User, Smart meter
Pre-Conditions	The user has the app downloaded on their phone which is compatible with the system. The user has an account and is logged in. The user has a smart meter.
Flow of Events	Connecting to a smart meter: 1) (Start) User starts the app and navigates to the points page 2) User starts the scanning process by pressing "add smart meter" 3) The system scans for nearby devices and displays a list of the devices found 4) User selects their smart meter from the list 5) The system starts the verification process to validate the smart meter 6) The system successfully verifies the smart meter 7) Database manager saves smart meter to the database 8) The system displays a message saying "successfully connected" 9) User presses the "OK" button found under the message 10) The system returns the user to the dashboard (End) After a smart meter is connected: 1) (Start) The app awakens at 00:00 2) The system takes in the energy reading for the previous day and the average daily energy usage from the smart meter 3) The system compares the two figures and awards 0.5 points per unit the day reading is less than the daily average

	4) Database manager updates the user's total points in the database by adding the points earned to it 5) Database manager updates the accumulated points counter for points earned by saving energy in the database 6) The system sends a notification which contains information on how many points the user earned by saving energy on the previous day 7) The system goes back to sleep (End)
Alternate Flow	1) (From step 5 in connecting to smart meter) The system fails to verify the smart meter and displays "Connection failed" 2) User presses the "OK" button found under the message 3) The system returns the user to the dashboard (End)
Post Conditions	The number of points the user has increases. The accumulated points counter for points earned by saving energy is increased. A smart meter is connected to the user's account and added to the database.
Scenario	Zora Isagani is a 38-year-old doctor who spends her days with patients and paperwork. One day, a coworker recommends Zora an article about the climate crisis. Having read the article Zora makes it her mission to do her part for the world to help tackle the climate crisis. However, having no prior knowledge in this subject Zora doesn't know where to begin. Luckily, at the end of the article, the article recommends TreeConnect. Without delay, Zora downloads the app and sets up an account. Upon logging in, she learns more about the climate crisis and learns how trees combat the changing climate. She also learns that the app allows users to sponsor trees using points and becomes delighted as she doesn't have the funds to sponsor a tree. Upon learning all this, Zora makes it her goal to sponsor a tree by the end of the year. Zora has chosen to earn points by saving energy at home as this is the most convenient way for her to earn points. Once at home, she opens the app and connects her smart meter to the app. Over the following weeks, Zora tries her best to reduce her energy consumption at home. One way she does this is by ensuring lights are turned off when not in use. Although she struggled to earn points in the first few weeks as she wasn't very good at remembering to turn off the lights, she slowly improved and started to earn points and she was able to see how many points she earned each day by the notifications that she would receive at midnight every day. After six weeks of dedicated efforts, Zora has managed to earn enough points to be able to sponsor a tree allowing her to achieve her goal.

Use case 3: Browse trees catalogue.

Introduction	The system allows users to be able to view all trees available for sponsorship and further information about each tree upon selection. The system also allows users to be able to narrow down their search by allowing users to apply filters.
Actors	User, Database manager, Server (system)
Pre-Conditions	The user has the app downloaded on their phone which is compatible with the system. The user has an account and is logged in.
Flow of Events	1. (Starts) User starts the app and navigates to the sponsorship page 2. Database manager fetches the tree catalogue (i.e. all trees available for sponsorship) and sends it to the system 3. The system displays the tree catalogue. The trees are displayed with their names, picture and cost in GBP and points 4. User presses the "filter" button 5. The system displays the filter options 6. User chooses to filter by cost and chooses a max amount of £15 7. The system sends the filter request to the database manager 8. Database manager filters the tree catalogue by removing all trees that cost over £15 to sponsor and sends it to the system 9. The system displays the filtered tree catalogue

	10. User selects a tree from the tree catalogue 11. Database manager fetches the cost and the description/information of the tree and sends it to the system 12. The system displays the tree along with its cost and description and it also displays a sponsor button 13. User presses the “sponsor” button (End)
Alternate Flow	1. (From step 4) User chooses to filter by location and chooses “Birmingham” 2. The system sends the filter request to the database manager 3. Database manager filters the tree catalogue by removing all trees that will not be planted in Birmingham and sends it to the system 4. The system displays the filtered tree catalogue 5. User selects a tree from the tree catalogue 6. Database manager fetches the cost and the description/information of the tree and sends it to the system 7. The system displays the tree along with its cost and description and it also displays a sponsor button 8. User presses the “sponsor” button (End)
Post Conditions	
Scenario	Yanni Lovita is a 35-year-old teacher who has been using the app for the last 2 years. As Yanni is quite knowledgeable in the climate crisis, Yanni has made it his goal to sponsor a tree every month which he has yet to fail. Yanni enjoys learning and spends quite a lot of time on TreeConnect even when not sponsoring a tree. Yanni spends his time on the tree catalogue where he enjoys acquiring new knowledge about different types of trees as the tree catalogue has a vast amount of information about each tree such as its value to wildlife, details about its colour and even some mythology and symbolism facts. Yanni suddenly wakes up from his sleep panicking as there are only 15 minutes remaining until the end of the month and due to him being very busy throughout the month, he had forgotten to sponsor a tree. Panicking, Yanny quickly checks his bank account and sees that he only has £4.50 in his bank. With only 10 minutes remaining, he quickly opens TreeConnect and is determined to get a tree sponsored in the next 10 minutes so that his streak of sponsoring a tree every month is not broken. As Yanni has blocked the app from using his location and the fact that he doesn’t own a smart meter means that Yanni has accumulated a total of 0 points over the past 2 years meaning he doesn’t have the option to use points to sponsor a tree. Realising that it will take him quite a bit of time to find a tree that cost less than £5 to sponsor Yanni starts further panicking. However, as Yanni is a long-time user and is familiar with the app, he remembers that the app has a filter option. Swiftly applying the filter to only view tree that cost less than £5 to sponsor, Yanni selects the first tree that appears and quickly sponsors it. With only 2 minutes remaining Yanni has managed to not lose his streak, and this was all thanks to the filter option.

Use case 4: Earning points by cycling. (The app doesn’t need to be open for the user to earn points by cycling).

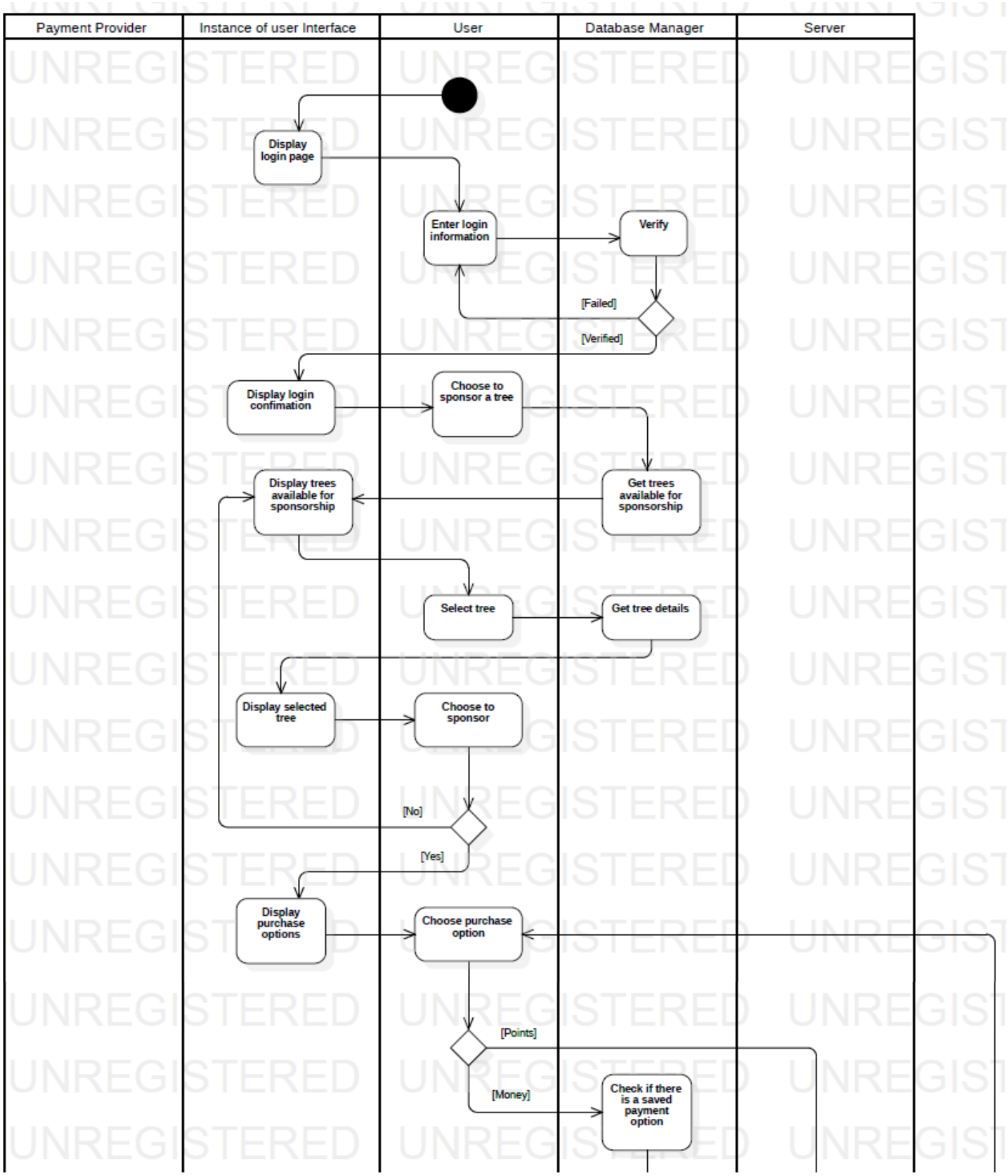
Introduction	The system allows the user to earn points by cycling. The system can check what type of transportation is used by the user and what the distance travelled is by using google maps which it will initialise when it detects movement.
Actors	User, Server (system), Database manager, Google maps

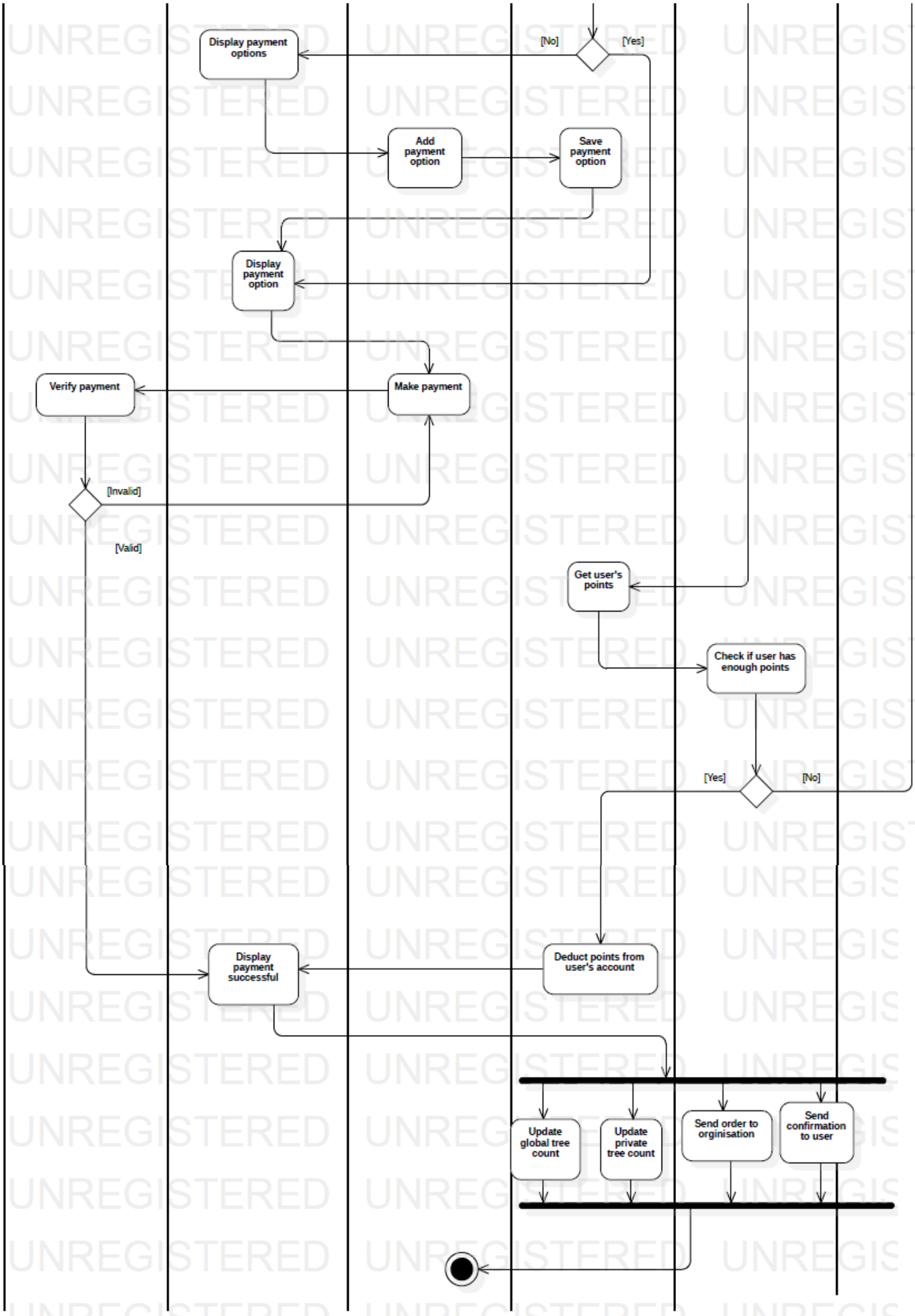
Pre-Conditions	The user has the app downloaded on their phone which is compatible with the system. The user has Google maps on their device. The user has given permission for the app to user their location (i.e. location tracking is enabled). The user has a bike. The user has internet. The user uses only one form of transportation throughout their commute and doesn't stop during the commute.
Flow of Events	1. (Start) User starts commuting 2. The system detects the movement, awakens and initialises Google maps 3. The systems requests google maps to provide it with the type of transport the user is using 4. Google maps returns that the user is cycling 5. The system request Google maps to track the user's commute 6. User finishes commuting 7. The system detects that the movement has stopped and request the distance cycled from google maps 8. Google maps returns the distance cycled 9. The system awards the user with 10 points per km cycled 10. Database manager updates the user's total points in the database by adding the points earned to it 11. Database manager updates the accumulated points counter for points earned by cycling in the database 12. Database manager updates the total distance cycled in the database 13. The system sends a notification which contains information on how many points the user earned by cycling and the total distance cycled 14. The system goes back to sleep (End)
Alternate Flow	1. (From step 3) Google maps returns that the user is driving 2. The system goes back to sleep (End)
Post Conditions	The number of points the user has increases. The accumulated points counter for points earned by cycling increases. The total distance cycled is increased.
Scenario	Fergie Talip, is a 19-year-old jobless student who has had an account with TreeConnect for a total of 8 months now, but he has yet to sponsor a tree as he lacks the funds to do so. Fergie has turned to earning points so that he may be able to sponsor a tree but due to his busy schedule and lack of a smart meter at home, he currently doesn't have a way to earn points. However, he has recently started university which is 3km aways from his home. Due to Fergie's parents not being able to drop him of at university, he has decided to get a bike and now cycles to and from university. Fergie keeps his phone, which has the app downloaded, in his pocket each time he cycles so that the app can detect each time he starts and finishes cycling so that it can calculate the distance travelled by cycling and thus reward him the correct number of points. Each time Fergie completes a journey on his bike, he receives a notification alerting him on the number of points he earned. It has now been a year since Fergie started university and he has cycled a total of 600km and has managed to earn 6000 points. He can now afford to sponsor a tree and he is able to do this without the need of money.

B4: Activity Diagram

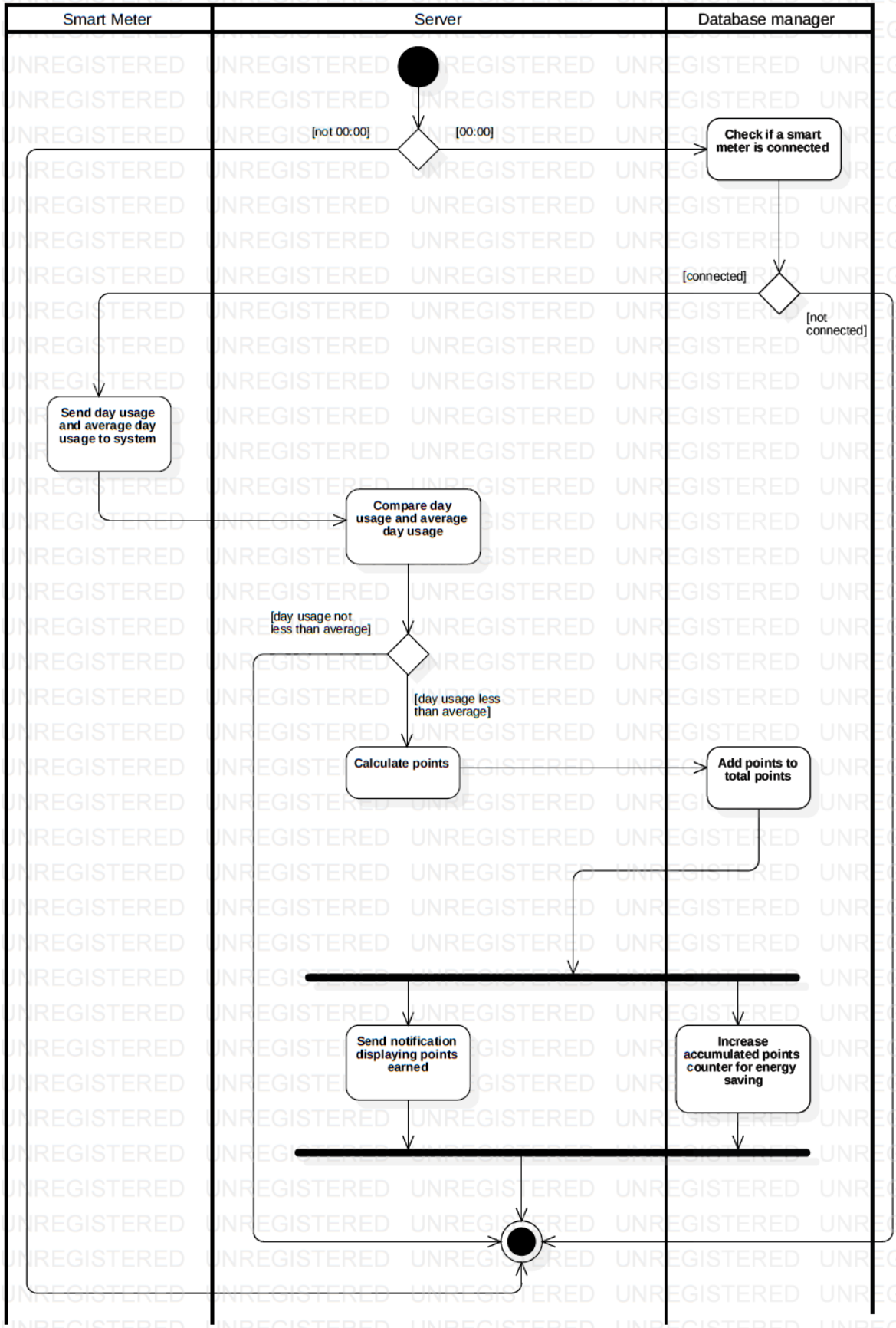
Activity Diagram 1: The following shows how a tree can be sponsored by a user.

An actor called 'instance of user interface' has been included in the following diagram as this represents the component of the system resposible for displaying information to the user on a screen and thus it acts as a buffer between the user and the server. That is, the 'instace of user interface' retains information. This allows the server to perform other tasks independently ensuring that the user's screen remains unchanged until the server makes alterations to it.




Display payment options
[No]
[Yes]
Add payment option
Save payment option
Display payment option
Verify payment
Make payment
[Invalid]
[Valid]
Get user's points
Check if user has enough points
[Yes]
[No]
Display payment successful
Deduct points from user's account
Update global tree count
Update private tree count
Send order to orginisation
Send confirmation to user

Activity Diagram 2: The following shows how points are earned by saving energy at home which is measured using a smart meter. It is assumed that the user is logged into the app.



B5: Class Analysis

Noun-Verb Analysis -

To identify the classes that could possibly be used to create the system, we use a Natural Language Analysis method know as Noun-Verb analysis, which is done on the original specification for the project. **Nouns will be in bold,** and verbs will be underlined (Note: *repeat nouns and verbs or close synonyms will be italicised and not included in the noun-verb analysis*).

Support Functionality

The *system* must require *admin and support account* creation to be done only through an **invite link** from an *existing admin account*. The **system** would allow users to contact **support** or **customer service**. The *system* would allow multiple **admin and support accounts** to be created but only one super admin must exist. The *system* would allow **admin privileges** to be expanded/limited or revoked.

The *system* should add an **expiration period (30 mins)** to *generated invite links* and limit each *link* to only one *admin account* creation. The system must allow admins to login with a **uniquely generated ID and password**. If needed, they can help users with **access**, **sponsorship** or close their *accounts* if there is a **security breach**. The *system* must keep a **log of admin login sessions** for accountability. The *system* should generate a *unique ID* for *each admin account* upon *account creation* and only allow to be signed in with this *ID and password*. One *unique ID* per *account*. The system should lock the *admin account* after three failed **login attempts**. To access their account, they must contact the **super admin** to gain *access* again. The system should require a different **password requirement** for *admins* and **users**. For *admins*, **10-characters minimum: 2 upper cases, special character and 2 numbers**.

Login Functionality

The *system* must allow *users* to create *accounts* by accepting their **email address** and *password*. The system must allow the *user* to enter their **personal details** when *creating an account*. The system must prompt the *user* to meet the *password requirements* when *creating an account*. The *system* must hash *password* at **client end** before sending it to the **database**. The *system* must validate the entered *email address* by sending a 4-**digit verification code** to their *email address*. The *system* must be able to verify and accept the *4-digit code* entered by the *user*. The *system* must allow *users* to log in to their *accounts* using the *email address* and *password*. The system must allow *users* to reset their *password* by sending a **password reset link** to the *email* the *user* inputs if the user has forgotten their password.

The *system* must authenticate *users' credentials (email and password)* to grant access to the *platform*. When resetting the password, the system must censor first 4 characters of *the email address* using "*" for security purposes. The system must require minimum of 8 characters, 1 upper case, 1 lower case and a special character while inputting a *password*. The system must check if *email address* is already associated with an *account*. Each *email* must only be associated with one *account*. When a **login from a new device** is detected, the *system* should alert the *account holder* through their preferred **method of contact**. The *system* must *enable users to reset their password when they forget their password (*passwords must always meet requirements).

Dashboard Functionality

The *system* must enable access to **account functionality** for **user management**. The *system* must provide access to **points functionality**. The *system* should use **data analytics** to provide *users* with **sponsorship recommendations** that are in line with their interests based on **behavioural data**. The *system* should display **information about the climate change** and how their efforts are important in combating this. The *system* should provide information about different **types of trees** that are available for sponsorship. The *system* should display real-time *data* on the **total number of trees sponsored** by the user and by all users globally through our system. The system should display real-time *data* on the **total number of points accumulated** by the *user*.

Account Functionality

The *system* should allow users to control their **privacy settings**. The *system* could allow users to set up additional **security measures** like **two-factor authentication**. The *system* should allow the user to update their *personal details*. The *system* should allow the user to enter/update their **payment details**. The *system* must enable users to turn on/off **location tracking**. The *system* must allow *users* to log out of their *account*. The *system* must allow *users* to delete their *account*. The *system* must allow *users* to manage their **notification preference**. The *system* must allow the *user* to change their preferred **method of contact**. The *system* could allow users to choose a preferred **theme** (dark/light mode).

The system must ensure that *privacy settings* management such as *location tracking* and *analytical data collected* must be easily accessible to *users*. The *system* should ensure process of updating *personal information* should be straightforward and intuitive. **Name text-fields** cannot be left empty and must contain only letters. When a *user updates their email address*, the *system* must *verify via a 4-digit code that should be sent to the users email address* and the user can enter this *code* in **account functionality**. The *system* must ask for permission to save *payment details*. If a *payment detail* is updated, the *system* must remove completely any previous *payment detail* saved. When the user *updates their method of contac*t, the system must implement the change effect immediately. The *system* must support an error free *notification preference* choice to ensure the *user* is not disturbed or annoyed. The *system* could support **text-to-speech** and **increased font size** to allow **accessibility** to users with **disability**. The *system* must adhere to **data privacy and data retention policies** upon *account deletion*.

Tree Functionality

The *system* must display all available *trees* for *sponsorship* and their price in **GBP** and **points**. The *system* must deduct *points* from the *user's account* if *points* were used to *sponsor a tree*. The *system* must provide a range of **payment services** (Apple pay, debit/credit card etc) to allow the *user* to pay for *sponsoring a tree*. The *system* must communicate **tree sponsorship orders** to the **organisation(s) responsible for planting the sponsored trees**. The *system* must send immediate **confirmation for successful orders** to the user's *preferred contact method*. The *system* would send users **updates** on sponsored trees. The system should continuously track the *total number of trees sponsored* and update the *count* in real-time to *provide accurate data*. The system should provide **information about different types of trees that are available for sponsorship** and inform *users* about their **environmental impact**. The *system* could allow *users* to add a **personal touch** (such as request a

plaque with a name of a beloved one) to *sponsored trees* at **checkout**. The *system* could offer a search feature for finding trees or sponsorship details.

The *system* must keep up to date *data of sponsorship details* and *display this in real-time to users* so they can make a well-informed choice. The *system* should *provide clear, concise and accurate data about each species of trees*. The *system* could allow *users* to browse *trees for sponsorship* through filtering **species** or **location**. The *system* must be able to handle multiple *tree sponsorship orders* from multiple *users* without **performance degradation** or **errors**. The *system* should comply with relevant **financial regulations** and *laws* when processing payments. The system should display each *tree* in a visually appealing **tile** with *cost in both points and money* displayed on **bottom right corner**. The *system* should *send a confirmation through their preferred contact method* and display the *sponsored tree* on the **user-interface** immediately after sponsoring a tree.

Points Functionality

The *system* must display how many points each **activity** is worth. The *system* must contain an **option which explains how to earn points**. The *system* must be able to calculate **distance walked or cycled** and use this information to award points. The *system* must be able to connect to a **smart meter**. It should *use this information to calculate the number of points to award the user with*. The *system* must accurately keep track of *accumulated points* based on the *user activity*. The system must send a notification when the user earns points by either *cycling, walking* or **by being energy efficient**. The *system* must display **real-time updates** on *points earned* and a **chart of what activities those points came from**.

The *system* must *reward users 10 points for each KM cycled*. The system *rewards 0.1 points for each step taken*. The system must *collect smart meter reading data at 00:00 am* and *reward 0.5 points for each unit of energy they are below their average use* (which will be collected from their smart meter). The system must have a **standard conversion rate** from point to money and enable *users* to see how much money their points translate to. 10000 points = £10.

Noun/verb	Accepted	Reason
Support		
System	Class User: sendRequest() receiveRequest() Class ServerHandler: sendRequest() receiveRequest() queryDatabase()	Two classes representing the two components of the system: User and ServerHandler. At the server end there are methods that are constantly running in series in a loop: one to receive requests from the client and another to send requests to the client. The same thing happens at the client end. At the server end there is also the queryDatabase method which sends an SQL query to the database.
Allow users to contact support or customer service	User: contactSupport()	Method within user class that allows users to contact support
Admin and support accounts	Class Admin Class Support inherits Admin	Admin class represent admin account. Support class inherits functionality from admin class
Would allow multiple admin and support accounts to be created	No	Too general a method
Allow admin privileges to be expanded/limited or revoked	Class Superadmin inherits admin: addPrivilege() removePrivilege()	Superadmin class controls admin privileges. These can be added or removed
require admin and support account creation to be done only through an invite link from an existing admin account.	Admin: generateInviteLink() Class SupportManagement: List<String> inviteLinks	An admin may generate an invite link, which will be stored at the server end. SupportManagement is a class at the server end that manages support functionality. It stores invite links generated by admins. These will include a time stamp for reference of the method below.
Add an Expiration period (30 mins)	SupportManagement: deleteOldInviteLinks()	This is constantly running at the server end. It looks at timestamps and deletes the links that are older than 30 minutes.
Limit each link to only one admin account creation	Admin: enterInviteLink() SupportManagement: checkInviteLink()	The first method allows the invite link to be entered. This will contact the server to check the invite link, and if it is valid it will be deleted from the

	activateAccount()	aforementioned list of invite links. Then the account will be activated using the activateAccount method. Login details are stored in the database, with passwords of course hashed.
Allow admins to login	Admin: login() logout()	Methods that allow admins to log in and log out.
Uniquely generated ID and password	No	Done in activateAccount method
Have access to user management tools	No	Method is too general.
Can help users with access, sponsorship or close their accounts	Admin: accessAccount() manageSponsorships() closeAccount()	Three methods to manage account access, sponsorships and the closure of accounts.
Security breach	No	N/A
Keep a log of admin login sessions for accountability	SupportManagement: List<String> loginSessions logSession()	This list will track admin login sessions. Once an admin logs in, the logSession method will be triggered that logs the session information to the list.
only allow to be signed in with this ID and password	SupportManagement: validateLoginDetails()	Method that validates ID and password when logging in
Lock the admin account	SupportManagement: lockAccount() Admin: int loginAttempts boolean isLocked	If a login attempt is made unsuccessfully, it will increment the loginAttempts method. If this is at least 3, the account will be locked and the loginAttempts attribute is set to zero. After a successful login attempt, this is also set to zero.
Three failed login attempts	No	Implemented above.
Contact the super admin	No	This is done by telephone, not by the system.
gain access again	Superadmin: unlockAccount()	This method unlocks the admin account.
Require a different password requirement for admins and users	No	N/A
10-characters minimum: 2 upper cases, special character and 2 numbers	No	Implemented when uniquely generating password.
Login		
login	Class LoginManagement	Class that manages login functionality
Allow users to create accounts	LoginManagement: createAccount()	Method which will take in the necessary information and

		create an account for the user. Account details will be stored in the database
Accepting their email address and password	LoginManagement: isValidEmailAddress() isValidPassword()	Methods within the loginManagement class isValidEmailAddress() will makes sure the entered email address is valid IsValidPassword() will check the password inputted against the password requirements to see if it is valid Valid email address and password is necessary for account creation
Allow the user to enter their personal details when creating an account	LoginManagement: enterDetails()	Allows user to enter their details.
Prompt the user to meet the password requirements when creating an account	LoginManagement: promptValidPassword()	Method which uses the isValidPassword() method mentioned above, if the password isn't valid, a message will be displayed to the user outlining what makes a valid password and/or what is missing from their inputted password to make it valid
Hash password at client end before sending it to the database	LoginManagement: hashPassword()	Method which will encrypt the password before storing it in the database to ensure privacy of the password
Validate the entered email address by sending a 4-digit verification code to their email address	LoginManagement: send4DigitCode()	This method generated a random 4-digit code which will be sent to the user's email address. The code will need to be used to activate the account
Allow users to log in to their accounts using the email address and password	Class UserAccount: boolean accountVerified login()	UserAccount is a class that handles account and login functionality on the user end. Method in the UserAccount class which will allow the user to use the email address and password created to login to the app once they have activated their account (when accountVerified is True)
Allow users to reset their password by sending a	LoginManagement: forgotPassword()	Method in the loginManagement class which will activate once the 'Forgot

password reset link to the email the user inputs		Password' button is clicked. The method will send a link to the user's email address to change their password
Authenticate users' credentials (email and password)	No	Email address authenticated in the isValidEmailAddress() method. Password authenticated in the isValidPassword() method
Grant access to the platform	No	N/A
Censor first 4 characters of the email address using "*"	UserAccount: censorEmailAddress()	Method in the UserAccount class which will replace the first 4 characters of the user's email address with * on the user-interface
Require minimum of 8 characters, 1 upper case, 1 lower case and a special character while inputting a password	No	Part of the isValidPassword method
Must check if email address is already associated with an account	LoginManagement: accountExist()	Method in the LoginManagement class called by the isValidEmailAddress method which will check in the database if there is an account which uses the inputted email address
Login from a new device	LoginManagement: checkDevices()	Method in the LoginManagement class which tries to find a match between the devices being used and devices used previously which are saved in the database
When a login from a new device is detected, the system should alert the account holder through their preferred method of contact	LoginManagement: newDeviceDetected()	Uses method checkDevices above to see if the login is from a new device. If the user has logged in from a new device, a message will be sent via their preferred method of contact stating that a new device has logged into their account
Dashboard		
Enable access to account functionality	No	Not relevant.
User management	UserAccount: userManagement()	userManagement allows user management for the user.
Provide access to points functionality	No	Will be done in the points section where this is clearer.

Use data analytics	User: trackSponsorships()	trackSponsorships tracks trees the user has sponsored.
Provide users with sponsorship recommendations that are in line with their interests	User: generateSuggestions()	This method will generate suggestions for trees to sponsor based on the tracked sponsorships as per the previous method.
Behavioural data	No	Done in trackSponsorships
Display information about the climate change and how their efforts are important in combating this	User: displayClimateChageInfo() Class TreesManagement: getClimateChangeInfo()	Displays climate change information as described. TreesManagement is a server class that manages tree functionality. The second method gets climate change information from the external database
Provide information about different types of trees that are available for sponsorship	User: treesForSponsorship() TreesManagement: getTreesForSponsorship()	Method which displays the data of all the trees available for sponsor to the user-interface. It will contact the server for this information and the server will user the getTreesForSponsorship method to get this data from the database and return it to the user.
Display real-time data on the total number of trees sponsored by the user and by all users globally through our system	User: displayTreeData() TreesManagement: getTreeData()	Methods which gets statistical tree data from the database and outputs them.
Display real-time data on the total number of points accumulated	User: displayPoints()	Method which will display the total number of points accumulated since activating their account. There is also a breakdown by activity stating metrics and points per activity.
Account		
Allow users to control their privacy settings	No	Done is userManagement method.
Allow users to set up additional security measures	No	Done in userManagement method.
Two-factor authentication	Class UserPreferences: boolean 2fa	UserPreferences is a client-side data class that stores user preferences.

		Boolean that is true if 2FA is enabled and false otherwise
Allow the user to update their personal details	UserAccount: updateDetails() Class AccountManagement: updateDetails()	The first method will be triggered when the user wishes to update their details. The server will then be contacted and it will update the details in the database. AccountManagement is a server class that manages account functionality.
Allow the user to enter/update their payment details	No	Implemented above
Enable users to turn on/off location tracking	UserPreferences: boolean locationTrackingEnabled	Boolean that checks if location management is enabled.
Allow users to log out of their account	UserAccount: logout() boolean isLoggedIn	logout is a method that allow user to log out of their account. There will also be a flag for each user stating whether they are currently logged in.
Allow users to delete their account	UserAccount: deleteAccount() AccountManagement: deleteAccount()	Allows user to delete their account. Removes their User object from the system. The first method contacts the server which triggers the second method.
Allow users to manage their notification preference	UserPreferences: boolean notificationsEnabled	Boolean flag that indicates if notifications are enabled.
Allow the user to change their preferred method of contact	UserPreferences: String preferredContact	Stores preferred method of contact.
Allow users to choose a preferred theme (dark/light mode)	UserPreferences: boolean darkModeEnabled	Boolean that dictates whether dark mode is enabled
Ensure that privacy settings management such as location tracking and analytical data collected must be easily accessible	No	This depends on the design of the system, not functionality
Ensure process of updating personal information should be straightforward and intuitive	No	As above
Name text-fields cannot be left empty and must contain only letters	UserAccount: isNameValid()	isNameValid method checks the name against the stated requirements. If it is valid, it is stored. Otherwise, it is not

		stored and will be flagged on the user interface.
User can enter this code in account functionality	No	Related to design not functionality.
Ask for permission to save payments details	UserAccount: getPaymentStoringPermission( )	This method is triggered once payment details are entered on the user interface. It will ask the user for storage permission.
Remove completely any previous payment detail saved	No	This is done once payment details are updated anyway.
Implement the change effect immediately	No	As above
Support an error free notification preference choice	ServerHandler: sendNotification()	This method sends notifications to users' phones.
Ensure the user is not disturbed or annoyed	No	N/A
Support text-to-speech and increased font size	UserPreferences: boolean textToSpeechEnabled boolean increasedFontSizeEnabled	These two attributes flag if text-to-speech and increased font size are enabled.
Allow accessibility to users with disability	No	N/A
Adhere to data privacy and data retention policies upon account deleted	No	N/A
Trees		
Display all available trees for sponsorship and their price in GBP and points	No	Done in displayTreeData() method above.
The system must deduct points from the user's account if points were used to sponsor a tree	Class UserPoints: int numPoints Class UserPayment: deductPoints()	UserPoints is a data class that contains users' points data. The numPoints attribute stores the number of points the user has. UserPayment is a client-sider class that handles payments. The deductPoints method deducts points from users.
Provide a range of payment services	UserPayment: launchApplePay() launchGooglePay() launchPayPal() payByCard() payByPoints()	The launch method launches different payment providers we are using. There is also the option to pay by card and points.
Allow the user to pay for sponsoring a tree	No	Done above.

Communicate tree sponsorship orders to the organisation(s) responsible for planting the sponsored trees	Classs TreeManagement: sendOrderDetails()	TreeManagement is a class that handles tree functionality. This method sends order details to organisations.
Send immediate confirmation for successful orders to the user's preferred contact method	UserPayment: sendConfirmation()	Sends order confirmation to the users.
Send users updates on sponsored trees	TreeManagement: sendUpdates()	Sends updates to users regarding tree updates.
Continuously track the total number of trees sponsored	TreeManagement: int numSponsored updateNumSponsored()	There is an attribute storing the number of trees sponsored as well as a method that updates in fixed intervals.
Update the count in real-time	No	Done above
Provide information about different types of trees that are available for sponsorship	No	Done in displayTreeData method
Inform users about their environmental impact	No	Done in displayTreeData method.
Allow users to add a personal touch to sponsored trees at checkout	User: checkout()	Method that allows user to check out.
Offer a search feature for finding trees or sponsorship details	User: search()	Method that searches for tree based on parameter.
Keep up to date data of sponsorship details	ServerHandler: enterSponsorshipDetails()	Allows sponsorship details of each tree to be entered
Allow users to browse trees for sponsorship through filtering species or location	User: applyFilters()	Method that filters results displayed.
Handle multiple tree sponsorship orders from multiple users without performance degradation or errors	No	Related to server design.
Comply with relevant financial regulations and laws when processing payments	No	N/A.
Display each tree in a visually appealing tile	No	Related to design of UI.
Cost in both points and money displayed on bottom right corner	No	Related to UI.
Display the sponsored tree on the user-interface	No	As above.
Points		
Display how many points each activity is worth	No	Already done in displayPoints method

Contain an option which explains how to earn points	No	Done above
calculate distance walked or cycled	UserPoints: calculateDistanceTravelled()	Method that calculates distance travelled as described and awards points.
Use this information (distance walked or cycled) to award points	No	Done above.
Connect to a smart meter	UserPoints: readSmartMeterReadings()	Reads smart meter readings and applies points accordingly.
Accurately keep track of accumulated points based on the user activity	UserPoints: int stepsDistance int stepsPoints double cyclingDistance int cyclingPoints double energyUsedToday double energyAverage int energyPoints	Variables that store steps and cycling distance and points.
Send a notification when the user earns points	No	Done in sendNotification method
By being energy efficient	No	N/A.
Display real-time updates on points earned and a chart of what activities those points came from	No	Done in displayTotalPoints method.
Standard conversion rate	No	Implemented in the calculate method and the smart meter method above.
Enable users to see how much money their points translate to	No	Already visible on dashboard.

CRC Cards

Using the noun-verb analysis above, we have created CRC (Class-Responsibilities-Collaborators) cards that outline the responsibilities each class has. This will better inform our object diagram.

User	
Responsibilities	**Collaborators**
Provides user interface (UI) which displays trees and points data	UserPoints, ServerHandler, TreeManagement
Provides functionality to contact support	
Tracks user sponsorships and generates suggestions based on this	
Contains classes that store data by categories and perform actions by category.	UserPreferences, UserPoints, UserPayment, UserAccount
Sends and receives server requests	ServerHandler
Allows payments to take place	UserPayment, ServerHandler

ServerHandler	
Responsibilities	**Collaborators**
Stores user attributes in the database	
Handles requests to and from the client (user)	User
Allows notifications to be sent to users	User
Allows tree/sponsorship details to be entered	
Contains management classes that handle different requests	SupportManagement, LoginManagement, AccountManagement, TreeManagement, PointsManagement

Admin	
Responsibilities	**Collaborators**
This class represents admins such that each admin has its own class	
Generates invite links for the creation of admin and support accounts. Allows them to be entered.	ServerHandler, SupportManagement
Allows admins to be logged in and out of their account	
Has access to user management tools	User
Sends data to and from the server	ServerHandler

Support	
Responsibilities	**Collaborators**
Subclass of Admin that contains extra support-specific functionality	Admin

Superadmin	
Responsibilities	**Collaborators**
Subclass of Admin	Admin
Allows admins and support to gain access if they are locked out of their account	Admin
Alllows admin privileges to be added and revoked	Admin

SupportManagement	
Responsibilities	**Collaborators**
Stores generated invite links	Admin
Keeps track of admin login sessions	Admin
Locks admin accounts	Admin

LoginManagement	
Responsibilities	**Collaborators**
Allows users to create accounts and log in to them.	ServerHandler, UserAccount
Validates login credentials entered for both registering and logging in	ServerHandler, UserAccount
Allows users to reset their password	ServerHandler, UserAccount
Detects logins from new devices and sends confirmations to email addresses accordingly	ServerHandler
Allows users to log out of their accounts	ServerHandler, UserAccount

UserAccount	
Responsibilities	**Collaborators**
Allows updating of user details	ServerHandler, AccountManagement
Handles logging in and logging out	ServerHandler, LoginManagement, AccountManagement

UserPreferences	
Responsibilities	**Collaborators**
Stores data about user preferences	User

AccountManagement	
Responsibilities	**Collaborators**
Updates user details	ServerHandler
Allows account deletion	ServerHandler, UserAccount

UserPoints	
Responsibilities	**Collaborators**
Keeps track of points data	User, ServerHandler, PointsManagement
Constantly reads smart meter readings	
Calculates points accumulated by the user based on distance information	UserPoints

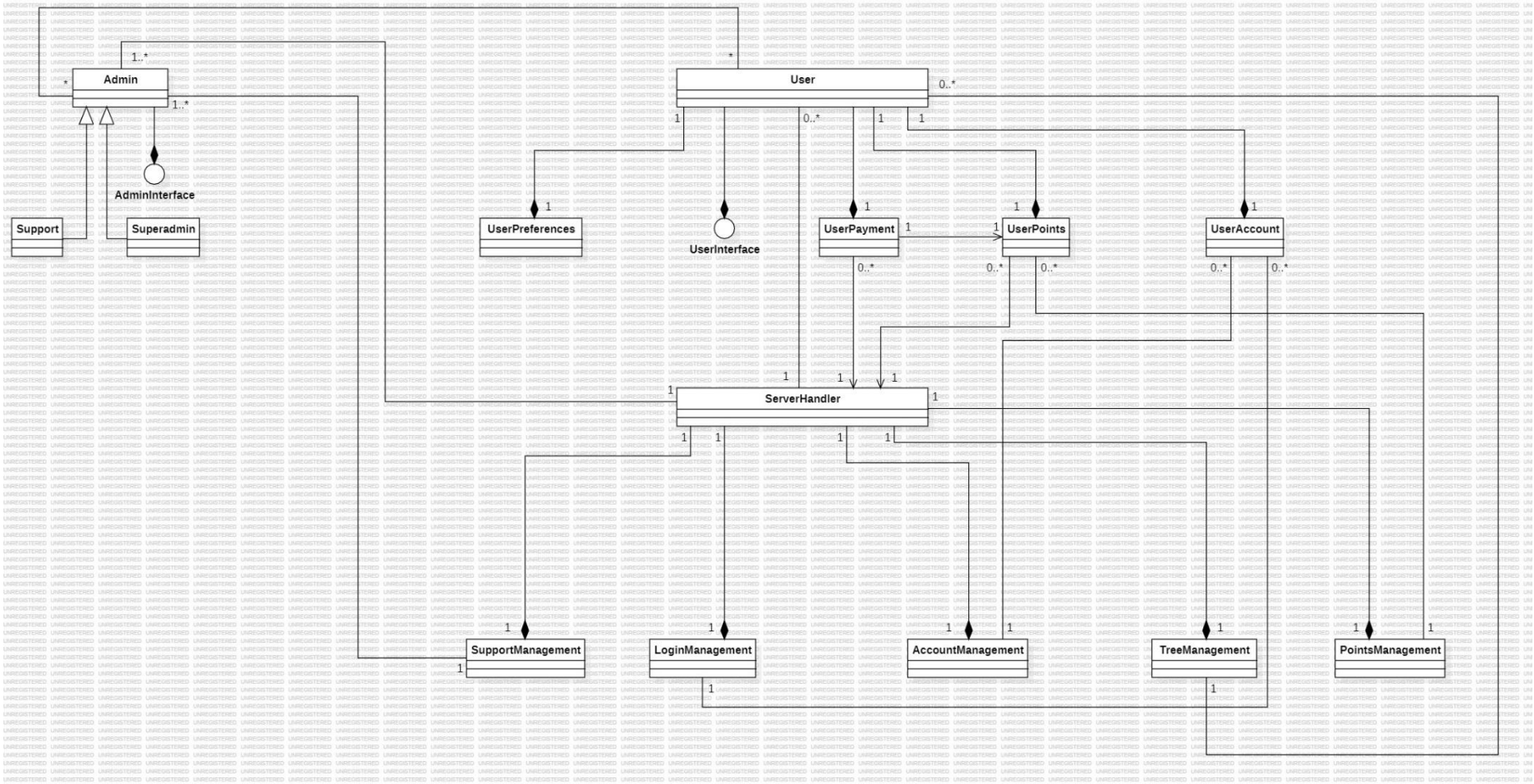
PointsManagement	
Responsibilities	**Collaborators**
Deducts points on purchase	ServerHandler, UserPoints

UserPayment	
Responsibilities	**Collaborators**
Facilitates payments via a payment platform	User
If payment is done via points, these are deducted from the tally	UserPoints
Sends confirmation of payments to users' preferred method of contact	UserPreferences, ServerHandler

TreeManagement	
Responsibilities	**Collaborators**
Sends order confirmations to users	User
Sends users updates on their sponsored trees	User
Tracks the number of trees sponsored	

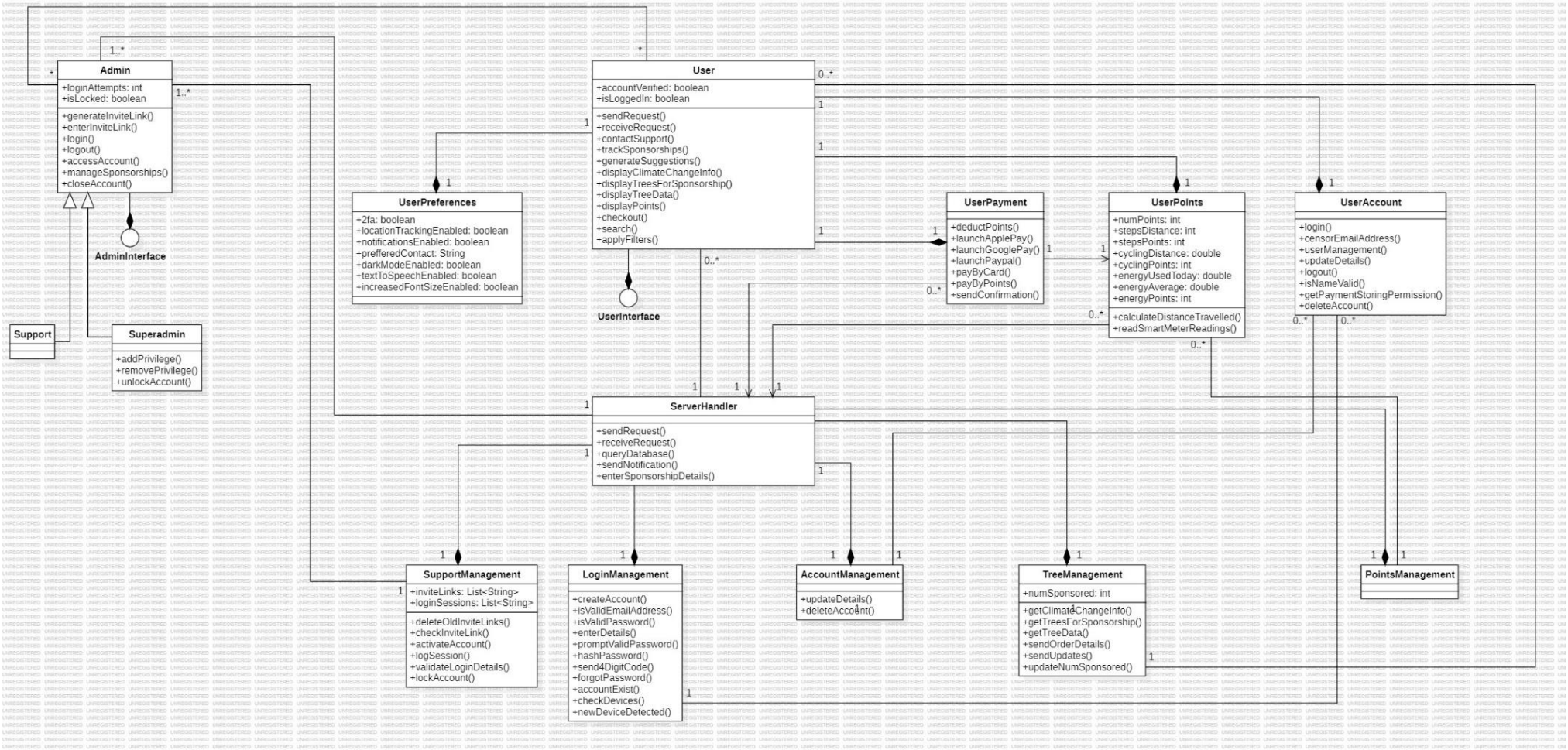
First-Cut Class diagram

Using the noun-verb and responsibility-driven analyses above, we will produce a diagram below that shows all the classes visually alongside their relationships.



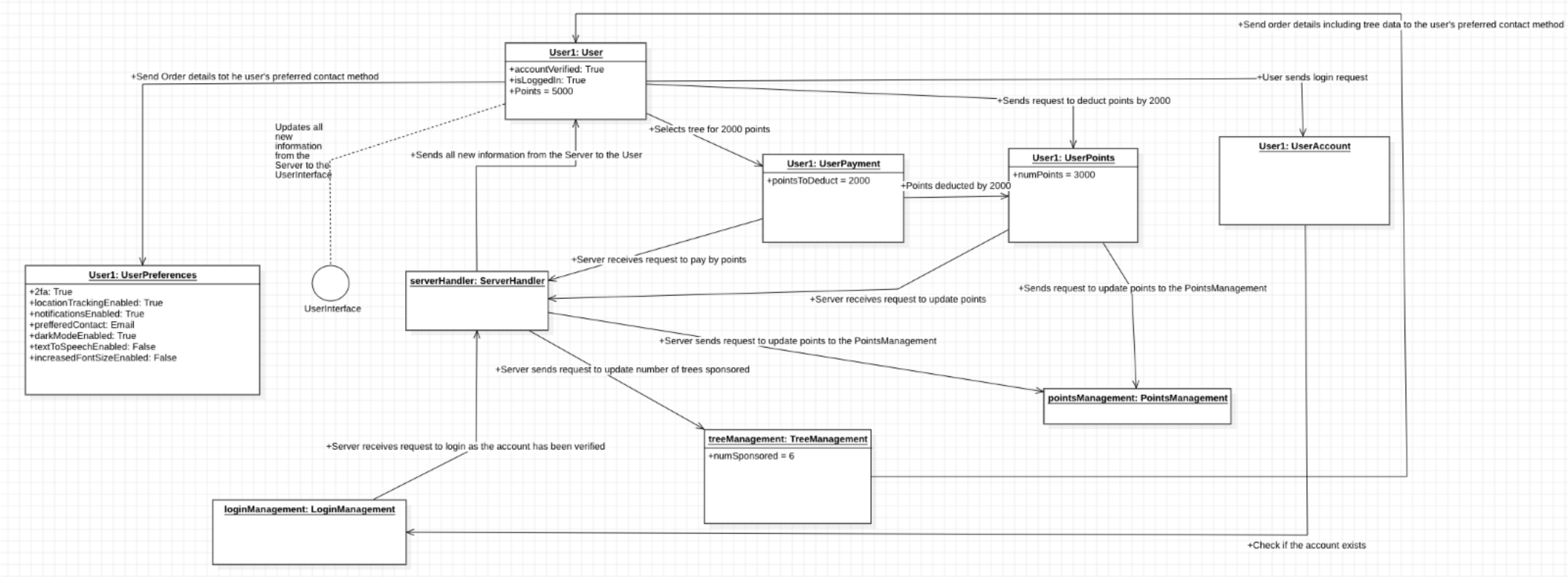
Detailed class diagram

We filled these classes with methods and attributes based on those identified in the noun-verb analysis.



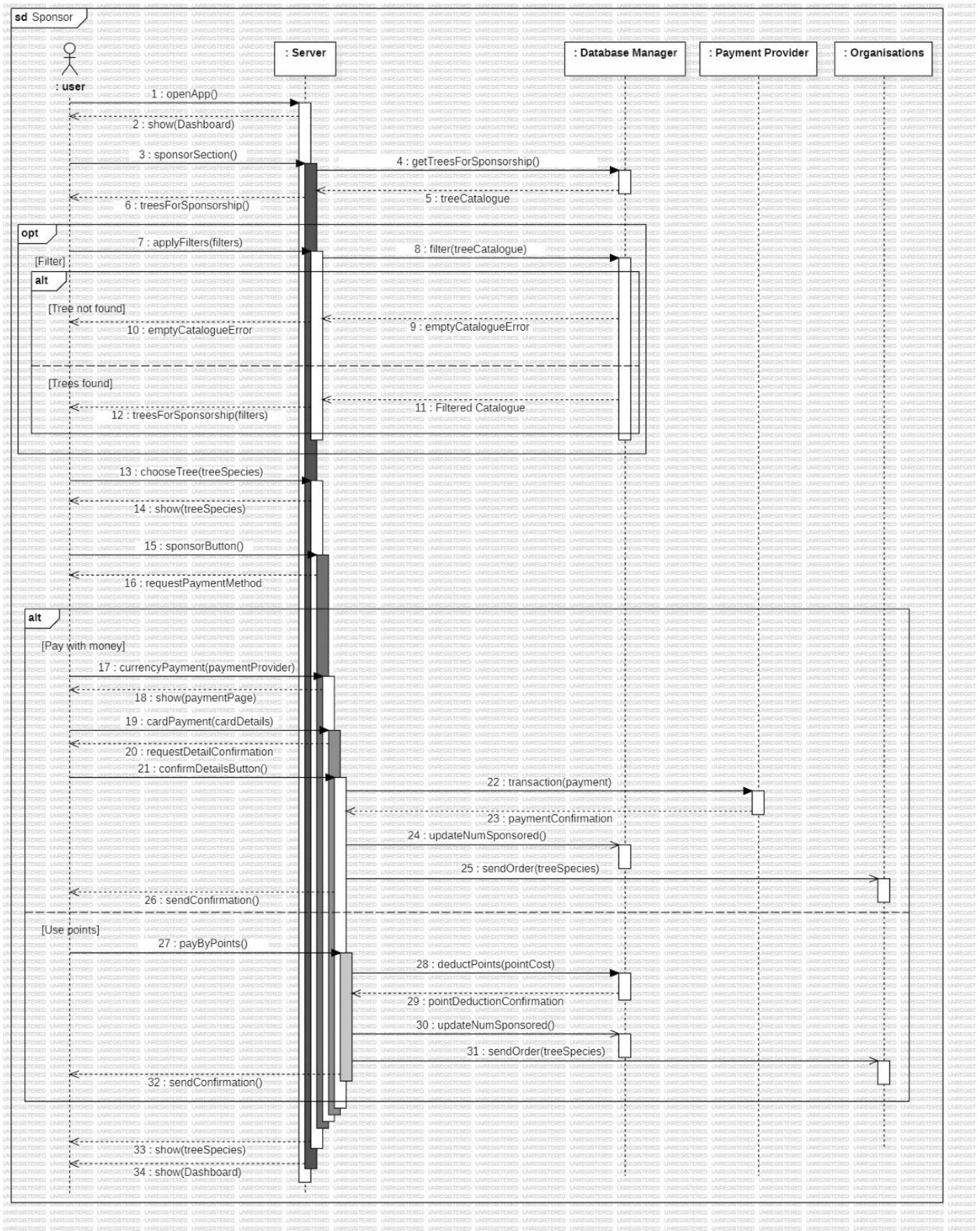
B6: Object diagram

Tree sponsorship scenario. The user attempts to login, and after checking that the account exists in the LoginManagement, the login is successful and updated in the UserInterface. Upon logging in, the user notices he has 5000 points after walking/cycling/smart meter readings before the scenario. The user decides to sponsor a tree worth 2000 points. Upon selecting the tree, the user's points are deducted by 2000, and updated in the external PointsManagement Class/Database, and the number of trees sponsored updates from 5 to 6 in the TreeManagement Class/Database.



B7: Sequence Diagram

Sequence Diagram 1:



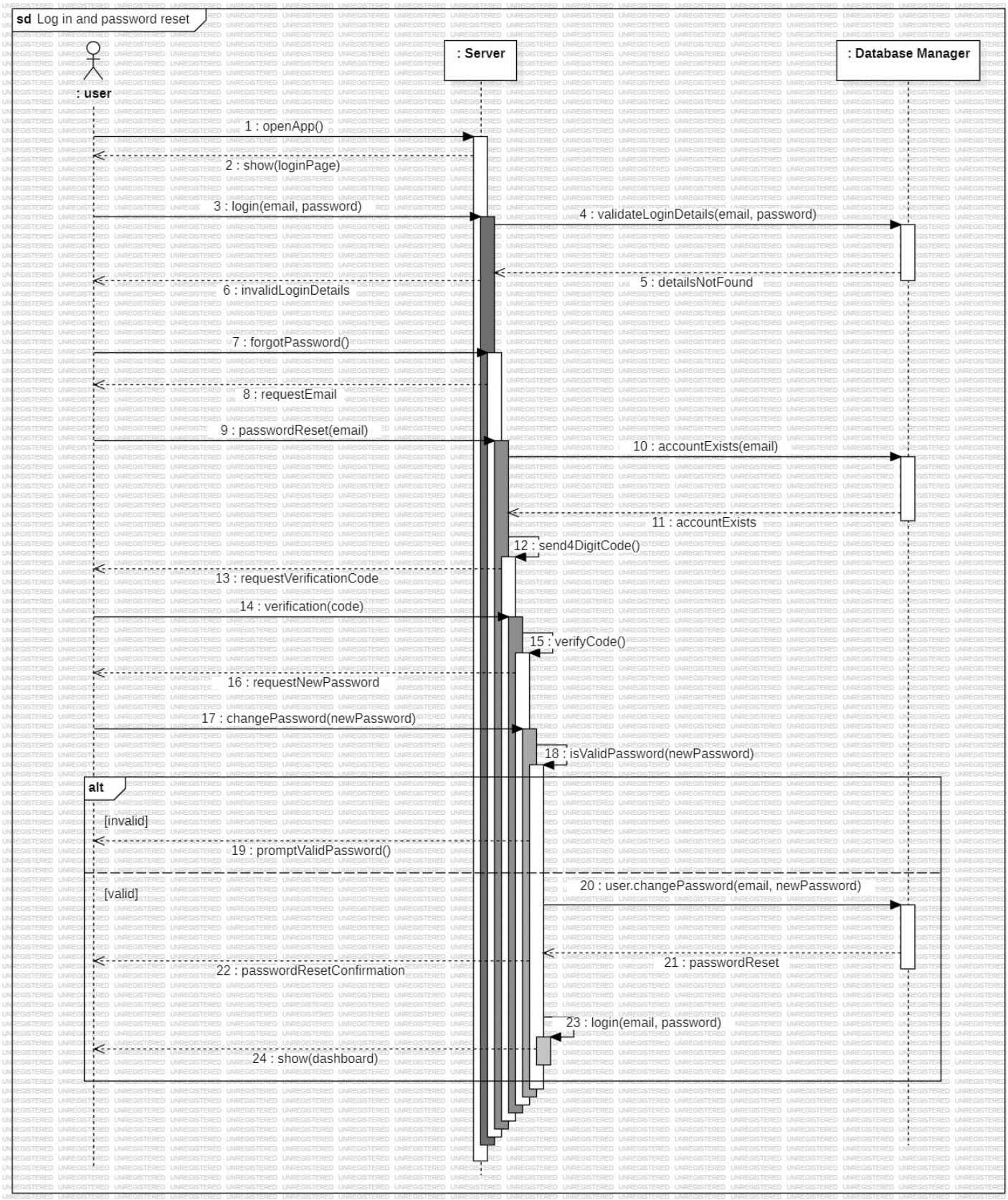
In this diagram, the user sponsors a tree either using the points they have earned or pay by money.

- The use case starts once the user opens the app. This displays the Dashboard to the user.
- User clicks the “Sponsor” section on the Dashboard, which requests server for tree catalogue. Server requests tree catalogue from database manager. Server receives tree catalogue from database manager. Server displays tree catalogue to User.
- User has an option to request the server to filter catalogue. Server requests database manager to filter the catalogue.
  - If trees not available after filter, database manager sends catalogueEmptyError to server. Server displays “catalogue empty error” message to User.
  - If trees are available after filter, database manager sends filtered catalogue to server. Server displays filtered catalogue to User.
- User selects and requests a tree to server. Server displays tree species description.
- User selects “Sponsor” button. Server displays and prompts user to choose payment options.
  - If the user selects a pay with money option and selects “Payment”, this is requested to the server. Server displays payment page and prompts user to input payment details.
  User inputs details and selects payment button, this is sent to server. Server prompts user to confirm payment.
  User sends confirmation to server. Server requests transaction of the payment to the payment provider. The payment provider sends confirmation of payment to the server. The server sends asynchronous request to the database manager to update trees available to sponsor. Server sends asynchronous order to the organization. Server sends payment confirmation to the user.
  - If user selects pay with points, user send option to server. Server then sends deduct points to update database manager. Database manager sends confirmation of point deduction to server. The server sends asynchronous request to the database manager to update trees available to sponsor. Server sends asynchronous order to the organization. Server sends confirmation of point deduction to user.
- Server displays page with the tree and receipt ordered to user.
- Server redirects user to the dashboard.

Assumption:

- The user is logged in to an account on the app.
- If using filter option, user chooses the correct tree catalogue filter requirements in the first attempt.
- Transaction of payment from the payment provider is successful.
- Point deduction was successful.

Sequence Diagram 2:



In this diagram, user attempts to login but has wrong password. User must reset password to login.

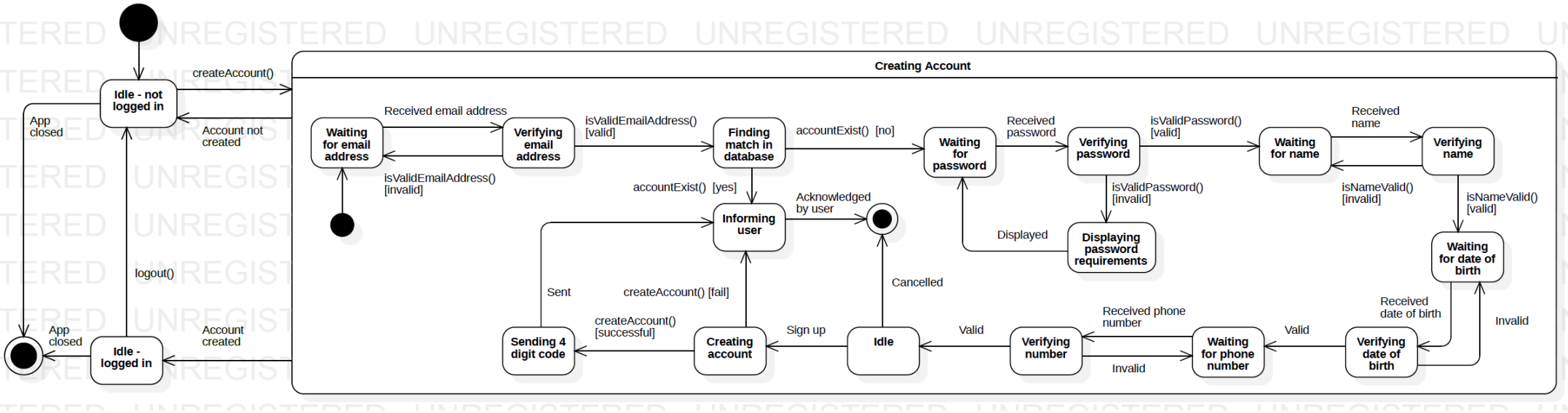
- The use case starts when the user opens the app. The server shows the login page to the user, with the choice on whether to sign up, log in or reset password.
- User enters login details and selects the “login” button. Server checks to see if email and password match an account stored in database manager. In this case, the details do not match an account due to an invalid password, so the database denies login. Server shows “incorrect login details” message to user.
- User chooses to reset password with the “forgot password” link. The server redirects the user to the 'Reset your password' page which prompts user for an email.
- User enters email address and select reset. Server checks if account exists in database manager. Database manager confirms that an account with this email exists, to the server. The server generates and sends a 4-digit verification code to the email address. Server prompts user for the verification code.
- User enters the code and sends it to the server. The server self-checks the code, to see if it matches, which it matches. The server redirects the user to the password reset page.
- User resets password and the password is sent to the server. The server self-checks password constraints on the new password.
  - if invalid password is entered, server prompts user to enter a valid password.
  - if valid password is entered, the server requests the database manager to change the password for the given email. The database manager sends password reset confirmation to the server. The server sends “password reset confirmation" to the user. The server will login to the app, using the email provided and the new password. The server redirects the user to the account Dashboard.

Assumptions:

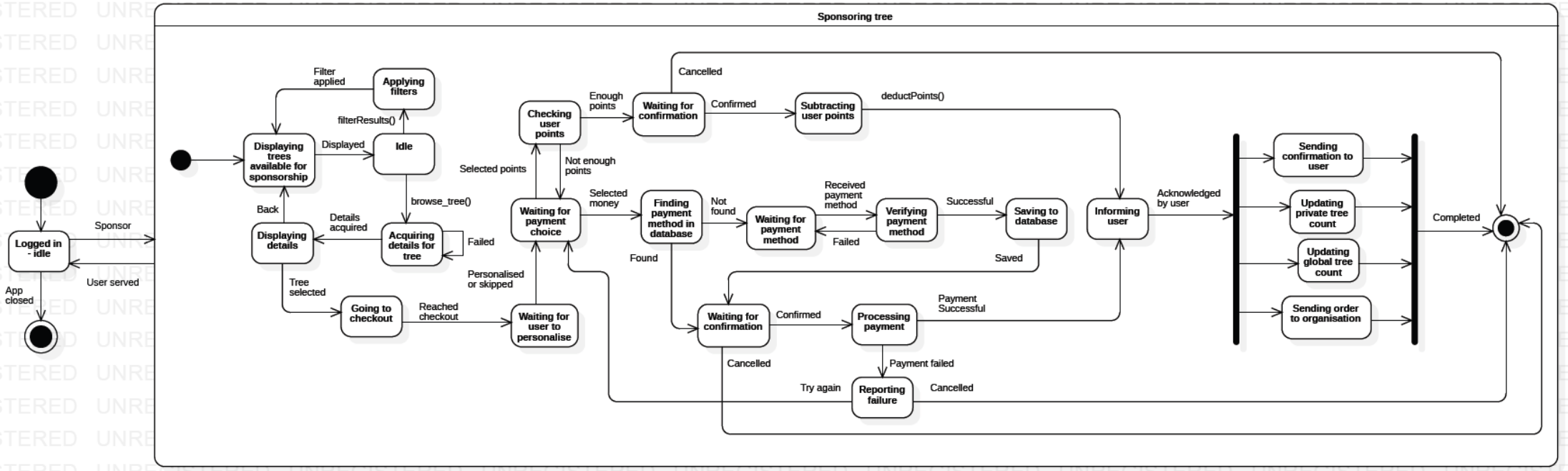
- Account associated with email exists.
- Valid verification code is entered.
- Phone verification is disabled.
- System Log in is not on a new device.

B8: State Machine Diagram

State Machine Diagram 1: The following state machine showcases the states of the machine when a new user tries to create an account. This is an important part of the system as without an account the app is not accessible to the user. It can be seen from the state machine that the system supports a step-by-step signup process. That is, the user is guided through the registration process one step at a time and at the end of each step, (typically) there is a verification process (or other checks) of the entered information and these tests must be passed before the system allows the user to proceed to the next step. This approach is intended to help improve user experience by breaking down the registration process into small clear steps allowing users to focus on one set of information at a time reducing the chances of errors from the user. This is one way the system is designed to be user friendly. Note that this process also speeds up the process of account creation as many verification/checking steps are performed beforehand.



State Machine Diagram 2: The following state machine showcases the states of the machine when a user tries to sponsor a tree using either points or money. It starts of by allowing the user to browse all the trees available for sponsorship and once the user has selected a tree, the desired payment is taken. It is assumed that the user has an account, is successfully logged in and is on the dashboard.



Section C - Software Architecture Style, Modelling and Evaluation (Unit 3 and Unit 4)

C1: Component Diagrams

Diagram 1: Client-server architecture.

In a Client-Server Architecture, the client (usually a mobile or web application) communicates with a server that hosts the backend logic, database interactions, and other centralized services.

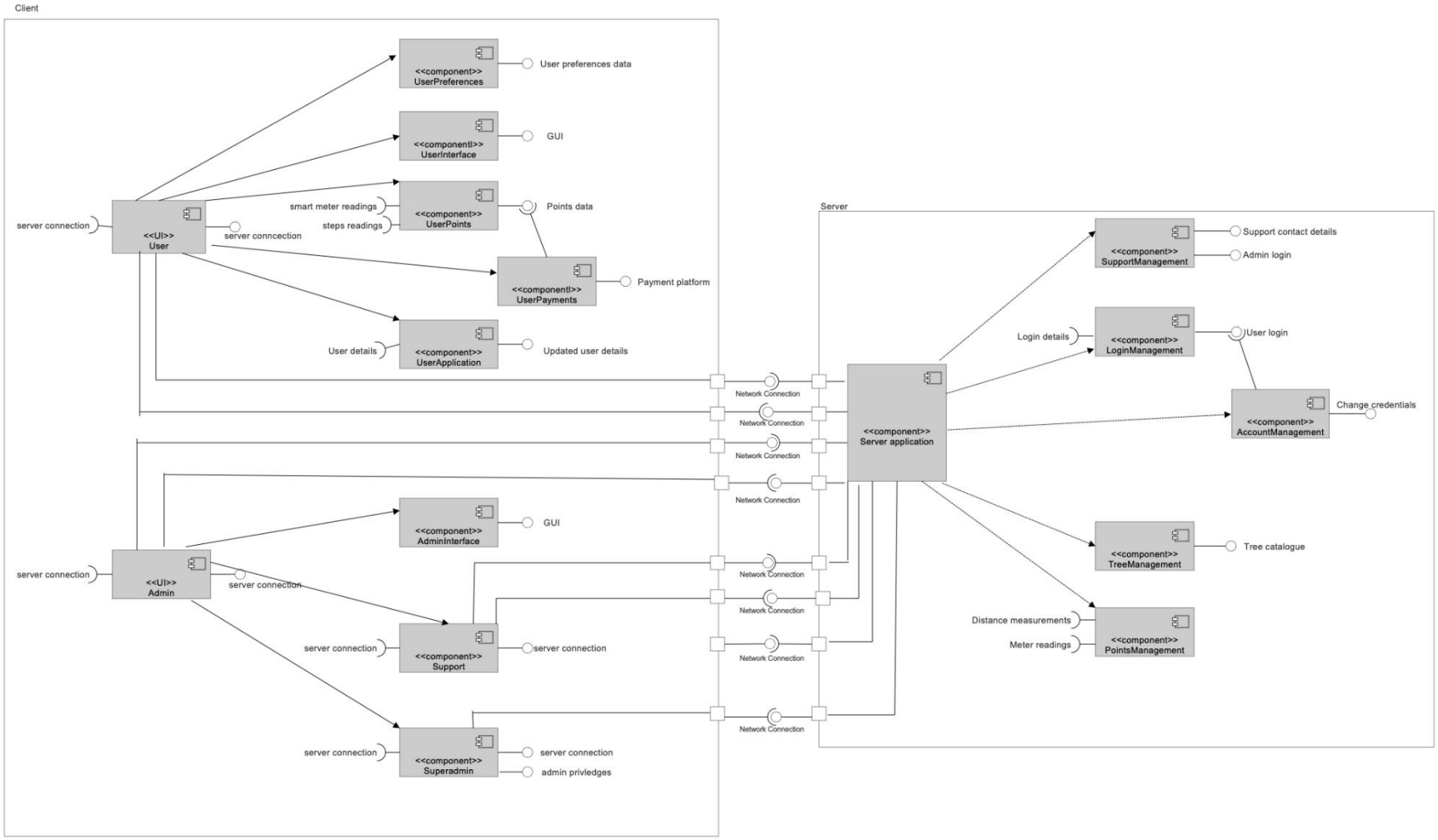
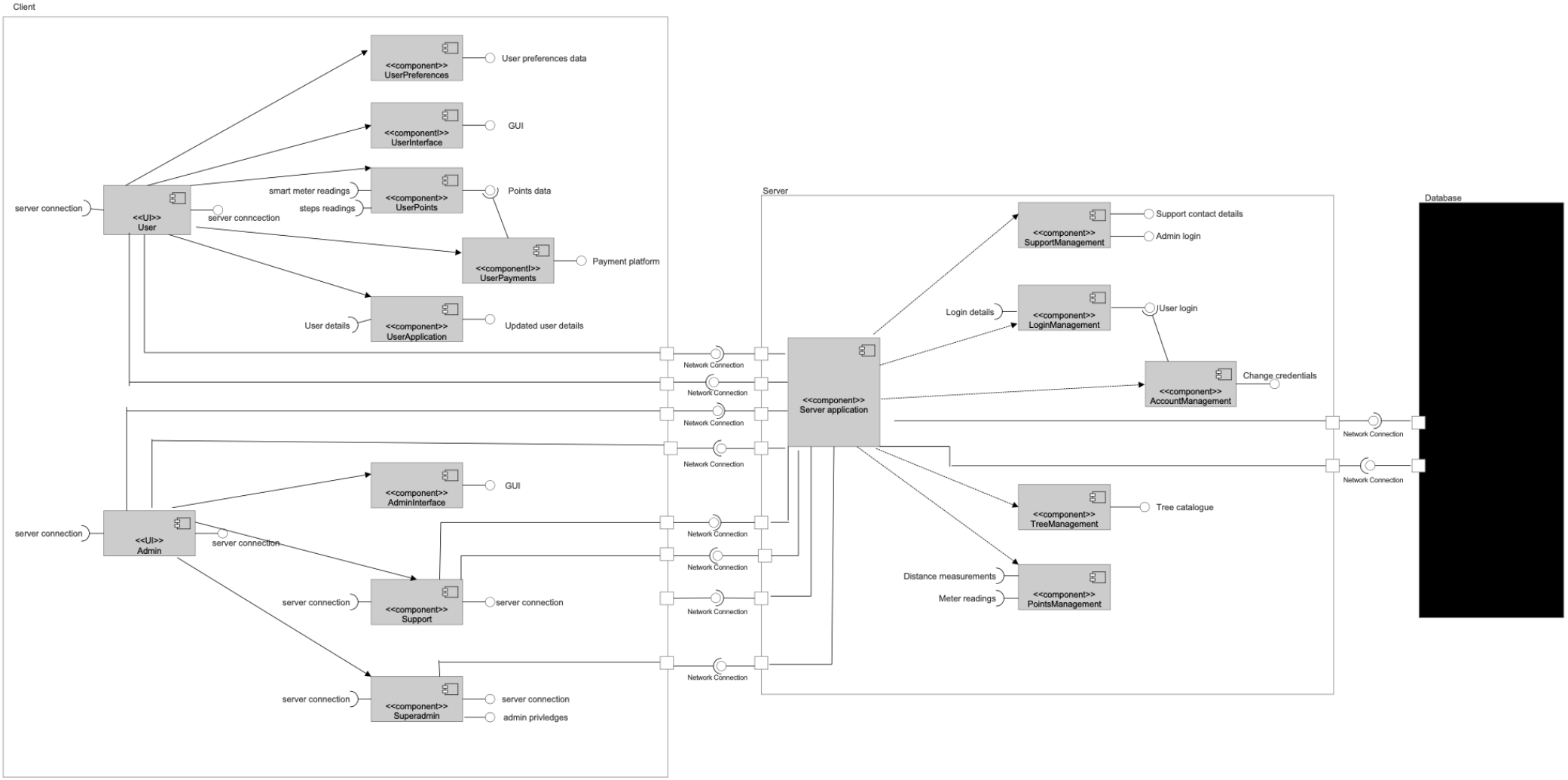


Diagram 2: Three tier architecture.

A three-tier architecture consists of a client layer, a server layer, and a database layer. In this setup, the client sends requests to the server, which processes the requests and interacts with the database to retrieve or store data. **Note:** the database is outsourced



C2: Deployment Diagrams

Diagram 1: Client-server architecture.

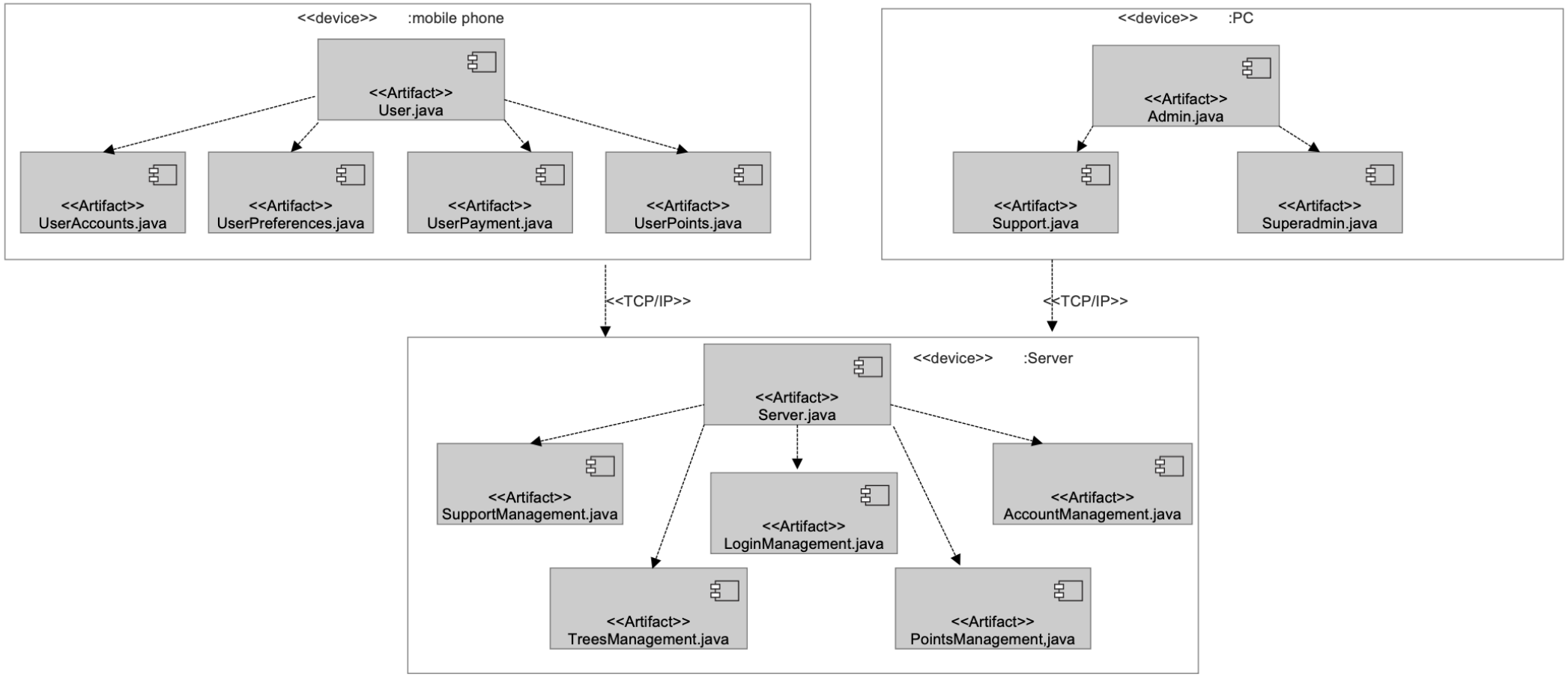
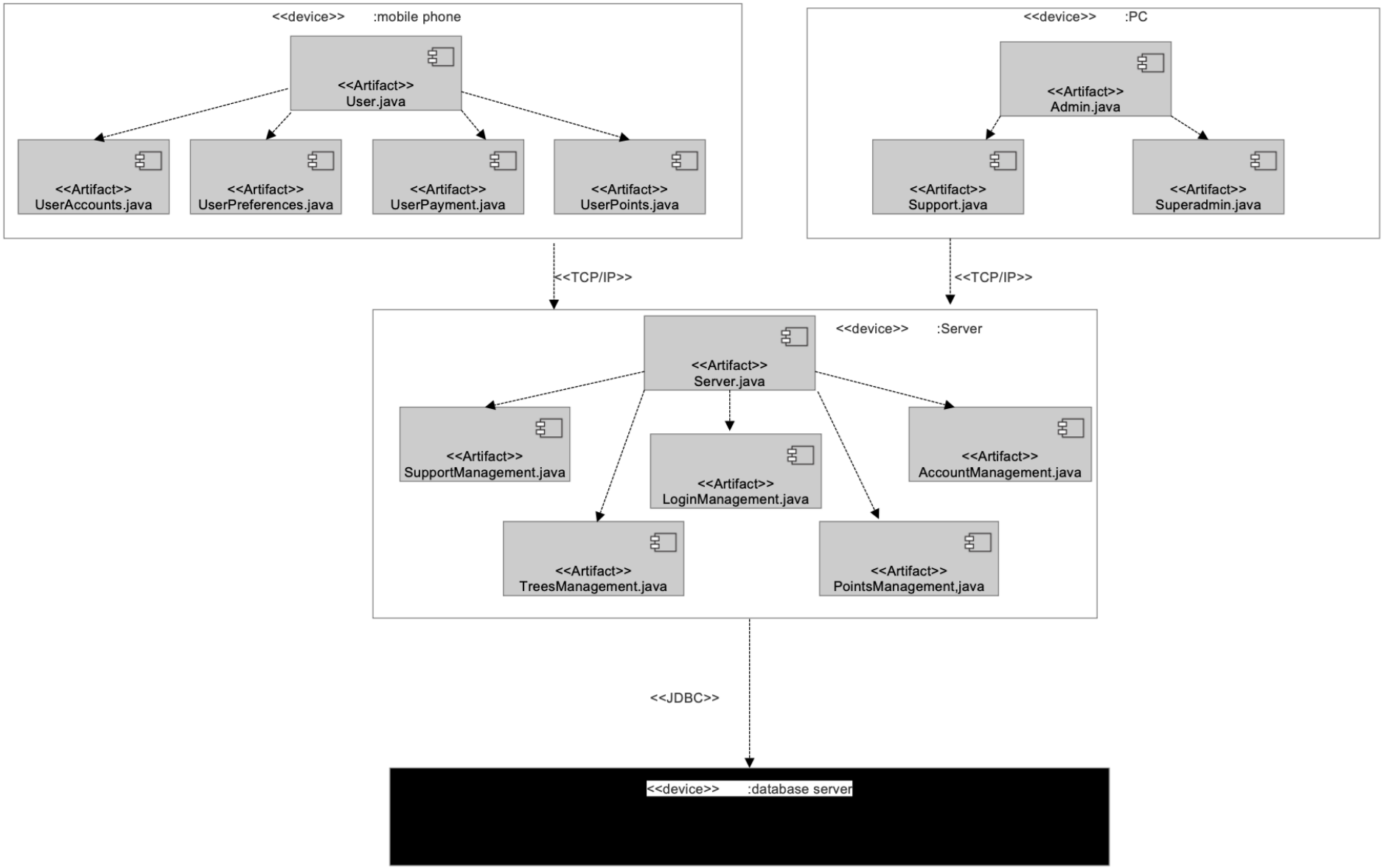


Diagram 2: Three tier architecture.
Note: the database is outsourced.



C3: Comparison

We are integrating two pivotal architectural models in software design: the Client-Server architecture and the Three-Tier architecture. These models are foundational in structuring and organizing software systems, each presenting distinct advantages and considerations.

The client-server architecture is a simpler model where two main components interact: the client, which requests services, and the server, which processes these requests. Its simplicity makes it easier to set up and manage. Centralisation of resources and data management in the server simplifies maintenance and updates. However, this centralisation also means that clients heavily depend on the server, creating potential bottlenecks, especially as the user base grows. Moreover, the centralised data storage poses significant security risks if the server is compromised. While the client-server model is scalable to a point, its efficiency decreases with massive user growth, making it less ideal for very large-scale applications.

In contrast, the three-tier architecture enhances scalability and performance by distributing the workload across three distinct tiers. It offers flexibility and maintainability, as each layer can be developed and maintained independently. This separation also enhances security by isolating the data layer from the client. However, this architecture is more complex to design, implement, and maintain than the client-server model. It incurs potentially higher initial development and ongoing maintenance costs due to its complexity and additional resources. Additionally, the extra layer can introduce performance overhead, especially if not optimally designed.

For our app, with our complex integrations and a broad scope, the three-tier architecture emerges as the ideal choice. Its ability to separate concerns across different layers allows for efficient management of diverse functionalities, from user interactions to data analytics. This architecture enhances scalability, a vital feature for an app designed to attract a large and environmentally conscious user base. Additionally, the inherent security benefits of the three-tier model, crucial for handling sensitive payment processing and user data, align perfectly with the app's requirements. While this approach may demand a higher initial investment in development and maintenance, the long-term advantages of scalability, security, and flexibility justify this choice, making it the most suitable architecture for the ambitious and multifaceted nature of our app.

Section D – Software Testing (Unit 5)

Test Plan

1. Introduction

TreeConnect is an innovative platform committed to encouraging users to adopt sustainable practices and raise their level of environmental consciousness. The software encourages users to walk, cycle, and to use energy saving methods, with a primary focus on increasing awareness and sponsoring trees to be planted. Users are rewarded points for their environmentally friendly activities, which they can use to sponsor trees. Also, users have the opportunity to sponsor trees using money. Additionally, we utilize data analytics algorithm for users to see their statistics breakdown and monitor their carbon offset as well recommending them sponsor opportunities based on their app use behaviour. TreeConnect wants to foster a meaningful, community-driven approach and sense of responsibility.

Project Name: TreeConnect
Document version: 1.0
Document Owner: Group 25
Date: 01/12/2023

2. Testing Objectives

1) Compatibility testing – ensure that the system functions correctly across different devices/operating systems
2) Performance testing – assess the systems buttons responsiveness
3) Functional testing – ensure that the user can register with a valid email and password, login, change name, filter tree catalogue by cost and sponsor a tree using their points and a valid card

3. Features to be Tested

- Functional requirements:

1) Feature: User registration – ensure that an account can only be created if the password meets the 4 password requirements and if an email not already associated with an account is used
2) Feature: Login Functionality – ensure that an account can be accessed using only valid credentials (email and password)
3) Feature: Change personal details – ensure that the user can update their name only if a valid input is provided
4) Feature: Earning points by cycling – confirm that the user is awarded the correct number of points for cycling
5) Feature: Filtering tree catalogue – verify that relevant trees are displayed based on the chosen cost
6) Feature: Sponsoring using points – ensure that a user can sponsor a tree if the user has enough points
7) Feature: Sponsoring using card – ensure that the user can sponsor a tree using a valid card

- Non-Functional requirements:

1) Feature: Compatibility, Evaluate the portability of the application by ensuring it runs on iOS 10 and above and Android 10 and above
2) Feature: Response time of app – verify that the system takes less than 3 seconds to open
3) Feature: Response time of buttons – verify that the system takes less than a second to respond when either the 'Sponsor', 'Account', 'Support' or 'Points' button is pressed from the dashboard

4. Test Strategy

We will employ a diverse array of testing strategies to comprehensively assess the app, ensuring a thorough and holistic testing approach that covers a broad spectrum of scenarios.

Black-box testing:
Black-box testing will be conducted for **User Registration** and **Login**, **Change Personal Details**, and **Earning Points.** This approach will carefully examine the functionality based on inputs and outputs without getting into the specifics of the internal code. The goal is to make sure that these parts meet the standards and operate as intended.

White-box testing:
We will use **White-box** testing for **Sponsoring Trees, App Response Time**, **Security** and **Compatibility with Different Devices**. This method entails a thorough examination of the code routes, security protocols, and internal organisation of the application. To guarantee a reliable and safe application, it seeks to detect any potential weaknesses in the code design, evaluate response times, and locate security vulnerabilities.

Unit Testing:
We will use **Unit testing** for **User Registration** and **Login**, **Changing personal information**, **Earning points and Filter functionality** when Browsing Trees Catalogues. This will guarantee correctness and dependability, it will concentrate on individual components and independently test their performance.

Integration Testing:
Integration testing will be done on **Sponsoring both money and Points** to evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.

System Testing:
To test the overall functionality of the system and ensure that it adheres to the intended design and specified requirements, a thorough system testing phase will cover all features with Various Devices.

Acceptance Testing:
This test will be carried out on all features to check whether the application successfully satisfies the needs of the user and meets the specified requirements.

Performance Testing:
The primary goal of this test is to test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing will be done under various network loads and conditions.

Security Testing:
To evaluate security measures, thorough security testing will be carried out with the goal of locating and resolving any potential vulnerabilities and guaranteeing strong user data protection. For example, to ensure that passwords are strong enough and encrypted when storing, each email is only and only linked to one account and there is a security layer provided during checkout when using money to sponsor.

Usability Testing:
This test will concentrate on ensuring a user-friendly experience across various functionalities to achieve user friendliness.

5. Test Cases

See table below.

6. Schedule

Test Preparation: December 15, 2023- December 25, 2023
Testing Execution: December 25, 2023- December 30, 2023
Defect Reporting: December 30, 2023- December 31, 2023
Test Summary Report: January 6, 2024

7. Test Completion Criteria

Upon completion of testing, all critical defects found during the assessment have been addressed and retested, ensuring their successful resolution. The application aligns well with the predetermined acceptance criteria, meeting both functional and non-functional requirements as specified in the project guidelines. Comprehensive test coverage guarantees the thorough examination of all vital functionalities within the application. Furthermore, the application exhibits stability, reliability, and satisfactory performance across various scenarios and workloads, therefore concluding that the test coverage meets all the predefined criteria, and the app is ready for the next phase.

8. Assumptions

Ash tree costs £10/10,000 points to sponsor.
Ash and Oak trees are trees that are available for sponsorship on the app and can be found using the search feature.
There is no saved payment method on the account.
yanni@gmail.com is the email address of a registered user of the app.
All empty columns are to be filled during/after testing.

9. Appendix

Not Applicable.

Test Case ID	Test Scenario	Test Case Title	Pre-requisites	Test Strategy	Test Steps	Test Data	Expected Results (ER)	Actual Result	Priority	Result	Comment
TC_UR_001	(TS_001) User Registration	Verify user registration with a **valid email** (email not already in use) and **valid password** (8 characters long, containing at least 1 capital, 1 lower and 1 special character)	1 – Have the app downloaded and open on a compatible device 2 – **Valid email**, **valid personal information** (name, date of birth, phone number) and a **valid password** required	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – Press "Sign up" button 2 – **Enter valid email** address into the 'Email address' field <Refer Test Data> 3 – **Enter valid password** into the 'Password' field <Refer Test Data> 4 - **Enter valid name** into the 'Name' field <Refer Test Data> 5 – **Enter valid date of birth** into the 'Date of Birth' field <Refer Test Data> 6 – **Enter valid phone number** into the 'Phone Number' field <Refer Test Data> 7 – Press "Sign up" button (Verify ER)	**Email:** johnsmith@gmail.com **Name:** John Smith **Date of Birth:** 23-04-1988 **Phone number:** 07829829606 **Password:** 2Qq5aXr!	1 – User should be **navigated to the signup page** 2 – User should be **navigated to the verification page** where they are required to input 4-digit code				
TC_UR_001	(TS_001) User Registration	Verify user registration with an **invalid email** (**email already in use**)	1 – Have the app downloaded and open on a compatible device	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – Press "Sign up" button 2 – **Enter invalid email address** into the 'Email address' field <Refer Test Data> (Verify ER)	**Email:** yanni@gmail.com	As soon as the email is entered, a **message** with the text '**Email already in use**' should be displayed				
TC_UR_001	(TS_001) User Registration	Verify user registration with an **invalid password** (8 characters long, containing at least 1 lower and 1 special character but **missing capital letter**)	1 – Have the app downloaded and open on a compatible device 2 – **Valid email required**	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – Press "Sign up" button 2 – **Enter valid email address** into the 'Email address' field <Refer Test Data> 4 – **Enter invalid password** into the 'Password' field <Refer Test Data> (Verify ER)	**Email:** johnsmith@gmail.com **Password:** 2qq5axr!	As soon as the invalid password is entered a **message** with the text '**Invalid password: Missing Capital Letter**' should be displayed				

TC_UR_001	(TS_001) User Registration	Verify user registration with an **invalid password** (8 characters long, containing a least 1 capital and 1 special character but **missing lowercase letter**)	1 – Have the app downloaded and open on a compatible device 2 – **Valid email required**	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – Press "Sign up" button 2 – **Enter valid email address** into the 'Email address' field <Refer Test Data> 4 – **Enter invalid password** into the 'Password' field <Refer Test Data> (Verify ER)	**Email:** johnsmith@gmail.com **Password:** 2QQ5AXR!	As soon as the invalid password is entered a **message** with the text '**Invalid password: Missing Lowercase Letter**' should be displayed				
TC_UR_001	(TS_001) User Registration	Verify user registration with an **invalid password** (containing at least 1 capital, 1 lower and 1 special character but **not 8 characters long**)	1 – Have the app downloaded and open on a compatible device 2 – **Valid email required**	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – Press "Sign up" button 2 – **Enter valid email address** into the 'Email address' field <Refer Test Data> 4 – **Enter invalid password** into the 'Password' field <Refer Test Data> (Verify ER)	**Email:** johnsmith@gmail.com **Password:** 2Qq5	As soon as the invalid password is entered a **message** with the text '**Invalid password: Not 8 Characters Long**' should be displayed				
TC_UR_001	(TS_001) User Registration	Verify user registration with an **invalid password** (8 characters long, containing at least 1 capital and 1 lowercase character but **missing special character**)	1 – Have the app downloaded and open on a compatible device 2 – Valid email required	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – Press "Sign up" button 2 – **Enter valid email address** into the 'Email address' field <Refer Test Data> 4 – **Enter invalid password** into the 'Password' field <Refer Test Data> (Verify ER)	**Email:** johnsmith@gmail.com **Password:** 2Qq5aXrm	As soon as the invalid password is entered a **message** with the text '**Invalid password: Missing Special Character**' should be displayed				
TC_LF_002	(TS_002) Login Functionality	Verify logging into the app using **valid credentials** (i.e. valid email and valid password)	1 – Have the app downloaded and open on a compatible device	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Security Testing:** locate and resolving any potential	1 – **Enter valid email address** into the 'Email address' field <Refer Test Data> 2 – **Enter valid password** into the 'Password' field <Refer Test Data>	**Email:** johnsmith@gmail.com **Password:** 2Qq5aXr!	User should be **logged** in and **taken** to the **dashboard**				

			2 – **Login credentials** for an **existing account are required**	vulnerabilities and guaranteeing strong user data protection	3 – Press 'Login' button (Verify ER)						
TC_LF_002	(TS_002) Login Functionality	Verify logging into the app using **invalid credentials** (i.e. **invalid email** and **invalid password**)	1 – Have the app downloaded and open on a compatible device	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – **Enter invalid email address** into the 'Email address' field <Refer Test Data> 2 – **Enter invalid password** into the 'Password' field <Refer Test Data> 3 – Press 'Login' button (Verify ER)	**Email:** yannilovita@gmail.com **Password:** On$o98x!	Warning **message** with the text '**Error: Invalid email and/or Password**' should be displayed				
TC_LF_002	(TS_002) Login Functionality	Verify logging into the app using **invalid credentials** (i.e. **valid email** but **invalid password**)	1 – Have the app downloaded and open on a compatible device 2 – **Email address for an existing account required**	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – **Enter valid email address** into the 'Email address' field <Refer Test Data> 2 – **Enter invalid password** into the 'Password' field <Refer Test Data> 3 – Press 'Login' button (Verify ER)	**Email:** johnsmith@gmail.com **Password:** On$o98x!	Warning **message** with the text '**Error: Invalid email and/or Password**' should be displayed				
TC_LF_002	(TS_002) Login Functionality	Verify logging into the app using **invalid credentials** (i.e. **valid password** but **invalid email**)	1 – Have the app downloaded and open on a compatible device 2 – **Password for an existing account required**	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Security Testing:** locate and resolving any potential vulnerabilities and guaranteeing strong user data protection	1 – **Enter invalid email address** into the 'Email address' field <Refer Test Data> 2 – **Enter valid password** into the 'Password' field <Refer Test Data> 3 – Press 'Login' button (Verify ER)	**Email:** yannilovita@gmail.com **Password:** 2Qq5aXr!	Warning **message** with the text '**Error: Invalid email and/or Password**' should be displayed				
TC_UN_003	(TS_003) Updating Name	Verify updating name by using a **valid name** (**letters only**)	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Account' page	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Acceptance Testing**: To ensure the specified requirements are met. **Usability Testing:** To ensure a user-friendly experience.	1 – Press 'Edit Profile' button 2 – **Enter valid name** into the 'name' field <Refer Test Data> 3 – Press 'Update' button (Verify ER)	**Name:** John Smith	1 – User should be navigated to the 'Profile' page 2 – User should be navigated to the 'Account' page and the **updated name should be displayed at the top of page**				

TC_UN_003	(TS_003) Updating Name	Verify updating name by using an **invalid name** (**containing symbols and numbers**)	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Account' page	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance. **Acceptance Testing**: To ensure the specified requirements are met. **Usability Testing:** To ensure a user-friendly experience.	1 – Press 'Edit Profile' button 2 – **Enter invalid name** into the 'name' field <Refer Test Data> 3 – Press 'Update' button (Verify ER)	Name: J0hn 5m!th	1 – User should be navigated to the 'Profile' page 2 – Warning **message** with the text '**Invalid name**' should be displayed				
TC_CP_004	(TS_004) Earning points by cycling	Verify that the correct number of points are awarded when cycling for **integer distances**	1 – Have the app downloaded on a compatible **device** that **has google maps and be logged in** 2 – **Ensure that location permissions are given to the app** 3 – A route which is **50km** long and is suitable for cycling is required	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Integration Testing:** to evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.	1 – Secure the device on the user 2 – Cycle the route provided <Refer Test Data> 3 – Check the notifications from the app (Verify ER)	A route which is **50km** long and is suitable for cycling	**Notification** with the text '**Congratulations earned 500 points by cycling 50km**' should be received				
TC_CP_004	(TS_004) Earning points by cycling	Verify that the correct number of points are awarded when cycling for **decimal distances**	1 – Have the app downloaded on a compatible **device** that **has google maps and be logged in** 2 – **Ensure that location permissions are given to the app** 3 – A route which is **35.52km** long and is suitable for cycling is required	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Integration Testing:** to evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.	1 – Secure the device on the user 2 – Cycle the route provided <Refer Test Data> 3 – Check the notifications from the app (Verify ER)	A route which is **35.52km** long and is suitable for cycling	**Notification** with the text '**Congratulations earned 355.2 points by cycling 35.52km**' should be received				
TC_FT_005	(TS_005) Filtering Tree Catalogue by Cost	Verify that all **relevant trees are displayed** when a filter of a **maximum valid cost of £15** is applied	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials**	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components	1 – Press the filter button 2 – **Choose to filter by 'Cost'** 3 – **Choose to filter by 'GBP'** 4 – **Enter valid cost** in 'Maximum' field <Refer Test Data>	**Cost:** 15	1 – Filter options should be displayed 2 – **All and only** trees **costing more than £15** should be **removed** from the tree catalogue				

			3 – Navigate to the 'Sponsor' page	to independently test its performance.	5 – Press 'Apply' button (Verify ER)						
TC_FT_005	(TS_005) Filtering Tree Catalogue by Cost	Verify that all relevant trees are **displayed** when a filter of a **minimum valid cost of £5** is applied	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Sponsor' page	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance.	1 – Press the filter button 2 – **Choose to filter by 'Cost'** 3 – **Choose to filter by 'GBP'** 4 – **Enter valid cost** in 'Minimum' field <Refer Test Data> 5 – Press 'Apply' button (Verify ER)	**Cost:** 5	1 – Filter options should be displayed 2 – **All and only** trees **costing less than £5** should be **removed** from the tree catalogue				
TC_FT_005	(TS_005) Filtering Tree Catalogue by Cost	Verify that all relevant trees are **displayed** when a filter of a **maximum valid cost of 16,000 points** is applied	1 – Have the app downloaded and open on a compatible device 2 – Login using existing credentials 3 – Navigate to the 'Sponsor' page	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance.	1 – Press the filter button 2 – **Choose to filter by 'Cost'** 3 – **Choose to filter by 'Points'** 4 – **Enter valid cost** in 'Maximum' field <Refer Test Data> 5 – Press 'Apply' button (Verify ER)	**Cost:** 16000	1 – Filter options should be displayed 2 – **All and only** trees **costing more than 16,000 points** should be **removed** from the tree catalogue				
TC_FT_005	(TS_005) Filtering Tree Catalogue by Cost	Verify that all relevant trees are **displayed** when a filter of a **minimum valid cost of 6,000** points is applied	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Sponsor' page	**Black-box Testing:** This will carefully examine the functionality based on inputs and outputs. **Unit testing:** concentrate on individual components to independently test its performance.	1 – Press the filter button 2 – **Choose to filter by 'Cost'** 3 – **Choose to filter by 'Points'** 4 – **Enter valid** cost in 'Minimum' field <Refer Test Data> 5 – Press 'Apply' button (Verify ER)	**Cost:** 6000	1 – Filter options should be displayed 2 – **All and only** trees **costing less than 6,000 points** should be **removed** from the tree catalogue				
TC_SP_006	(TS_006) Sponsoring Tree using Points	Verify sponsoring a tree using **valid number of points**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Sponsor' page 4 – **10,000 points are required** 5 – **Existing tree costing maximum 10,000 points required**	**White-box Testing: To** thoroughly examine the internal organisation of the application **Integration Testing:** to evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.	1 – **Enter tree name** into 'Search' field <Refer Test Data> 2 – Press on the button having search icon 3 – Press on the searched tree and press 'Sponsor' button 4 – **Select 'Points'** as the **payment method** 5 – Press 'Sponsor' button	**Tree:** Ash	1 – **Searched tree** should be **displayed** in the search result 2 – Further **information** about the tree should be **displayed** along with a '**Sponsor**' **button** 3 – User should be **navigated** to the **checkout page** 4 – User should be **navigated** to the **confirmation page** and a **message** with text '**Thank you**' should be displayed				

TC_SP_006	(TS_006) Sponsoring Tree using Points	Verify sponsoring a tree using **invalid number of points**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Sponsor' page 4 – **Account is required to have less than 10,000 points** 5 – **Existing tree costing 10,000 points required**	**White-box Testing: To** thoroughly examine the internal organisation of the application **Integration Testing:** to evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.	1 – **Enter tree name** into 'Search' field <Refer Test Data> 2 – Press on the button having search icon 3 – Press on the searched tree and press 'Sponsor' button 4 – **Select 'Points' as the payment method** 5 – Press 'Sponsor' button	**Tree:** Ash	1 – **Searched tree** should be **displayed** in the search result 2 – Further **information** about the tree should be **displayed** along with a '**Sponsor**' **button** 3 – User should be **navigated** to the **checkout page** 4 – Warning **message** with the text '**Error: Not enough points**' should be displayed				
TC_SC_007	(TS_007) Sponsoring Tree using Card	Verify sponsoring a tree using a **valid card**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – Navigate to the 'Sponsor' page 4 – **Card details of a valid card are required**	**White-box Testing:** To thoroughly examine the internal organisation of the application **Integration Testing:** To evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.	1 – **Enter tree name** into 'Search' field <Refer Test Data> 2 – Press on the button having search icon 3 – Press on the searched tree and press 'Sponsor' button 4 – **Select card as the payment method** 5 – **Enter valid card** number into 'Card Number' field <Refer Test Data> 6 – **Enter valid expiry date** into 'Expiry' field <Refer Test Data> 7 – **Enter valid CVV** into 'CVV' field <Refer Test Data> 8 – Press 'Save' button 9 – Press 'Sponsor' button (Verify ER)	**Tree:** Oak **Card number:** 1857 0925 0266 0638 **Expiry:** 01/30 **CVV:** 834	1 – **Searched tree** should be **displayed** in the search result 2 – Further **information** about the tree should be **displayed** along with a '**Sponsor**' **button** 3 – User should be **navigated** to the **checkout page** 4 **– Pop up window** will be **displayed** asking for card details 5 – **Pop up window** will **close** and a **message** with the text '**Payment Method Saved**' should be displayed 6 – User should be **navigated** to the **confirmation page** and a **message** with text '**Thank you**' should be displayed				
TC_SC_007	(TS_007) Sponsoring Tree using Card	Verify sponsoring a tree using an **invalid card** (**expired card**)	1 – Have the app downloaded and open on a compatible device	**White-box Testing:** To thoroughly examine the internal organisation of the application	1 – **Enter tree name** into 'Search' field <Refer Test Data> 2 – Press on the button having search icon	**Tree:** Oak **Card number:** 1857 0925 0266 0638 **Expiry:** 01/12 **CVV:** 834	1 – **Searched tree** should be **displayed** in the search result 2 – Further **information** about the				

			2 – **Login using existing credentials** 3 – Navigate to the 'Sponsor' page	**Integration Testing:** To evaluate how well various modules and functions work together. This will ensure that these functionalities operate and communicate with each other without problems.	3 – Press on the searched tree and press 'Sponsor' button 4 – **Select card as the payment method** 5 – **Enter valid card number** into 'Card Number' field <Refer Test Data> 6 – **Enter invalid expiry date** into 'Expiry' field <Refer Test Data> 7 – **Enter valid CVV** into 'CVV' field <Refer Test Data> 8 – Press 'Save' button (Verify ER)		tree should be **displayed** along with a '**Sponsor**' **button** 3 – User should be **navigated** to the **checkout page** 4 – **Pop up window** will be **displayed** asking for card details 5 – Warning **message** with the text '**Invalid Payment Method**' should be displayed				
TC_AP_008	(TS_008) App Portability	Verify app **runs** on **iOS 10 and above**	1 – Have the app **downloaded** on an **iOS device** with the **operating system version 10 or above**	**White-box Testing:** To thoroughly examine the internal organisation of the application **System Testing:** To test the overall functionality of the system and ensure that it adheres to the intended design and specified requirements, a thorough system testing phase will cover all features with Various Devices.	1 – Open the app 2 – **Verify** that the app **launches without errors** 3 – **Navigate** through **various features** and **functionalities** within the app 4 – **Check** for any **visual anomalies** or **layout issues specific** to the **iOS environment** 5 – Close the app (Verify ER)	Not Applicable	1 – The **app should run smoothly on iOS devices** with **version 10 and above** 2 – All **features** and **functionalities** should be **accessible** and **responsive** 3 – **No visual anomalies** or **layout issues** should be **observed**				
TC_AP_008	(TS_008) App Portability	Verify app **runs** on **Android 10 and above**	1 – Have the app **downloaded** on an **Android device** with the **operating system version 10 or above**	**White-box Testing:** To thoroughly examine the internal organisation of the application **System Testing:** To test the overall functionality of the system and ensure that it adheres to the intended design and specified requirements, a thorough system testing phase will cover all features with Various Devices.	1 – Open the app 2 – **Verify** that the app **launches without errors** 3 – **Navigate** through **various features** and **functionalities** within the app 4 – **Check** for any **visual anomalies** or **layout issues specific** to the **Android environment** 5 – Close the app (Verify ER)	Not Applicable	1 – The **app should run smoothly on Android devices** with **version 10 and above** 2 – All **features** and **functionalities** should be **accessible** and **responsive** 3 – **No visual anomalies** or **layout issues** should be **observed**				
TC_RA_009	(TS_009) Responsive Time of App	Verify that it takes **less than 3 seconds**	1 – Have the app downloaded on a	**White-box Testing:** To thoroughly examine the	1 – **Open** the app and **start** the **stopwatch** at the **same time**	Not Applicable	All **ten** stopwatch **readings** should be **less than 3 seconds**				

		to **open** the app on an **Android device**	compatible Android device 2 – A **stopwatch** is **required**	internal organisation of the application **Performance Testing**: To test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing should be done under various network loads and conditions.	2 – **Stop** the **stopwatch** as soon as the **login page loads** and **note** the **time** 3 – **Close** the **app** and **reset** the **stopwatch** 4 – **Repeat steps 1 to 3 ten** times (Verify ER)						
TC_RA_009	(TS_009) Responsive Time of App	Verify that it takes **less than 3 seconds** to **open** the app on an **iOS device**	1 – Have the app downloaded on a compatible iOS device 2 – A **stopwatch** is **required**	**White-box Testing:** To thoroughly examine the internal organisation of the application **Performance Testing**: To test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing should be done under various network loads and conditions.	1 – **Open** the app and **start** the **stopwatch** at the **same time** 2 – **Stop** the **stopwatch** as soon as the **login page loads** and **note** the **time** 3 – **Close** the **app** and **reset** the **stopwatch** 4 – **Repeat steps 1 to 3 ten** times (Verify ER)	Not Applicable	All **ten** stopwatch **readings** should be **less than 3 seconds**				
TC_RB_010	(TS_010) Response Time of Buttons	Verify **response time** of '**Sponsor**' **button** is **less than 1 second**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – A **stopwatch** is **required**	**White-box Testing:** To thoroughly examine the internal organisation of the application **Performance Testing**: To test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing should be done under various network loads and conditions.	1 – **Press the 'Sponsor' button** and **start** the **stopwatch** at the **same time** 2 – **Stop** the **stopwatch** as soon as the **sponsorship page loads** and **note** the **time** 3 – **Go back on the dashboard and reset the stopwatch** 4 – **Repeat steps 1 to 3 ten times** (Verify ER)	Not Applicable	All **ten** stopwatch **readings** should be **less than 1 second**				
TC_RB_010	(TS_010) Response Time of Buttons	Verify **response time** of '**Account**' **button** is **less than 1 second**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – A **stopwatch** is **required**	**White-box Testing:** To thoroughly examine the internal organisation of the application **Performance Testing**: To test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing should be done	1 – **Press the 'Account' button** and **start** the **stopwatch** at the **same time** 2 – **Stop** the **stopwatch** as soon as the **account page loads** and **note** the **time** 3 – **Go back on the dashboard and reset the stopwatch** 4 – **Repeat steps 1 to 3 ten times** (Verify ER)	Not Applicable	All **ten** stopwatch **readings** should be **less than 1 second**				

				under various network loads and conditions.							
TC_RB_010	(TS_010) Response Time of Buttons	Verify **response time** of '**Support**' **button** is **less than 1 second**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – A **stopwatch** is **required**	**White-box Testing:** To thoroughly examine the internal organisation of the application **Performance Testing**: To test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing should be done under various network loads and conditions.	1 – **Press the 'Support' button** and **start** the **stopwatch** at the **same time** 2 – **Stop** the **stopwatch** as soon as the **support page loads** and **note** the **time** 3 – **Go back on the dashboard and reset the stopwatch** 4 – **Repeat steps 1 to 3 ten times** (Verify ER)	Not Applicable	All **ten** stopwatch **readings** should be **less than 1 second**				
TC_RB_010	(TS_010) Response Time of Buttons	Verify **response time** of '**Points**' **button** is **less than 1 second**	1 – Have the app downloaded and open on a compatible device 2 – **Login using existing credentials** 3 – A **stopwatch** is **required**	**White-box Testing:** To thoroughly examine the internal organisation of the application **Performance Testing**: To test the App's Response Time to guarantee peak performance. To verify speed and responsiveness, testing should be done under various network loads and conditions.	1 – **Press the 'Points' button** and **start** the **stopwatch** at the **same time** 2 – **Stop** the **stopwatch** as soon as the **points page loads** and **note** the **time** 3 – **Go back on the dashboard and reset the stopwatch** 4 – **Repeat steps 1 to 3 ten times** (Verify ER)	Not Applicable	All **ten** stopwatch **readings** should be **less than 1 second**				

Section E – Usability and Prototyping (Unit 6)

E1: Prototype

Figure 1: Login page

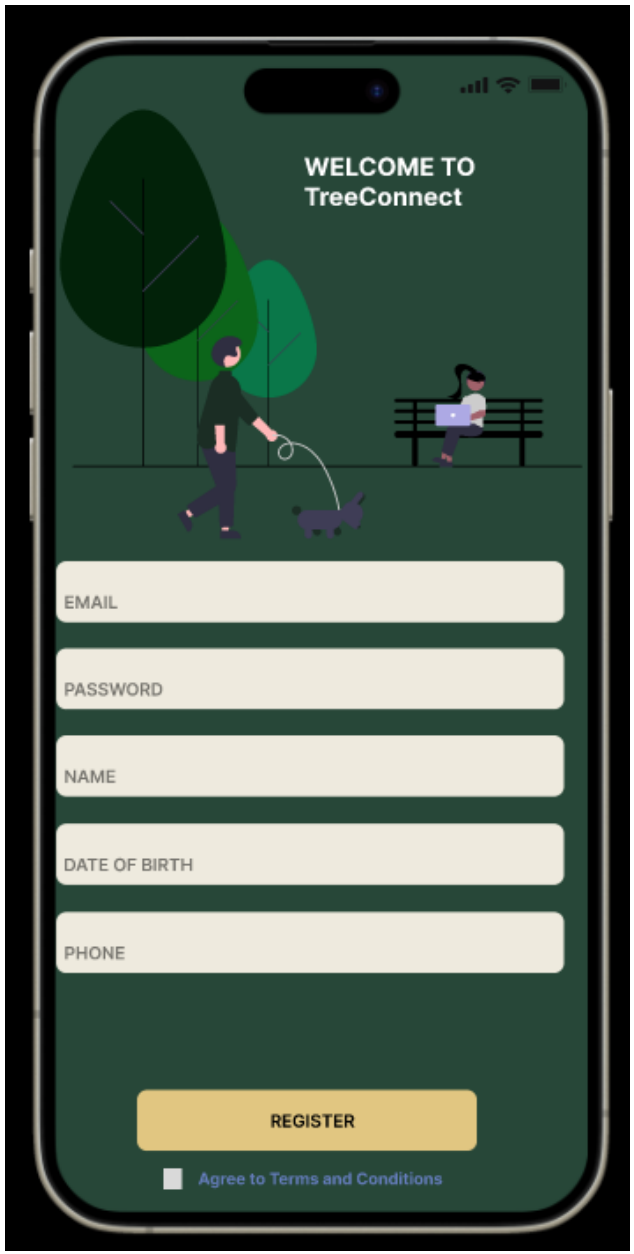


Figure 2: Sign up page

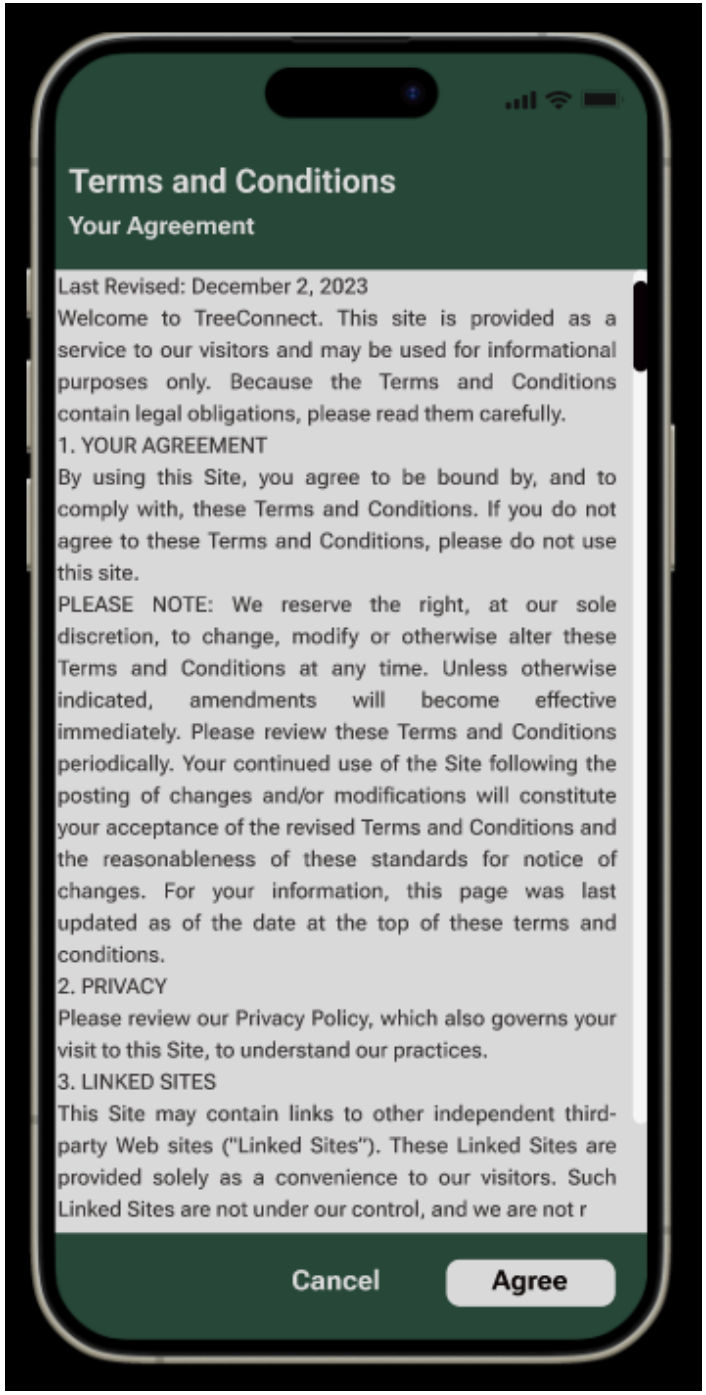


Figure 3: Terms and Conditions page

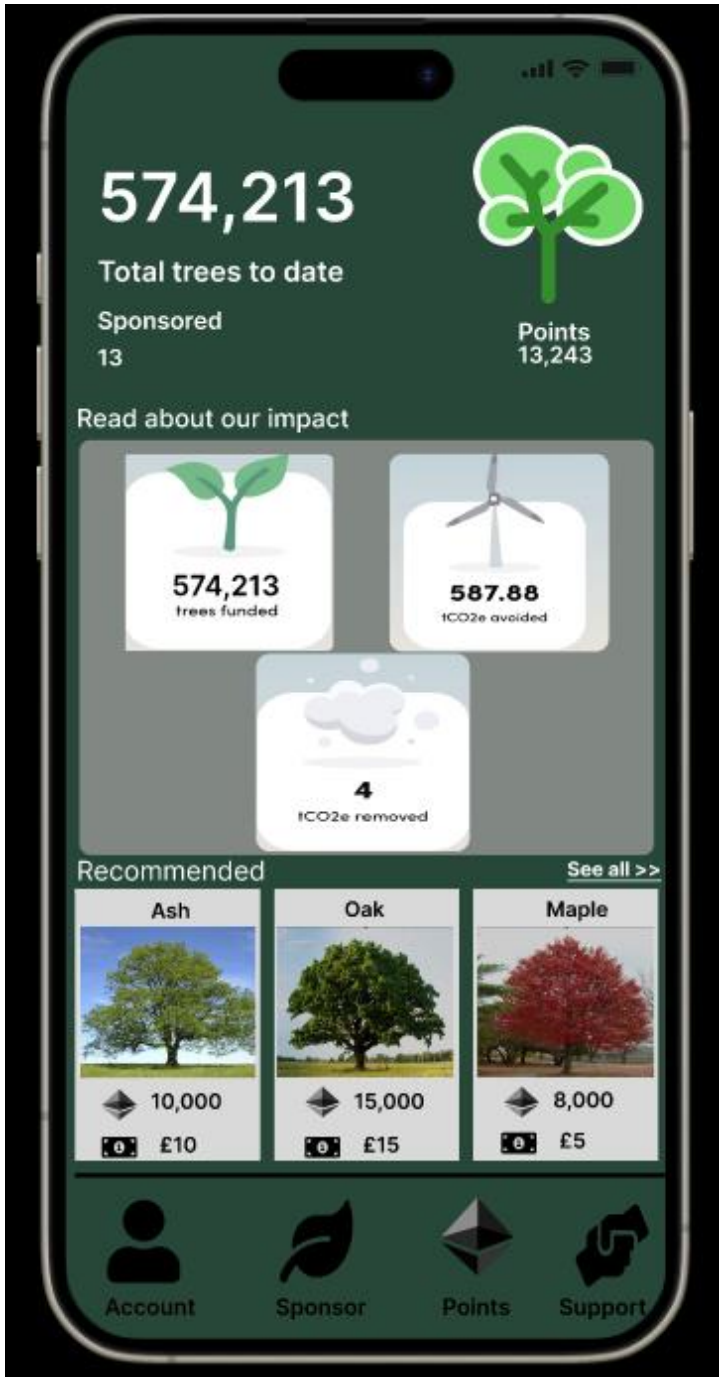


Figure 4: Dashboard

Figure 5: Sponsorship page (Containing Tree Catalogue)

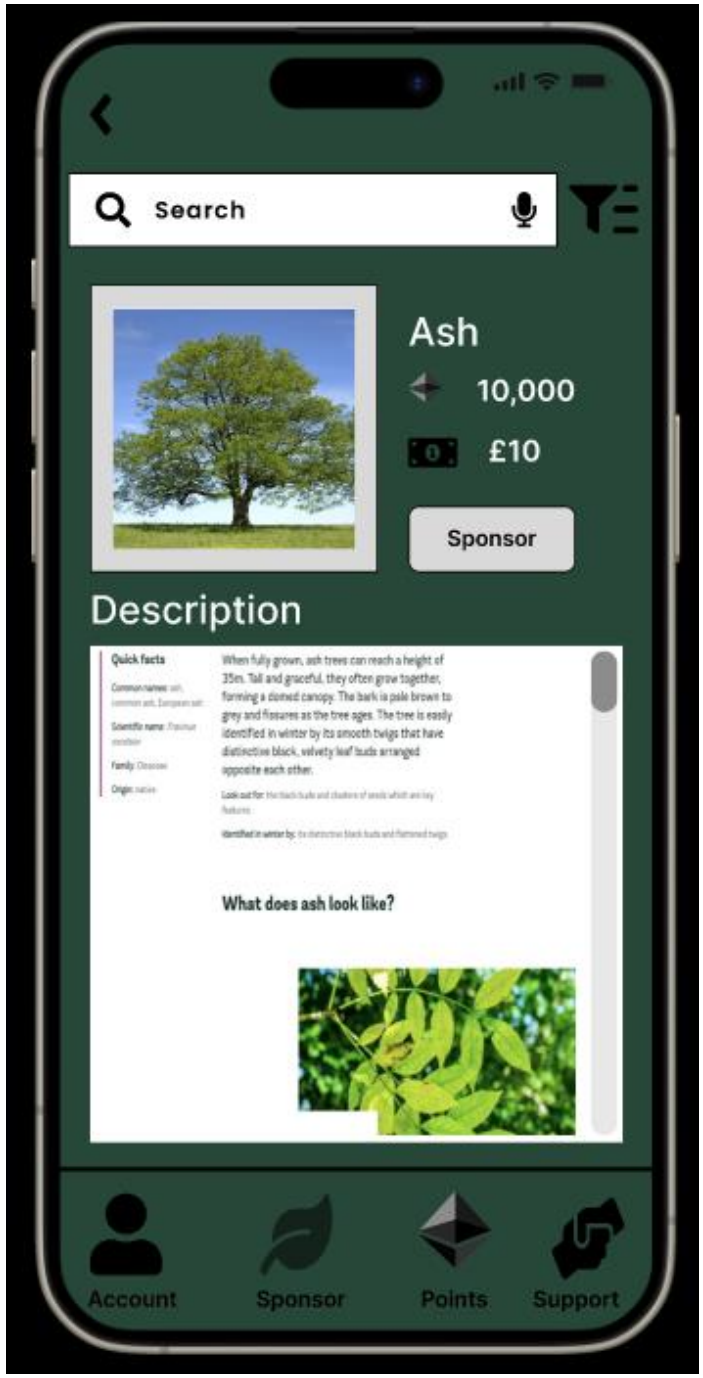


Figure 6: Further details of tree shown when Tree is selected

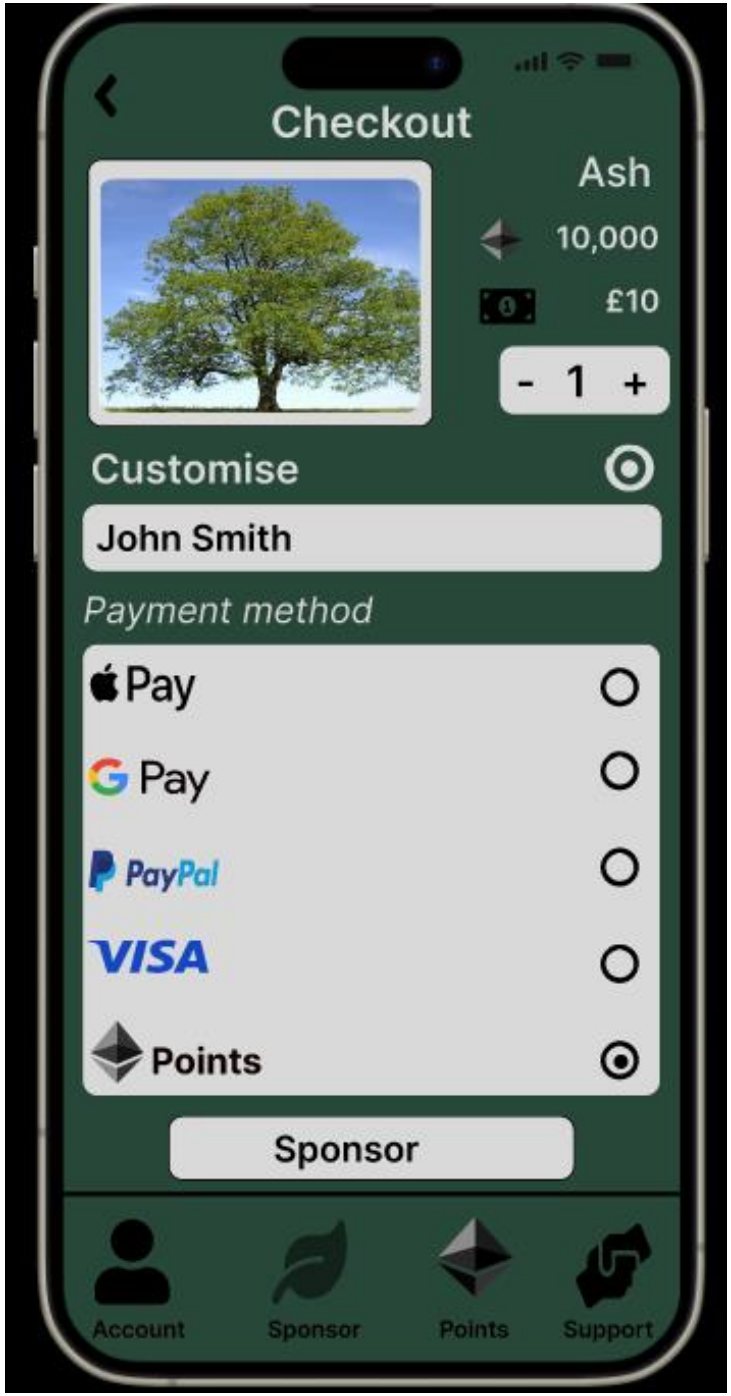


Figure 7: Checkout page

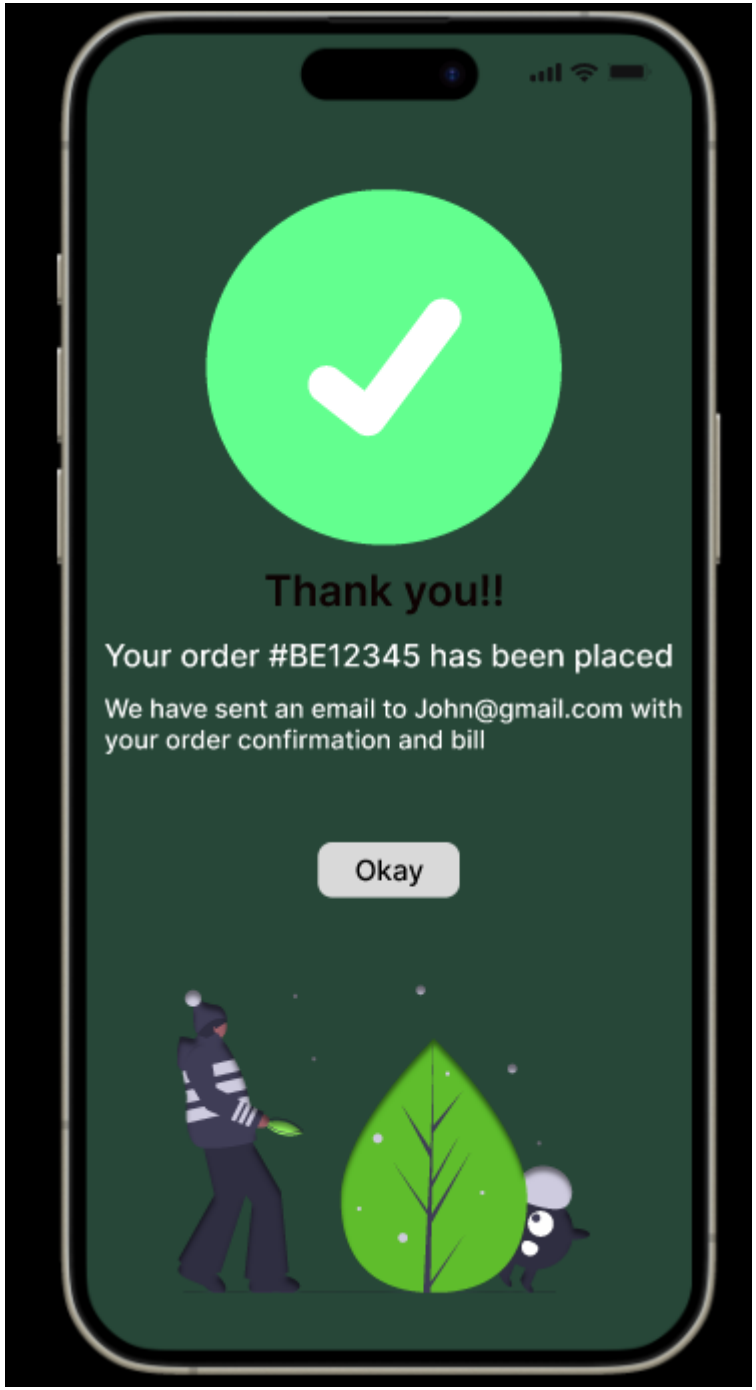


Figure 8: Confirmation page

Figure 9: Illustrating notification received when finishing cycling

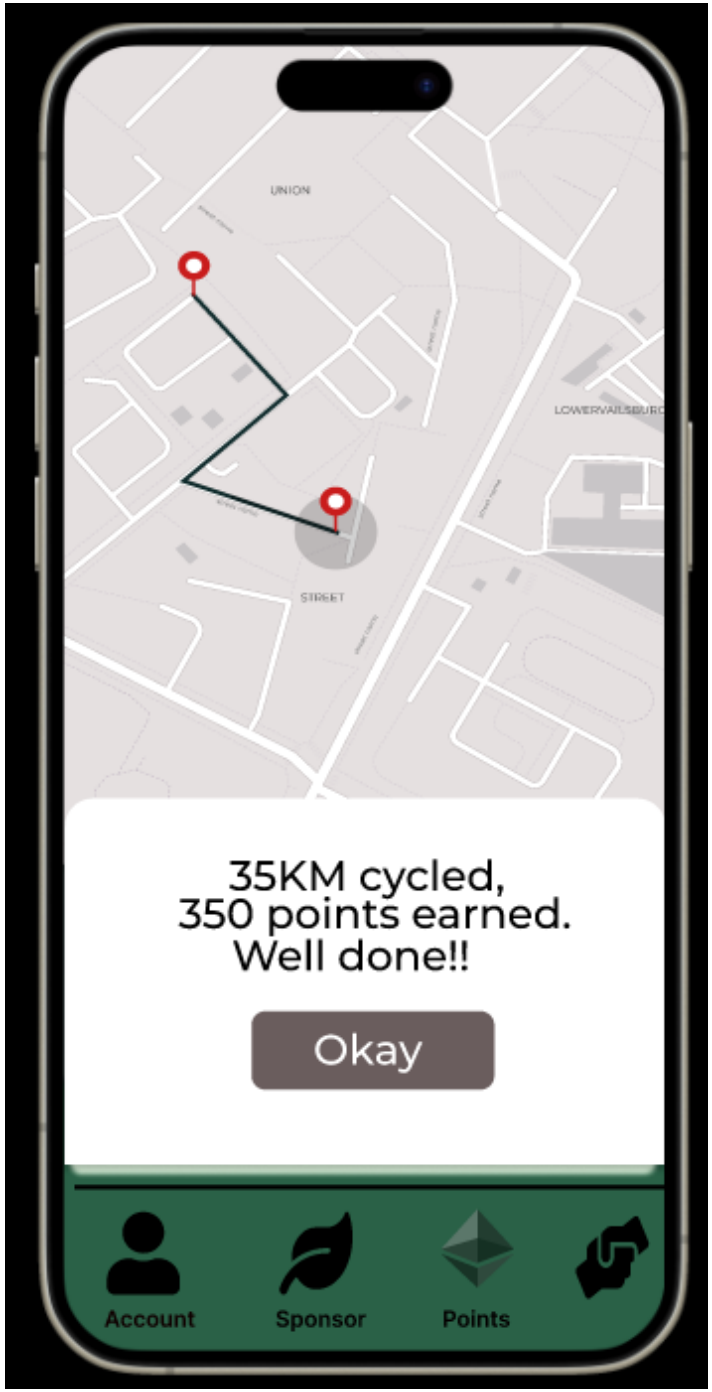


Figure 10: Map feature showing journey and other statistics

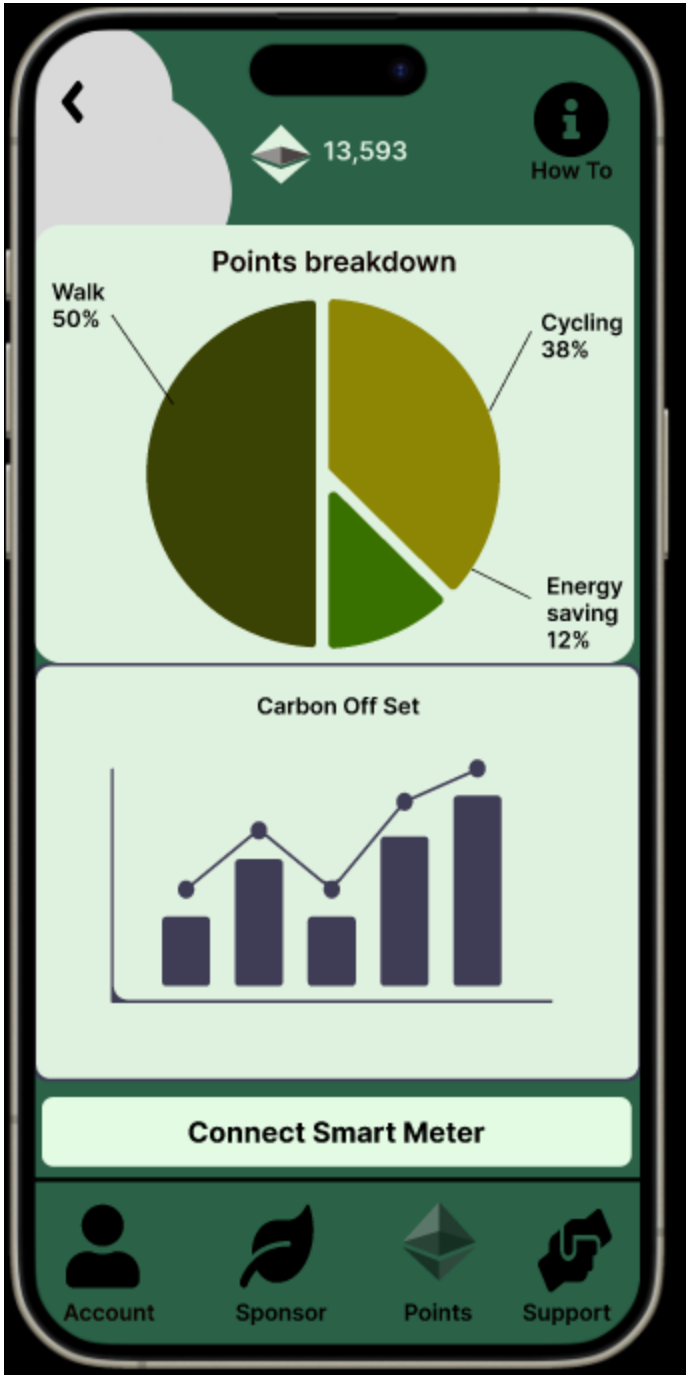


Figure 11: Points page

Figure 12: 'How to' pop up window

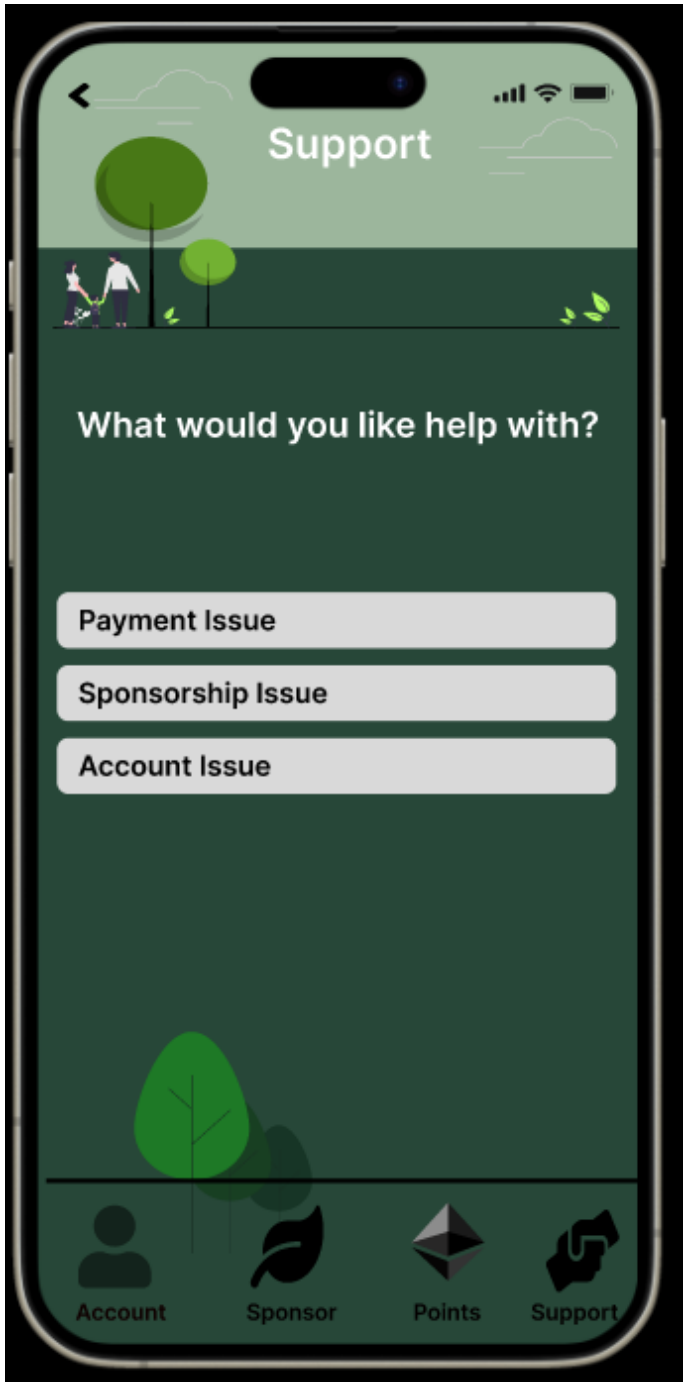


Figure 13: Support page

Interactive Prototype:
https://www.figma.com/proto/o3m7YXRHAOI5o8AJTA0ci9/Untitled?type=design&node-id=18-304&t=AHzgU1vbqrsmzrSH-0&scaling=scale-down&page-id=0%3A1&starting-point-node-id=8%3A5

Section F – Ethics and Professional Practice (Unit 8)

The ICEE/ACM Code of Ethics provides principles which must be followed to uphold ethical principles and prevent unethical behaviours. Our app tackles Birmingham's environmental initiative gap, by strictly upholding the IEEE/ACM Code of Ethics.

Privacy:

By adopting techniques such as hashing passwords and implementing measures to anonymise sensitive information like emails through secure practices (e.g. blocking with "*"), we prioritise user privacy which embodies section 1.6 of the ACM Code of Ethics, "Respect Privacy". Our commitment to employing robust encryption methods when handling user data ensures that personal data remains confidential and hidden from any attacks, upholding the principles of integrity and privacy protection as outlined in the Code of Ethics. Through this, we cultivate a secure environment that not only complies with best practices, but also respects and safeguards the privacy rights of our users.

Public Good:

Our project not only contributes to a sustainable future, but also follows the ethical responsibilities in the ACM Code of Ethics. We ensure that our project is centred around the interest of the Public which embodies section 3.1 of the ACM Code of Ethics, "Ensure that the public good is the central concern during all professional computing work". By addressing public interest through environmental sustainability, our project actively fulfils this ethical obligation by not only allowing users to actively engage in environmental conservation through planting trees, but also by promoting healthier lifestyles through walking or cycling, this way we address public interest with tangible, positive impacts.

Transparency and Accountability:

By providing users with clear and accessible information regarding our operations, data handling practices and environmental impact of our initiatives, we ensure transparency in all aspects of our computing work. We also keep accountability at the forefront of our project by establishing mechanisms for users to report concerns and provide feedback through our customer support email or phone number. These features reflect our commitment to section 2.8 of the ACM Code of Ethics "Access computing and communication resources only when authorised or when compelled by the public good". Through our transparent practices and a culture of accountability, we aim to foster a trusting and responsible environment for our users and stakeholders.

Accessibility:

Our system incorporates additional features such as text-to-speech functionality and increased font size in addition to the default accessibility features on devices. These accommodations have been set to ensure that users with additional needs can access and interact with our application seamlessly, adhering to the principles of section 1.4 of the ACM Code of Ethics, "Be fair and take action not to discriminate". Prioritising inclusivity through these features reflects our commitment to a fair user experience, regardless of different individual capabilities. Our vision extends beyond creating a platform tailored for a specific audience, we aim to cultivate a technological environment that cherishes diversity and embraces inclusivity.

Project Management and Moderation

Our team approached the project management process with a good strategy to ensure timely completion and high-quality deliverables. Each member played a significant role in different phases of the coursework, collaborating effectively to meet the set deadlines.

Additionally, the team proactively sought feedback to enhance the project's quality. To gather valuable insights and recommendations, we arranged structured feedback sessions with TAs. These sessions were crucial in identifying areas for improvement in our coursework. Here is a breakdown of each members contribution:

Name: Prince Alexander John **Email:** pxx276@student.bham.ac.uk **I.D:** 2427675

1. Provided the foundational groundwork for Non-functional Requirements in A2.
2. Actively participated in revamping A2 based on feedback from TAs.
3. Independently worked on the 'Sponsorship' use case in B2 and its scenario in B3.
4. Contributed to the initial production phase of E2 as well as setting up the environments and tools needed.
5. Attended all in person meetings and gave ideas on how to improve the project.

Name: Dami Olugbodi **Email:** dxo265@student.bham.ac.uk **I.D:** 2409165

1. Contributed significantly to conceptualizing various ideas for A2, specifically the Login Functionality.
2. Independently worked on half of the use cases and scenarios in B2 and B3 (before the TA's suggested to revise them)
3. Took the lead in creating initial templates for section D, later revising and refining these templates based on feedback.
4. Attended all in person meetings and gave ideas on how to improve the project.

Name: Bhuvan Praveen Kumar **Email:** bxp267@student.bham.ac.uk **I.D:** 2408367

1. Played an integral role in the creation of the original A2 and actively participated in its revision after advice from TAs.
2. Took the initiative and independently completed on both diagrams of B7 to high standards, implementing changes promptly following feedback.
3. Contributed to the initial production phase of E2 as well as the editing, dedicating after work hours to ensure its completion.
4. Attended all in person meetings and gave ideas on how to improve the project.

Name: Amar Purewal **Email:** axp1144@student.bham.ac.uk **I.D:** 2480763

1. Initially provided one of the use cases (Profile on B2), but later was removed due to changes advised from TAs.
2. Worked on providing component and deployment diagrams that were used for C1 C2
3. Worked on C3, rewrote many times due to change of diagrams due to feedback from TAs.
4. Attended all feedback sessions.

Name: Zakaria Hussein (Co-Scrum Master) **Email:** zxh023@student.bham.ac.uk **I.D:** 2344619

1. Independently worked and completed A1.
2. Collaborated on A2 and contributed significantly to writing up the team's ideas as well as making additions and improvements.
3. Independently worked and completed B1.
4. Independently worked on and completed the test plan for D as well contributing to the table.
5. Provided valuable input and effort towards E2.
6. Organised team meetings, ensuring that the group stayed aligned promoting progress.

7. Actively documented feedback received and communicated key points to the team.
8. Attended all in person meetings and gave ideas on how to improve the project.

Name: Muhammad Ali Shah (Co-Scrum master) **Email:** mas248@student.bham.ac.uk **I.D:** 2403348

1. Independently worked and completed B5 (which was the single biggest task)
2. Collaborated extensively on A2, C, and E2, contributing valuable insights and efforts toward these sections' completion.
3. Organized and managed the delegation of tasks, ensuring equitable distribution of workloads among team members.
4. Maintained detailed records of task assignments and progress, facilitating a clear understanding of ongoing activities within the team.
5. Oversaw the tracking of project progress, maintaining a comprehensive overview of the team's advancement toward set goals and deadlines.
6. Read other teammates work to ensure its up to standard and give them support.
7. Attended all in person meetings and gave ideas on how to improve the project.

Name: Awais Rafique **Email:** axr593@student.bham.ac.uk **I.D:** 2055367

1. Independently completed 3 of the tables in B2 and its associated scenario in B3
2. Independently worked on and completed B4 as well as B8.
3. Played a crucial role in collaboration across multiple sections: A2, B2, B3, D, E1, and E2, showcasing adaptability and versatility.
4. Organised all feedback sessions.
5. Volunteered repeatedly to help the team move forward and progress.
6. Read other teammates work to ensure its up to standard and give them support.
7. Independently formatted the final document for the team (with many revisions made along the way).
8. Attended all in person meetings and gave ideas on how to improve the project.

Name: Mohammed Adnan Hoq **Email:** mxh1223@student.bham.ac.uk **I.D:** 2437604

1. Independently worked on and completed B6 and F.
2. Collaborated and provided essential groundwork on A2, B5, and E2
3. Produced the creative and unique idea for E2 that would set the team apart from other teams.
4. Attended all in person meetings and gave ideas on how to improve the project.

No Of Feedbacks: 8

References

ACM (n.d.) *The Code ACM Code of Ethics and Professional Conduct*. Available at: https://www.acm.org/binaries/content/assets/membership/images2/fac-stu-poster-code.pdf.

Better (2016) *How Long Will it Take My New Red Maple to Grow to Full Size?* Available at: https://www.bhg.com/gardening/trees-shrubs-vines/trees/how-long-will-it-take-my-new-red-maple-to-grow-to-full-size/.

Lorenzo Stilo (2019) *ResearchGate | share and discover research*. Available at: https://www.researchgate.net/figure/Pie-chart-showing-the-result-of-the-survey-question-regarding-importance-of-safety_fig5_350340210.

Stephens, D. (2021) *A Guide to Green Ash Trees*. Available at: https://www.treekc.com/company-blog/a-guide-to-green-ash-trees.

Suwak, M. (2022) *How to Grow and Care for Oak Trees | Gardener's Path*. Available at: https://gardenerspath.com/plants/landscape-trees/grow-oak/.

Woodland Trust (2019) *A-Z of British trees*. Available at: https://www.woodlandtrust.org.uk/trees-woods-and-wildlife/british-trees/a-z-of-british-trees/.